UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

LIANNA MARA CASTRO DUARTE

FORENSE COMPUTACIONAL EM AMBIENTE DE REDE BASEADO NA GERAÇÃO DE ALERTAS DE SISTEMAS DE DETECÇÃO DE INTRUSOS AUXILIADO PELA ENGENHARIA DIRIGIDA POR MODELOS

São Luís

2012

LIANNA MARA CASTRO DUARTE

FORENSE COMPUTACIONAL EM AMBIENTE DE REDE BASEADO NA GERAÇÃO DE ALERTAS DE SISTEMAS DE DETECÇÃO DE INTRUSOS AUXILIADO PELA ENGENHARIA DIRIGIDA POR MODELOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão, como requisito para a obtenção do grau de mestre em Engenharia de Eletricidade.

Orientador: Ph.D. Zair Abdelouahab

São Luís

2012

Duarte, Lianna Mara Castro

Forense Computacional em Ambiente de Rede Baseado na Geração de Alertas de Sistemas de Detecção de Intrusos Auxiliado pela Engenharia Dirigida por Modelos / Lianna Mara Castro Duarte. – São Luís, 2012.

115 f.

Orientador: Ph.D. Zair Abdelouahab.

Impresso por computador (fotocópia).

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís, 2012.

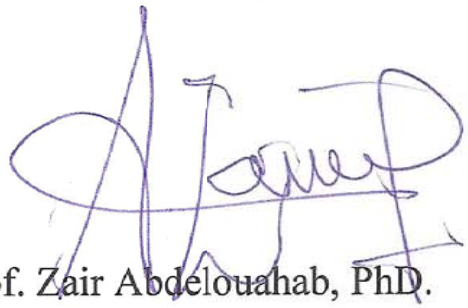
1. Redes de Computadores - Ferramentas de Segurança 2. Computação Forense 3. MDA 4. IDS I. Título.

CDU 004.056-53

FORENSE COMPUTACIONAL EM AMBIENTE DE REDE BASEADO NA GERAÇÃO DE ALERTAS DE SISTEMAS DE DETECÇÃO DE INTRUSOS AUXILIADA PELA ENGENHARIA DIRIGIDA POR MODELOS

Lianna Mara Castro Duarte

Dissertação aprovada em 19 de outubro de 2012.



Prof. Zair Abdelouahab, PhD.
(Orientador)

Prof. Slimane Hammoudi, Dr.
(Membro da Banca Examinadora)

Prof. Denivaldo Cicero Pavão Lopes, Dr.
(Membro da Banca Examinadora)

Aos meus pais Elda e Gilmar, a meu marido Márcio e minhas irmãs Lanna, Liannara e Lannara pelo apoio e incentivo em todos os momentos.

Resumo

Mesmo o grande progresso das técnicas utilizadas pelos sistemas de proteção como *firewalls*, sistemas de detecção de invasão e antivírus para detecção e prevenção de ataques, não são suficientes para eliminar a ameaça dos ciberataques. Mesmo ataques que existem há décadas ainda alcançam sucesso, e as vulnerabilidades bem conhecidas continuam a existir e reaparecer na Internet e redes corporativas [1]. As tecnologias de detecção de intrusão atuais fornecem informações ricas sobre um ataque. No entanto, o principal foco de detecção de intrusão centra-se no fato da segurança ter sido comprometida. A computação forense, por outro lado, tenta entender e explicar o que aconteceu com o ambiente de segurança e como uma violação de segurança pode acontecer [2]. No entanto, existe uma carência de mecanismos investigativos que possam trabalhar em sinergia com estes sensores e identificar não só os atacantes, mas as ações maliciosas que foram executadas. A falta de padronização no processo de realização da forense computacional e de rede [3], assim como a heterogeneidade das ferramentas e o fato de que os tipos de arquivos de *logs* dependem dos desenvolvedores, faz com que haja uma grande variedade nos formatos destes alertas de segurança. Além disto, o conhecimento empregado na investigação dos incidentes fica restrito aos analistas de segurança de cada caso. Esta dissertação propõe, de forma geral, o desenvolvimento de um modelo baseado na forense computacional que possa ser aplicado em ambiente de rede para trabalhar em conjunto com o IDS NIDIA [4] e IDSs heterogêneos associando aos alertas informações sobre procedimentos que podem ser executados para a investigação dos incidentes utilizando ferramentas existentes. A metodologia empregada para o desenvolvimento deste trabalho utilizou inicialmente de pesquisa bibliográfica para atingir os objetivos propostos, oriundas de livros, teses, dissertações, artigos científicos e documentos hipermídia, seguida de levantamento das informações para a elaboração da solução e uma análise de ferramentas que pudessem auxiliar no processo de modelagem e implementação do protótipo que foi auxiliado pela Arquitetura Dirigida por Modelos.

Palavras-chave: Computação Forense, MDA, IDS.

Abstract

Even the great progress of techniques used by protection systems as firewalls, intrusion detection systems and antivirus to detect and prevent attacks are not enough to eliminate the cyber-attacks threat. Known attacks for decades still achieve success, and well-known vulnerabilities continue to exist and reappear on the Internet and corporate networks [1]. The intrusion detection technologies we have today provide rich information about attacks. However, the main focus of intrusion detection focuses on the fact that security has been compromised. The computer forensics, on the other hand, attempts to understand and explain what happened to the security environment and how a security violation can happen [2]. However, there is a lack of investigative mechanisms to work synergistically with these sensors and identify not only the attackers, but the malicious actions that were performed. The lack of standardization in the process of computer and network forensics [3], as well as the heterogeneity of tools and the fact that the log/alert files depend on developers, causes a large variety in the formats of these security alerts. Moreover, the knowledge used in the incidents investigation still restricted to security analysts in each case. This work proposes, the development of a model based on computer forensics that can be applied in a network environment to work with IDS NIDIA [4] and heterogeneous IDSs associating information to alerts about procedures that can be performed to investigate the incident using existing tools. The methodology used to develop this was initially use literature to achieve the proposed objectives, derived from books, theses, dissertations, research papers and hypermedia documents, followed by the gathering of information for the development of the solution and analysis tools that could assist in the implementation and modeling the prototype, that was assisted by Model Driven Architecture.

Keywords: Computer Forensics, MDA, IDS.

Agradecimentos

Agradeço a Deus em primeiro lugar que nunca me abandonou nas horas mais difíceis.

Agradeço à minha família pelo apoio, incentivo e por não me permitirem desistir nos momentos de fraqueza. Especialmente a meu pai Gilmar e minha mãe Elda, pelo exemplo e pelo apoio em todos os momentos da minha vida.

Ao meu marido Márcio pelo amor, paciência e incentivo nestes anos em que estive ausente, me apoiando, conversando e cuidando de mim nos momentos difíceis (Te amo!).

Ao meu Orientador professor Zair Abdelouahab, pela oportunidade, pela orientação, paciência, compreensão e dedicação dispensadas à realização deste trabalho.

A minha avó Pedrina, meu exemplo de força, que sempre foi muito especial na minha vida e que partiu durante o desenvolvimento deste trabalho.

A meus tios e primos pelas orações, abraços e palavras de incentivo.

Agradeço a UFMA e ao Programa de Pós-Graduação em Engenharia de Eletricidade, especialmente ao Alcides sempre disposto a ajudar no que precisamos.

À CAPES pelo apoio e contribuição financeira dada durante todo o desenvolvimento do trabalho aqui apresentado.

À professora Dra. Maria da Guia, pelas dicas preciosas.

Agradeço a equipe do Laboratório de Sistemas em Arquiteturas Computacionais (LABSAC) da UFMA. Em especial, Eduardo Devidson, César, Ariel, Frederico, Mário e a principalmente a meu irmão de coração Raimundo Neto e a minha "bff" Valéria, que foram muito importantes nesta caminhada.

Agradeço especialmente aos amigos que fiz e que quero levar para a vida toda: Amélia, Jéssica e Vladimir, pelo apoio, pelas palavras de incentivo, pelas conversas, pelos abraços, sorrisos e lágrimas.

Agradeço a minha amiga Karina, a seus pais Francisca e Robinson e seu irmão Vinícius que abriram as portas de sua casa, e de sua família, me acolhendo e me dando todo suporte quando cheguei a São Luís.

Agradeço a equipe dos laboratórios LESERC e LSD da UFMA. Em especial a Ruy Guilherme, Berto e Márcio pelas palavras de incentivo.

Agradeço a equipe de limpeza da UFMA, especialmente a seu César pelo cuidado.

Agradeço à equipe da UAPI/UFPI pelo apoio, especialmente a meus companheiros Anathalia e Franklhes, pela parceria e pelo incentivo.

A todos os que contribuíram, incentivaram, torceram e lembraram de mim em suas orações, para que eu conseguisse concluir esta etapa da minha vida.

"Entrega teu caminho ao Senhor, confia nele, e o mais ele fará".

Salmo 37,5

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API *Application Programming Interface*.

APIDS *Application Protocol-Based Intrusion Detection Systems*.

ATL *Atlas Transformation Language*.

CAP *Common Alerting Protocol*.

CEE *Common Event Expression*.

CERT.br Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil.

CIDF *Comon Intrusion Detection Framework*.

CNA *Candidate Numeration Authority*.

CRUD *Create, Read, Update and Delete*.

CSIRT *Computer Security Incident Response Team*.

CVE *Common Vulnerabilities and Exposures*.

CWM *Common Warehouse Metamodel*.

DARPA *Defense Advanced Research Projects Agency*.

DFRWS *Digital Forensic Research Workshop*.

DSL *Domain Specific Language*.

EJB *Enterprise Java Beans*.

EJB-QL *Enterprise JavaBeans Query Language*.

EMF *Eclipse Modeling Framework*.

GNF *General Network Forensics*.

HIDS *Host-Based Intrusion Detection Systems*.

HQL *Hibernate Query Language*.

IDE *Integrated Development Environment*.

IDMEF *The Intrusion Detection Message Exchange Format*.

IDS *Intrusion Detection System*.

IETF *Internet Engineering Task Force*.

IIDB *Incidents of Intrusion and Forensic Information DataBase*.

IODEF *Incident Object Description and Exchange Format*.

IPS *Intrusion Prevention System*.

M2C *Mode-to-Code*.

M2M *Model-to-Model*.

MDA *Model Driven Architecture*.

MDE *Model Driven Engineering*.

MOF *Meta Object Facility*.

MVC *Model View Controller*.

NIDIA *Network Intrusion Detection System Based on Intelligent Agent*.

NIDS *Network-based Intrusion Detection Systems*.

OMG *Object Management Group*.

PIM *Platform Independent Model*.

PSM *Platform Specific Model*.

RADB *Reaction DataBase*.

SNF *Strict Network Forensics*.

STDB *Strategy DataBase*.

TMF *Textual Modeling Framework*.

UML *Unified Modeling Language*.

XMI *XML Metadata Interchange*.

XML *eXtensible Markup Language*.

Sumário

1 Introdução

Este capítulo descreve o cenário em que o trabalho foi desenvolvido trazendo: a problemática, a motivação, os objetivos e a apresentação dos capítulos desta dissertação.

1.1 Cenário e Definição do Problema

A sociedade hoje é cada vez mais dependente da infraestrutura tecnológica e uma boa ilustração deste fato é a dimensão que a Internet tem na vida moderna. Hoje, a comunicação, o comércio e entretenimento são quase todos baseados em sistemas de rede, onde as vantagens em se utilizar a rede mundial de computadores são inúmeras. Isto acontece, principalmente, devido ao rompimento das barreiras geográficas de espaço e tempo, o que possibilita compartilhar informações em tempo real.

Entretanto, estas facilidades e toda esta potencialidade que tornaram a internet um grande mercado em expansão, atraem também indivíduos maliciosos. Estes, enxergam na internet a possibilidade de executar ações ilícitas para os diversos fins, e desenvolvem métodos cada vez mais sofisticados para executar tais ações.

Carlos Lopes, diretor do Instituto das Nações Unidas para Formação e Pesquisa (UNITAR), afirma que pelo menos 431 (quatrocentas e trinta e um) milhões de pessoas no mundo foram diretamente afetadas, em 2011, por algum tipo de ataque cibernético [17]. Lopes afirma ainda, que os ataques cibernéticos estão experimentando "um crescimento exponencial" durante os últimos anos, principalmente, com ações contra centros de tecnologia de inteligência, que eram menos vulneráveis por apresentarem sistemas de segurança mais sofisticados que os demais [17].

No Brasil, o Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil (CERT.br) mantém estatísticas sobre notificações de incidentes a ele reportados. A Figura 1.1, que mostra os incidentes reportados ao CERT.br de 1999 a 2011, demonstra que o total de notificações recebidas em 2011 foi de 399.515 (quase três vezes maior que o total de 2010), o que comprova não só o crescimento dos incidentes,

como também a preocupação das organizações em descobrir o que de fato ocasionou tais incidentes.

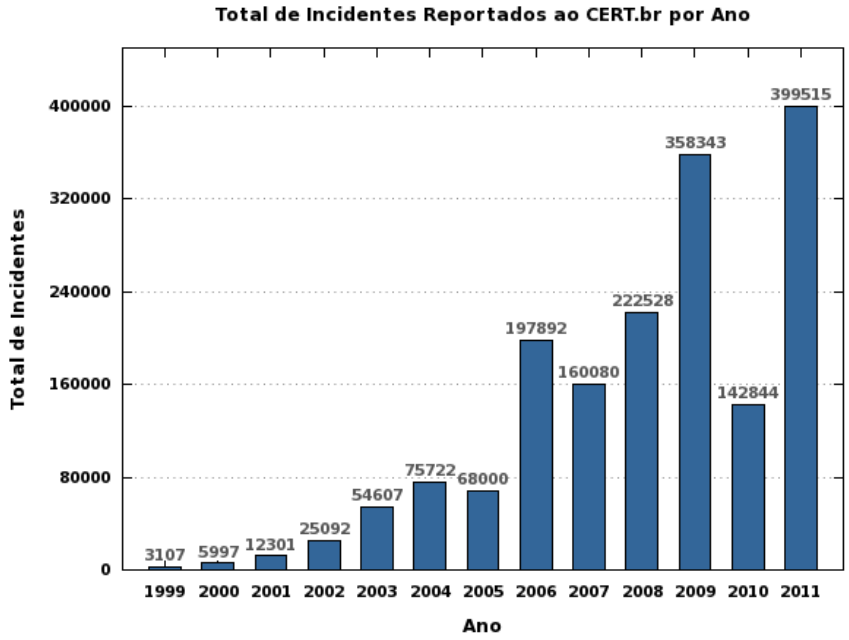


Figura 1.1: Incidentes reportados ao CERT.br em 2011 [5]

Dentro do contexto apresentado, percebe-se que, mesmo o grande progresso das técnicas utilizadas pelos sistemas de proteção como *firewalls*, sistemas de detecção de invasão e antivírus para detecção e prevenção de ataques, não são suficientes para eliminar a ameaça dos ciberataques. Mesmo ataques que existem há décadas ainda alcançam sucesso, e as vulnerabilidades bem conhecidas continuam a existir na Internet e redes corporativas, onde as melhores práticas de segurança da informação são constantemente ignoradas [1].

As tecnologias de detecção de intrusão atuais fornecem informações ricas sobre um ataque. No entanto, o principal foco de detecção de intrusão centra-se no fato da segurança ter sido comprometida. A análise forense, por outro lado, tenta entender e explicar o que aconteceu com o ambiente de segurança e como uma violação de segurança pode acontecer [2]. Todavia, existe uma carência de mecanismos investigativos que possam trabalhar em sinergia com estes sensores e identificar não só os atacantes, mas as ações maliciosas que foram executadas.

A falta de padronização no processo de realização da forense computacional e de rede [3], assim como a heterogeneidade das ferramentas e o fato de que os tipos de arquivos de *logs* dependem dos desenvolvedores, faz com que haja uma grande variedade nos formatos destes alertas de segurança. Este fato, dificulta, além da integração

de *Intrusion Detection System* (IDS)s diferentes em uma arquitetura unificada [2], a análise das informações, a extração destas de forma clara [18], e o compartilhamento das informações entre as instituições de combate ao crime. Além disto, o conhecimento empregado na investigação dos incidentes fica restrito aos analistas de segurança de cada caso.

1.2 Objetivo Geral e Específicos

Esta dissertação propõe, de forma geral, o desenvolvimento de um modelo baseado na forense computacional que possa ser aplicado em ambiente de rede para trabalhar em conjunto com o IDS *Network Intrusion Detection System Based on Intelligent Agent* (NIDIA) [4] e IDSs heterogêneos associando aos alertas informações sobre procedimentos que podem ser executados para a investigação forense dos incidentes. De modo mais específico, espera-se:

- Propor um modelo, baseado nas etapas de forense computacional, que dentro de um ambiente preparado para forense de rede, possa trabalhar sinergicamente com o IDS;
- Apresentar uma estratégia para a padronização dos alertas de segurança da rede que agregue soluções para a investigação dos eventos utilizando *Model Driven Architecture* (MDA);
- Melhorar, baseado nas técnicas aplicadas de forense de rede, a base de conhecimento e prover respostas aos incidentes de segurança;
- Demonstrar o uso da forense computacional em ambiente de rede como complemento aos mecanismos tradicionais de segurança.

1.3 Organização do Trabalho

Esta dissertação está estruturada em 6 (seis) capítulos os quais são descritos a seguir:

Este primeiro capítulo que trata de uma breve introdução sobre o cenário, objetivos a serem atingido; o segundo capítulo que apresenta o referencial teórico necessário para o entendimento deste trabalho; no terceiro capítulo é feito um breve resumo de algumas soluções apresentadas na literatura semelhantes ao que está sendo proposto; no quarto capítulo é realizada a descrição da abordagem proposta sua modelagem; no quinto capítulo é apresentada a estratégia de implementação da solução proposta e o estudo de caso; no sexto capítulo são feitas as considerações finais e sugestões para trabalhos futuros.

O trabalho é finalizado com a apresentação das referências bibliográficas que fundamentaram teoricamente o estudo proposto.

2 Fundamentação Teórica

Este capítulo tem como objetivo apresentar uma visão geral dos conceitos fundamentais que serviram de base para o desenvolvimento deste trabalho. Inicialmente, faz-se um levantamento sobre os conceitos de segurança da informação. Em seguida, são abordados os conceitos e objetivos da forense computacional. Em continuidade ao capítulo, apresenta-se os conceitos sobre sistemas de detecção de intrusos e alertas de segurança (modelos existentes). O capítulo é finalizado com os conceitos de *Model Driven Engineering* (MDE) e MDA relevantes para a padronização dos alertas através de transformações e desenvolvimento dos componentes com a geração de código dos modelos.

2.1 Segurança da Informação

Com o crescimento das redes de comunicação, globalização dos negócios, e expansão de mercados, a maioria das organizações torna o funcionamento de seus negócios cada dia mais dependentes da Internet. Toda a tecnologia empregada para isto traz muitos benefícios, que auxiliam empresas a apoiar as atividades do dia a dia. No entanto, apareceram também indivíduos mal intencionados, e grupos criminosos buscando atacar empresas e indivíduos ou cometer algum tipo de fraude, o que tornou fundamental a adoção de procedimentos para garantir a segurança da informação.

Segurança da Informação é a proteção da informação de diversos tipos de ameaças (possibilidade de um agente ou fonte de ameaça [1] explorar acidentalmente ou propositalmente uma vulnerabilidade [2] específica) para garantir a continuidade dos negócios, minimizar os danos, maximizar o retorno dos investimentos e as oportunidades de negócio [20] [21].

[1] uma intenção e método objetivando a exploração de uma vulnerabilidade ou uma situação e método que pode acidentalmente disparar uma vulnerabilidade [19]

[2] falha ou fraqueza de procedimento, *design*, implementação, ou controles internos de um sistema que possa ser acidentalmente ou propositalmente explorada, resultando em uma brecha de segurança ou violação da política de segurança do sistema [19]

O principal propósito da segurança da informação é preservar três características básicas: confidencialidade (os dados são acessados por quem tem direito a acessa-los), integridade (dados devem ser realmente verdadeiros para que possam ser processados corretamente) e disponibilidade (os dados devem ser acessados quando necessário) [20] [21].

Muitas empresas acreditam que apenas com a compra de equipamentos e *softwares*, podem criar uma infraestrutura segura. No entanto, *firewalls*, IDS, antivírus, são apenas algumas das ferramentas disponíveis para ajudar a proteger uma rede e seus dados [22]. A segurança envolve, além dos aspectos tecnológicos e processuais, aspectos humanos, jurídicos e de negócio [23], o que torna fundamental a adoção de políticas de segurança, a utilização e configuração adequada das ferramentas de segurança adotadas (*firewall*, IDS/*Intrusion Prevention System* (IPS), etc.), a implantação de algoritmos de criptografia, entre outros. Além disto, é fundamental que sejam feitos o monitoramento adequado e verificação contínua da integridade do sistema, além da utilização de procedimentos de *backup*, para possibilitar a proteção da informação em diversos níveis [24].

Segundo Fry [1], uma segurança de sucesso dentro da empresa demanda foco e este acompanhamento focado requer ajustes consideráveis e um rápido reconhecimento do tráfego benigno, onde sem ajuste e uma forte compreensão do ambiente, a equipe de monitoramento de rede poderá perder tempo e não encontrar soluções alternativas.

No entanto, por mais que controles e ferramentas sejam implantados para proteger os ativos (qualquer coisa que tenha valor para a organização [20]) da rede, é importante ressaltar que um sistema nunca está totalmente seguro, pois à medida que os sistemas de segurança se sofisticam, as técnicas de invasão evoluem e, na grande maioria das vezes, mesmo com a geração de alertas dentro da infraestrutura de segurança, estes não são averiguados de forma adequada [25].

Neste contexto, organizações tem a responsabilidade de manter um controle completo sobre os dados e informações relevantes que ficam armazenados em seus equipamentos. Isto faz com que seja necessária a implantação de estratégias que possam prover subsídios para que procedimentos investigativos sejam executados e focados na necessidade e no modelo de segurança implantado pela organização [26].

2.1.1 Gestão de Riscos e Políticas de Segurança e Ataques

Na grande maioria das organizações é preciso demonstrar a necessidade de manter a informação segura e provar a importância de investir em procedimentos de segurança para os ativos. Uma forma de mostrar esta necessidade é analisar os riscos [3] e vulnerabilidades.

A gestão de riscos em segurança da informação é uma das dimensões que o processo de segurança deve tomar dentro de uma organização [27] [28]. Ela compõe um conjunto das dimensões que possibilitam que o processo de segurança da informação aconteça de maneira eficiente, eficaz e contínua ao longo do tempo [28]. Vários padrões utilizados em segurança [29] [27] [28] [30] dão suporte ao processo de gestão de riscos e o abordam como um dos principais pontos para a definição do sistema de gestão de segurança da informação.

O campo da segurança da informação é bastante jovem (comparado a indústrias como economia, finanças, e seguros), e quando tenta-se aplicar uma disciplina madura como a gestão de riscos podem existir lacunas que precisam ser superadas [21]. O risco é definido pela ISO\IEC27002 [27] como a possibilidade de uma determinada ameaça explorar vulnerabilidades de um ativo ou de um conjunto de ativos de maneira a prejudicar a organização. A medida de risco é definida como a combinação da probabilidade de um evento indesejável e a sua consequência.

Teoricamente, riscos de maior prioridade, considerando o aspecto do impacto (mudança adversa no nível obtido dos objetivos de negócios [28]), deverão ser minimizados. Desta forma, para Beal [31], a gestão de risco é o conjunto de processos que permite às organizações identificar e implementar as medidas de proteção necessárias para diminuir os riscos a que estão sujeitos os seus ativos de informação, e equilibrá-los com os custos operacionais e financeiros envolvidos.

Mesmo que existam várias ameaças de segurança orientadas para o mesmo ativo, cada uma terá um risco diferente e cada um vai precisar de uma avaliação de risco diferente [32]. Uma vez que os riscos forem definidos é possível determinar onde maiores esforços deverão ser empregados e como focar o monitoramento destes de forma a direcionar adequadamente os esforços e recursos para evoluir continuamente

[3] probabilidade de uma fonte de ameaça explorar uma vulnerabilidade, resultando em um impacto para a organização [19]

o sistema de gestão de segurança da organização. Ou seja, teremos metas para a segurança que deverão ser comunicadas a todos os usuários, equipes e gerentes através de um conjunto de regras de segurança (política de segurança).

Uma política de segurança é um instrumento importante para proteger uma organização contra ameaças à segurança da informação que a ela pertence ou que está sob sua responsabilidade. Uma ameaça à segurança é compreendida neste contexto como a quebra de uma ou mais de suas três propriedades fundamentais (confidencialidade, integridade e disponibilidade).

O objetivo de uma política de segurança da informação é prover uma orientação e apoio da direção para a segurança da informação, de acordo com os requisitos do negócio e com as leis e regulamentações pertinentes [28]. A política deve especificar os mecanismos através dos quais os requisitos de segurança possam ser alcançados. Outro propósito é oferecer um ponto de referência a partir do qual se possa adquirir, configurar e auditar sistemas computacionais e redes [27] de forma que uma tentativa de utilizar um conjunto de ferramentas de segurança na ausência de pelo menos uma política de segurança implícita não faz sentido.

No contexto deste trabalho as políticas estão focadas na implementação por sensores de segurança dentro da rede e o monitoramento é feito na verificação dos eventos gerados por características determinadas. Um evento de segurança da informação é uma ocorrência identificada de um estado de sistema, serviço ou rede, indicando uma possível violação da política de segurança da informação ou falha de controles, ou uma situação previamente desconhecida que possa ser relevante para a segurança da informação [20], [27].

Um incidente de segurança pode ser um simples ou uma série de eventos de segurança da informação indesejados ou inesperados, que tenham uma grande probabilidade de comprometer as operações do negócio e ameaçar a segurança da informação [20], [27]. Ressaltamos que os alertas de segurança, também serão tratados como eventos de segurança no contexto deste trabalho.

Percebe-se então, que independentemente do modelo e padrões de segurança adotados por uma organização, a gestão de riscos é uma forma importante para avaliar como deve ser feita a segurança dos ativos. Ela pode ser usada também como uma estratégia para melhor conhecer os ativos do negócio e direcionar os recursos e

esforços para determinar os controles que possam ser implementados na implantação do sistema de gestão de segurança da informação.

Para pensar na proteção, organizações precisam identificar os ativos a serem protegidos e ponderar seus valores em relação ao negócio. Desta forma, podem ser definidas as políticas e os controles que serão implementados para minimizar os riscos dos ativos e o impacto que seu comprometimento pode trazer à organização. No contexto deste trabalho o processo de gestão de riscos é importante para a preparação do ambiente, onde os ativos a serem monitorados devem ser analisados para direcionar este monitoramento para perfis específicos de ameaças/ataques.

Desta maneira, quando analisamos o que exatamente faz um ataque podemos dividi-lo de acordo com o tipo de ataque que ele representa, o risco que representa, e os controles que podem ser utilizados para mitigá-los [32]. Este processo auxilia o gerenciamento de ameaças de segurança, que é uma técnica utilizada no auxilio ao gerenciamento dos sistemas de segurança de uma organização, para analisar os relatórios dos sensores de monitoramento, como os IDSs, *firewall* e outros sensores de varredura [25].

Este gerenciamento pode ajudar a reduzir os falsos positivos dos sensores, desenvolver técnicas rápidas de resposta para contenção e avaliação de ameaças, correlacionar e escalar falsos positivos através múltiplos sensores e plataformas, e desenvolver análise intuitiva, executar a computação forense e gerar relatórios de gerenciamento [32]. Dentre as técnicas utilizadas para gerenciamento de ameaças de segurança estão: a avaliação de risco e análise forense [32].

2.2 Forense Computacional

No contexto das redes de computadores a coleta de eventos é sempre necessária para apoiar as investigações e é recomendado que mesmo quando não está sendo feito o monitoramento ativo dos eventos, os mesmos sejam coletados para apoiar respostas a incidentes [20] [33] [34] [35]. Para ajudar a determinar os melhores alvos para monitoramento de segurança, ou seja, elicitar as fontes que devem ser monitoradas, as políticas de segurança devem ser bem definidas e analisadas constantemente, além da

documentação da rede como um todo. Deve-se conhecer muito bem o ambiente a ser monitorado [1].

A Computação Forense busca por evidências para reconstruir as ações que levaram o sistema de um estado seguro ao momento que a invasão foi detectada. Ela consiste de uma série de técnicas e procedimentos, realizados metodologicamente, para coleta de evidências a partir de equipamentos de informática e vários dispositivos de armazenamento em mídia digital, que podem ser apresentados em um tribunal em um formato coerente e significativo [36].

Com o uso das técnicas de computação forense pode-se rastrear invasores além de atacar criminosos digitais; fazer complementação em *firewall* e sistemas de detecção de invasão; montar uma estrutura complementar de segurança da rede e que trabalhe sinergicamente com os sensores de segurança da rede. No entanto, não existe padronização entre os sistemas e metodologias para forense computacional, e em geral, cada novo sistema para esta finalidade não se baseia nos sistemas anteriores, em vez disso, cada novo projeto começa tudo de novo [3].

A computação forense é dividida principalmente em forense estática e forense dinâmica. A forense estática coleta e analisa os dados depois que o crime foi cometido (*post-morten*). Já a forense dinâmica, baseada em rede, captura e analisa os dados na rede depois que uma evidência parcial do crime for obtida [37].

Ferramentas e técnicas forenses são mais frequentemente utilizadas no contexto de investigações criminais e tratamento de incidentes de segurança usados para responder a um evento pela investigação dos sistemas suspeitos coletando e preservando evidências, reconstruindo eventos e avaliando o estado atual de um evento [38] [34]. Porém, técnicas e ferramentas forenses também são úteis para muitos outros tipos de tarefas como:

- **Solução de Problemas Operacionais**: encontrar a localização física e virtual de uma máquina com configuração de rede incorreta, resolução de problemas funcionais com aplicações, registro e análise do sistema operacional atual e definições de configuração para um *host* se configuram como alguns destes problemas;

- **Monitoramento de *Logs***: análise de entradas de registro e correlacionamento de *logs* entre vários sistemas. Isto pode ajudar a lidar com os incidentes, identificar violações de políticas, auditoria e outros esforços [34];

- **Recuperação de dados**: existem dezenas de ferramentas que podem recuperar dados perdidos de sistemas incluindo os dados que foram apagados acidentalmente ou propositalmente ou ainda modificados [34];

- **Aquisição de dados**: algumas organizações usam ferramentas de análise forense para coletar dados de *hosts* que estão sendo redistribuídos ou recolhidos. Por exemplo, quando um usuário deixa a organização, os dados da estação de trabalho dos usuários podem ser coletados e armazenados, caso seja necessário utilizalos no futuro. A mídia das estações de trabalho podem então ser higienizadas para remover todos os dados originais do usuário [34].

- **Investigação e Auditoria / Conformidade Regulamentar**: informações confidenciais devem ser protegidas e certos registros devem ser mantidos para fins de auditoria. Além disso, quando as informações protegidas são expostas a outras partes (em caso de fusões e aquisições), as organizações podem ser obrigados a notificar outros órgãos ou indivíduos afetados. A forense pode ajudar as organizações a realizarem a auditoria e cumprir tais requisitos [34].

2.2.1 Forense de Rede

A Forense de Rede é uma parte especial da Forense Computacional usada para descrever a tarefa de analisar informações coletadas em redes ativas a partir de invasões diferentes, auditorias e capacidade de monitoramento para o propósito de proteção [39], [16]. Ela geralmente se refere à coleta e análise dos dados da rede: tráfego da rede; *logs* de *firewalls* e IDSs, etc. [3], [16]. Ou seja, ela está voltada para a forense digital em ambientes de rede.

Palmer [16] a define como o uso de técnicas cientificamente comprovadas para: coletar, fundir, identificar, examinar, correlacionar, analisar e documentar evidências digitais. O autor afirma ainda que estas ações são executadas a partir de processamentos de múltiplos ativos e fontes de transmissão digital. O propósito é desvendar fatos relacionados com intenções planejadas, ou mensurar o sucesso de atividades

não autorizadas, que possam significar *romper, corromper e / ou comprometer* componentes do sistema. As informações podem ainda ser utilizadas para auxiliar na resposta ou recuperação destas atividades.

De acordo com Ren e Jin [40], existem dois tipos de análise forense de rede: *General Network Forensics* (GNF) e a *Strict Network Forensics* (SNF). Na GNF o propósito é realizar a análise apenas para aumentar a segurança e descobrir os personagens da rede que podem guiar estratégias de gerenciamento de *firewalls* e IDSs. Já a SNF é a intersecção entre a informática e a ciência forense propriamente dita. Esta tem com o objetivo satisfazer os princípios legais e incluir etapas que satisfaçam o processo legal, utilizando-se de técnicas de computação e de rede que possam ser validadas judicialmente.

Para identificar ataques a forense de rede pode lidar com a captura e inspeção dos pacotes que passam por um determinado nó da rede, de forma que os pacotes possam ser inspecionados *online* ou armazenados em disco para análise posterior, respeitando a preservação dos dados antes de sua manipulação. Analistas podem usar os dados do tráfego da rede para reconstruir e analisar ataques na rede e o uso inapropriado dela, assim como resolver vários tipos de problemas operacionais.

O conteúdo das comunicações carregado pelas redes, assim como mensagens de *email* ou áudio, também pode ser coletado para dar suporte a investigações [41]. Para atingir tal objetivo é necessário que sejam implementados controles que possibilitem que os dados da rede sejam analisados, uma vez que a natureza da rede é dinâmica. Ou seja, a rede precisa estar preparada para que forense computacional possa extrair informações suficientes e válidas.

No Capitulo 4 deste trabalho é proposto um modelo de forense computacional em ambiente de rede baseado no monitoramento de *logs* do IDS, este usado como gatilho inicial de acordo com perfis predefinidos de ataque. Associado a estes perfis, consideramos o modelo do ataque e o modelo de segurança do ambiente para coletar informações nos demais sensores atribuídos ao *host* afetado pelo evento. O modelo proposto utiliza-se de técnicas e ferramentas para forense computacional de rede para lidar com os incidentes de segurança, gerar alertas para outras IDSs (através de transformações dos alertas) ou, até mesmo, a criação de um roteiro de ações forenses a serem executadas.

2.3 Sistemas de Detecção de Intrusos e Alertas de Segurança

Detecção de invasão é o processo de monitorar os eventos que ocorrem em um sistema de computador ou rede e analisa-los para sinalizar possíveis incidentes, que são violações ou ameaças iminentes de violações de políticas de segurança, uso aceitável de políticas ou práticas padronizadas de segurança [41] [32].

Um Sistema de Detecção de Intrusão (IDS) é o *software* que automatiza o processo de detecção de invasão. Existem vários tipos de IDSs, incluindo IDSs baseadas em *hosts* (*Host-Based Intrusion Detection Systems* (HIDS)), IDS baseada em protocolos de aplicação (*Application Protocol-Based Intrusion Detection Systems* (APIDS)), e IDS baseado em rede (*Network-based Intrusion Detection Systems* (NIDS)) [32] [4]. Estes ainda podem ser classificados quanto à maneira que o IDS processa os dados para que seja feita a identificação das atividades intrusivas:

- Detecção por Abuso: está relacionada a identificação de comportamentos intrusivos correspondentes a exploração de vulnerabilidades conhecidas, comumente chamadas de assinaturas de ataque [4]. O objeto principal da detecção por abuso foca em usar um sistema especialista para identificar intrusões baseado em uma base de conhecimento pré-determinado [4] [42] [43] [44] e podem ainda ser classificados como:
  - Baseado em Assinatura: assinaturas ou atributos que caracterizam um ataque são armazenados para referência em análises posteriores para identificar intrusões [43];
  - Baseado em Regras: sistema baseado em regras utiliza um conjunto de "*if-then*" de regras para caracterizar ataques informáticos [43];
  - De transição de estado: nesta abordagem IDSs tentam identificar a intrusão usando uma máquina de estado finito que é deduzida a partir da rede. Os estados do IDS correspondem a diferentes estados da rede e um evento "transita" nesta máquina de estados finita. Uma atividade identifica uma intrusão se as transições de estado na máquina de estados finita da rede refletir a um estado de sequela [43].

- Detecção por análise de estado de protocolo: este método compara perfis predeterminados de definições geralmente aceitas de atividade benignas de um protocolo para cada estado protocolo com eventos observados para identificar desvios [43];

- Detecção por Anomalia: este método é baseado na observação de desvios de comportamentos esperados em atividades relevantes dos usuários ou processos do sistema monitorado, onde o detector de anomalia deve ser capaz de distinguir entre a anomalia e o comportamento normal [4] [43] [45]. Esta categoria pode utilizar-se de vários métodos para detecção de comportamentos anômalos como:

  - Métodos estatísticos: comportamento do usuário na rede é monitorado medindo as estatísticas certas variáveis ao longo do tempo [43];
  - Baseados em distância: tentam superar as limitações da abordagem estatística quando os dados são difíceis de estimar nas distribuições multidimensionais [43];
  - Baseados em Regras: em sistemas baseados em regras, o IDS tem definido o conhecimento do comportamento normal do usuário/rede e intrusões são identificadas por comparação deste comportamento predefinido como normal com as atividades atuais do usuário/rede [43];
  - Baseados em perfis: similar ao método baseado em regras, no entanto, o perfil de comportamento normal é construído para diferentes tipos de tráfegos de rede, usuários e todos os dispositivos e desvios destes perfis significam intrusões [43];
  - Baseados em modelos: outras abordagens baseadas em desvio normal e comportamento anormal é modelá-las, mas sem criar vários perfis para eles. Nos métodos baseados em modelo, os pesquisadores tentam modelar os comportamentos normais e/ou anormais de forma que os desvios entre estes significam intrusões [43].

IDSs baseados em assinatura são incapazes de caracterizar ataques lentos que se estendem em um período de tempo longo. Além disto, apenas ataques cujas assinaturas estejam armazenados em seu banco de dados podem ser detectados. No entanto a detecção por assinatura apresenta uma taxa baixa de falsos positivos.

Já IDSs baseados em anomalia tem a capacidade de detectar atividades maliciosas desconhecidas. No entanto ataques que não alterem significantemente as características do sistema podem passar despercebidos, além da potencialidade de gerar uma alta taxa de falsos positivos (geralmente mais elevada do que os baseados em assinatura) [4] [44] [45]. Quando é detectada uma invasão IDSs geram alertas para que respostas/providências sejam tomadas a cerca da ação detectada.

2.3.1 Alertas de Segurança

Alertas e *logs* de segurança são algumas das peças mais importantes entre os dados analíticos da infraestrutura de segurança. No entanto, a heterogeneidade das ferramentas e o fato de que os tipos de arquivos de *logs* dependem dos desenvolvedores, faz com que haja uma grande variedade nos formatos destes alertas. Desta forma, vários esforços já foram e são empregados para padronização destes artefatos [46], [47], [48], [49], entre outros.

O *Comon Intrusion Detection Framework* (CIDF) foi um esforço desenvolvido na tentativa de permitir que diferentes componentes de detecção e resposta a intrusão interoperassem e compartilhassem informações e recursos de forma que componentes para detecção de invasão pudessem ser reusados em outros sitemas [46]. Era um projeto de pesquisa da *Defense Advanced Research Projects Agency* (DARPA) para a utilização por pesquisadores, já foi descontinuado e influenciou a criação do *The Intrusion Detection Message Exchange Format* (IDMEF)

O IDMEF tem o propósito de ser um formato de dados padrão que os sistemas de detecção de intrusão podem usar para comunicar alertas. Este formato é uma proposta do *Internet Engineering Task Force* (IETF) e é definido na RFC 4766 [50], onde é justificada a necessidade de uma arquitetura e de comunicação padronizada dos sistemas de detecção de intrusos.

Outro esforço com objetivo de complementar o IDMEF é o *Incident Object Description and Exchange Format* (IODEF) [48]. Ele tem o propósito de definir uma representação de dados, provendo um *framework*, para o compartilhamento de informações referentes a incidentes de segurança, comumente trocadas entre grupos de resposta a incidentes de segurança em computadores [51] [48].

Outra proposta de padronização dos dados é o *Common Event Expression* (CEE) [49]. Este busca padronizar a forma como eventos em computadores são descritos, logados e trocados [49]. Segundo Chuvakin [52] este padrão tem uma chance maior que qualquer padrão de ser adotado por ter aprendido as lições das outras tentativas (por isto não tem foco na compatibilidade com os demais esforços de padronização) e considera o IDMEF o como um padrão que falhou. A iniciativa CEE é um esforço recente liderado pelo Mitre, e, atualmente, conta com membros de empresas renomadas como: Cisco, HP/ArcSight, McAfee, NIST, and Microsoft, além de alguns produtos como Tripwire e Sensage já iniciaram o uso de versões do CEE em seus produtos.

A iniciativa CEE recomenda que a indústria coordene quatro áreas para facilitar a transmissão e interpretação de *logs* [49]:

- Criar uma taxonomia de evento para definições de *logs* uniformes e precisos que leve a uma representação comum do evento;
- Criar sintaxes de *log* utilizando um único dicionário de dados para fornecer detalhes específicos do evento de forma consistente;
- Padronizar mecanismos flexíveis de transporte de eventos para dar suporte a vários ambientes;
- Propor recomendações de *log* para os eventos e atributos gerados pelos dispositivos.

Já o *Common Alerting Protocol* (CAP) [53] é um formato genérico que tem como objetivo principal prover uma única mensagem que tenha a capacidade de ativar todos os tipos de sistemas de alerta. Esses alertas de emergência têm por finalidade informar ao público sobre possíveis perigos ou ameaças que possam causar algum tipo de dano simples e abrangente para a troca de todos os tipos de alertas de emergência [54]. Este formato foi estendido por [51] como proposta de codificação dos alertas emitidos por grupos de resposta a incidentes de segurança em computadores no projeto do IDS NIDIA.

Outro formato importante referente aos alertas é o *Common Vulnerabilities and Exposures* (CVE). Este surgiu da necessidade de padronizar os relatos de vulnerabilidades [55], e atualmente é fortemente adotado pelos produtos de segurança. O objetivo do CVE é facilitar o compartilhamento de dados e a comparação entre os diversos

bancos de informação sobre vulnerabilidades existentes. Basicamente, o problema encontrado era o uso do nome da vulnerabilidade como índice de pesquisa nessas bases.

No padrão CVE, um relato de vulnerabilidade passa por um processo de verificação de veracidade das informações e coleta de mais fatos sobre a vulnerabilidade. Esse processo tem as seguintes fases [56]:

- Descobrimento: o potencial ponto de exposição é descoberto;
- Publicação: a informação é anunciada publicamente como uma candidata a vulnerabilidade. Após o anuncio a informação é enviada para o CVE *Candidate Numeration Authority* (CNA);
- Identificação: a candidata é verifica para se saber se já não é conhecida, evitando duplicações. Caso não seja, é numerada com um identificador chamado "CAN" (*candidate*) no formato "CAN-<ano em quatro digitos>-<numero em quatro digitos>";
- Proposta: a candidata é discutida por um grupo formado por representantes de importantes organizações como CISCO, Symantec, SUN, Microsoft e pesquisadores, entre outros. A decisão do grupo pode ser pela aceitação, rejeição, modificação, além de poder não opinar ou pedir mais tempo para tomar a decisão.

Após estas fases, são realizadas mais discussões e decisões até que se chegue a um consenso para a publicação das decisões de forma que quando um relato é aceito, o resultado é uma entrada que consiste em: um nome, descrição, referências e o identificador padrão do CVE. Desta forma o CVE aumenta a efetividade dos sistemas de segurança de maneira confiável além de possibilitar a busca por informações em várias fontes sem a necessidade de se saber o nome dado à vulnerabilidade em cada base de dados [55].

No entanto, mesmo com estes esforços há uma grande heterogeneidade nos formatos adotados pelos desenvolvedores nos sensores de segurança. Estes ainda podem gerar alertas diferentes dependendo de suas configurações [38]. Neste contexto, nenhum dos formatos apresentados foi plenamente adotado por sensores de segurança ou mesmo por grupos de resposta a incidentes de segurança em computadores.

2.4 O IDS NIDIA

O projeto NIDIA é uma proposta de IDS multiagentes [4], baseado na noção de sociedade de agentes inteligentes. Arquiteturalmente, ele é composto por camadas, onde cada camada possui atividades a desempenhar que são executadas através do comportamento dos agentes que a compõem [4]. Estes são também responsáveis pela comunicação entre as camadas trocando informações importantes para desempenhar suas atividades.

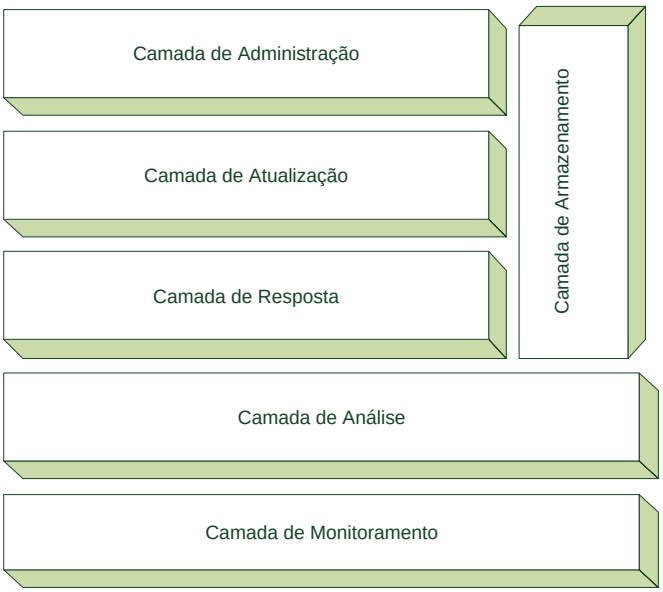


Figura 2.1: Modelo em camadas do IDS NIDIA

Na Figura 2.1 é apresentada a arquitetura do NIDIA, onde as funcionalidades das camadas são [57]:

- Camada de Monitoramento: responsável por capturar a ocorrência de eventos no meio exterior e fornecer informações sobre o mesmo para o resto sistema.
- Camada de Análise: responsável pela análise de eventos recebidos da camada de monitoramento. Nesta camada, os eventos coletados são formatados de maneira que padrões de ataques possam ser identificados e posteriormente a confirmação de um ataque;
- Camada de Reação: responsável por tomar contramedidas caso um incidente de segurança seja detectado;

- Camada de Atualização: responsável pela atualização das bases de informações. As consultas poderão ser feitas diretamente de qualquer camada, porém, inserções devem ser feitas somente através desta camada. Ela terá também a responsabilidade de manter a integridade e consistência das informações armazenadas

- Camada de Administração: responsável pela administração e integridade de todos os agentes do sistema;

- Camada de Armazenamento: responsável por manter de forma persistente informações provenientes das demais camadas. Nesta camada, localizam-se as bases de dados utilizadas pelo NIDIA. Segue um descrição das mesmas:

  - *Strategy DataBase* (STDB) é a base de dado responsável por registrar as estratégias adotadas por uma organização qualquer em relação à sua política de segurança.
  - *Reaction DataBase* (RADB) estão contidas as informações referentes as ações que devem ser tomadas de acordo com a severidade do ataque detectado.
  - *Incidents of Intrusion and Forensic Information DataBase* (IIDB) registra os danos causados por ataques bem sucedidos e tentativas de ataques. Este contem informações que podem ser úteis na identificação de tentativas de ataques provenientes de uma mesma origem ou domínio ou simplesmente serem usadas em investigações futuras.

Por trabalhar com arquitetura multiagentes, o IDS NIDIA pode ter um domínio administrativo composto por várias redes locais, monitorados por apenas um IDS NIDIA e diversos Agentes Sensores instalados em pontos estratégicos destas redes [54]. Desde sua criação o projeto do IDS NIDIA [4] vem recebendo várias inovações, principalmente devido ao constante crescimento e variedade dos ataques, e trabalha com a detecção de ataques por assinaturas e por anomalias.

Estas inovações veem melhorando a potencialidade do IDS acrescentando várias funcionalidades como: atualização automática do mecanismo de detecção de ataques [58]; o modelo de sistema de resposta de intrusão (IRS) [59]; a atualização de forma automática usando *web services* [51]; serviço de IDS remoto [60]; mecanismos de tolerância a falhas [42] para o próprio IDS; proposta para integração das bases de dados

[54]; detecção de ataques em redes *wireless* [61]; detecção de ataques em dispositivos móveis [57], detecção de ataques de *botnets* [62], entre outros.

2.5 Engenharia Dirigida Por Modelos (MDE)

A MDE é uma abordagem promissora para o desenvolvimento de sistemas e aplicações complexas [63] [64]. A MDE é uma metodologia de desenvolvimento de *software* que se concentra na criação de modelos ou abstrações, mais perto de alguns conceitos do domínio particular. Ela é utilizada para aumentar a produtividade, maximizando a compatibilidade entre os sistemas, simplificando o processo de *design*, e promovendo a comunicação entre indivíduos e equipes de trabalho sobre o sistema [65].

Cada modelo é descrito usando uma linguagem particular de modelagem como a *Unified Modeling Language* (UML) ou uma *Domain Specific Language* (DSL). A metodologia MDE tem oferecido uma promissora aproximação entre a falta de habilidade das linguagens de terceira geração e a complexidade das plataformas em expressar efetivamente os conceitos de domínio [66].

2.5.1 Model Driven Architecture (MDA)

O MDA é um *framework* para desenvolvimento de *software* definido pelo *Object Management Group* (OMG), cujo foco principal é dar a visão de como o *software* pode ser desenvolvido colocando a modelagem no centro do processo de desenvolvimento [63]. A chave para esse processo é que cada etapa da geração é automatizada o máximo possível [6].

A arquitetura MDA foi desenvolvida a partir de tecnologias e padrões abertos, bem estabelecidos, independentes de plataforma e também construídos pelo grupo OMG. São eles:

1. *Meta Object Facility* (MOF): define uma metalinguagem comum para a especificação de outras linguagens [67];

2. UML: define uma metalinguagem derivada da MOF, para descrever sistemas orientados a objetos [67];

3. *Common Warehouse Metamodel* (CWM): define uma metalinguagem, derivada da MOF, para descrever sistemas de *data warehousing* e outros relacionados. Permite o intercâmbio de metadados entre diferentes repositórios e bancos de dados de uma corporação através de interfaces padrão [67];

4. *XML Metadata Interchange* (XMI): define os meios de compartilhamento de modelos derivados da MOF. Define um mapeamento de modelo UML para *eXtensible Markup Language* (XML), sua distribuição, intercâmbio de modelos UML entre ferramentas e plataformas UML [67].

As tecnologias MDA utilizadas permitem separar a arquitetura de uma aplicação de sua implementação, possibilitando a criação de projetos de melhor qualidade para as aplicações, além de permitir portabilidade para outras plataformas. A partir dai, são alcançados benefícios relacionados à facilidade de migração entre plataformas [6] [66], maior flexibilidade para geração de código e interoperabilidade e portabilidade em nível de modelos [68] [66].

A representação do *framework* MDA básico está ilustrada na Figura 2.2: a primeira etapa é a construção de um modelo com um alto nível de abstração, independente de qualquer tecnologia. Esse modelo é chamado de Modelo Independente de Plataforma (*Platform Independent Model* (PIM)); a segunda etapa, considerada a mais complexa, é a transformação do PIM gerado na etapa anterior em um ou mais Modelos Específicos de Plataforma (*Platform Specific Model* (PSM)), este é mais específico para o sistema em termos de tecnologia de implementação, como um modelo de banco de dados ou um modelo *Enterprise Java Beans* (EJB); a terceira e última etapa é a transformação de um PSM em código.

A MDA dita que os artefatos a serem gerados durante o desenvolvimento devem ser principalmente o resultado de transformações que os modelos sofrem a partir de um estado de alto nível, preferencialmente independente da plataforma de implementação (PIM), e passa por estágios de mais baixo nível onde são criados modelos específicos para a plataforma alvo (PSM) até chegar aos artefatos finais (código por exemplo).

Como mostrado no *framework* MDA básico na Figura 2.2, é necessário que a cada etapa sejam aplicadas regras de transformação. Basicamente existe duas categorias de transformações. A primeira é a geração automática/semi-automática de um

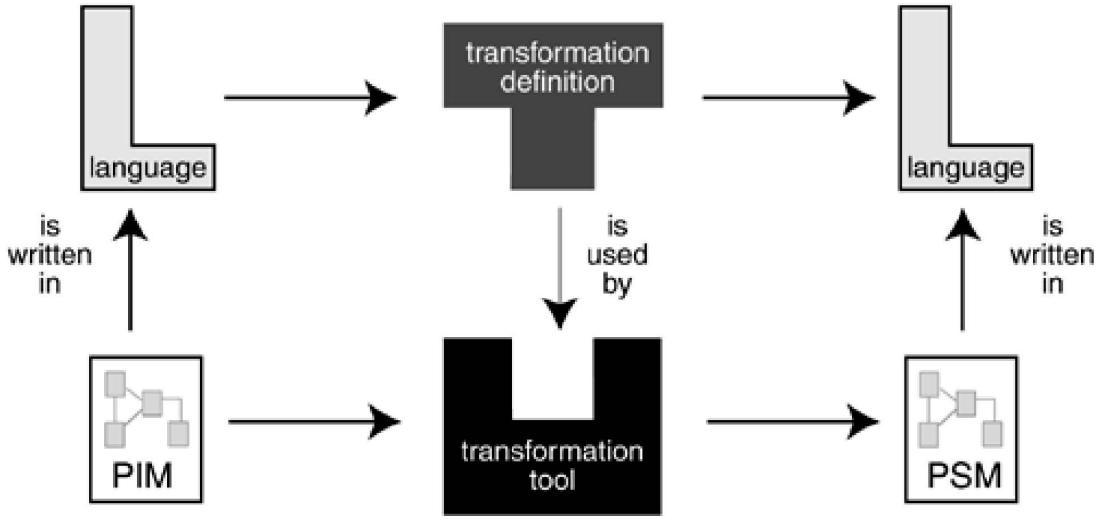


Figura 2.2: Framework MDA básico [6]

modelo de destino a partir de um modelo de origem (ambos os modelos são definidos através de um metamodelo). Esta transformação é denominada *Model-to-Model* (M2M); e a segunda é a geração automática/semiautomática de código-fonte a partir de um modelo de entrada. Esta transformação é denominada *Mode-to-Code* (M2C).

As transformações são definidas por um conjunto de regras que juntas descrevem como um modelo pode ser transformado em um ou mais modelos. Elas podem ser classificadas considerando duas características: quanto aos formalismos dos modelos de entrada e saída, e quanto ao nível de abstração [6]. Segundo Mens e Gorp [69] as características quanto ao formalismo são:

- Exógena: são transformações entre modelos expressos em linguagens diferentes, ou seja, transforma um modelo para um formalismo diferente. Por exemplo:
  - Síntese: transformação de um modelo de mais alto nível em um de mais baixo nível. O exemplo típico é a geração de *bytecode* a partir de um código-fonte Java ou geração de código-fonte a partir de um modelo;
  - Engenharia Reversa: o inverso da síntese. Extrai o nível superior a partir de uma especificação em um nível inferior;
  - Migração: transforma um programa escrito em uma linguagem para outra mantendo o mesmo nível de abstração.
- Endógena: são transformações entre modelos expressos na mesma linguagem. Por exemplo:

  - Otimização: transformação destinada a melhorar certas qualidades operacionais (desempenho) do sistema, sempre preservando a sua semântica;
  - *Refactoring*: mudança da estrutura interna do *software* para melhorar a qualidade do *software* (reutilização, modularidade, adaptabilidade, etc.) sem alterar o seu comportamento.

As características quanto ao nível de abstração podem ser [69]:

- Horizontal: transformação onde os modelos de destino e origem estão no mesmo nível de abstração. Os exemplos típicos são *refactoring* e migração;
- Vertical: transformações onde os modelos de destino e origem estão em diferentes níveis de abstração. Os exemplos típicos são geração de código e engenharia reversa.

Entre as linguagens em que as transformações podem ser escritas está *Atlas Transformation Language* (ATL), que além de ser uma DSL para especificar transformações M2M [70] é também um *kit* de ferramentas [71]. No campo do MDE, ATL fornece meios para produzir um conjunto de modelos alvo a partir de um conjunto de modelos de origem [71]. A ATL é uma linguagem híbrida, declarativa e imperativa, onde um programa de transformação escrito com esta linguagem é composto de regras que definem como os elementos do modelo de origem são encontrados e navegam para criar e inicializar os elementos dos modelos de destino [70] [71].

Eclipse Modeling Framework (EMF)

O *Eclipse Modeling Framework* (EMF) faz parte do *Eclipse Modeling Project*, um subprojeto do projeto Eclipse, que é um projeto de *software open source*, cujo objetivo é fornecer uma plataforma de ferramentas altamente integradas. EMF permite aos desenvolvedores construir rapidamente aplicações robustas baseadas em modelos surpreendentemente simples [72].

EMF é um *framework* de modelagem e ambiente de geração de código para ferramentas e outras aplicações baseadas em modelos de dados estruturados, que unifica Java, UML e XML, como mostrado na Figura 2.3. O EMF explora todas as facilida-

des providas pelo Eclipse, e reúne em um único ambiente as tecnologias que dão base à MDA.

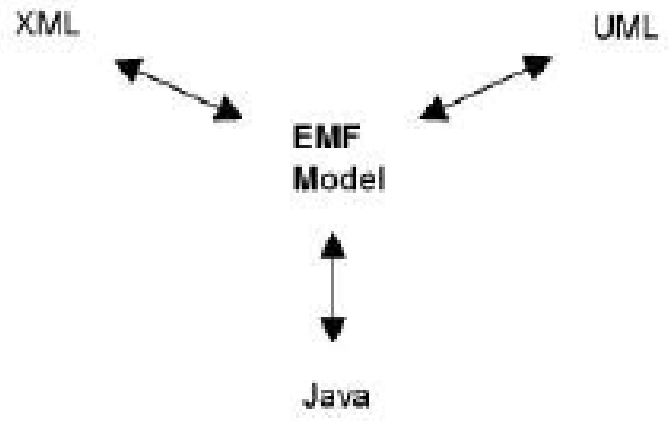


Figura 2.3: EMF unifica Java, XML e UML

Este *framework* foi essencial para o desenvolvimento deste trabalho. A partir dele foram descritos os metamodelos, assim como as regras feitas em ATL para a execução das transformações. O modelo utilizado para representar os modelos de EMF é chamado Ecore.

O Ecore é um meta modelo incluso no EMF para descrever modelos (meta-metamodelo), dar suporte à execução para os modelos, ele inclui notificações de mudanças, suporte à persistência com serialização em XMI, e uma *Application Programming Interface* (API) reflexiva para manipular objetos EMF genericamente [72].

Xtext

Xtext é um *framework* para desenvolvimento de DSL(Linguagens específicas de domínio) e outras linguagems de programação de forma textual [73]. Ele é fortemente integrado com o EMF e utiliza a plataforma Eclipse para prover um ambiente integrado de desenvolvimento - *Integrated Development Environment* (IDE).

Uma das facilidades do *Xtext* que pode ser utilizada neste trabalho foram os *parsers* e serializadores gerados. Diferente de outros geradores de *parsers* (como por exemplo JavaCC [74] ou mesmo o ANTLR [75] que é utilizado pelo Xtext), ele agrega mais do que apenas um *parser* e analisador léxico de uma gramática de entrada [73]. A linguagem gramatical (baseada no EBNF [76]) é usada para descrever e gerar: modelos

ECORE (opcional); o *lexer* para ler os seus modelos a partir de texto (o que permitiu que os alertas fossem utilizados diretamente como modelos de entrada); serializador para escrever os modelos de volta ao texto, um *link* para estabelecer referências cruzadas entre os elementos dos modelos; uma implementação da interface do EMF *Resource* com suporte total para carregar e salvar modelos EMF e uma integração da linguagem com o Eclipse IDE [73].

Teneo

Para facilitar o gerenciamento dos modelos foi utilizado o *plugin* Teneo, que é uma solução para persistir modelos em base de dados para o EMF. O Teneo integra o EMF com soluções existentes de persistência, combinando a forte funcionalidade de geração de código do desenvolvimento orientado por modelos com o poder do armazenamento, *caching* e consultas dos *softwares* sofisticados de mapeamento objeto-relacional e *1118-1590-5341-6933-7029-3016* de persistência [77]. Ele suporta a criação automática do EMF para mapeamentos relacionais, de forma que os objetos EMF possam ser armazenados e recuperados usando consultas avançadas (*Hibernate Query Language* (HQL) ou *Enterprise JavaBeans Query Language* (EJB-QL)), útil para o protótipo devido à quantidade de modelos a gerenciar, criando um repositório de modelos em uma base de dados relacional.

2.6 Considerações Finais

Este capítulo apresentou uma revisão dos principais conceitos e tecnologias utilizadas para o desenvolvimento deste trabalho. Foi apresentada uma introdução sobre segurança da informação abordando sua importância e os benefícios que podem ser alcançados com abordagens como a gestão de riscos e a adoção de políticas de segurança.

Dando continuidade ao capítulo, foi apresentada uma introdução sobre forense computacional e seus conceitos básicos. Foram abordados também conceitos sobre sistemas de detecção de intruso e abordagens para padronização de alertas de segurança. Foi apresentado ainda, o IDS NIDIA com suas principais características, sua arquitetura e funcionamento.

O capítulo foi finalizado mostrando conceitos de engenharia dirigida por modelos. Foi dado enfoque à MDA e ao EMF, com as tecnologias e *plugins* utilizados no desenvolvimento do protótipo da solução proposta neste trabalho, através da geração de código e transformações entre modelos.

O capítulo a seguir apresentará os principais trabalhos que serviram embasamento para o desenvolvimento desta dissertação.

3 Trabalhos Relacionados

Este capítulo visa apresentar os principais trabalhos serviram embasamento para o desenvolvimento desta dissertação. Os trabalhos foram separados em duas categorias:

- Trabalhos de modelos e *frameworks* de forense computacional
- Trabalhos que sugerem reuso do conhecimento forense, modelagem de ataques e implementações relacionadas à ideia proposta

3.1 Modelos e *Frameworks* para Forense Computacional

Antes de realizar qualquer procedimento investigativo, faz-se necessário estabelecer métodos para a investigação [78], [38]. Dentro da forense digital, isto não é diferente, e para dar suporte a este processo foram propostos vários modelos de processos investigativos.

Dentre os modelos existentes foram destacados alguns trabalhos [16] [79] [7] [8] [9] [10] [11] [13], que tiveram maior influência em relação à definição das etapas, objetivos das etapas e da taxonomia utilizada no modelo proposto e utilizado neste trabalho.

3.1.1 *Digital Forensic Research Workshop*

Palmer [16], no relatório técnico da *Digital Forensic Research Workshop* (DFRWS), enfatiza que o monitoramento e a análise dos dados de redes e de sistemas vivos (*live systems*) serão essenciais para a aplicação da lei. O trabalho enfatiza ainda a importância dos IDSs no processo de análise forense de rede, uma vez que são fundamentais como fontes de entrada devido à captura de dados a partir de grande variedade de fontes e à centralização das informações obtidas.

Neste trabalho é proposto um modelo para forense computacional que utiliza sete etapas (identificação, preservação, coleta, examinação, análise, apresentação e decisão), como visto na Figura 3.1, que ilustra as etapas e as técnicas ou métodos candidatos de cada etapa. O trabalho é importante uma vez que é o primeiro na literatura a oficialmente definir forense de rede e sugerir adaptar os procedimentos de forense computacional para a forense de rede.

Identification	Preservation	Colection	Examination	Analysis	Presentation	Decision
Event/Crime Detection	Case Management	Preservation	Preservation	Preservation	Documentation	
Resolve Signature	Imaging Technologies	Approved Methods	Traceability	Traceability	Expert Testimony	
Profile Detection	Chain of Custody	Approved Software	Validation Techiniques	Statistical	Clarification	
Anomalous Detection	Time Synch.	Approved Hardware	Filtering Techiniques	Protocols	Mission Impact Statement	
Complaints		Legal Authority	Patternt Matching	Data Mining	Recomended Countermeasure	
System Monitoring		Lossless Compression	Hiden Data Discovery	Timeline	Statistical interpratation	
Audit Analysis		Hiden Data Extraction	Link			
Etc.		Data Reduction		Spacial		
		Recovery Techiniques				

Tabela 3.1: Processo investigativo para forense digital segundo a DFRWS [16]

3.1.2 *An Abstract Digital Forensics Model (ADFM)*

Reith [79] analisa o desenvolvimento do processo forense digital e propõe um modelo abstrato para os procedimentos. No trabalho são propostas nove etapas onde os objetivos de cada etapa podem ser vistos na Tabela 3.2.

O trabalho é importante uma vez que complementa o modelo proposto em [16] com a etapa de preparação e faz um mapeamento dos objetivos de cada etapa. No entanto, a preparação ainda é uma etapa realizada após a descoberta do incidente.

Etapa	Objetivo
Identificação	Reconhecer um incidente a partir de indicadores e determinar o seu tipo.
Preparação	Preparar ferramentas, técnicas, mandados de busca e autorizações de monitoramento e apoio da gestão.
Abordagem e Estratégia	O objetivo da estratégia deve ser formulação dinâmica de uma abordagem baseada no impacto de espectadores e a tecnologia específica. Deve-se maximizar a coleta de evidências minimizando o impacto para a vítima
Preservação	Isolar, proteger e preservar o estado da evidência física e digital. Inclui impedir pessoas de usar o dispositivo digital ou permitir que outros dispositivos eletromagnéticos sejam usados dentro de um raio que possa afetar o dispositivo
Coleta	Gravar a cena física e duplicar provas digitais utilizando procedimentos padronizados e aceitos.
Examinação	Buscar sistematicamente e em profundidade provas relacionadas com a suspeita de crime. Centra-se na identificação e localização de possíveis provas. Constrói uma documentação detalhada para análise
Análise	Determinar a significância, reconstruir fragmentos de dados e tirar conclusões baseadas em evidências encontradas. Pode levar várias iterações de exame e análise para apoiar uma teoria crime. A distinção de análise é que não pode exigir altas habilidades técnicas para realizar e, assim, mais pessoas podem trabalhar neste caso.
Apresentação	Resumir e apresentar explicações de conclusões. Este deve ser escrito em termos leigos, usando uma terminologia abstrata, onde a terminologia deve referenciar os detalhes específicos.
Retorno de evidência	Garantir que a propriedade física e digital seja devolvida ao proprietário adequado, bem como determinar o que e como provas criminais devem ser removidas.

Tabela 3.2: Etapas e objetivos do modelo abstrato de forense digital

3.1.3 *Incident Response Process (IRP)*

Mandia [7] propõe uma metodologia de resposta a incidentes. Esta tem o objetivo básico de impedir respostas desarticuladas e não coesas para confirmar (ou não) a ocorrência de um incidente.

Neste modelo existe uma preocupação em considerar os aspectos dos profissionais de segurança corporativa e também preocupações dos agentes da lei [7]. A metodologia é composta por sete etapas (Figura 3.1) cujos objetivos estão descritos na Tabela 3.3.

Etapa	Objetivo
Preparação pré-incidente	Preparar a organização e o *Computer Security Incident Response Team* (CSIRT) (Equipe de Resposta a Tratamento de Incidentes de Segurança) antes do acontecimento do incidente
Detecção de Incidentes	Identificar um potencial incidente de segurança
Resposta Inicial	Realizar uma investigação inicial com os detalhes básicos que envolvem o incidente, a montagem da equipe de resposta a incidentes, e notificar os indivíduos que precisam de saber sobre o incidente
Formulação de estratégia de resposta	Com base nos resultados dos fatos conhecidos, determinar a melhor resposta e buscar aprovação da gerência, além de determinar as ações civis, criminais, administrativas ou outras que possam ser apropriadas
Investigar o Incidente	Realizar uma coleta completa de dados. Rever os dados recolhidos para determinar o que aconteceu, quando aconteceu, quem fez, e como pode ser prevenido no futuro
Relatório	Relatar com precisão as informações sobre a investigação do incidente
Resolução	Empregar as medidas de segurança e mudanças nos procedimento de segurança de acordo com as lições aprendidas e desenvolver soluções a longo prazo para os problemas identificados

Tabela 3.3: Etapas e objetivos da metodologia de resposta a incidentes

Diferente do modelo proposto por Reith [79] o modelo proposto por Mandia [7] enfatiza que a preparação deve ser feita antes mesmo do conhecimento do incidente, e não apenas após a identificação do mesmo. No modelo é colocado como passo

inicial da fase de preparação pré-incidente a identificação do risco, observando quem são os ativos, proporcionando o melhor direcionamento dos recursos/esforços para os ativos mais críticos à organização.

Percebe-se também uma preocupação inicial em aprender com os incidentes ocorridos para a implementação de novas medidas de segurança. Neste trabalho é enfatizado também a importância das políticas de segurança para dar continuidade ou suspender um procedimento investigativo relativo a um incidente.

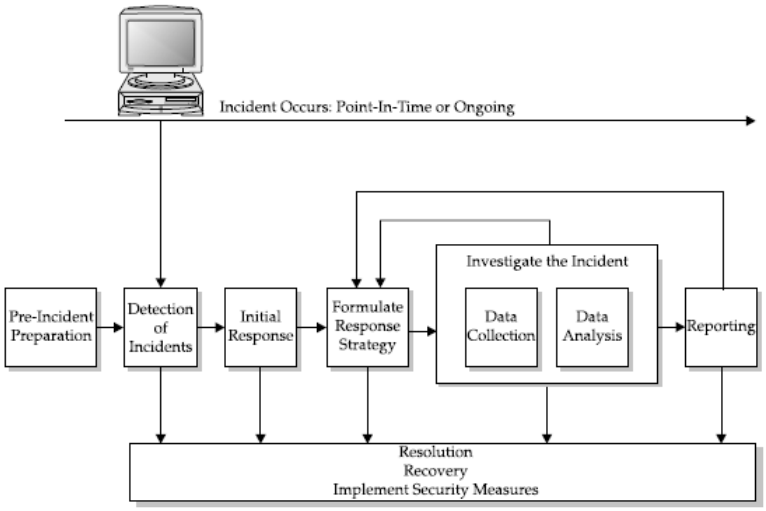


Figura 3.1: Modelo de processo investigativo de [7]

3.1.4 *The Investigative Process Model (IPM)*

Casey e Palmer [8] propuseram o Modelo de Processo Investigativo (Figura 3.2) e delinearam categorias para o gerenciamento de um caso, onde investigadores e examinadores devem trabalhar juntos para "escalar" as etapas de baixo para cima de forma sistemática, determinada em um esforço para apresentar uma história convincente depois de atingir o patamar mais alto [8].

Figura 3.2: Categorias do modelo de processo investigativo [8]

Etapa	Objetivo
Acusação ou alerta de incidente	Ponto de partida para que a investigação seja iniciada (pode ser desencadeada por eventos em cenários tradicionais de aplicação da lei, uma acusação ou alerta incidente automatizado)
Avaliação de Valor	Deve ser feita uma triagem para que os recursos vitais sejam concentrados sobre os problemas mais graves ou onde eles possam ser mais eficazes
Protocolos para cena do crime/incidente	Conservar e documentar o estado e a integridade dos itens (digitais ou não) na cena do crime através do uso de protocolos, práticas e procedimentos para minimizar a chance de erros, descuidos, ou lesões. O produto ou a saída deste estágio é uma cena segura com conteúdos mapeados e registrados (fotografias e diagramas básicos etc.)
Identificação ou Apreensão	Fazer escolhas informadas, decisões fundamentadas sobre o que aproveitar e estar preparado para documentar e justificar as ações sobre as evidências. Pode-se ainda localizar e apreender provas fisicamente para posterior processamento por um examinador de provas digitais.
Preservação	Identificar ítens voláteis para que permaneçam inalterados. Devem ser tomadas ações apropriadas para garantir a integridade das potenciais provas (físicas e digitais). Devem ser empregados métodos e ferramentas para garantir a integridade das evidências. A saída desta fase é geralmente um conjunto de cópias duplicadas de todas as fontes de dados digitais, onde serão executados os procedimentos investigativos
Recuperação	Extrair dados que foram apagados, escondidos, camuflados, ou que estejam de alguma forma indisponível para visualização utilizando o sistema operacional nativo e sistema de arquivos residente. O objetivo é identificar e, se possível tornar visível, todos os dados que podem pertencer a um tipo de dado particular. A saída fornece o máximo de conteúdo para os investigadores.
Colheita	Recolher dados e metadados sobre todos os objetos documentados nas fases de preservação e recuperação, agrupando dados e características e formulando hipóteses
Redução	Reduzir os dados coletados eliminando itens irrelevantes para a investigação, tendo como saída um conjunto menor de informações que tenham potencial maior de conter
Organização e Busca	Organizar o conjunto reduzido de materiais, agrupando-os em unidades significativas. O objetivo principal é permitir que o investigador encontre e identificar os dados de forma mais fácil durante a etapa de análise.
Análise	Análise detalhada dos dados identificados pelas atividades anteriores.Executando subatividades como: Avaliação(analisar para tentar determinar fatores como meio, a motivação, oportunidade); Experimentação(realizar experimentos para testar hipóteses); Fusão e correlação ; e Validação (conclusões fundamentadas que os investigadores se propõem a apresentar a juristas ou outros tomadores de decisão como "prova positiva"para a acusação ou absolvição)
Relatório	Fornecer uma visão transparente do processo de investigação, relatórios finais devem conter detalhes importantes de cada etapa, incluindo a referência a protocolos
Persuasão e Testemunho	Inclui técnicas e métodos utilizados para ajudar o analista e / ou especialista de domínio a traduzir detalhes tecnológicos e de engenharia para uma narrativa compreensível

Tabela 3.4: Etapas do modelo de processo investigativo

O modelo sugere etapas diferentes dos modelos já citados até aqui, como por exemplo a avaliação de valor (*assessment of worth*) e a redução dos dados. Onde a primeira é importante para concentrar os recursos vitais sobre os problemas mais graves, ou de onde estes recursos possam ser mais eficientes.

O modelo aborda ainda, que se deve considerar a política da organização e observar se fatores que contribuem para a gravidade de um problema incluem ameaças de danos físicos, potencial de perdas significantes, ou ainda, risco de comprometimento maior do sistema ou interrupções.

Puderam ser identificadas neste trabalho características que remetem à análise de risco de ativos, além do correlacionamento das informações, que não haviam sido sugeridos pelos trabalhos anteriores.

3.1.5 *Extended Model of Cybercrime Investigations (EMCI)*

Ciardhuáin [9] propõe o Modelo Estendido de Investigação de Crime Digital (Figura 3.3). O modelo abrange os modelos já referenciados e, segundo o autor, pode apoiar o desenvolvimento de ferramentas, técnicas, formação e certificação / credibilidade de investigadores e ferramentas.

Além disto, pode fornecer uma estrutura unificada para estudos de caso/-lições aprendidas materiais a serem compartilhados entre os investigadores e o desenvolvimento de normas, testes de conformidade e investigação de melhores práticas [9]. No entanto, a descrição das etapas deste modelo são visivelmente voltadas para o processo investigativo utilizando *hosts*, além de não abordar a preparação do ambiente organizacional.

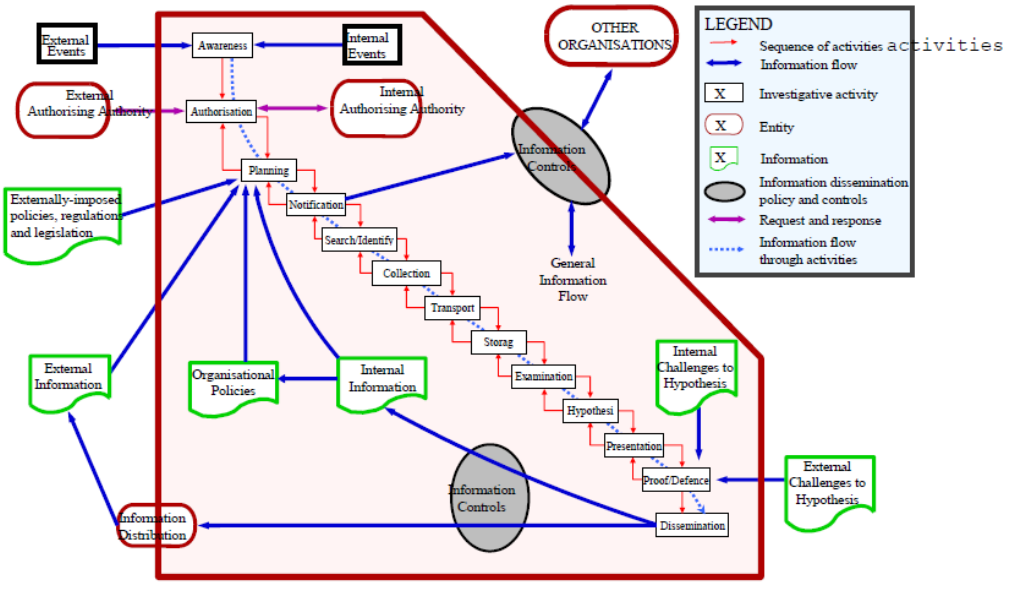


Figura 3.3: Modelo estendido de investigação de crime digital [9]

Etapa	Objetivo
Conscientização	Conscientização de que a investigação é necessária. Normalmente criada por eventos externos (um crime é relatado para a polícia ou um auditor é solicitado a realizar uma auditoria), ou ainda por eventos internos (alerta de um IDS)
Autorização	Obtenção de autorização (da organização e/ou policial) para iniciar a investigação)
Planejamento	Etapa influenciada por informações de dentro e de fora da organização. Do lado de fora, os planos serão influenciados por regulamentos e legislação que definem o contexto geral da investigação, informações coletadas pelos investigadores a partir de outras fontes externas, e de dentro da organização com as estratégias da organização, políticas, e informações sobre as investigações anteriores.
Notificação	Informar ao objeto de investigação ou outras partes interessadas que a investigação está ocorrendo
Busca e Identificação de Evidência	Localização e identificação de evidências(descoberta do computador usado, rastreio de IPS, etc)
Coleta	Atividade em que a organização investiga toma posse das evidências de uma forma que possam ser preservadas e analisadas, por exemplo, de imagem de discos rígidos ou a apreensão de computadores inteiros
Transporte	Evidências devem ser transportadas para um local adequado para exame posterior. Pode ser simplesmente a transferência física de computadores apreendidos para um local seguro, ou ainda a transmissão de dados através de redes
Armazenamento	Evidências encontradas devem ser armazenadas para examinação posterior de forma que seja preservada sua integridade
Examinação	Uso de técnicas para encontrar e interpretar os dados significativos. Uso de técnicas automatizadas para apoiar o investigador são obrigatórias.
Hipótese	Baseado na examinação deve-se construir hipóteses sobre os fatos acontecidos, documentando a hipótese com relatórios
Apresentação	A hipótese deve ser apresentado a outras pessoas (gestores da organização ou um juri)
Prova/Defesa	Os investigadores terão de provar a validade da hipótese e defendê-la. Pode requerer reanalisar evidências
Disseminação	Algumas informações podem ser disponibilizadas apenas dentro da organização, enquanto outras informações podem ser mais amplamente divulgadas, de acordo com as políticas da organização.

Tabela 3.5: Etapas e objetivos do modelo estendido de investigação de crime digital

3.1.6 *Objectives-based Framework for the Digital Investigations Process (OBFDIP)*

Beebe [10] elaborou o *Framework* Orientado a Objetivos para o Processo de Investigação Digital. O *framework* é organizado em camadas (Figura 3.4) na tentativa de simplificar o processo de forense digital.

Segundo o autor, a prioridade no desenvolvimento do modelo é a sinergia com os modelos anteriores para forense computacional e é introduzido o uso de tarefas orientadas a objetivos onde as metas de investigação são utilizadas para selecionar as tarefas de análise.

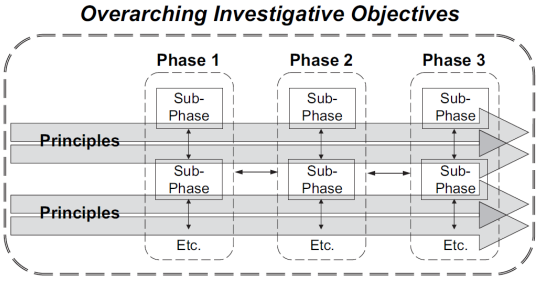


Figura 3.4: Estrutura do *framework* e modelo genérico proposto por [10]

O trabalho detalha a primeira camada do *framework* que é composta por seis fases, como pode ser visto na Figura 3.5, em conformidade com modelos anteriores, mostrando também as interações permitidas entre as fases. As subfases devem ser colocadas nas camadas inferiores fornecendo especificidades e granularidade, orientada por princípios e objetivos [10]. No entanto, as demais camadas não são especificadas.

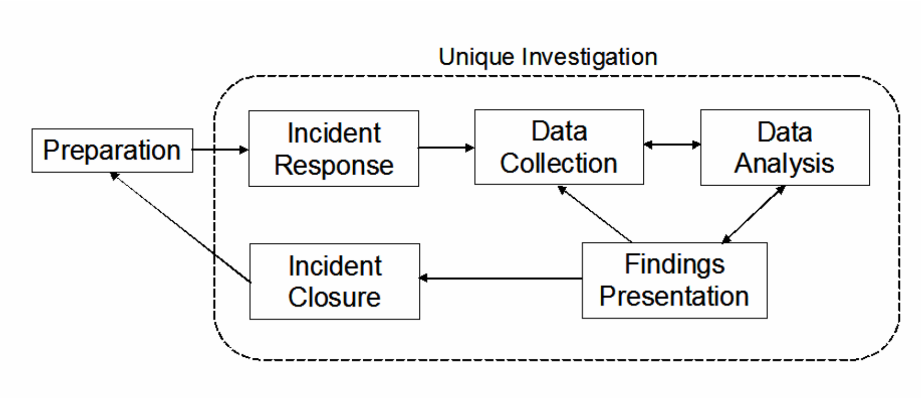


Figura 3.5: Fases da primeira camada do *framework* orientado a objetivos para o processo de investigação digital [10]

3.1.7 *A Generic Framework for Network Forensics (GFNF)*

Pilli [11] propõe um modelo de processo genérico para forense digital, na tentativa de formalizar uma metodologia específica para forense baseada em rede. O modelo é também utilizado em *survey* do mesmo autor [12] e envolve as fases de preparação, detecção, resposta a incidentes, coleta, preservação, exame, analise, investigação e apresentação, executando um ciclo como pode ser visto na Figura 3.6. O autor sugere o uso de ferramentas pré-existentes para apoiar as fases descritas.

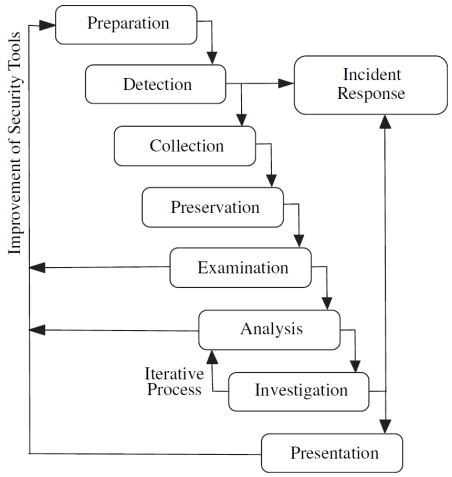


Figura 3.6: Modelo de processo genérico para forense de rede [11] [12]

Etapa	Objetivo
Preparação	Forense de Rede é aplicável somente aos ambientes onde as ferramentas de segurança de rede (sensores) como sistemas de detecção de intrusão, analisadores de pacotes, *firewalls*, software de medição de fluxo de tráfego são implantados em pontos estratégicos da rede. Autorizações necessárias e garantias legais são obtidas de modo que a privacidade não seja violada.
Detecção	Observar os alertas gerados por várias ferramentas de segurança, indicando uma violação de segurança ou violação da política. Analisar eventos não autorizados e anomalias
Resposta ao Incidente	Etapa iniciada com base na informação recolhida para validar e avaliar o incidente. Depende do tipo de ataque identificado e é guiado política da organização, restrições legais e de negócios. Iniciar um plano de ação de defesa e ação de recuperação de dano. Decide-se aqui continuar ou não a investigação do incidente
Coleta	Coletar os dados dos sensores usados para coletar o tráfego. Deve haver um procedimento bem definido usando hardware confiável e ferramentas de software, para recolher o máximo de provas causando o mínimo impacto
Preservação	Armazenar os dados originais obtidos como somente leitura em um dispositivo de *backup*. Dever ser gerado um *hash* de todos o dados. As cópias dos dados serão analisadas ??e os dados de tráfego originais permanecem intactos
Examinação	Integrar e fundir os dados (traços) obtidos a partir de vários sensores de segurança para formar um grande conjunto de dados em que a análise pode ser realizada. Procura-se metodologicamente por indicadores específicos do crime nos dados recolhidos. É dado um *feedback* para melhorar as ferramentas de segurança.
Análise	Correlacionar e classificar indicadores para deduzir observações importantes usando os padrões de ataque. Pode utilizar técnicas estatísticas, mineração de dados, e *soft compting* para buscar dados que combinem com os padrões dos ataques. Dá um *feedback* para as ferramentas de segurança.
Investigação	Determinar o caminho a partir de uma rede vítima ou sistemas para sistemas intermediários e vias de comunicação até o ponto de origem do ataque. Pode exigir algumas características adicionais a partir da fase de análise
Análise	Apresentar as observações em uma linguagem compreensível, proporcionando explicação dos vários procedimentos usados ??para chegar à conclusão, assim como a proporcionar documentação sistemática para requisitos legais.

Tabela 3.6: Etapas e objetivos do modelo de processo genérico para forense de rede

3.1.8 *A Multi-component View of Digital Forensics (MCVDF)*

Diferente dos modelos apresentados até aqui, Grobler [13] enfatiza a necessidade de um *framework* de gestão global para o processo de forense que possa:

- Preparar as organizações para investigações pela identificação pró-ativa de a disponibilidade de provas admissíveis, além de processos relevantes para forense [13];
- Usar ferramentas e técnicas de forense digitais para melhorar estruturas de governança nas organizações [13];
- Reunir e analisar provas durante o curso de ataques e [13];
- Investigar incidentes com sucesso para determinar a causa raiz dos incidentes e processar um executor [13].

Baseado nos requisitos apresentados acima se percebe que grande parte dos modelos de processo para forense digital tradicional não são suficientes para atendelos. O autor propõe então um *framework* composto por três componentes:

- Componente proativo [1](ProDF): preparação forense de uma organização para assegurar o sucesso, o custo efetivo investigações com o mínimo de interrupção de negócios garantindo que 'boas' provas e processos estejam no local e disponíveis quando necessárias para uma investigação ou conforme necessário durante o fluxo normal dos negócios [13];

- Componente ativo (ActDF): referente à capacidade de uma organização reunir (identificar, coletar e preservar) evidência digital em um ambiente vivo para possibilitar uma investigação bem sucedida [13];

- Componente reativo (ReDF): lida com a forense digital tradicional, com as etapas que são executadas após a ocorrência de um ataque utilizando técnicas analíticas e de investigação para a preservação, identificação, extração, documentação, análise e interpretação de mídia digital, armazenada digitalmente ou codificados para evidenciar, e/ou analisar a causa-raiz e a apresentação de evidências digitais provenientes de fontes digitais a fim de facilitar ou promover a reconstrução de incidentes [13].

O autor define então os objetivos de cada um deles como pode ser visto na Tabela 3.7.

Etapa	Objetivos
ProDF	Tornar-se pronto para forense digital; Aprimorar os programas de Governança e segurança de TI da organização, avaliando a eficácia dos controles, medidos em relação a TI e os objetivos Informações de Segurança (relacionado aos objetivos do negócio); Melhorar informações de segurança com o uso das ferramentas de forense digital para melhorar a eficácia e eficiência na organização.
ActDF	Coletar evidências vivas relevantes em um sistema vivo ou produção ambiente, utilizando ferramentas e tecnologias apropriadas; Minimizar o efeito e impacto de um incidente em curso, e Fornecer um ponto de partida significativo para uma investigação reativa dentro dos parâmetros do sistema de controle de risco da organização
ReDF	Determinar a causa-raiz do incidente; vincular um autor ao incidente; minimizar o impacto de um incidente; investigar o incidente com sucesso.

Tabela 3.7: Síntese dos objetivos dos componentes (MCVDF)

Uma vez definidos os objetivos Globler mostra como deve ser feita a interação entre os diferentes componentes (Figura 3.7) para proporcionar uma visão geral da forense digital.

[1]*proactive*: (de uma pessoa, política ou ação) criar ou controlar uma situação, fazendo com que algo aconteça ao invés de responder a ele depois que ele aconteceu: ser pró-ativa na identificação e prevenção de problemas potenciais [80]

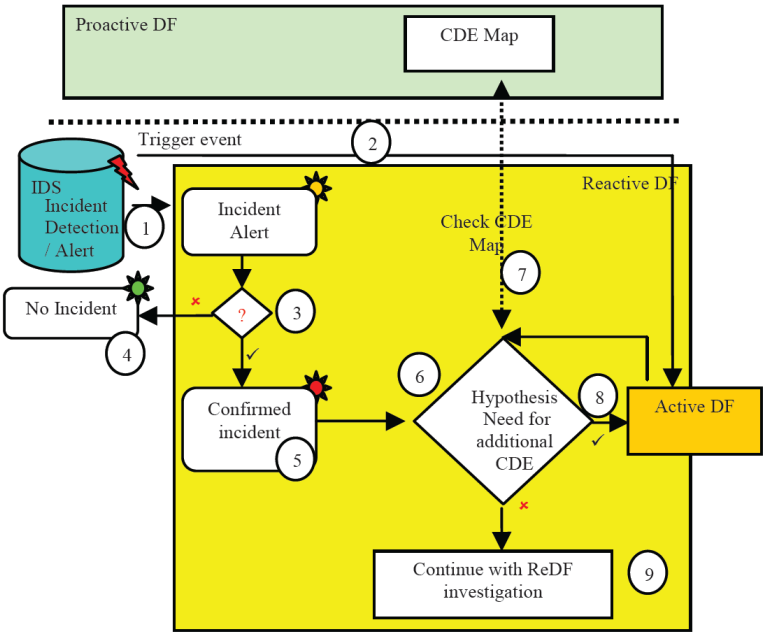


Figura 3.7: Relacionamento entre os componentes de forense digital [13]

3.2 Reuso do Conhecimento Forense, Modelagem de Ataques e Implementações

Nesta seção serão abordados alguns trabalhos que estão relacionados à modelagem e implementação da solução proposta. Os trabalhos abordam o uso de ontologias e MDA para reutilização do conhecimento forense e modelagem de ataques.

3.2.1 *Weaving Ontologies to Support Digital Forensic Analysis*

Hoss e Carver [14] propõem uma abordagem ontológica para o desenvolvimento de ferramentas para análise de forense computacional. A idéia é integrar conhecimento forense e legal utilizando "tecelagem" de ontologias.

No entanto nenhuma ontologia específica é proposta no artigo, e os autores apenas mostram uma estrutura abstrata para as ontologias necessárias. Como pode ser visto na Figura 3.8, é proposto o uso de cinco ontologias especializadas: ontologia do crime, ontologia forense dos dispositivos, ontologia legal, ontologia do dispositivo digital e uma ontologia para a integração das informações, mas sem nenhum modelo concreto proposto.

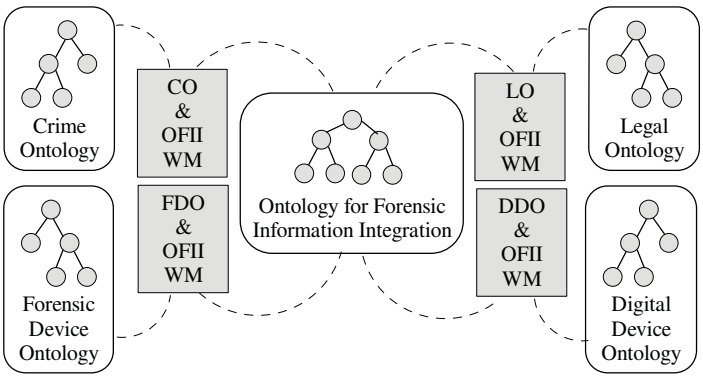


Figura 3.8: Ontologias para a integração de informações forenses [14]

3.2.2 *Method Ontology for Intelligent Network Forensics Analysis*

Saad e Traore [15] propõe uma ontologia que representa o domínio da forense de rede para a resolução dos problemas. Segundo o autor, a ontologia proposta possui conhecimento de 11.000 atividades maliciosas e 30 métodos de forense de rede para resolver problemas, combinando o conhecimento do domínio forense e o conhecimento da resolução de problemas.

Neste trabalho são abordados três tipos principais de conhecimento: objetivos de solução de problemas; conhecimento para resolução de problemas para o processo de forense de rede; e conhecimento real sobre o domínio da forense de rede. No trabalho é exemplificada a representação ontológica de um ataque de FTP de múltiplas etapas onde a representação ontológica do ataque pode ser visto na Figura 3.9.

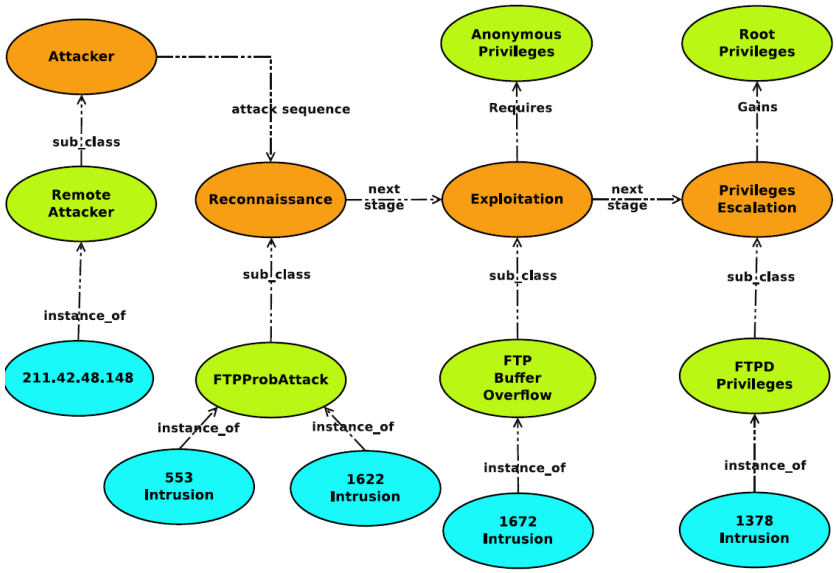


Figura 3.9: Representação ontológica para um ataque FTP [15]

No entanto, a construção e manutenção da ontologia se torna um processo complexo uma vez que o domínio de forense de rede possui um grande número de conceitos envolvidos que possuem relações complexas [15]. As relações ontológicas gerais estão mostradas na Figura 3.10.

Relation-Name	Subject-Class	Object-Class
Executes	Attacker	Attack
Exploits	Attacker	Vulnerability
Uses	Attacker	Malicious
Located-At	Attacker	Location
Has-A	Attacker	Motive
Leaves	Attacker	Evidence
Uses	Attacker	Malicious
Target	Attacker	Asset
Gains	Attacker	Privilege
Compromises	Attacker	System
Requires	Attack	Vulnerability
Elevates	Attack	Privilege
Proved-By	Attack	Evidence
Causes	Attack	Impact
Triggered-By	Attack	Malicious
Affects	Attack	Asset
Traced-To	Attack	Malicious
Has-A	Attack	Objective
Exist-In	Vulnerability	Asset
Requires	Attack	Privileges
Extracted-From	Evidence	Asset

Figura 3.10: Relações ontológicas gerais [15]

3.2.3 *A Generic Metamodel for IT Security Attack Modeling for Distributed Systems*

Miede [81] propõe um metamodelo genérico de segurança de TI. O objetivo do trabalho é modelar ataques de sistemas distribuídos. O metamodelo proposto busca capturar os conceitos principais envolvidos em segurança e seus relacionamentos, focado nos ataques de forma que o conhecimento sobre os ataques não seja exclusivo dos atacantes [81].

O metamodelo é dividido em três partes básicas: núcleo (ilustra os conceitos básicos de segurança), contramedidas (conceitos relacionados às medidas para respostas a ataques) e ataques (ilustra os conceitos relacionados a ataques), dando uma visão panorâmica e pouco específica, apresentando os componentes de ataques e suas possíveis relações.

A utilização do metamodelo é ilustrada no artigo com exemplos que utilizam apenas partes do metamodelo. Estas referem-se aos ataques na tentativa de descrever metadados, fases, métricas, requisitos, nível de abstração, ferramenta, modo do ataque, atacante e ação do ataque como pode ser visto na Tabela 3.8.

Ataque P2P *Distributed Hashtables* (DHTs)	
Attack Metadata	Name: Incorrect Lookup Routing [18]. Description: The misbehaving peer denies correct forwarding of received lookup requests. As a result, the provider of the requested content cannot be found. The misbehaving peer may route requests to arbitrary peers or simply drop them Related Attacks/ Vulnerabilities: Partitioning [18] References: [18], [19]. Related Countermeasures: Improve robustness of the lookup process [18] or distribute copies of the objects in the network [20].
Attack Phases	Identification and Vulnerabilities Analysis: Not necessary. Gain access: Sybil Attack or Incorrect Routing Updates [18] to become responsible for a large amount of content or specific content Perform attack: Drop or redirect lookup request. Complete attack: Detection may be prevented by colluding misbehaving peers
Requirements	Prerequisites: Attacker must have access to the P2P system. Context: The attack can only be performed upon receiving a lookup request Dependencies: The misbehaving peer must be used to forward the lookup request for the particular content or must be directly responsible for it. Resources: Depending on the desired effect, from one up to an arbitrary number of misbehaving peers
Attack Metric	Frequency/ Reproducibility: Depending on the frequency of received lookup requests. May be performed on each request or statistically Gain/ Loss: If successful, the attack results in a Denial of Service on the attacked content. The chance of finding a specific object decreases with the number of misbehaving peers in the P2P system Impact/ Size: Depending on the structure of the overlay (ring, tree,. . .) and on the number of connections maintained by each peer
Abstraction Level	P2P overlay, application layer
Attack Tool	Modified P2P client
Attack Mode	Active
Attack Type	Criminal attack (destructive attack)
Attacker	Resources: Access to the P2P system, i. e., a device with an Internet connection Expertise: Advanced expertise in form of knowledge on the particular P2P overlay is required to implement an attack tool. Only basic knowledge on P2P systems required if tools are already available Access: Not relevant Risk Aversion: Low risk since detection is difficult Objectives/ Motivation: Denial of Service Location: Any

Tabela 3.8: Instância de um ataque P2P de *distributed hashtables* (DHTs)

3.3 Considerações Finais

Neste capítulo foram mostrados os trabalhos relacionados à solução proposta. Inicialmente foram mostrados modelos e *frameworks* propostos para computação forense que serviram de fundamentação para o processo proposto.

Em seguida, foram vistos trabalhos que abordam a modelagem do domínio de segurança através do reuso do conhecimento forense e modelagem de ataques que serviram de inspiração para o desenvolvimento e implementação da solução proposta nesta dissertação.

Inicialmente, pudemos perceber que, mesmo existindo várias propostas de modelos de processos para a execução de forense computacional, ao longo dos anos não se estabeleceu um padrão entre estes, e mesmo, muitas vezes, utilizando taxono-

mias parecidas, os objetivos, etapas e relacionamentos entre etapas ainda não são bem definidos.

Além disto, grande parte destes modelos são projetados para serem guiados por um especialista humano, ou ainda possuem foco apenas no *host*. Mesmo *frameworks* que trabalham diretamente com os pacotes acabam trabalhando de forma isolada em relação aos demais sensores de segurança da rede.

A Tabela 3.9 mostra um mapeamento das etapas dos trabalhos citados [16] [79] [7] [8] [9] [10] [11] [13].

DFRWS [16]	ADFM [79]	IRP [7]	IPM [8]	EMCI [9]	OBFDIP [10]	GFNF [11]	MCVDF [13]
-	Preparation (post incident)	Pre-incident Preparation	-	Awareness, Autho- rization, Planning	Preparation	Preparation & Authoriza- tion	Preparation (ProDF)
Identification	Identification	Detection of incidents	Incident Alerts or accusation	Notification	-	Detection	Incident Detection & Alert
-	-	-	Assessment of worth	-	-	-	-
-	Approach & Strategy	Initial Response, Formulate Strategy	-	-	Incident Response	Incident Response	Incident Response
Collection	Collection	Investigation (Data Collection)	Incident/Crime Scene Protocol, Identification or Seizure	Search & Identification, Collection	Data Collection	Collection	Evidence Acquisition (ReDF)
Preservation	Preservation	-	Preservation	Transport, Storage	-	Preservation	Evidence acquisition (seizure) (ReDF)
Examination	Examination	-	Recovery, Harvesting, Reduction, Organization & Search	Examination	-	Examination	Event Reconstruction
Analysis	Analysis	Investigation (Data Analysis)	Analysis	Hypothesis	Data Analysis	Analysis & Investigation	Analysis
-	-	-	-	-	-	-	Service Restoration
Presentation	Presentation	Reporting	Reporting	Presentation	Findings Presentation	Presentation & Review	Present Findings
Decision	Returning Evidence	Resolution	Persuasion & Testimony	Proof of Defense, Dissemination	Incident Clousure	-	Dissemination of results

Tabela 3.9: Mapeamento das etapas dos trabalhos

Na seção 3.2 foram apresentados os trabalhos que inspiraram e serviram de base para a modelagem e implementação da abordagem proposta. Estes tratam de soluções de reutilização do conhecimento forense e modelagens no domínio da segurança.

No primeiro trabalho citado ([14]), é proposta uma abordagem com a utilização da tecelagem de ontologias para o desenvolvimento de ferramentas forenses. No entanto, não é apresentada nenhuma ontologia. O trabalho apenas sujere que as ontologias a serem integradas sejam divididas em cinco categorias: ontologia do crime, ontologia forense dos dispositivos, ontologia legal, ontologia do dispositivo digital e uma ontologia para a integração das informações.

Já o trabalho [15] utiliza a abordagem ontológica para a análise de ataques. O trabalho utiliza uma ontologia genérica e torna-se limitada, segundo o próprio autor, por ter um grau de complexidade relativamente elevado, uma vez que a manutenção

da ontologia proposta é complexa. Devido à natureza genérica da ontologia apresentada neste trabalho, torna-se difícil expandir e analisar os conceitos de forma específica uma vez que os ataques não são devidamente categorizados. Este fato impede que sejam adicionados conceitos específicos à analise de cada categoria de ataque.

Dando prosseguimento foi mostrado o trabalho [81], que utiliza uma abordagem baseada na MDA propondo um metamodelo genérico de segurança com o foco em sistemas distribuídos [81]. No trabalho é mostrado um metamodelo genérico para segurança de TI, onde o autor tenta abordar de uma forma geral o cenário de segurança para sistemas distribuídos com o foco nos ataques. O autor utiliza como exemplo os conceitos relacionados aos ataques distribuídos não utilizando o metamodelo completamente. Neste conceitos específicos relativos à forense computacional não são abordados e a forense computacional é apenas citada como um tipo de resposta, de forma que os conceitos relacionados a ela não são abordados no metamodelo.

O escopo abordado neste trabalho é a execução do processo investigativo basicamente no ambiente de rede apoiado pela MDA(com a geração de código e realização de transformações dos alertas).

Os trabalhos aqui apresentados contemplam a execução do processo investigativo nos escopos voltados a *hosts*(incluindo mídias) e rede e podem ser ainda classificados como ativo, proativo e reativos como pode ser visto na Tabela 3.10.

Características	Trabalhos
Host	[16] [79] [7] [8] [9] [10]
Rede	[7] [8] [11]
Reativo	[16] [79] [7] [8] [9] [10] [11] [13]
Ativo	[11] [13]
Proativo	[7] [13]

Tabela 3.10: Características encontradas nos trabalhos

No capítulo seguinte será apresentada a abordagem proposta para a realização da investigação forense em ambiente de rede baseada na geração dos alertas.

4 Investigação Forense em Ambiente de Rede

Este capítulo se dedica a uma visão da solução proposta nesta dissertação, onde é apresentada a abordagem para investigação forense no ambiente de rede. Inicialmente, é discutido o processo proposto e as etapas que devem ser realizadas para que a investigação possa acontecer. Em seguida é mostrada a abordagem utilizada e a modelagem da solução.

4.1 Processo Genérico de Investigação

Mais cedo ou mais tarde quase toda rede poderá experienciar uma violação de segurança da informação [38], uma vez que invasões não podem ser completamente prevenidas [82]. Diante disto, procedimentos que possam, não só identificar a ocorrência de ações maliciosas, como também entende-las e aprender com elas, são essenciais.

Neste contexto, foram criados vários métodos para realização da investigação de ações maliciosas de forma que existem várias técnicas investigativas provadas e métodos para a computação forense. Como pode ser visto no Capítulo 3 Seção 3.1, vários modelos e *frameworks* vêm sendo propostos ao longo dos anos, mas sem obter um consenso.

Por este motivo, e para adaptar os procedimentos da abordagem utilizada, montamos um processo para a investigação no ambiente de rede. Este é inspirado nos modelos e *frameworks* relacionados, e terá como estímulo da investigação a geração de alertas dos sensores IDS configurados e relacionados aos ativos (Figura 4.1).

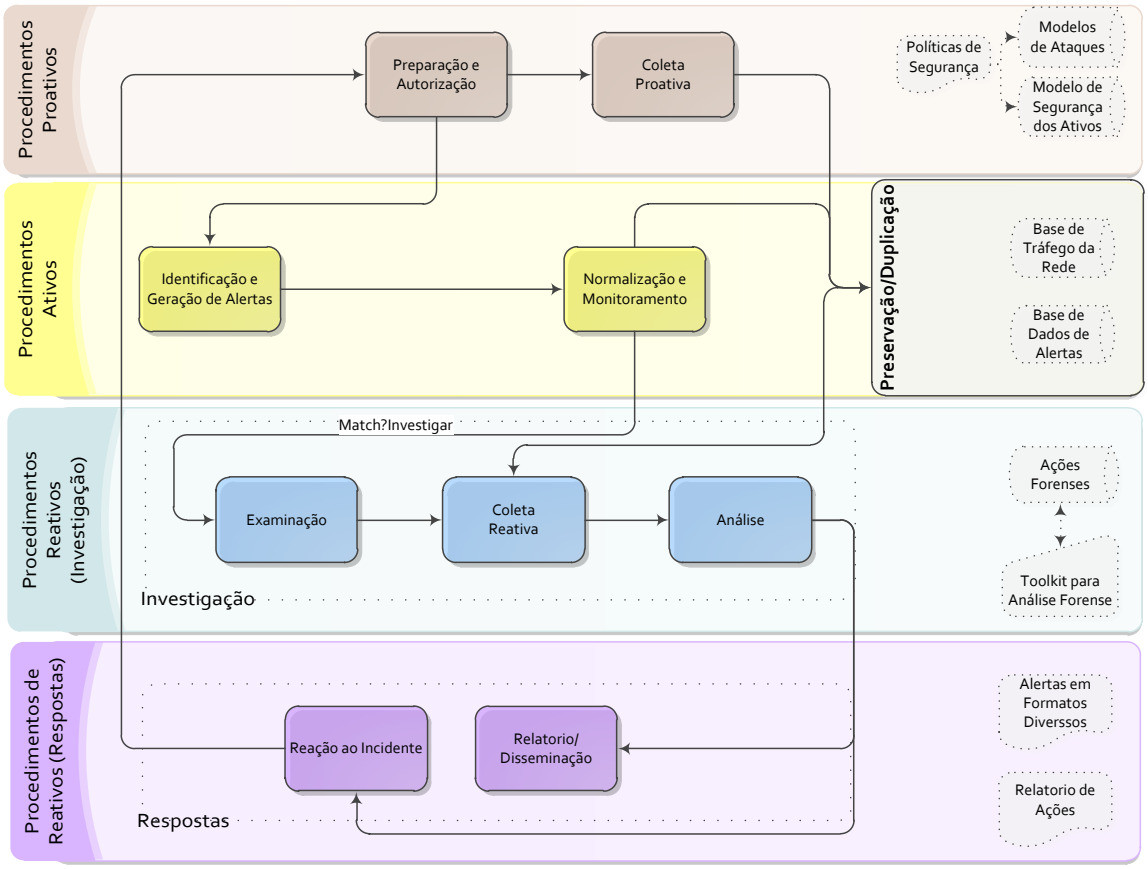


Figura 4.1: Processo genérico de investigação no ambiente de rede baseado na geração de alertas

O objetivo também é reutilizar conceitos e ferramentas que já funcionam na rede, de forma a facilitar a investigação de riscos monitorados pelas IDS, proporcionando capacidade reativa de cunho investigativo. São utilizadas basicamente as etapas: preparação e autorização; identificação e geração de alertas; coleta proativa; preservação/duplicação; normalização e monitoramento; investigação (examinação, coleta reativa e análise) e respostas (reação ao incidente e relatório/disseminação). As próximas seções caracterizam as etapas do processo genérico apresentado.

4.1.1 Preparação e Autorização

Na rede, para que os procedimentos forenses possam ser aplicados de maneira satisfatória, é importante que o ambiente esteja preparado. Ou seja, o processo deve ser iniciado antes que o incidente seja detectado. Desta forma, a primeira etapa é a de **preparação e autorização**.

O objetivo principal desta etapa é que existam subsídios para recuperar evidências e artefatos utilizados em um ataque ou violação de segurança. Para isto, é preciso que existam mais que apenas sensores de segurança na rede como IDS, analisadores de pacotes, *firewalls*, entre outros, posicionados de forma estratégica. Aqui também deve ser feita a regulamentação do ambiente (considerando a legislação e criando procedimentos relacionados à segurança da informação).

No Brasil, não existem normas específicas que regem a forense computacional; existem apenas normas gerais que abrangem todos os tipos de perícia contidas no Código de Processo Penal [83], e que salvo algumas peculiaridades, podem ser adotadas para forense computacional.

Além disto, deve-se evitar o problema de violação de privacidade (garantida pela Constituição Federal de 1988 a todos os cidadãos brasileiros [84]). Este problema pode geralmente ser contornado através de políticas de segurança claras e de conhecimento de todos os usuários, que abordem a vistoria de arquivos, *e-mails* e outros dados pessoais que possam trafegar pela rede.

Para tanto, nesta etapa devem ser definidas, de forma geral, as estratégias de segurança que irão ser adotadas na organização e sua regulamentação. Um outro aspecto importante da preparação do ambiente é o fato de precisar estar em constante adaptação, ou seja, deve ser focada na melhoria contínua da segurança do ambiente, uma vez que o avanço das tecnologias, assim como das técnicas dos ataques estão em constante atualização.

Desta forma, como toda fase de planejamento, esta é a mais delicada e trabalhosa do processo e requer um esforço conjunto da organização. Uma vez que esta não seja executada de maneira adequada, pode fazer com que os dados existentes na rede não sejam suficientes para a investigação. Boas práticas para a execução desta etapa podem ser encontradas em padrões e normativas como [20] [27] [28] [29] [30]. E, uma vez definidas estas estratégias, é importante que sejam devidamente documentadas.

Para nossa abordagem, são artefatos importantes:

- Documentação de autorização para monitoramento do tráfego de rede. Onde os documentos que devem ser de conhecimento dos usuários ou grupos de usuários (contendo por exemplo políticas restritivas), devem ser devidamente assinados

garantindo a não violação de privacidade e autorização da organização para execução dos procedimentos;

- As políticas de segurança definidas [1] para os ativos;
- Definição dos níveis de proteção dos ativos (análise de riscos para os ativos) e perfil de monitoramento (riscos aceitos para os ativos) e modelagem dos ataques monitorados;
- Seleção e posicionamento dos sensores/ferramentas que serão utilizadas para a segurança da rede;
- Mapeamento das políticas definidas para os ativos a fim de configurar os sensores adequadamente e direcionar os esforços de monitoramento/análise;
- Estratégias para continuidade do negócio, como a definição de respostas, ferramentas e ações forenses.

O ambiente (rede e ativos em geral) e as pessoas envolvidas devem estar, então, preparados para que possa ser realizada uma investigação forense, independente de como esta será iniciada.

4.1.2 Identificação e Geração de Alertas

A partir da definição das estratégias e posicionamento dos sensores de segurança na etapa anterior, os dados que trafegam pela rede devem ser examinados pelos mesmos. Uma vez que este tráfego é processado pelos sensores, quando há a ocorrência de uma violação de segurança devem ser gerados alertas/*logs* que indiquem este acontecimento.

Boas práticas recomendam que estes alertas gerados sejam armazenados em um servidor distinto para prover confiabilidade e evitar que sejam destruídos durante uma invasão do sistema. Além disto, evita a possibilidade de um atacante realizar o ataque de *logging* [2], minimizar possíveis interferências dos relógios dos sistemas (facilitando o correlacionamento dos eventos).

[1] Intrusões e ações maliciosas devem ser definidas na política de segurança. Caso não seja definido o que é e o que não é permitido no sistema, não consegue-se entender o que é uma intrusão ou uma violação de política de segurança

[2] Ataque de negação de serviço contra o próprio sensor, que consiste em gerar eventos em excesso até que o disco onde são armazenados os *logs* fique cheio e o sistema trave em conseqüência disto [5]

Deve ser definido também o tempo de disponibilidade *on line* dos alertas e *logs* na rede. Uma das técnicas recomendáveis para o gerenciamento e armazenamento *off line* dos alertas é utilizar *checksums* (código usado para verificar a integridade de dados) criptográficos dos arquivos que serão armazenados. Os *checksums* devem ser mantidos também separados dos *logs* para que possam ser usados para verificar a integridade destes, caso venham a ser necessários para ajudar na investigação de incidentes de segurança descobertos posteriormente [27].

No ambiente de rede é comum termos sensores heterogêneos como IDS diferentes, entre outros. Esta heterogeneidade é vista também nos formatos dos alertas, e, como abordado no Capítulo 2 Seção 2.3.1, existem vários formatos propostos para estes alertas de segurança. Uma vez que o estímulo da abordagem são alertas gerados é preciso que estes estejam em um formato comum. Para isto, foi acrescentada no modelo investigativo a etapa de **normalização e monitoramento** dos alertas.

4.1.3 Normalização e Monitoramento

Para facilitar o gerenciamento das informações contidas nos alertas, nesta etapa os alertas deverão ser transformados para um formato comum antes de sua manipulação, ou seja, toda a manipulação de informações contidas nos alertas deverá ser feita nas cópias normalizadas e os alertas em seus formatos originais devem ser preservados.

O monitoramento tem o objetivo principal de direcionar os recursos disponíveis para a investigação de acordo com os riscos aceitos na fase de preparação, através da definição de perfis específicos, de acordo com o nível de proteção desejável para o ativo. Devem ser especificadas as condições para início da investigação. Estas podem ser ocorrências como: geração de um alerta específico, geração de uma sequência de alertas em uma restrição de tempo, identificação de uma sequência de palavras-chave nos alertas gerados, entre outros que possam ser definidos.

Esta etapa está diretamente ligada à preservação/duplicação dos alertas, uma vez que os componentes para a realização dela deverão ser posicionados junto ao servidor de *logs*.

4.1.4 Preservação/Duplicação

Apesar de não existir legislação específica para forense computacional no Brasil, existem algumas passagens no Código de Processo Penal referentes a perícia que devem ser seguidas. Dentre elas pode-se destacar uma e a sua possível abordagem computacional, estritamente ligada a esta etapa:

- Art. 170 Nas perícias de laboratório, os peritos guardarão material suficiente para a eventualidade de nova perícia. Sempre que conveniente, os laudos serão ilustrados com provas fotográficas, ou microfotográficas, desenhos ou esquemas [83].

Apesar da abordagem ser direcionada à rede, a etapa de **preservação /duplicação** tem o objetivo de manter os dados originais inalterados. Os dados que trafegam pela rede ou segmento monitorado devem ser preservados e armazenados para auxiliar procedimentos investigativos posteriores, principalmente devido à grande volatilidade dos dados na rede.

Além disto, os dados devem ser duplicados antes da manipulação ou correlacionamento; todas as operações que possivelmente venham a ser executadas nos alertas ou tráfego de rede gravado, devem ser executadas em suas cópias. Os dados originais devem ser mantidos de forma segura para garantir sua integridade (utilização de *checksum* criptográficos, por exemplo), obedecendo um tempo determinado pelas políticas de segurança.

4.1.5 Coleta Proativa e Reativa

No processo proposto a coleta é dividida em duas etapas: proativa e reativa. A coleta proativa trata da coleta do tráfego da rede de forma que este seja preservado (Seção 4.1.4) e possa ser utilizado em processamentos posteriores. Os sensores de coleta devem ser posicionados estrategicamente considerando além do ativo a topologia utilizada na rede.

Já a coleta reativa, que é realizada na fase de investigação, trata da coleta de dados complementares (alertas, *dumps*, etc.) baseados nos dados da examinação do alerta.

4.1.6 Investigação

Na etapa de normalização e monitoramento quando há uma identificação da ocorrência de um perfil monitorado deve ser requisitada a investigação do evento. A etapa de **investigação** é composta basicamente por 3 (três) processos: examinação; coleta reativa e análise. O objetivo da etapa de investigação é identificar o caminho que foi percorrido pelo atacante dentro da rede até que o ativo atacado fosse comprometido (princípio básico da forense computacional).

Para isto, os indícios da ocorrência de atividades maliciosas devem ser examinados (etapa de examinação do perfil que deu início à investigação) com o objetivo de extrair informações relevantes para iniciar a etapa de coleta reativa, que deve coletar os demais alertas e/ou registros de tráfego da rede baseados nestas informações.

Uma vez que estes dados sejam coletados da-se início à etapa de análise das informações. Nesta etapa os alertas devem ser agrupados e relacionados às etapas do modelo de ataque. Quando baseado em monitoramento de perfis, esta etapa também deve ser direcionada a eles. Ou seja, procedimentos analíticos devem ser específicos ao tipo de ataque monitorado.

4.1.7 Respostas

A etapa de respostas está relacionada às respostas dos riscos aceitos. Esta foi dividida em duas subcategorias: resposta ao incidente e relatório/disseminação.

A etapa de **resposta ao incidente**, é onde devem estar relacionadas as respostas direcionadas aos incidentes detectados (procedimentos adicionais que devam ser realizados nos ativos, como por exemplo o isolamento de um ativo e substituição por seu *backup*, ou ainda o redirecionamento do tráfego malicioso para um ativo *honeypot* semelhante ao ativo atacado, entre outros).

O relatório de execução de procedimentos é um outro processo importante, não só para validar os resultados obtidos, como para a conformidade com um outro artigo do Código Penal importante para a perícia, que pode e deve ser aplicado no âmbito computacional:

- Art. 171 - Nos crimes cometidos com destruição ou rompimento de obstáculo a subtração da coisa, ou por meio de escalada, os peritos, além de descrever os vestígios, indicarão com que instrumentos, por que meios e em que época presumem ter sido o fato praticado [83].

Além disso, nesta etapa pode-se fazer a **disseminação** do incidente através da geração de alertas nos formatos de outras IDSs para atualização de suas bases de dados e/ou criação de novas regras, e/ou ainda, relatar alertas associados às ações executadas para a investigação.

4.2 Mecanismo de Investigação Forense em Ambiente de Rede

Diante das etapas do processo proposto, foi modelado o mecanismo para auxiliar sua execução no ambiente de rede, o qual está ilustrado na Figura 4.2. A abordagem busca investigar ataques cujos riscos foram aceitos e são processados pelos sensores IDSs, desta forma foram criados quatro componentes:

- Componente Administrativo: com os conceitos referente a manutenção das informações no mecanismo, como modelo de segurança dos ativos, modelo dos ataques, ações forenses e configuração dos monitores;
- Componente Monitor/Normalizador: monitora a geração de alertas, executa sua normalização e verifica se está em conformidade com o perfil definido. Este componente deve ser colocado junto a fonte de alertas, no caso da abordagem aqui utilizada, ficará junto ao servidor de *logs* centralizado, onde ficarão as bases de dados dos alertas, e tráfego da rede;
- Componente de Investigação: componente configurado em um servidor específico para a investigação, onde devem estar configuradas as ferramentas úteis ao processo;
- Componente de Reação: este componente ficará agregado ao componente investigativo e deverá ser acionado após a análise para execução de modificações de políticas e/ou disseminação do ataque.

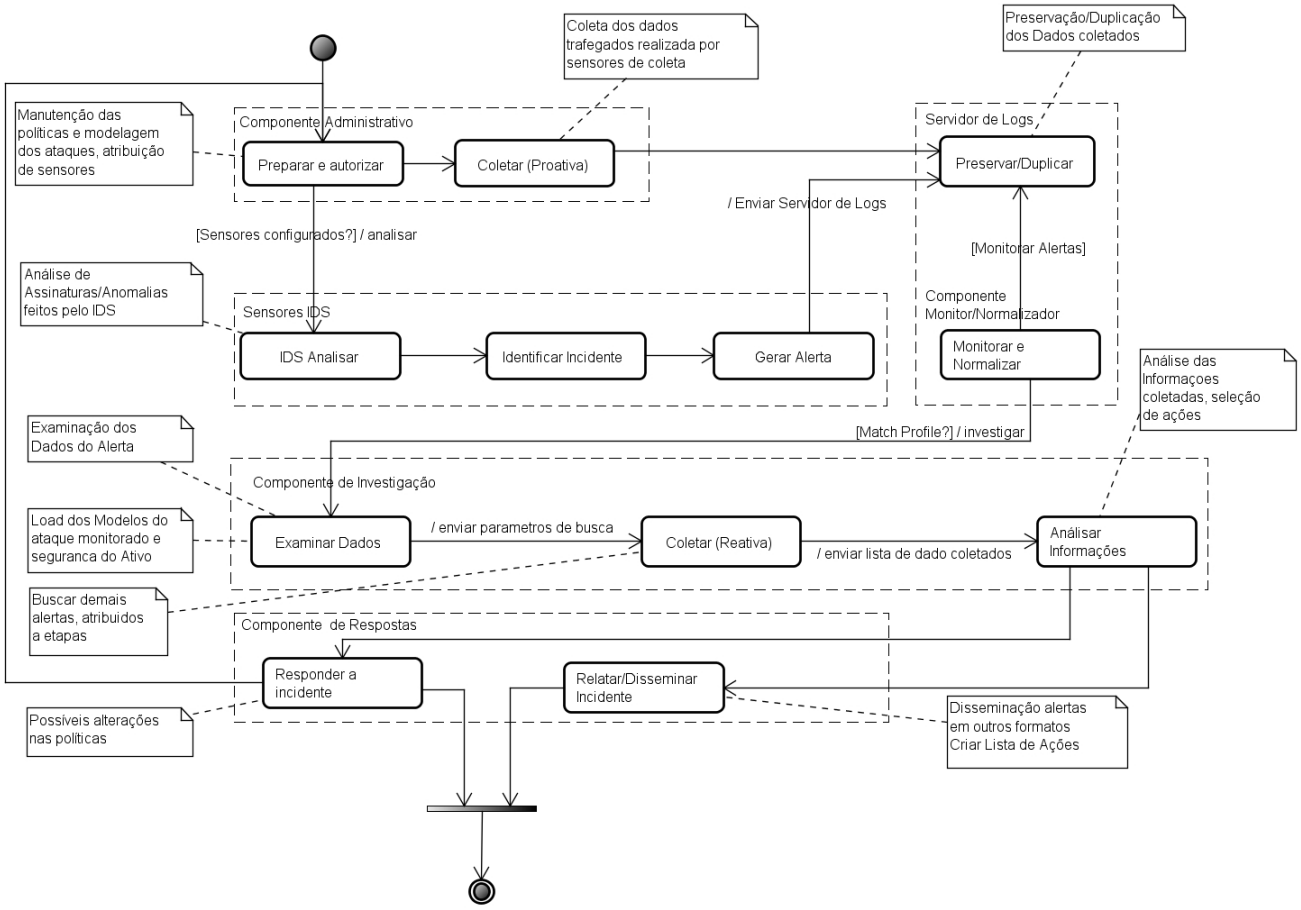


Figura 4.2: Abordagem utilizada para o mecanismo

Para agilizar o processo de desenvolvimento do protótipo, garantir a flexibilidade da aplicação e possível reutilização dos conceitos aplicados (além da execução de transformações, e demais benefícios trazidos pela MDE), foram desenvolvidos metamodelos para os conceitos utilizados na investigação utilizando a abordagem MDA.

Para a abordagem a utilização do MDA é importante, inicialmente, para a geração de código e instanciação dos modelos agilizando o processo de desenvolvimento. Além disso, permite a criação dos repositórios com os modelos de ataques, da segurança aplicada aos ativos, das ações forenses, além de possibilitar a normalização dos alertas para um modelo genérico através das transformações entre modelos. As próximas seções descrevem os componentes e os metamodelos criados.

4.2.1 Componente Administrativo

Para auxiliar a etapa de preparação, e disponibilizar ao mecanismo informações relevantes para a abordagem (políticas, configuração dos sensores, etc.), foram

criados os seguintes artefatos: um metamodelo genérico de segurança dos ativos; o metamodelo genérico para modelagem de ataques; e metamodelo genérico de ações forenses.

Metamodelo Genérico de Segurança dos Ativos

O objetivo básico da utilização deste metamodelo é fazer com que as informações de segurança aplicada aos ativos no ambiente monitorado estejam disponíveis. Para isto, é necessário saber que políticas de segurança estão empregadas ao ativo monitorado e os sensores envolvidos.

A Figura 4.3, mostra o relacionamento entre os conceitos abordados. É importante salientar que a instanciação destes conceitos deve referir-se à políticas e/ou controles implementáveis pelos sensores disponíveis na rede, ou ainda pelo Sistema Operacional. O metamodelo foi construído conforme o metametamodelo Ecore.

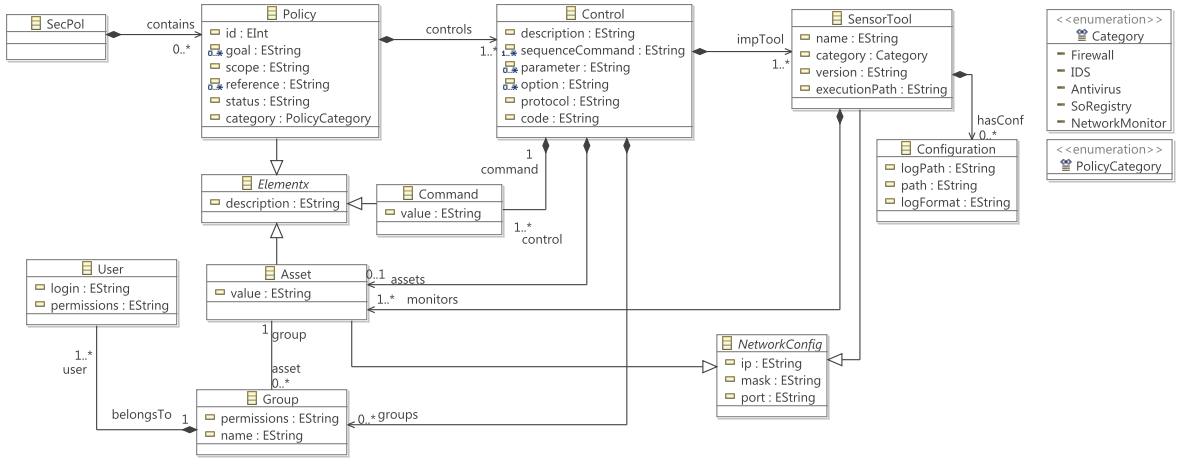


Figura 4.3: Metamodelo genérico de segurança dos ativos

Este metamodelo deve ser instanciado baseado nas políticas estabelecidas. A Tabela 4.1 mostra a descrição das classes com os conceitos básicos abordados neste metamodelo.

CONCEITO		PROPRIEDADE	DESCRIÇÃO	TIPO
Element	Classe abstrata com informações comuns a elementos do metamodelo.	description	descrição do elemento	EString
NetworkConfig	Classe abstrata com informações de configuração na rede	ip	endereço IP	EString
		mask	mascara de rede	EString
		port	porta	EString
SecPol	Classe que contém as políticas de segurança (elemento principal)			EString
Policy	Classe referente às políticas de segurança	id	id da política	EString
		goal	objetivo	EString
		scope	scopo	EString
		references	referências normativas	EString
		status	status de funcionamento	EString
Control	Controles implementados nas políticas	sequenceCommand	sequência de comandos	EString
		parameter	parametros	EString
		option	opções de execução	EString
		protocol	protocolo	EString
		code	código adicional	EString
SensorTool	Sensor que implementa a política	name	nome da ferramenta	EString
		category	categoria	EString
		version	versão	EString
		executionPath	path de execução	EString
Configuration	Classe com as configurações básicas do sensor	logPath	path do log	EString
		path	path da configuração	EString
		logFormat	formato do log	EString
Asset	Classe com as caracteríticas básicas dos ativos	id	identificador do ativo	EString
		value	nivel de proteção	EString
Group	Informações de grupos	permissions	permissões de grupo	EString
User	Informações sobre usuários	id	id do usuário	EString
		login	login do usuário	EString
		permissions	permissões de usuário	EString
Category	Enumeração com as categorias de sensor	-	Firewall | IDS | Antivirus | So-Registry | NetworkMonitor	Enum

Tabela 4.1: Elementos do metamodelo segurança dos ativos

Metamodelo Genérico de Modelagem de Ataques e Metamodelo de Ações Forenses

Uma vez que forem estabelecidas políticas dentro do ambiente monitorado, é também importante definir como uma delas é violada. Para isto foi definido um metamodelo genérico para modelagem de ataques Figura 4.4.

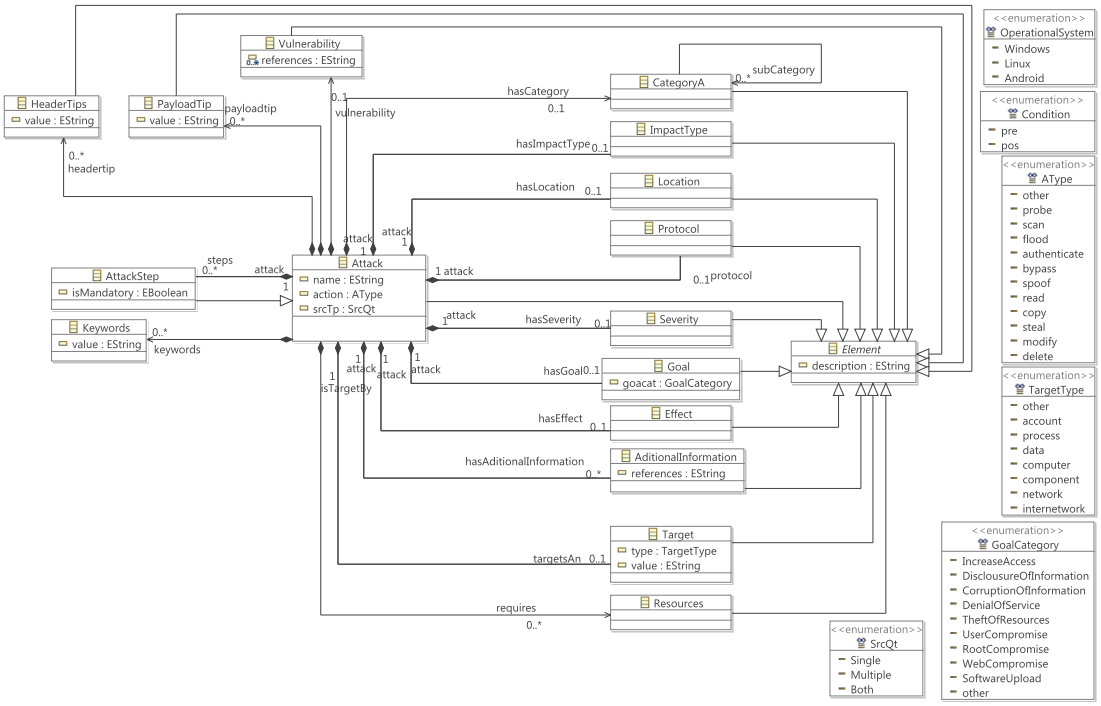


Figura 4.4: Metamodelo genérico para modelagem de ataques

Independente das tecnologias utilizadas para a segurança do ambiente, é necessário que haja conhecimento dos ataques ou ações que são consideradas maliciosas dentro da rede. Isto auxilia a compreender o que é "dito" pelos sensores utilizados, assim como estabelecer perfis de monitoramento e direcionar os recursos disponíveis (ferramentas e pessoas). Além disto, possibilita melhorar as ações de resposta configuradas nos sensores.

Para a execução da forense computacional o conhecimento sobre os ataques é fundamental, uma vez que mesmo utilizando diferentes técnicas, ações ou *exploits*, o princípio básico de funcionamento da maior parte dos ataques geralmente é o mesmo, mesmo que a vulnerabilidade explorada seja diferente e as ferramentas utilizadas sejam outras.

O metamodelo proposto, Figura 4.4 visa principalmente modelar múltiplas etapas, baseado no *modus operandi* básico de ataques, modelando os cenários. Os conceitos abordados estão relacionados na Tabela 4.2

Além de modelar os ataques foram modeladas também ações forenses que possam ser executadas para investigar suas etapas. A Figura 4.5 representa os conceitos modelados, onde a classe *ForensicAction* representa a ação, com seu objetivo, tipo de alvo (onde a ação deve ser executada). Além disto, a classe *ForensicTool* representa

CONCEITO		PROPRIEDADE	DESCRIÇÃO	TIPO
Element	Classe abstrata com informações comuns a elementos do metamodelo.	description	descrição do elemento	EString
Attack	Classe com informações do ataque herda de element (elemento principal)	id		EString
AttackStep	Passos do ataque herda de element	type		Condition
ImpactType	Tipo de impacto do ataque herda de element			
Location	Localização herda de element			
Protocol	Protocolo utilizado no ataque			
Category	Categoria de classificação (taxonomia) do ataque			
Severity	Severidade calculada para o ataque			
Goal	Objetivo no ataque			
Effect	Efeito causado			
AditionalInformation	Informações adicionais			
Target	Alvo do ataque	type		Estring
Condition	Enumeração referente a pre/pós condição da etapa do ataque		Pre/Pós	Enum
OperationalSystem	Enumeração de Sistemas Operacionais			Enum

Tabela 4.2: Elementos do metamodelo genérico modelagem de ataques

as ferramentas adicionadas ao *toolkit* e relacionadas à execução de uma ação e sua plataforma, com seu nome, *path* de instalação e comando a ser executado.

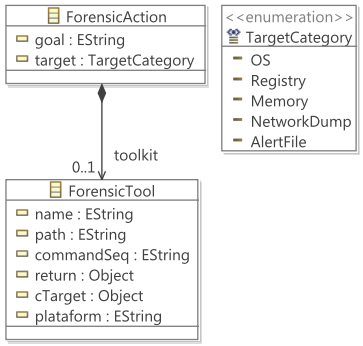


Figura 4.5: Metamodelo genérico para ações forenses

4.2.2 Componente Monitor/Normalizador

Muito se discute na comunidade sobre os padrões para alertas de segurança e *logs*, como já falado no Capítulo 2 Seção 2.3. No entanto, como comentado anteriormente, até então, não há um padrão plenamente adotado para a geração de alertas de segurança, e quando se trabalha com ambientes heterogêneos isto pode se tornar um grande problema. Por este motivo, foi elaborado um metamodelo genérico para estes eventos, como pode ser visto na Figura 4.6.

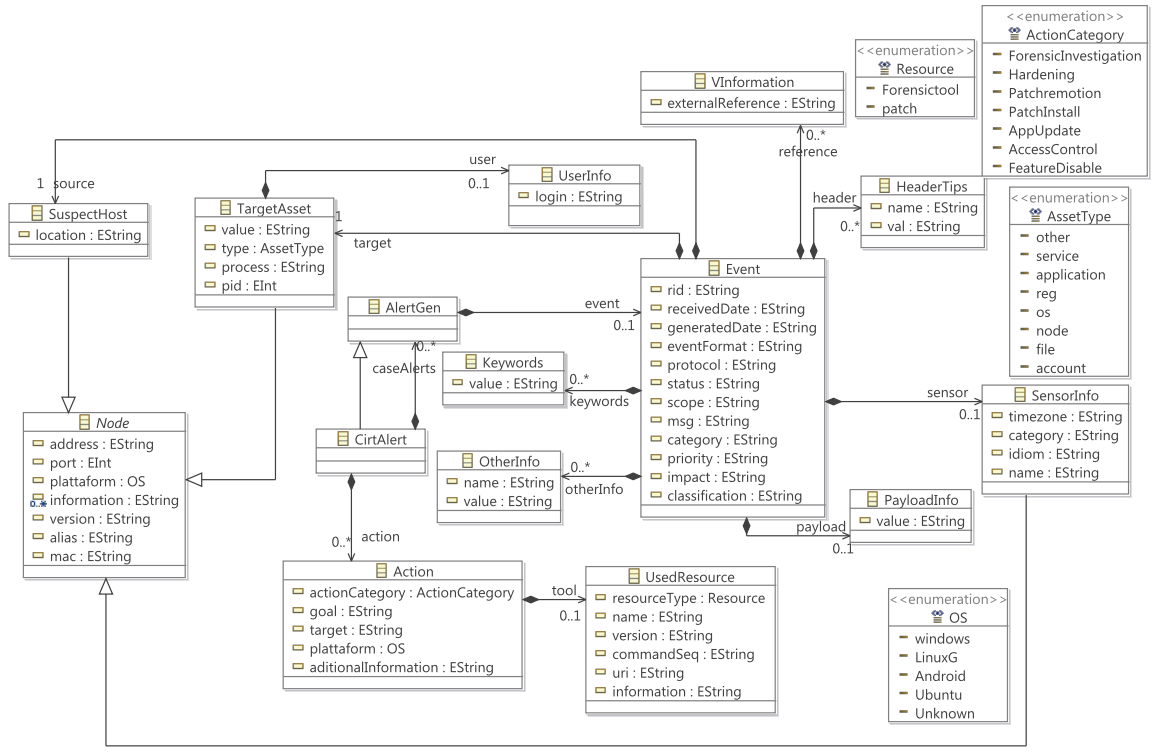


Figura 4.6: Metamodelo genérico para alertas de segurança

O objetivo é manter a interoperabilidade entre os alertas das soluções utilizadas pelos sensores, assim como associa-los às ações forenses que possam ser executadas, para extrair as evidências relacionadas a eles. A Tabela 4.3 descreve os principais conceitos deste metamodelo.

O componente Monitor/Normalizador deve ser colocado junto ao servidor de *logs*. Como os alertas originais devem ser preservados, quando o servidor recebe um alerta este processo faz uma cópia do arquivo, executa sua transformação e salva o alerta normalizado em um repositório. Este processo é replicado para cada tipo de alerta monitorado. Para que este funcione, é preciso que estejam feitos os metamodelos dos alertas dos sensores envolvidos, assim como escritas as regras de transformação para o modelo genérico de alertas.

Com o alerta normalizado, o componente monitor/normalizador, verifica se este se encaixa no perfil em que foi configurado, caso haja uma correspondência faz a requisição para processo de investigação onde é enviado para o componente de investigação o alerta e/ou lista de alertas que a iniciaram, como pode ser visto no diagrama de atividades da Figura 4.7.

CONCEITO		PROPRIEDADE	DESCRIÇÃO	TIPO
AlertGen	Classe que contém os elementos básicos (mínimos) de um alerta/log de segurança. Inicialmente foi especializado em duas classes: IdsAlert e CirtAlert			
IdsAlert	Além das Informações de um alerta genérico aborda também os conceitos específicos de um evento de segurança alertados por sensores IDS			
CirtAlert	Contém as informações dos eventos e informações de ações destinadas a CIRTs			
Element	Classe abstrata com informações comuns a alguns alementos do metamodelo	description	descrição do elemento	EString
Node	Classe abstratas com informações comuns aos nós.	address	endereço do nó (Ip ou Mac)	EString
		port	porta do nó	EInt
		alias	nome ou alias do nó	EString
		information	Informacoes adicionais sobre o nó (decoy, spoof, etc.)	EString
		plattaform	Plataforma do nó	OS
		version	Versão do nó	EString
SuspectHost	Classe que mostra as informações do nó suspeito de ser o atacante. Herda as informações da classe Node.	location	localização do nó	EString
AttackInfo	Mostra as informações do ataque. Herda as informações de Element.	priority	prioridade do ataque	EInt
		impact	impacto do ataque	EString
VInformation	Classe que mostra ífnormações da vulnerabilidade explorada.	externalReference	informações/referências sobre a vulnerabilidade explorada	EString
TargetAsset	Classe que mostra as informações do ativo atacado.Herda as informações da classe Node.	value	valor estimado do ativo atingido	EInt
		type	tipo de ativo	AssetType
		process	processo atingido	EString
Event	Classe que mostra as informações recebidas no evento de segurança gerado pelo sensor. Herda as informações da classe Element.	receivedDate	timestamp do recebimento do evento	EString
		generatedDate	timestamp da geração do evento	EString
		id	identificador do evento	EString
		eventFormat	formato do evento	EString
		protocol	protocolo do evento	EString
		aditionalInformation	informacoes adicionais sobre o evento	EString
		referenceBinary	arquivo binario de referencia	Estring
		status	status do evento	EString
		scope	escopo do evento	EString
		msg	mensagem descritiva do evento	EString
SensorInfo	Classe que mostra as informações do sensor de segurança que gerou o evento. Herda as informações da classe node	timeZone	timezone utilizado no sensor	EString
		category	categoria do sensor	Estring
		idiom	idioma usado nos eventos do sensor	EString
		name	nome identificador do sensor	EString
UserInfo	Classe que descreve as informações de usuário que possam vir no log	login	login identificador do usuário	EString
Header	Classe que contém as informações do cabeçalho que possam vir no evento	field	campo do cabeçalho	EString
		value	valor do campo	EString
PayloadInfo	Classe que contém as informações de payload que possam vir no evento	value	payload do evento	Estring
Action	Classe que contém informações relacionadas a ações ligadas a eventos de segurança	actionCategory	Categoria da ação	ActionCategory
		goal	Objetivo da ação	EString
		target	Alvo da ação	EString
		plattaform	Plataforma da ação	OS
		aditionalInformation	Informações adicionais	EString
UsedResource	Recurso utilizado na execução de ações	version	Versão do recurso	EString
		commandSeq	Sequência de comandos	EString
		uri	Uri do recurso	EString
		information	Informações adicionais	EString
		name	nome do recurso	EString
		resourceType	Tipo de recurso	Resource
Keywords	Palavras chaves que possam identificar o evento	value	valor da palavra chave	EString

Tabela 4.3: Elementos do metamodelo genérico de alertas

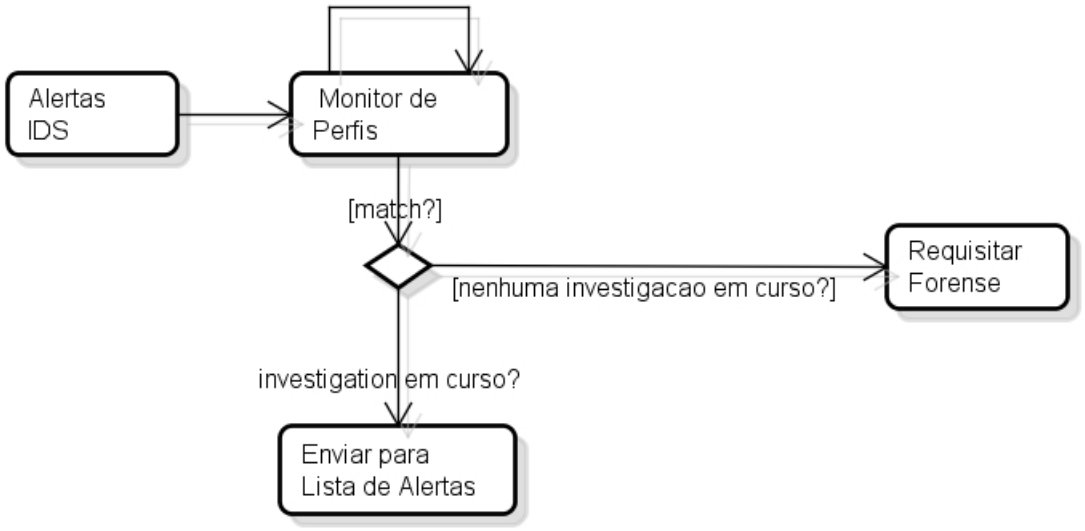


Figura 4.7: Monitor de logs

Este componente é essencial na abordagem proposta, uma vez que IDSs diferentes geralmente geram alertas em formatos diferentes, e com mensagens diferentes, para alertas correlatos. Além disto, alguns alertas são definidos com poucas informações, como origem, destino, nome e hora do evento. Outros proveem informações mais ricas como portas, serviços, processos, informações do usuário, entre outras.

Inicialmente, apenas dois tipos de alertas podem ser normalizados no componente: alertas em formatos XML e alertas em formato de texto plano, de forma que temos dois processos distintos de transformação no componente.

Para realizar a transformação reversa dos alertas (texto para modelo) em formato XML, foi necessário a injeção de código XML para que este possa ser serializado conforme o metamodelo XML, disponível nos exemplos de transformações do projeto Eclipse [85]. A partir do modelo com o XML injetado é escrita a transformação em ATL para a transformação do modelo específico do alerta para o modelo genérico da Figura 4.6.

Já para alertas em texto plano a abordagem utilizada foi a criação de uma gramática que descreve o alerta. Esta foi criada com o auxilio do *framework* Xtext [73], onde foram utilizados por este componente o *parser*, serializador e o deserializador gerados pelo Xtext.

A Figura 4.8 mostra o diagrama de atividades que descreve o funcionamento deste componente. Onde é verificado a que categoria o alerta pertence (se XML ou texto plano) e depois é escolhida a estratégia de transformação.

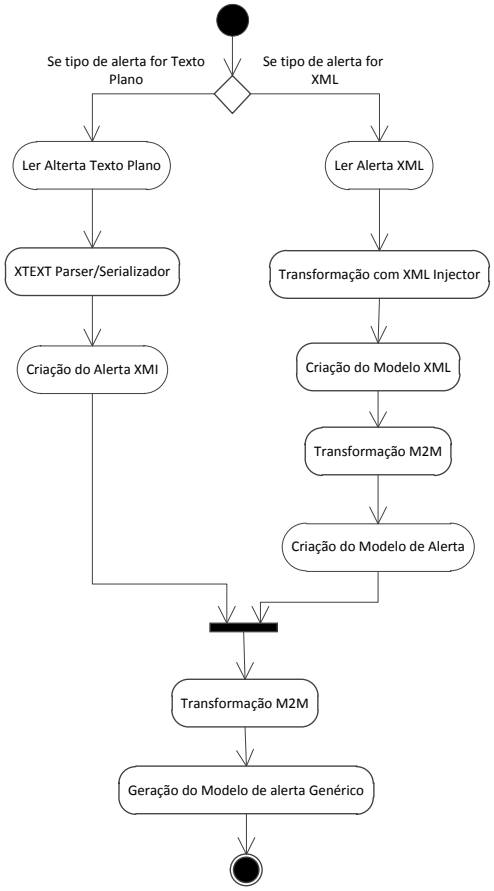


Figura 4.8: Processo de transformação dos alertas

4.2.3 Componente de Respostas - Disseminação

O funcionamento deste componente é similar ao processo de normalização executado no componente Monitor/Normalizador. No entanto, o objetivo é, a partir dos alertas, conforme o metamodelo genérico de alertas, gerar alertas em formatos diversos (específicos dos IDSs, como também compartilhar o relatório do caso, unindo os alertas utilizados na investigação e as ações forenses do tipo de ataque investigado).

Este componente de respostas foi limitado apenas à geração dos alertas em formatos diversos e relatório administrativo. A interação com a fase de planejamento

ainda é de responsabilidade do analista de segurança (decisões tomadas a partir das informações obtidas).

4.2.4 Componente de Investigação

Este componente é o núcleo do mecanismo de investigação proposto. A Figura 4.9 mostra as classes e os métodos principais que fazem parte deste componente assim como seus relacionamentos.

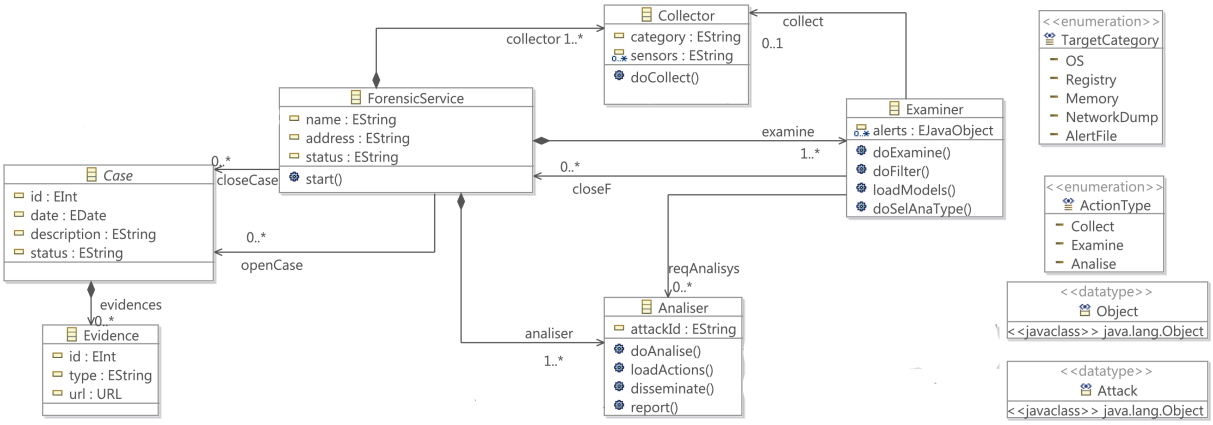


Figura 4.9: Componente de investigação

A classe *ForensicService* é a classe responsável pela configuração básica do serviço de investigação e gerenciamento dos processos de examinação, coleta e análise, onde o método *start()* é o responsável por iniciar o processo. Uma vez iniciada uma investigação é aberto um caso (*Case*), onde serão armazenadas algumas informações do caso.

A examinação é o primeiro procedimento realizado quando uma investigação é é iniciada, onde são examinadas as informações do perfil com o(s) alerta(s) que iniciou a investigação. A partir das informações filtradas no examinardor (doFilter()) serão carregados o modelo do ataque, o modelo de segurança relacionado ao ativo, as informações das políticas, e a lista dos demais sensores IDS que monitoram o ativo (método *loadModels* da classe *Examiner*).

Uma vez de posse destas informações, o examinador cria o filtro, onde o processo de coleta reativa é iniciado, para que os demais dados sejam coletados na base de alertas para a análise (inicialmente foram usados apenas os alertas dos IDSs).

Em seguida, os dados são passados para o analisador (*Analizer*). Esta classe pode ser extendida, de forma que possam ser executados procedimentos mais específicos relacionados ao cenário monitorado, como seleção de ações específicas ou execução de algoritmos mais complexos. O método *doAnalise* é o método que inicia a analise a partir do conjunto de alertas coletados. O método *loadActions* é o método que irá carregar as ações cadastradas para as etapas. O método *requestResponse* inicializa a geração das respostas.

4.3 Considerações Finais

Neste capítulo, apresentou-se a proposta de modelo para investigação forense de rede iniciada pela geração de alertas de IDSs. A partir desta apresentação podemos comparar o modelo proposto com os trabalhos relacionados apresentados no Capítulo 3.

A Tabela 4.4 mostra as características utilizadas como critério de comparação com os trabalhos apresentados na Seção 3.1.

CARACTERÍSTICAS	DFRWS [16]	ADFM [79]	IRP [7]	IPM [8]	EMCI [9]	OBFDIP [10]	GFNF [11]	MCVDF [13]	Modelo Proposto
Host	SIM	SIM	SIM	SIM	SIM	SIM	NÃO	NÃO	NÃO
Rede	NÃO	NÃO	SIM	SIM	NÃO	NÃO	SIM	SIM	SIM
Reativo	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM	SIM
Ativo	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO	SIM	SIM	SIM
Proativo	NÃO	NÃO	SIM	NÃO	NÃO	NÃO	NÃO	SIM	SIM
Conceitual	SIM	SIM	SIM	SIM	SIM	SIM	NÃO	SIM	NÃO

Tabela 4.4: Comparação entre os trabalhos - Modelos e frameworks para forense computacional

Já na Tabela 4.5 são mostrados alguns critérios de comparação com os trabalhos apresentados na Seção 3.2.

CARACTERÍSTICAS	Weaving Ontologies [14]	Method Ontology [15]	Generic Metamodel for IT [81]	Modelo Proposto
Baseado em Ontologias	SIM	SIM	NÃO	NÃO
Utilização de MDA	NÃO	NÃO	SIM	SIM
Reutilização de Ferramentas	NÃO	NÃO	NÃO	SIM
Divulgação dos Ataques	NÃO	NÃO	SIM	SIM
Utilização para Automatização	SIM	SIM	NÃO	SIM
Conhecimento Forense	SIM	SIM	NÃO	SIM
Alertas Diversificados	NÃO	NÃO	NÃO	SIM

Tabela 4.5: Comparação entre os trabalhos - Reuso do conhecimento forense / modelagem de ataques

Neste Capítulo foram apresentados também os metamodelos utilizados na construção do *mecanismo* utilizados para o desenvolvimento do protótipo da ferramenta. O próximo capítulo apresentará a implementação dos componentes do protótipo, um exemplo de utilização e os resultados obtidos.

5 Protótipo, Testes e Resultados

Para auxiliar o processo proposto na Seção 4.1 o mecanismo apresentado na Seção 4.2 foi implementado um protótipo do mecanismo de investigação. Este capítulo apresenta as ferramentas utilizadas, a arquitetura e implementação do protótipo. Posteriormente é apresentado o exemplo de utilização e os resultados obtidos nos testes dos componentes.

5.1 Arquitetura e Implementação

De acordo o modelo do mecanismo apresentado no Capítulo 4 o protótipo foi projetado para tornar o mecanismo utilizável. A Figura 5.1 mostra o diagrama de implantação para os componentes, onde pode ser visto como estes devem ser distribuídos.

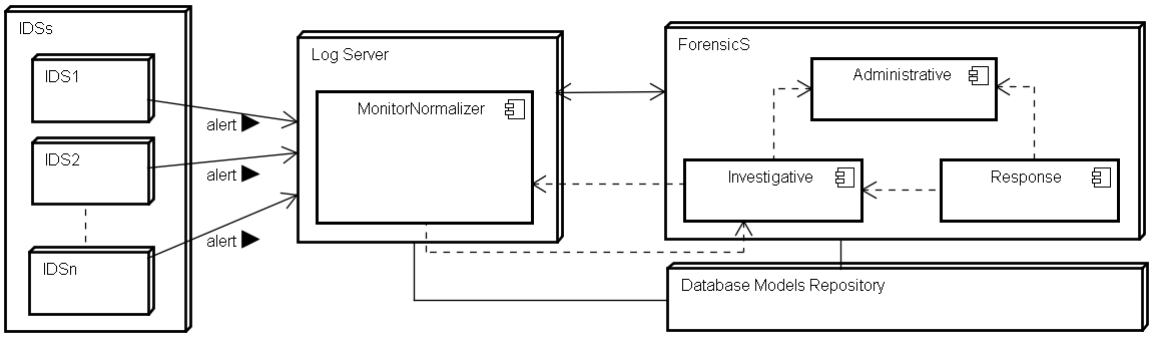


Figura 5.1: Diagrama de implantação

Ferramentas Utilizadas

O protótipo foi desenvolvido utilizando a abordagem de MDE, onde o *framework* MDA foi utilizado, através do EMF, com recursos como: geração de código em JAVA; transformações de texto para modelo; transformações modelo para modelo e modelo para código.

Para isto, foram utilizadas algumas ferramentas que auxiliaram o processo de modelagem e implementação, tais como:

- EMF: *framework* de modelagem e ambiente de geração de código para ferramentas e outras aplicações baseadas em modelos de dados estruturados, e explora todas as facilidades providas pelo Eclipse, juntando em um ambiente as tecnologias que dão base à MDA [72]. Este foi utilizado como ambiente para desenvolver os modelos, regras de transformação e gramáticas;

- ECORE: linguagem de metamodelagem que faz parte da EMF. É o modelo usado para representar modelos em EMF [86]. Os metamodelos foram feitos conforme o metamodelo ECORE;

- ATL: linguagem usada para as transformações dos modelos [70] [71]. Foi a linguagem utilizada para escrever as regras de transformação de modelo para modelo para a normalização dos alertas;

- *Xtext*: componente do *Textual Modeling Framework* (TMF) que suporta o desenvolvimento da gramática de uma DSL que pode ser usada para gerar um metamodelo baseado no ECORE e provê interoperabilidade com as tecnologias baseadas no EMF [87] [73]. Este foi utilizado para a definição das gramáticas dos alertas textuais, onde foram aproveitados o *parser*, o serializador e deserializador gerados;

- Teneo: solução de mapeamento objeto-relacional para persistência em base de dados para o EMF que pode ser integrado tanto com o *Hibernate*, quanto com o *EclipseLink* [77]. O teneo foi utilizado para auxiliar o mapeamento objeto-relacional e criação do repositório de modelos, utilizando a API gerada pelo EMF, juntamente Hibernate e JPA;

- Ferramentas diversas como: o IDS Snort 2.9 [88], Suricata IDS [89], *Wireshark* [90], Tcpslice [91], entre outras ferramentas presentes na distribuição Linux *Backtrack* 5 [92], utilizadas para a preparação do ambiente, montagem do *toolkit* e simulação do ataque;

- Linguagem Java [93], os *frameworks* JSF [94], o sistema de gerenciamento de banco de dados MySQL [95] e o servidor de aplicações Apache Tomcat 7 [96] utilizados no desenvolvimento do protótipo *stand alone*;

- Bibliotecas como IO [97] e Lang [98] do projeto Commons da Apache Software Foundation que foram utilizadas para a implementação do componente Monitor/Normalizador, possibilitando ao componente monitorar vários *paths* de *logs*;
- A biblioteca Jcraft [99] para o envio e execução dos comandos utilizando SSH2.

O protótipo foi pensado no formato de uma aplicação *stand alone* utilizando os modelos em tempo de execução. Do ponto de vista do projeto Eclipse [100], aplicações *stand alone* são aplicações que não se encaixam bem em um *shell* Eclipse, como: componentes do lado servidor, *applets*, aplicações com interface gráfica *Swing*, entre outros [72]. No entanto, para o EMF, são apenas mais código Java que pode explorar suas características [72].

A estrutura dos componentes do protótipo foi baseada na arquitetura *Model View Controller* (MVC), uma arquitetura muito usada atualmente no desenvolvimento de projetos de *software*, onde o código extra foi criado e adicionado ao código gerado pelo EMF. As camadas definidas na aplicação podem ser vistas na Figura 5.2.

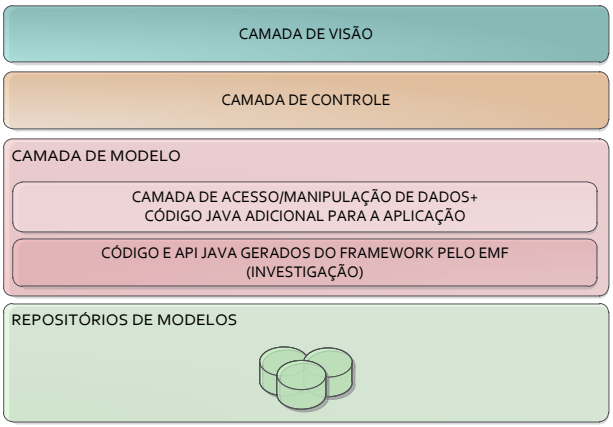


Figura 5.2: Arquitetura de implementação

A camada de Repositórios de Modelos é a camada responsável pelo armazenamento persistente das informações dos modelos utilizados. Nesta camada estão localizadas as bases de dados utilizados pelo protótipo. Esta foi implementada utilizando o sistema de gerenciamento de bando de dados remacional MySQL [95]. Para os repositórios fossem criados conforme os modelos utilizados foi utilizado o *framework* Teneo [77] para auxilio do mapeamento objeto relacional, onde a criação do *EntityMa-*

nager é feita através do *HbEntityDataStore* do Teneo para que seja feita a persistência dos dados no repositório.

A camada de Modelo representa os dados da aplicação e as regras do negócio que gerenciam o acesso e a modificação dos dados. Esta camada está dividida em duas subcamadas: a camada com o Código e API gerado pelo mecanismo de investigação através do EMF e a camada de Acesso/Manipulação de Dados, onde fica o código adicional, criado para utilização dos modelos em conjunto com o repositório e implementação das classes para o exemplo.

A Camada de Controle, por definição, é a camada responsável por receber a entrada de dados e fazer a validação de acordo com as especificações dos modelos. Nesta implementação, a camada de controle foi implementada utilizando as anotações *ManagedBeans* para a integração com a Camada de Visão.

A Camada de Visão, responsável pela interação visual com os usários do protótipo, foi implementada utilizando páginas JSF, inicialmente apenas operações básicas de *Create, Read, Update and Delete* (CRUD) para os dados do componente Administrativo no repositório.

Para a geração o código Java conforme os modelos dos componentes mostrados no Capítulo 4 foram feitas algumas modificações, para que o código gerado pudesse ser compatível com o modelo de implantação. Como há a necessidade de serializar os objetos para o tráfego na rede, antes de gerar o código a partir do modelo criado foi criada uma interface chamada *SerializableEObject* que extende as classes *org.eclipse.emf.ecore.EObject* e *java.io.Serializable*. O motivo é que a classe *EObject*, que por padrão é estendida pelos elementos raiz do modelo gerador, não é serializavel.

O modelo gerador (*ForensicActions.genmodel*) foi então criado para que fosse feita a geração do código Java do mecanismo. Este foi feito a partir do modelo do componente de investigação apresentado na Seção 4.2.4, por ser o elo entre os demais modelos que compõem o mecanismo, gerando o código de todos os componentes. O *genmodel* foi personalizado, onde uma das alterações relevantes foi a alteração da propriedade *Root Extends Interface* do *.genmodel* onde foi colocado o caminho da interface *SerializableEObject* anteriormente citada. A Figura 5.3 mostra a estrutura de código gerada, assim como algumas propriedades do *.genmodel*.

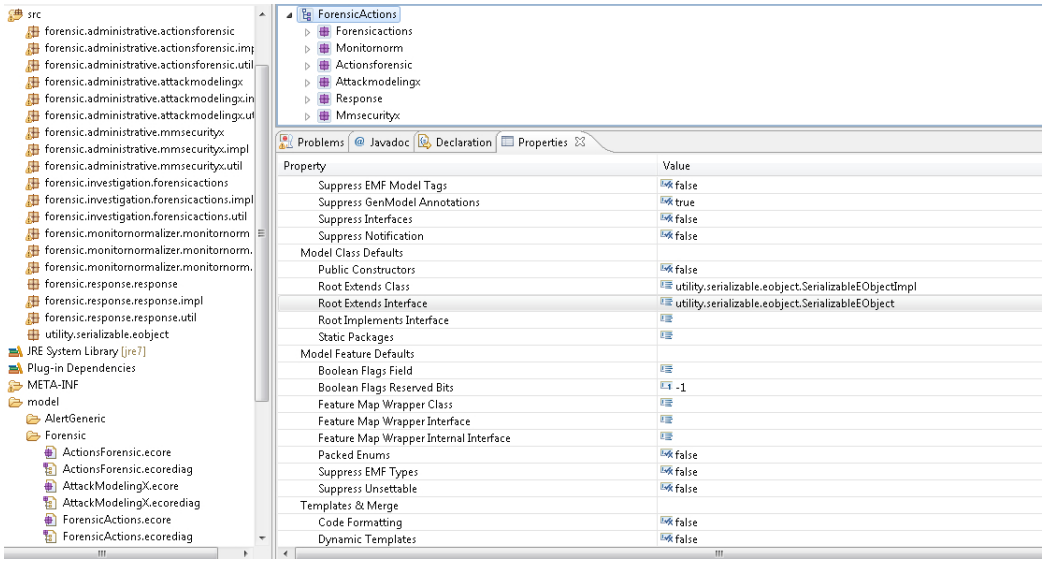


Figura 5.3: GenModel e geração de código no EMF

Com o código do projeto gerado, este foi retirado do EMF, foram acrescentadas todas as depedências (bibliotecas do EMF, Teneo, ATL, Xtext e Commons IO, etc.) necessárias à implementação do mecanismo, implementadas as metaoperações dos componentes e e acrescentadas as classes complementares criadas. Para que o código implementado nas metaoperações não fossem sobrescritos, no caso de regeração do código, as anotações *@generated* foram substituídas pela anotação *@generated not*.

A Figura 5.4 mostra o diagrama de pacotes do protótipo, onde pode-se vizualizar como ficou, de forma geral, a estrutura do projeto e as principais classes adicionais inseridas.

No pacote *views* foram acrescentadas as páginas JSF [94] criadas para a administração do protótipo. Já no pacote *controllers* foram colocados os *beans* gerenciados para cada uma das classes principais do componente administrativo, para facilitar a manipulação das informações.

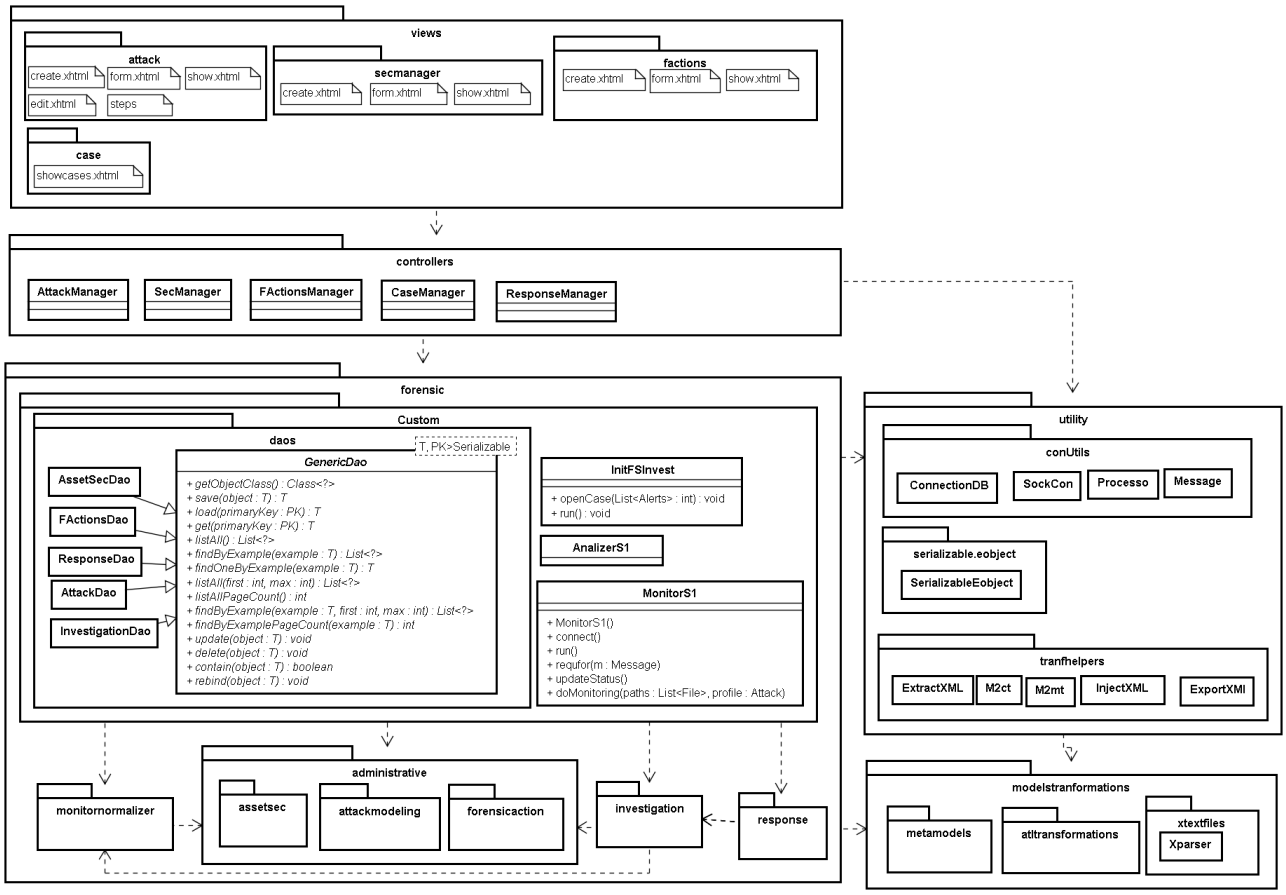


Figura 5.4: Diagrama de pacotes do protótipo

O pacote *forensic* contém o código que foi gerado pelo modelo de investigação no EMF. Neste foram implementadas as metaoperações do componente Monitor/Normalizador, como por exemplo o método *doMonitoring* onde foi inserido um *FileAlterationListener* (Commons IO [97]) para a lista de arquivos de alertas monitorados e implementado o médoto *onFileChange* que requisita a normalização (doNormalize) sempre que novos alertas são gerados nos *paths* monitorados.

No componente Monitor/Normalizador após a etapa de normalização é feita a identificação do cenário, verificado se os alertas fazem parte do ataque monitorado (perfil). Como os alertas em seus formatos originais nem sempre possuem todas as informações do modelo genérico de alertas apresentado na Seção 4.2.2 (Figura 4.6), a exemplo exemplo da classificação do alerta, que pode ser fundamental na iden-

tificação de alertas correlatos, foi utilizado o algorítmo mostrado na Figura 5.5 para a identificação dos ataques.

Entrada: A, alerta no formato genérico
Saída: L, lista de alertas de acordo com a identificação do cenário
Dados: matchScenery,profile, assetname, stepList, LIMIT

```
1  p = monitor.getProfile();
2  stepList = monitor.getProfile().getHasSteps();
3  enquanto !matchScenery faça
4      se A.getEvent().getTarget() == monitor.getAsset() então
5          se (A.getEvent().getMsg() == p.getMsg()) e (stepList==null) então
6              se p.srcTp.equals(multiple) então
7                  occurrence++;
8                  se aoccurrence == LIMIT então
9                      matchScenery = true;
10
11             fim
12             senão
13                 se findKeywords(A)==true então
14                     matchScenery = true;
15                 fim
16             fim
17         fim
18         senão
19             Iterator it = monitor.getProfile.getHasSteps().iterator();
               enquanto it.hasnext() and matchSteps==false faça
20                 st = it.next(); se findSteps(st, A) então
21                     matchSteps = true;
22                 fim
23             fim
24             matchScenery = true;
25         fim
26     fim
27 fim
```

Figura 5.5: Algoritmo para identificação do cenário

Onde inicialmente é identificado o destino do alerta, caso seja o ativo monitorado será então verificada a compatibilidade da mensagem do alerta, caso seja igual à descrição do perfil (descrição do ataque) e não hajam passos para este, será verificado o limite de ocorrências da mensagem para o ativo, caso seja igual é identificada a ocorrência do cenário. É feita ainda uma nova comparação entre as *keywords* do ataque e a mensagem, e neste caso, se forem encontradas considera-se que a mensagem é equivalente.

Caso existam etapas, estas são comparadas à mensagem até que sejam identificados, seguindo os mesmo critérios de comparação explicados anteriormente para descrição principal do ataque através do método findSteps(st, A).

Também no pacote *forensics* foi inserido o pacote *custom* que contém o código adicional ao código gerado, onde o pacote *daos* contém as classes que foram utilizadas para manipular os modelos e repositórios e configurações adicionais. A classe principal que este pacote contém é a classe *GenericDao*. Esta é uma classe abstrata genérica com os métodos básicos principais para manipulação e acesso ao repositório de dados (*save*, *update*, *delete*, *listAll*, *findByExample*, entre outros mostrados no diagrama), onde são passados para seu construtor o nome da classe e o tipo da chave primária.

Ainda no pacote *custom* pode-se identificar as classes *MonitorS1*, *InitSInvest* e *AnalizerS1*.Para este protótipo, a classe *MonitorS1* e a classe *InitSInvest* foram implementadas usando *sockets* e *threads* (cliente e servidor respectivamente). A classe MonitorS1 é a classe que instância um objeto do tipo *MonitorNormlizer* para enviar a mensagem de requisição de início de investigação (*requFor(m:Message)*) contendo a requisição e os alertas identificados para o cenário. A classe *InitSInvest* instância a classe *ForensicService*, que por sua vez dá inicio aos procedimentos de investigação. Já a classe *AnalizerS1* estende a classe *Analizer* do componente de investigação, onde foi implementado o primeiro cenário.

O pacote *utility* contém as classes utilitárias aos componentes como as classes para acesso ao repositório de modelos, *helpers* para a execução das transformações, arquivos de configuração dos componentes, das mensagens enviadas, entre outros. No diagrama podemos identificar a classe ConnectionDB responsável por fazer a conexão com o repositórios de modelos, utilizando o *framework* Teneo. A classe *SockCon* com as configurações de conexão por *sockets* usadas nas classes *MonitorS1* e *InitSInvest*, a

interface *SerializableEobject* e as classes *helpers* para as transformações e exportação de XMI, utilizadas pelo MonitorNormalizer e para as Respostas (geração de alertas em outros formatos).

O pacote *modelstransformations* foi utilizado para organizar os modelos e transformações. Neste, o pacote *metamodels* contém os metamodelos dos alertas e demais metamodelos necessários às tranformacoes dos alertas, o pacote *atltransformations* contém as transformações escritas em ATL (assim como os arquivos .asm). Já pacote *xtextfiles* contém as classes que referenciam os projetos feitos utilizando o Xtext (cujos projetos foram exportados como .jars para o projeto do mecanismo).

5.2 Exemplo e Resultados

Para a execução do exemplo foi configurado um segmento simplificado de rede, com uma topologia *Dual Home* e um *firewall bridge*. A topologia utilizada na simulação pode ser vista na Figura 5.6. Foram utilizadas máquinas virtuais com o *software* de virtualização VMware [101] para emular os *hosts* da rede em um ambiente controlado para a realização dos testes.

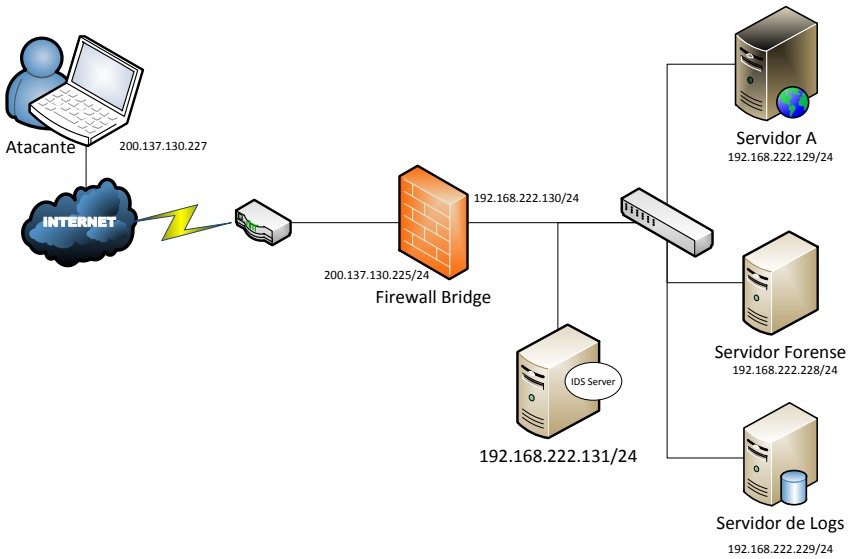


Figura 5.6: Topologia utilizada na simulação

Para o *firewall bridge* foi utilizado um servidor virtual com Ubuntu Server 10.04 onde a configuração de *bridge* foi feita com o pacote *bridge-utils* e o *firewall* com

IpTables [102] (inicialmente configurados apenas *forward* de portas e NAT transparente) com duas interfaces de rede em modo *bridge* (eth0 - 200.137.130.225/24 e eth1 - 192.168.222.130/24). Ainda neste servidor foi configurado um *sniffer* com a ferramenta Wireshark em modo texto (tshark) [90], capturando os pacotes que trafegavam pelas interfaces para o ativo monitorado, a fim de preservar o tráfego da rede (coleta proativa).

O servidor NIDS (192.168.222.131/24) foi configurado com o Sitema Operacional Ubuntu Server 10.04, com a interface de rede em modo promíscuo, destinado à configuração do(s) IDS(s). Neste foram instalados os NIDS Snort, gerando os alertas no modo textual *Full* e o IDS Suricata, gerando alertas através do Syslog.

O servidor de endereço 192.168.222.229/24 é márquina configurado para ser o servidor de *Logs* com o Sitema Operacional Ubuntu Server 10.04 e recebe os alertas gerados pelas ferramentas. Já o servidor de endereço 192.168.222.228/24 é onde fica nossa aplicação, onde foi utilizada a distribuição BackTrack 5 e demais ferramentas necessárias o funcionamento do protótipo.

O servidor A (192.168.222.129/24) é um possível alvo de ataque. Neste foi instalada a distribuição Metasploitable, que é uma distribuição Ubuntu específica para testes de penetração que possui diversos serviços instalados, cujas vulnerabilidades podem ser exploradas pelo *framework* Metasploit [103]. O objetivo é que os procedimentos do cenário modelado obtenham sucesso, podendo assim gerar os alertas, a fim de testar as funcionalidades do protótipo. O servidor B é uma outra máquina vúlneravel. Neste foi instalado o Windows 2003 Server, que possui algumas vulnerabilidades, muito interessantes para testes de penetração.

A máquina usada pelo invasor utiliza a distribuição BackTrack 5 [92]. Os procedimentos de ataque foram realizados utilizando o *framework* Metasploit [103], e ferramentas como o Nmap disponíveis no Backtrack. Estes procedimentos foram executados apenas uma vez, no entanto, como o ambiente passou pela fase de preparação, com a instalação e configuração das ferramentas para monitoramento e captura do tráfego, o *dump* capturado foi armazenado (coleta proativa), e posteriormente, pode ser utilizado para repetições com o auxilio da ferramenta TCPReplay [104].

5.2.1 Cenário Modelado

Para a execução dos testes foi modelado um ataque exemplo ao Gerenciador de Aplicativos Tomcat 5.5 presente no Servidor A, funcionando na porta 8180, com o objetivo de gerar alertas para testar os componentes. O ataque consiste em obter credenciais a partir um ataque de brute force, execução de injetção de um *payload* malicioso no Tomcat (vulnerabilidade da configuração padrão), utilização do *shell* ganho para escalada de privilégios no Linux.

Os passos executados foram:

1. Identificação do alvo e busca de vulnerabilidades (*probing*);
2. Bruteforce com dicionário para tentativas de *login* no serviço;
3. Exploração da vulnerabilidade de configuração padrão do Tomcat;
4. *Deploy* de um .war mal intencionado para a obtenção da *shell*;
5. Envio de comandos após obtenção do shell no Tomcat;
6. Escalada de privilégios do usuário para se tornar administrador - *upload* de *exploit*;

Desta forma para iniciar a utilização do protótipo é necessário:

1. Instanciar as informações do ataque que será monitorado (cenário);
2. Instanciar as informações da política de segurança do ativo a ser monitorado para investigação;
3. Ter implementado a classe do analisador (extendida da classe *Analiser*) do cenário;
4. Definir as ações forenses que podem ser executadas;
5. Configurar e iniciar o componente Monitor/Normalizador;
6. Iniciar o serviço de investigação;

A Figura 5.7 mostra parte do formulário utilizado no protótipo para a instanciação e edição dos dados do ataque no protótipo e a Figura 5.8 mostra o fragmento do XMI do ataque exportado do repositório.

Figura 5.7: Exemplo de formulário - inserção/edição de dados do ataque

```
<administrative:Attack xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:administrative="http://attackmo
  <targetsAn description="Tomcat Service" type="component" value="3"/>
  <hasImpactType description="medium"/>
  <protocol description="TCP"/>
  <hasGoal description="Unauthorized access to the server" goacat="TheftOfResources"/>
  <hasEffect description="Integrity, disponibility and/or confidentiality compromisseds"/>
  <hasAditionalInformation references="doc.emergingthreats.net/2008453"/>
  <hasLocation description="remote"/>
  <requires description="user and password"/>
  <steps description="Tomcat Brute force scan for passwords" name="User Information Probe" action="scan" srcTp="Multiple">
  </steps>
  <steps description="A malicious war is deployed on the server with a shell" name="Tomcat  Remote War Deployed" isMandatory="true">
  </steps>
  <steps description="Exploit Deployed Obtain Privilege" name="Remote File Uploaded" isMandatory="true">
    <steps description="Remote GCC code Compilation" name="Remote code Compilation Attempted" isMandatory="true"/>
  </steps>
  <steps description="An exploit is uploaded to  privilege scalation - identity verification" name="Server user Privilege Scalation" isMandatory="true">
  </steps>
  <keywords value="Brute Force Scan Detected"/>
  <keywords value="Tomcat Unspecified Access"/>
  <keywords value="Tomcat Malicious Shell"/>
  <vulnerability description="weak configuration">
    <references>http://apache.tomcat</references>
```

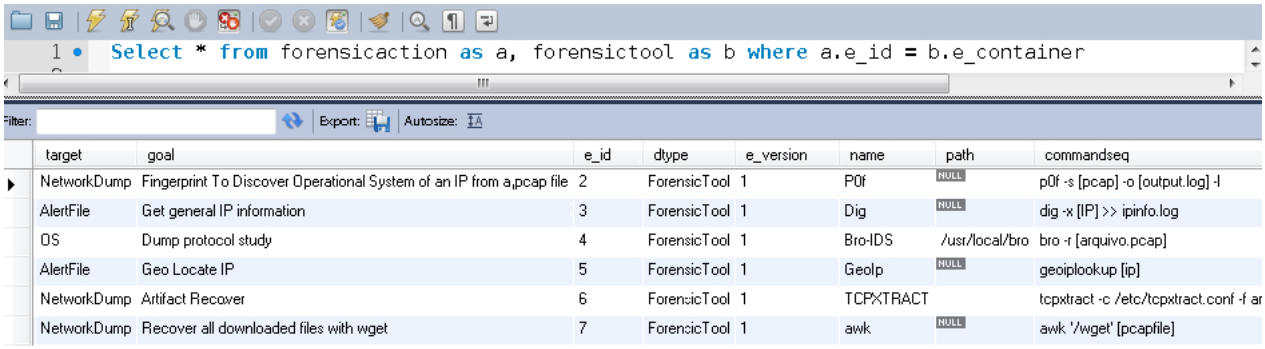
Figura 5.8: Fragmento do XMI do ataque

Na Figura 5.9 podemos visualizar um fragmento política modelada para o ativo que será monitorado no repositório e os sensores atribuídos que irão monitora-

los. Já na Figura 5.10 podemos ver algumas ações que foram adicionadas ao repositório.

```
<?xml version="1.0" encoding="ASCII"?>
<mmsecurityx:SecPol xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mmsecurityx="http://mmsecurityx/1.0" xsi:sch
  <contains description="Investigating Tomcat Scenery1 Attack - .war deploy" id="1" scope="Tomcat Server" status="Active">
    <goal>Monitor Malicious Deployment</goal>
    <goal>Investigate Actions</goal>
    <controls description="UserInformationProbeControl" protocol="TCP" code="">
      <impTool executionPath="" name="Snort" version="2.9" monitors="//@contains.0/@controls.0/@assets">
        <hasConf/>
      </impTool>
      <impTool name="Suricata" monitors="//@contains.0/@controls.0/@assets">
        <hasConf/>
      </impTool>
      <attack href="AttackTomcat.attackmodelingx#//@steps.0"/>
      <assets ip="192.168.222.128" mask="255.255.255.0" port="8180" description="TomCat Aplication Server" group="//@contains.0/@controls.0/@groups.0" value="4" name="Tomcat Ser
      <groups asset="//@contains.0/@controls.0/@assets" name="Admin">
        <user/>
      </groups>
    </controls>
    <controls description="AttackStepFileUpload">
      <impTool name="Snort">
        <hasConf path="/etc/snort/" logPath="/var/log/snort" logFormat=""/>
      </impTool>
      <impTool name="Suricata"/>
      <attack href="AttackTomcat.attackmodelingx#//@steps.2"/>
    </controls>
    <controls description="PrivilegeEscalation" code="">
      <impTool name="Snort"/>
      <impTool name="Suricata"/>
      <attack href="AttackTomcat.attackmodelingx#//@steps.3"/>
    </controls>
    <controls description="RemoteCodeCompilation">
      <impTool name="Snort"/>
      <impTool name="Suricata"/>
      <attack href="AttackTomcat.attackmodelingx#//@steps.2/@steps.0"/>
    </controls>
  </contains>
```

Figura 5.9: Fragmento da política modelada



```
1 •  Select * from forensicaction as a, forensictool as b where a.e_id = b.e_container
```

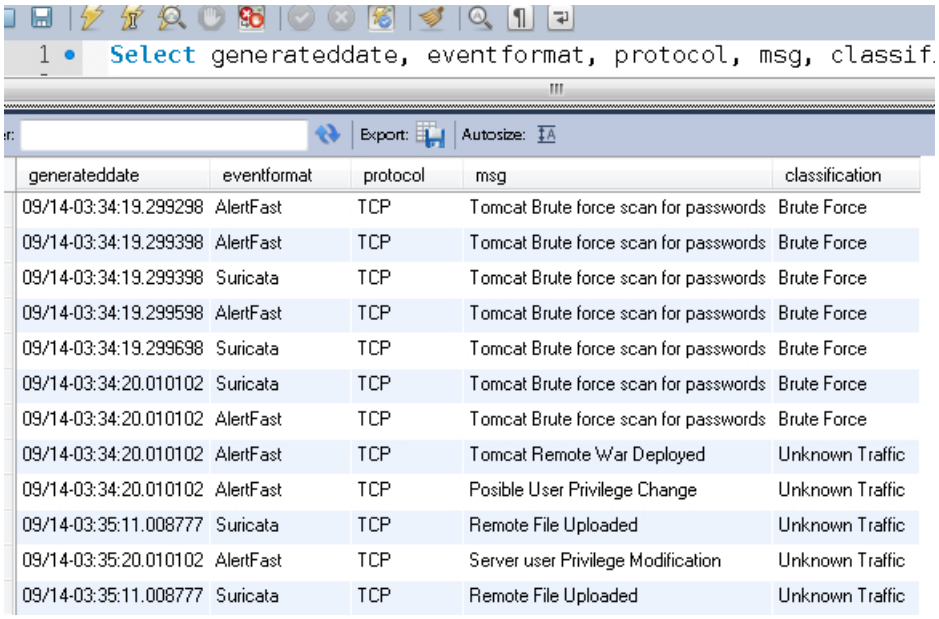
Filter: Export: Autosize:

target	goal	e_id	dtype	e_version	name	path	commandseq
NetworkDump	Fingerprint To Discover Operational System of an IP from a pcap file	2	ForensicTool	1	P0f	NULL	p0f -s [pcap] -o [output.log] -l
AlertFile	Get general IP information	3	ForensicTool	1	Dig	NULL	dig -x [IP] >> ipinfo.log
OS	Dump protocol study	4	ForensicTool	1	Bro-IDS	/usr/local/bro	bro -r [arquivo.pcap]
AlertFile	Geo Locate IP	5	ForensicTool	1	GeoIp	NULL	geoiplookup [ip]
NetworkDump	Artifact Recover	6	ForensicTool	1	TCPXTRACT		tcpxtract -c /etc/tcpxtract.conf -f ar
NetworkDump	Recover all downloaded files with wget	7	ForensicTool	1	awk	NULL	awk '/wget' [pcapfile]

Figura 5.10: Exemplo de ações disponíveis no repositório

Uma vez iniciado o teste, à medida que os alertas ocorrem estes vão sendo foram pelo componente Monitor/Normalizador. Na Figura 5.11 podemos ver alguns

alertas referente ao cenário que foram normalizados e salvos no repositório durante os testes.



```
1 • Select generateddate, eventformat, protocol, msg, classif
```

Export: Autosize:

generateddate	eventformat	protocol	msg	classification
09/14-03:34:19.299298	AlertFast	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:19.299398	AlertFast	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:19.299398	Suricata	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:19.299598	AlertFast	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:19.299698	Suricata	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:20.010102	Suricata	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:20.010102	AlertFast	TCP	Tomcat Brute force scan for passwords	Brute Force
09/14-03:34:20.010102	AlertFast	TCP	Tomcat Remote War Deployed	Unknown Traffic
09/14-03:34:20.010102	AlertFast	TCP	Posible User Privilege Change	Unknown Traffic
09/14-03:35:11.008777	Suricata	TCP	Remote File Uploaded	Unknown Traffic
09/14-03:35:20.010102	AlertFast	TCP	Server user Privilege Modification	Unknown Traffic
09/14-03:35:11.008777	Suricata	TCP	Remote File Uploaded	Unknown Traffic

Figura 5.11: Exemplo alertas normalizados no repositório

Ao final da execução dos procedimentos são gerados os alertas com a finalidade de divulgação. A Figura 5.12 mostra um fragmento do modelo de alerta para divulgação entre centros

```
<alertgeneric:CirtAlert xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:alertg
  <action goal="Geo Locate IP" target="AlertFile" plattaform="LinuxG" aditionalInformation="Returns Geographic Informations about attacker IP">
    <tool name="GeoIp" version="Any" commandSeq="geoiplookup [sourceIp]" information="Get the attacker geo location through IP address"/>
  </action>
  <action goal="Recover the Operational System from source attack IP" target="AlertFile" plattaform="LinuxG">
    <tool name="p0f" commandSeq="p0f -s [pcapfile] -o [outputlog] -l" information="Fingerprint To Discover Operational System of an IP from a p
  </action>
  <action goal="Recover  wgeted links from a dump" target="NetworkDump" plattaform="LinuxG" aditionalInformation="">
    <tool name="awk" commandSeq="awk '/wget' [pcapfile]"/>
  </action>
  <caseAlerts>
    <event rid="" receivedDate="2011/09/14-03:34:19.299298" generatedDate="09/14-03:34:19.299298" eventFormat="AlertFast" protocol="TCP" status
      <target address="200.137.130.225" port="8180"/>
      <source address="200.137.130.227" port="56932"/>
      <otherInfo name="sidInfo" value="[1:10000001:0]"/>
    </event>
  </caseAlerts>
  <caseAlerts>
    <event generatedDate="09/14-03:34:20.010102" eventFormat="AlertFast" protocol="TCP" msg="Tomcat Remote War Deployed" priority="3" classific
      <target address="200.137.130.225" port="8180"/>
      <source address="200.137.130.227" port="56932"/>
      <otherInfo name="sidInfo" value="[1:10000002:0]"/>
    </event>
  </caseAlerts>
  <caseAlerts>
    <event generatedDate="09/14-03:35:11.008777" eventFormat="Suricata" protocol="TCP" msg="Remote File Uploaded" priority="3" classification="
      <target address="200.137.130.225" port="8180"/>
      <source address="200.137.130.227" port="56932"/>
```

Figura 5.12: Fragmento de alerta com as ações para divulgação

5.3 Considerações Finais

Neste capítulo apresentamos as ferramentas utilizadas, a arquitetura e implementação do protótipo do mecanismo de investigação, assim como o exemplo e aplicação e os resultados obtidos nos testes dos componentes. O próximo capítulo apresenta os objetivos alcançados neste trabalho e suas limitações e sugestões para trabalhos futuros.

6 Conclusões e Trabalhos Futuros

No passado *hackers* tinham que criar suas próprias ameaças do zero. Este processo complexo limitava o número de invasores a um pequeno conjunto de cibercriminosos altamente qualificados [105]. No entanto o cenário hoje proporcionado pelo rompimento das barreiras geográficas de espaço e tempo possibilita a rápida divulgação de novas vulnerabilidades e maneiras de explorá-las.

Desta forma, alguns modos de exploração surgem rapidamente logo após alguns dias da descoberta de tais vulnerabilidades. Além disto, há hoje um forte suporte com *kits* para ataques que premitem adicionar rapidamente código para explorar as novas vulnerabilidades, automatizando e reutilizando informações para estas explorações.

Assim, temos uma realidade onde a disseminação de ataques com novas maneiras para atacar vítimas em potencial, e até mesmo o ressurgimento de vulnerabilidades já conhecidas, é mais rápida que a divulgação e/ou descoberta de ações que possam corrigir tais vulnerabilidades.

Neste contexto vêm sendo criadas diversos IDSs, cada um eficiente na sua especialidade. Desta forma, a manutenção de ambientes compostos por vários sistemas de detecção é cada vez mais comum. Cada um gera seus logs/alertas e relatórios próprios, disponibilizando grandes quantidades de informações, às vezes de difícil interpretação, onde nenhuma das tentativas de padronização obteve, até então, utilização efetiva por meio das soluções existentes no mercado.

Nessa perspectiva, mecanismos que auxiliem as investigações podem ajudar a mitigar riscos. A computação forense, que é uma ciência relativamente jovem e tem seus conceitos evoluindo a cada dia, se faz cada vez mais necessária e suas técnicas, juntamente com os sensores de segurança procedimentos ajudar a entender os procedimentos utilizados para a concretização de objetivos maliciosos, além de validar os resultados obtidos pelos sensores, e auxiliar a investigação de cenários conhecidos.

Contribuições do Trabalho

A principal contribuição desse trabalho foi propor uma modelo para forense computacional no ambiente de rede baseado nos alertas gerados pelos sensores IDS, com o desenvolvimento de uma abordagem e um mecanismo para a realização de forense computacional em ambiente de rede apoiado pela MDE.

Para tal tarefa, alguns resultados ou objetivos específicos foram alcançados, como foi sugerido no início deste trabalho de dissertação:

- A proposta de um modelo, baseado nas etapas de forense computacional, que dentro de um ambiente preparado para forense de rede, possa trabalhar sinergicamente com os alertas gerados pelos sensores IDS;
- Apresentar uma estratégia para a padronização dos alertas de segurança da rede que agregue soluções para a investigação dos eventos utilizando MDA;

Lições Aprendidas e Desafios

No decorrer da pesquisa e desenvolvimento do trabalho destacam-se algumas lições, as quais listamos a seguir:

- A computação forense é uma ciência ainda jovem em relação às demais ciências forenses e seus conceitos ainda estão em constante evolução, principalmente por ser aplicada a ambientes digitais que são dinâmicos por sua natureza;
- Mesmo cenários de ataques muito dinfundidos e conhecidos pela comunidade muitas vezes não são facilmente identificados por um único sensor de seguraça com suas configurações e regras padrões, o que requer esforço adicional como o uso diversas ferramentas, e conhecimento de certa forma avançado dos analistas de seguraça;
- Entender os cenários dos ataques, suas variações e manter-se atualizado nas novas técnicas utilizadas pelos potenciais atacantes é um grande desafio, principalmente devido à grande quantidade de conceitos envolvidos e à velocidade com que as formas de exploração são difundidas;

- O caos da heterogeneidade dos alertas/logs de segurança ainda não parece ter uma solução efetiva, mesmo as várias tentativas de padronização, e soluções como a utilização de transformações axiliadas pela Engenharia Dirigida por Modelos podem auxiliar na mitigação deste problema;
- A utilização de taxonomias diferentes na classificação dos ataques dificulta a identificação correta e agregação dos alertas;

Limitações

Em nossa proposta identificamos as seguintes limitações:

- O componente forense foi programado para execução de apenas um tipo de investigação por vez, não executando investigações concorrentes;
- A abordagem requer uma grande quantidade de armazenamento disponível devido à coleta proativa do tráfego, mesmo esta sendo direcionada ao ativo monitorado;
- A abordagem dependende dos IDSs utilizados e suas configurações para identificar o cenário modelado, que pode ser falha dependendo da habilidade evasiva do indivíduo malicioso e/ou má configuração dos sensores;
- A abordagem não exclui a necessidade de procedimentos investigativos adicionais.

Trabalhos Futuros

A partir deste trabalho inicial, identificamos algumas possibilidades de trabalhos futuros que poderiam ser desenvolvidos, tais como:

- Melhoria do algoritmo para reconhecimento dos padrões de ataque baseado nos alertas;
- Intensificar os testes com a modelagem de cenários de ataques mais complexos e com durações mais longas;

- Expansão do modelo genérico de alertas buscando incorporar mais tipos de alertas;
- Adicionar algoritmos diferentes para a examinação das informações dos alertas e geração de filtros buscando otimiza-los;
- Transformar o componente de investigação em um componente *multithread* para execução de investigações paralelas;
- Expandir os modelos utilizados melhorando, por exemplo, o gerenciamento da coleta de evidências;
- Acrescentar um módulo para geração dos *scripts* em linguagens diverssas para execução de procedimentos investigativos nos *hosts*;
- Colocar as funcionalidades do modelo de investigação como serviços;
- Adaptar a abordagem utilizada a uma abordagem autonômica;
- Integrar o modelo de investigação à base de alertas do IDS NIDIA.

Referências Bibliográficas

[1] FRY, C.; NYSTROM, M. *Security monitoring*. [S.l.]: O'Reilly, 2009. (O'Reilly Series). ISBN 9780596518165.

[2] K., B.; SY. Integrating intrusion alert information to aid forensic explanation: An analytical intrusion detection framework for distributive ids. *Information Fusion*, v. 10, n. 4, p. 325 – 341, 2009. ISSN 1566-2535. Special Issue on Information Fusion in Computer Security. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253509000153>.

[3] ALMULHEM, A. Network forensics: Notions and challenges. In: *Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium on*. [S.l.: s.n.], 2009. p. 463 –466.

[4] LIMA, C. F. L. *Agentes Inteligentes para Detecção de Intrusos em Redes de Computadores*. Dissertação (Mestrado) — Universidade Federal do Maranhão, Janeiro 2002.

[5] CERT.BR. Estatísticas dos Incidentes Reportados ao CERT.br. 2011. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em: 04 fev. 2012.

[6] KLEPPE, A.; WARMER, J.; BAST, W. *MDA Explained: The Model Driven Architecture: Practice and Promise*. 1rd. ed. [S.l.]: Indianapolis: Addison-Wesley, 2003.

[7] MANDIA, K.; PROSISE, C.; PEPE, M. *Incident Response & Computer Forensics*. 2nd. ed. [S.l.]: McGraw-Hill Professional, 2003. ISBN 0072131829.

[8] CASEY, E.; PALMER, G. Digital evidence and computer crime: forensic science, computers and the internet. In: ____. [S.l.]: Academic Press, 2004. cap. 4: The Investigative Process.

[9] CIARDHUAIN, S. O. An extended model of cybercrime investigations. *International Journal of Digital Evidence*, Volume 3, Issue 1, 2004.

[10] BEEBE, N. L.; CLARK, J. G. A hierarchical, objectives-based framework for the digital investigations process. *Digital Investigation*, v. 2, n. 2, p. 147 – 167,

2005. ISSN 1742-2876. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1742287605000307>.

[11] PILLI, E. S.; JOSHI, R.; NIYOGI, R. A generic framework for network forensics. *International Journal of Computer Applications*, Volume 1 - No. 11, 2010.

[12] PILLI, E. S.; JOSHI, R.; NIYOGI, R. Network forensic frameworks: Survey and research challenges. *Digital Investigation*, In Press, Corrected Proof, p. –, 2010. ISSN 1742-2876. Disponível em: <http://www.sciencedirect.com/science/article/B7CW4-4YP6SJY-1/2/10c0909fb97d0954de382e22c99fbc29>.

[13] GROBLER, C.; LOUWRENS, C.; SOLMS, S. von. A multi-component view of digital forensics. In: *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*. [S.l.: s.n.], 2010. p. 647 –652.

[14] HOSS, A.; CARVER, D. Weaving ontologies to support digital forensic analysis. In: *Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on*. [S.l.: s.n.], 2009. p. 203 –205.

[15] SAAD, S.; TRAORE, I. Method ontology for intelligent network forensics analysis. In: *2010 Eighth Annual International Conference on Privacy Security and Trust (PST)*. [S.l.: s.n.], 2010. p. 7–14.

[16] PALMER, G. A road map for digital forensic research. In: *Report From the First Digital Forensic Research Workshop (DFRWS)*. [s.n.], 2001. Disponível em: <http://www.dfrws.org/2001/dfrws-rm-final.pdf>. Acesso em: 10 maio 2011.

[17] JESUS, A. *Cibercrime afetou mais de 431 milhões de pessoas em 2011*. Techtudo. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2012/02/cibercrime-afetou-mais-de-431-milhoes-de-pessoas-em-2011.html>. Acesso em: 03 fev. 2012.

[18] SOUPPAYA, M. et al. Book, Online. *Guide to computer security log management [electronic resource] : recommendations of the National Institute of Standards and Technology / Murugiah Souppaya, Karen Kentc*. Draft. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD :, 2006. 1 v. (various pagings) : p. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf>.

[19] STONEBURNER, G.; GOGUEN, A.; FERINGA, A. Guide for Conducting Risk Assessments. In: NATIONAL (Ed.). *NIST Special Publication 800-30 - Revision 1*. NIST Special Publication 800-86: NIST, 2006. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf>. Acesso em: 13 jan. 2012.

[20] International Organization for Standardization. *ISO/IEC 27001:2005 Information technology - Security techniques - Specification for an Information Security Management System*. [S.l.], 2005.

[21] WHEELER, E. *Security Risk Management: Building an Information Security Risk Management Program from the Ground Up*. [S.l.]: Elsevier Science, 2011. ISBN 9781597496155.

[22] VACCA, J. A. *Network and System Security*. [S.l.]: Elsevier Science, 2010. (Syngress Media). ISBN 9781597495356.

[23] DEBI; ASHENDEN. Information security management: A human challenge? *Information Security Technical Report*, v. 13, n. 4, p. 195 – 201, 2008. ISSN 1363-4127. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1363412708000484>.

[24] WRIGHT, C.; FREEDMAN, B.; LIU, D. *The IT Regulatory and Standards Compliance Handbook: How to Survive Information Systems Audit and Assessments*. [S.l.]: Syngress Pub., 2008. ISBN 9781597492669.

[25] KIZZA, J. *A Guide to Computer Network Security*. [S.l.]: Springer, 2009. (Computer Communications and Networks). ISBN 9781848009165.

[26] FIGG, W.; ZHOU, Z. A computer forensics minor curriculum proposal. *J. Comput. Small Coll.*, Consortium for Computing Sciences in Colleges, USA, v. 22, p. 32–38, April 2007. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=1229637.1229642>.

[27] International Organization for Standardization. *ISO/IEC 27002:2005Information technology - Security techniques - Code of practice for information security management*. [S.l.], 2005.

[28] International Organization for Standardization. *ISO/IEC 27005 Tecnologia da Informacao - Tecnicas de Seguranca - Gestao de Riscos de Seguranca da Informacao*. [S.l.].

[29] LONG, J. O. *ITIL Version 3 at a Glance*. [S.l.]: Springer, 2008. ISBN 978-0-387-77392-6.

[30] LAHTI, C.; PETERSON, R. *Sarbanes-Oxley: IT compliance using COBIT and open source tools*. [S.l.]: Syngress Pub. Inc., 2005. (Safari Books Online). ISBN 9781597490368.

[31] BEAL, A. *Seguranca da Informacao: principios e melhores praticas para a protecao dos ativos de informacao das organizacoes*. [S.l.]: Sao Paulo: Editora Atlas, 2005. ISBN 9788522440856.

[32] ANDRESS, J. *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*. [S.l.]: Elsevier Science, 2011. (Syngress Media). ISBN 9781597496537.

[33] MANDIA, K. *Incident Response: Investigating Computer Crime*. 1st. ed. [S.l.]: McGraw-Hill Professional, 2001. ISBN 0072131829.

[34] KENT, K. et al. Guide to Integrating Forensic Techniques into Incident Response: Recommendations of the National Institute of Standards and Technology. In: NATIONAL (Ed.). NIST Special Publication 800-86, 2006. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf>.

[35] STRAUB, D.; GOODMAN, S.; BASKERVILLE, R. *Information security: policy, processes, and practices*. [S.l.]: M.E. Sharpe, 2008. (Advances in management information systems). ISBN 9780765617187.

[36] WILES, J.; CARDWELL, K.; REYES, A. *The best damn cybercrime and digital forensics book period*. [S.l.]: Syngress, 2007. ISBN 9781597492287.

[37] WENQI, W.; WEIGUANG, L. The research on forensic model based network. In: *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2009. (IWCSE '09), p. 119–122. ISBN 978-0-7695-3881-5. Disponível em: <http://dx.doi.org/10.1109/WCSE.2009.635>.

[38] CASEY, E. *"Intrusion Investigation,"in Handbook of Digital Forensics and Investigation*. [S.l.]: Academic, 2009. ISBN 9780123742674.

[39] CHEN, L. et al. Modeling and analyzing dynamic forensics system based on intrusion tolerance. In: *Computer and Information Technology, 2009. CIT '09. Ninth IEEE International Conference on*. [S.l.: s.n.], 2009. v. 2, p. 230 –235.

[40] REN, W.; JIN, H. Modeling the network forensics behaviors. In: *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on*. [S.l.: s.n.], 2005. p. 1 – 8.

[41] SCARFONE, P. M. K. Guide to Intrusion Detection and Prevention Systems (IDPS): Recommendations of the National Institute of Standards and Technology. In: NATIONAL (Ed.). Special Publication 800-94, 2007. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf>.

[42] SIQUEIRA, L. G. *Tolerancia a Falhas para o NIDIA: um Sistema de Deteccao de Intrusao Baseado em Agentes Inteligentes*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2006.

[43] SABAHI, F.; MOVAGHAR, A. Intrusion detection: A survey. In: *Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on*. [S.l.: s.n.], 2008. p. 23 –26.

[44] COLE, E.; KRUTZ, R.; CONLEY, J. *Network security bible*. [S.l.]: Wiley Pub., 2005. (Bible Series). ISBN 9780764573972.

[45] 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>.

[46] PORRAS DAN SCHNACKENBERG, S. S.-C. P. The common intrusion detection framework architecture (cidf). 1998. Disponível em: http://gost.isi.edu/cidf/drafts/architecture.txt.

[47] DEBAR D. CURRY, B. F. H. *The Intrusion Detection Message Exchange Format (IDMEF)*. March 2007.

[48] DANYLIW J. MEIJER, Y. D. R. *Incident Object Description and Exchange Format (IODEF)*. December 2007.

[49] MITRE. *Common Event Expression (CEE)*. MITRE. Disponível em: <http://cee.mitre.org/docs/cls.html>. Acesso em: 03 fev. 2012.

[50] WOOD, M.; ERLINGER, M. *Intrusion Detection Message Exchange Requirements*. IETF, mar. 2007. RFC 4766 (Informational). (Request for Comments, 4766). Disponível em: <http://www.ietf.org/rfc/rfc4766.txt>. Acesso em: 11 set. 2011.

[51] JUNIOR, F. A. P. *Proposta de Atualização Automática dos Sistemas de Detecção de Intrusão por Meio de Web Services*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2006.

[52] CHUVAKIN, A. *LogChaos: Challenges and Opportunities of Security Log Standardization*. NIST. Disponível em: <http://scap.nist.gov/events/2009/itsac/presentations/day2/Day2_CNMAL_Log_Standard_Challenges_Chuvakin.pdf>. Acesso em: 07 out. 2011.

[53] OASIS Standard. *Common Alerting Protocol Version 1.2*. July 2010. Disponível em: <http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf>. Acesso em: 10 ago. 2011.

[54] SILVA, E. C. C. *Gerenciamento e Integração das Bases de Dados de Sistemas de Detecção de Intrusões*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2006.

[55] MITRE. *Common Vulnerabilities and Exposures (CVE)*. MITRE. Disponível em: <http://cee.mitre.org/docs/cls.html>. Acesso em: 03 fev. 2012.

[56] ROHSE, M. *Vulnerability naming schemes and description languages: CVE, Bugtraq, AVDL and VulnXML*. SANS Institute, 2003. Disponível em: <http://www.sans.org/reading_room/whitepapers/threats/vulnerability_naming_schemes_and_description_languages_cve_bugtraq_avdl_and_vulnxml_1058>. Acesso em: 03 set. 2011.

[57] SILVA, A. L. da. *Modelo de IDS para Usuários de Dispositívos Móveis*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2008.

[58] DIAS, R. A. *Um Modelo de Atualização Automatica do Mecanismo de Detecção de Ataques para Sistemas de Detecção de Intrusão*. Dissertação (Mestrado) — Universidade Federal do Maranhão, Janeiro 2003.

[59] SANTOS, G. de Lourdes Ferreira dos. *Respostas Automaticas para Melhoria da Segurança em Sistemas de Detecção de Intrusos*. Dissertação (Mestrado) — Universidade Federal do Maranhão, Janeiro 2003.

[60] SILVA, M. L. C. *Modelo de IDS Remoto baseado na tecnologia de Agentes, Web Services e MDA*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2006.

[61] ATAIDE, R. L. da R. *Uma Arquitetura para a Detecção de Intrusos no Ambiente Wireless Usando Redes Neurais Artificiais*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2007.

[62] NETO, R. P. da C. *Sistema de Detecção de Intrusos em Ataques Oriundos de Botnets Utilizando Metodo de Detecção Hibrido*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2011.

[63] SCHMIDT, D. Guest editor's introduction: Model-driven engineering. *Computer*, v. 39, n. 2, p. 25 – 31, feb. 2006. ISSN 0018-9162.

[64] BLAIR, G.; BENCOMO, N.; FRANCE, R. Models@ run.time. *Computer*, v. 42, n. 10, p. 22 –27, oct. 2009. ISSN 0018-9162.

[65] CUONG, X. Q. N. V. Managing hardware verification complexity with aspect-oriented model-driven engineering. *International Journal of Modeling and Optimization*, Vol. 1, No. 1, 2011.

[66] MELLOR, S. J. e. a. *MDA distilled: principles of Model-Driven Architecture*. 1rd. ed. [S.l.]: Boston: Addison-Wesley, 2004.

[67] Object Management Group. *MDA guide version 1.0.1n*. [S.l.], June 2003. Disponível em: <http://www.omg.org/cgi-bin/doc?omg/03-06-01>. Acesso em: 07 jul. 2010.

[68] LOPES, D. C. P. *Model Driven Architecture*. http://www.dee.ufma.br/ dlopes/-course/ESUML/MDA.pdf: [s.n.], 2006. Disponível em: <http://www.dee.ufma.br/~dlopes/course/ESUML/MDA.pdf>. Acesso em: 11 dez. 2011.

[69] MENS, T.; GORP, P. V. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, v. 152, n. 0, p. 125 – 142, 2006. ISSN 1571-0661. Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005) Graph and Model Transformation 2005. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1571066106001435>.

[70] JOUAULT, F. et al. Atl: A model transformation tool. *Science of Computer Programming*, v. 72, n. 1-2, p. 31–39, 2008. ISSN 0167-6423. Special Issue on Second issue of experimental software and toolkits (EST). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167642308000439>.

[71] GROUP, Q.-M. *ATL - a model transformation technology*. julho 2010. Disponível em: <Availableathttp://www.eclipse.org/atl/>. Acesso em: 10 jan. 2012.

[72] STEINBERG, D. et al. *EMF: Eclipse Modeling Framework*. 2rd. ed. [S.l.]: Addison-Wesley Professional, 2008.

[73] The Eclipse Foundation. *Xtext 2.1 Documentation*. [S.l.], June 2009. Available at http://www.eclipse.org/Xtext/documentation/.

[74] JAVACC. *Java Compiler Compiler (The Java Parser Generator)*. Disponível em: <http://javacc.java.net/>. Acesso em: 20 fev. 2012.

[75] PARR, T. *ANTLR*. Disponível em: <http://www.antlr.org/>. Acesso em: 12 dez. 2011.

[76] WIRTH, N. *Extended Backus-Naur Form (EBNF)*. 1996.

[77] Eclipse Modeling Framework. *Teneo*. Disponível em: <http://wiki.eclipse.org/Teneo>. Acesso em: 10 fev. 2011.

[78] MCCLURE, S.; SCAMBRAY, J.; KURTZ, G. *Hacking 6 exposed*. [S.l.]: McGraw-Hill, 2009. (Hacking Exposed). ISBN 9780071613743.

[79] REITH CLINT CARR, G. H. G. M. An examination of digital forensic models. *International Journal of Digital Evidence*, v. 1, p. 1 – 12, 2002.

[80] PROACTIVE. In: Oxford Dictionaries. [s.n.], 2012. Disponível em: <http://oxforddictionaries.com/definition/proactive?region=us&q=proactive>. Acesso em: 28 fev. 2012.

[81] MIEDE, A. et al. A generic metamodel for it security attack modeling for distributed systems. In: *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*. [S.l.: s.n.], 2010. p. 430 –437.

[82] LAMIS, T. A forensic approach to incident response. In: *2010 Information Security Curriculum Development Conference*. New York, NY, USA: ACM, 2010. (InfoSecCD '10), p. 177–185. ISBN 978-1-4503-0202-9. Disponível em: <http://doi.acm.org/10.1145/1940941.1940975>.

[83] MOSSIN, H. *Comentarios ao Codigo de processo penal: a luz da doutrina e da jurisprudencia*. [S.l.]: Manole, 2005. ISBN 9788520421918.

[84] BRASIL. *Constituição (1988). Constituição da República Federativa do Brasil*. [S.l.]: Senado, 1988.

[85] ATL Transformations. Disponível em: <http://www.eclipse.org/m2m/atl/atlTransformations/>. Acesso em: 15 setembro 2011.

[86] BUDINSKY, F. *Eclipse modeling framework: a developer's guide*. Addison-Wesley, 2004. (The eclipse series). ISBN 9780131425422. Disponível em: <http://books.google.com.br/books?id=ff-9ZYhvPwwC>.

[87] GRONBACK, R. C. *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. 1. ed. [S.l.]: Addison-Wesley Professional, 2009. ISBN 0321534077, 9780321534071.

[88] SNORT. Disponível em: <http://www.snort.org>. Acesso em: 15 setembro 2011.

[89] SURICATA. Disponível em: <http://www.openinfosecfoundation.org/index.php/download-suricata>. Acesso em: 15 setembro 2011.

[90] WIRESHARK. Disponível em: <http://www.wireshark.org>. Acesso em: 15 setembro 2011.

[91] TCPSLICE. Disponível em: <http://sourceforge.net/projects/tcpslice>. Acesso em: 15 janeiro 2011.

[92] BACKTRACK5. Disponível em: <www.backtrack-linux.org>. Acesso em: 20 ago. 2011.

[93] ORACLE e Java - Tecnologias. Disponível em: <http://www.oracle.com/br/technologies/java/index.html>. Acesso em: 23 janeiro 2012.

[94] JAVA Server Faces Technology. Disponível em: <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>. Acesso em: 23 janeiro 2012.

[95] MYSQL. Disponível em: <http://www.mysql.com/>. Acesso em: 20 ago. 2011.

[96] APACHE. *Apache Tomcat*. Disponível em: <http://tomcat.apache.org>. Acesso em: 23 janeiro 2012.

[97] COMMONS IO. Disponível em: <http://commons.apache.org/io/>. Acesso em: 15 janeiro 2012.

[98] COMMONS Lang. Disponível em: <http://commons.apache.org/lang/>. Acesso em: 15 janeiro 2012.

[99] JCRAFT. Disponível em: <http://www.jcraft.com/>. Acesso em: 15 janeiro 2012.

[100] ECLIPSE. Disponível em: <http://www.eclipse.org>. Acesso em: 02 jun. 2012.

[101] VMWARE. *VMware Workstation*. Disponível em: <http://www.vmware.com/>. Acesso em: 14 janeiro 2011.

[102] IPTABLES - The netfilter.org "iptables"project. Disponível em: <http://www.netfilter.org/projects/iptables/index.html>. Acesso em: 15 setembro 2011.

[103] METASPLOIT Framework. Disponível em: <metasploit.org>. Acesso em: 16 ago. 2011.

[104] TCPREPLAY. Disponível em: <http://tcpreplay.synfin.net/>. Acesso em: 20 ago. 2011.

[105] INTERNET Security Threat Report - 2011 Thrends. Symantec Corp., 2012. Disponível em: <http://www.symantec.com/content/en/us/enterprise/

other_resources/b-istr_main_report_2011_21239364.en-us.pdf>.
Acesso em: 20 maio 2012.

A APÊNDICE 1 - Exemplo de Gramática Criada no Xtext - Snort (Full e Fast)

```
grammar br.com.snort.Snort with
org.eclipse.xtext.common.Terminals
import "http://www.eclipse.org/emf/2002/Ecore"
as ecore
generate snort "http://www.com.br/snort/Snort"

// Snort Alert Grammar - Full and Fast accepted
Alert:
	(snortAlert+=SnortAlert)+ ;
// Snort alerts  on this grammar are  Full or Fast
shaped
SnortAlert:
	AlertFull | AlertFast ;
//Composição de um alerta Full
AlertFull:
	snortMessage=SnortMessage
classification=Classification priority = Priority
	timestamp=TimeStamp1
(withMac=WithMac)?
	source=Address sourceport=Port '->'
destination=Address destinationPort=Port
	transportProtocol=UpperCase
	(specificInformation+=SpecificInformatio
n)*
	(externalReference+=ExternalReference
)* ;
//Composição de um alerta Fast
AlertFast:
	{AlertFast}
	timestamp=TimeStamp1
(snortMessage=SnortMessage)
(classification=Classification)? priority=Priority
	'{' transportProtocol=UpperCase'}'
source=Address sourceport=Port '->'
destination=Address destinationPort=Port  ;
//Alert message
SnortMessage hidden(WS):
	'[**]'sidInfo=SidInfo
description=Content'[**]' ;
//Message content
Content hidden(WS):
	(ID|UpperCase) ('.'|ID|WS|'/'|'-
'|UpperCase)* ;
//SidInfo - eventnumber:sid of signarute:
number of revisions
terminal SidInfo:
	'['INT':'INT':'INT']' ;
//terminal rule for alert classification
terminal Classification :
	'["Classification: ' -> '] ;
//terminal rule for priority
terminal Priority:
	'["Priority: ' INT	']' ;
//terminal rule for snort timestamp
terminal TimeStamp1:
	//08/12-02:34:11.023564
	INT '/' INT '-' INT ':' INT ':' INT '.' INT ;
//IPV6 components
terminal IpV6:
	Hex Hex Hex Hex ;
//IpV4 ou IpV6
terminal Ip:
	(((INT '.'INT '.'INT'.'INT)))
|(IpV6':'IpV6':'IpV6':'IpV6':'IpV6':'IpV6':'IpV6':'IpV
6)  ;
//hexadecimais
terminal Hex : (('0'
('x'|'X')|(('0'..'9'))|('a'..'f')|('A'..'F'))) ;
//porta
terminal Port ://returns ecore::EInt:
	':'INT ;
//Mac or IP address
Address returns ecore::EString :
	((Ip)|(MacAddress)) ;
//mac composition definition
terminal Mac:
	Hex (Hex)? ;
//mac address
terminal MacAddress:
	Mac':'Mac':'Mac':'Mac':'Mac':'Mac ;
//uppercase for strings
terminal UpperCase:
	('A'..'Z')+(WS)?;
//For alert Full with MAC
//8:0:3E:0:1:AF -> 1:0:5E:12:80:8 type:0x800
len:0x3E before source and dest. address
//AA:0:4:0:A:4 -> 0:0:21:C9:A2:B9 type:0x800
len:0x36 10.10.10.21:54635 -> 70.51.69.51:121
WithMac:
	(MacAddress '->' MacAddress)
((OpHex)|(OpInt))* ;
//tips options hex
terminal OpHex:
	ID':'(WS)?Hex(Hex)+ ;
//tips options int
terminal OpInt:
	ID':'(WS)?INT ;
//options for TCP protocol
terminal TcpOptions:
	'TCP Options ('INT') =>' ;
//prot specific information
SpecificInformation:
	((TcpOptions)?)((ID)|(OpHex)|(OpInt)|('D
F')|INT|Susp|UpperCase) ;
//terminal rule for external references
terminal ExternalReference:
	('["Xref =>' ->']');
//for specific information
terminal Susp:
	(('*')|('A'..'Z'))+;
//int definition
terminal INT returns ecore::EInt: ('0'..'9')+;
```

A APÊNDICE 2 - Exemplo de Transformação em ATL - Alerta Snort para Alerta Genérico

```
module Snort2AlertGen;
create OUT : AlertGeneric from IN : Snort;

helper context Snort!AlertFull def: reference(): Sequence (String) =  self.externalReference.asSequence();
helper context Snort!AlertFull def: headerInfo(): Sequence (String) =  self.specificInformation.asSequence();
rule Alert2AlertG{
    from alert:Snort!Alert
    to agen:AlertGeneric!AlertGen
    (      event <- alert.snortAlert   )
}
abstract rule sAlert2SecEvent {
        from sAlert: Snort!SAlert
        to
                genSecEvent: AlertGeneric!Event (
                        eventFormat <- sAlert.oclType().name,
                        generatedDate <- sAlert.timestamp,
                        protocol <- sAlert.transportProtocol.trim(),
                        classification <- sAlert.classification.substring(18,sAlert.classification.size()-1),
                        priority <- sAlert.priority.split('[^0-9]+').last(),
                        msg <-sAlert.snortMessage.description,
                        target <- genTargetAsset,
                        source <- genSuspectHost,
                        otherInfo <- genOtherInfo
                ),
                genSuspectHost: AlertGeneric!SuspectHost(
                        address <- sAlert.source,
                        port <- sAlert.sourceport.split(':').last().toInteger()

                ),
                genTargetAsset: AlertGeneric!TargetAsset(
                        address <- sAlert.destination,
                        port <- sAlert.destinationPort.split(':').last().toInteger()
                ),

                genOtherInfo : AlertGeneric!OtherInfo(
                        name <- 'sidInfo',
                        value <-sAlert.snortMessage.sidInfo
                )
}
rule sAlert2Full extends sAlert2SecEvent{
        from sAlert: Snort!AlertFull
        to
                genSecEvent: AlertGeneric!Event (
                        reference <- sAlert.reference() -> collect (e |thisModule.references(e)),
                        header <- sAlert.headerInfo() -> collect (e| thisModule.headers(e))
                )
}
lazy rule headers{
        from headerinfo: Sequence(String)
        to agen: AlertGeneric!HeaderTips(
                name <- headerinfo.split(':').get(0),
                val <- headerinfo.split(':').last()
        )
}
rule sAlertFast extends sAlert2SecEvent{
        from sAlert: Snort!AlertFast
        to
                genSecEvent: AlertGeneric!Event ()
}
lazy rule references
{
        from reference : Sequence(String)
        to agen: AlertGeneric!VInformation(
                externalReference <- reference)
}
```