FÁBIO DACÊNCIO PEREIRA

PROPOSTA E IMPLEMENTAÇÃO DE UMA CAMADA DE INTEGRAÇÃO DE SERVIÇOS DE SEGURANÇA (CISS) EM SOC E MULTIPLATAFORMA

São Paulo
2009

FÁBIO DACÊNCIO PEREIRA

Proposta e Implementação de uma Camada de Integração de Serviços de Segurança (CISS) em SoC e Multiplataforma

Tese de Doutorado apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia Elétrica

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. Edward David Moreno Ordonez

São Paulo
2009

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 05 de dezembro de 2009.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Pereira, Fábio Dacêncio
Proposta e implementação de uma camada de integração de serviços de segurança (CISS) em SoC e multiplataforma / F.D. Pereira. -- ed.rev. -- São Paulo, 2009.
149 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1. Circuitos lógicos 2. Segurança de redes 3. Arquitetura e organização de computadores I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II. t.

Dedicatórias

Dedico esta tese aos meus pais, Sebastião e Neusa, que me apoiaram em decisões e momentos difíceis, mas que principalmente comemoram conquistas e felicidades da vida. Estar aqui significa que o alicerce foi bem construído e os pilares são o amor, o respeito e o apoio. Muito obrigado por tudo que vocês me ensinaram.

Dedico todo este trabalho ao meu filho Pedro Luís. Seu nome expressa o seu ser. Pedro, oriundo de Pedra, Rocha, e Luís oriundo de Luz. Filho, sua força e luz são o meu norte para buscar novas conquistas e ser feliz.

Ao meu irmão Aquiles. Tenho orgulho desse cara. Quantas vezes já ouvi: “você é irmão do Aquiles!”

Dedico a minha namorada, Karina, que sempre me esperou. Quando eu chego, vejo que sem você, as coisas seriam bem mais difíceis. Obrigado pelo amor e carinho.

Agradecimentos

Agradeço primeiramente ao meu orientador e amigo, Edward, que por 11 anos me incentivou, orientou e apoiou sem restrições, muitas vezes, privando-se de um tempo que deveria ser dedicado à família ou ao descanso. No entanto, a sensação de transformar a vida das pessoas é um exercício nobre, que compete a poucos como você, meu amigo Edward.

Aos amigos, a quem recorri muitas vezes quando precisei de uma palavra de confiança e sinceridade. Cesar e Muzzi, meus companheiros para todos os assuntos e viagens. Já passamos por tudo! Aos amigos Rodolfo, Larissa, Paulo Nardi, Maycon e Didi. Eles talvez não saibam, mas quantas e quantas vezes citei seus nomes como exemplo.

A todos os colegas e amigos de trabalho. Fernando Netto (UNVEM), apoio sincero, profissionalismo e respeito, pessoa fantástica, que me ensinou muito e me ajudou em momentos difíceis. Aos Colegas da Design House LSI-TEC. Sinto-me honrado por ter feito parte de uma equipe onde se destacam a capacidade e inteligência de cada um. Equipe de ponta mesmo!

Aos professores, desde a graduação até o doutorado. Fui um privilegiado por ter tido estes notórios mestres, que influenciaram diretamente na minha carreira profissional e acadêmica.

A Deus, principio e razão de ser e de tudo

Deliberar é uma tarefa de muitos. Agir é tarefa de um só.
De Gaulle

Resumo

PEREIRA, F. D., **Proposta e Implementação de uma Camada de Integração de Serviços de Segurança (CISS) em SoC e Multiplataforma**, 2009, 149 f, Tese (Doutorado) - Engenharia Elétrica, Universidade de São Paulo –USP, São Paulo, 2009.

As redes de computadores são ambientes cada vez mais complexos e dotados de novos serviços, usuários e infra-estruturas. A segurança e a privacidade de informações tornam-se fundamentais para a evolução destes ambientes.

O anonimato, a fragilidade e outros fatores muitas vezes estimulam indivíduos mal intencionados a criar ferramentas e técnicas de ataques a informações e a sistemas computacionais. Isto pode gerar desde pequenas inconveniências até prejuízos financeiros e morais. Nesse sentido, a detecção de intrusão aliada a outras ferramentas de segurança pode proteger e evitar ataques maliciosos e anomalias em sistemas computacionais. Porém, considerada a complexidade e robustez de tais sistemas, os serviços de segurança muitas vezes não são capazes de analisar e auditar todo o fluxo de informações, gerando pontos falhos de segurança que podem ser descobertos e explorados.

Neste contexto, esta tese de doutorado propõe, projeta, implementa e analisa o desempenho de uma camada de integração de serviços de segurança (CISS). Na CISS foram implementados e integrados serviços de segurança como Firewall, IDS, Antivírus, ferramentas de autenticação, ferramentas proprietárias e serviços de criptografia. Além disso, a CISS possui como característica principal a criação de uma estrutura comum para armazenar informações sobre incidentes ocorridos em um sistema computacional. Estas informações são consideradas como a fonte de conhecimento para que o sistema de detecção de anomalias, inserido na CISS, possa atuar com eficiência na prevenção e proteção de sistemas computacionais detectando e classificando prematuramente situações anômalas.

Para isso, foram criados modelos comportamentais com base nos conceitos de Modelo Oculto de Markov (HMM) e modelos de análise de seqüências anômalas. A CISS foi implementada em três versões: (i) System-on-Chip (SoC), (ii) software JCISS em Java e (iii) simulador. Resultados como desempenho temporal, taxas de ocupação, o impacto na detecção de anomalias e detalhes de implementação são apresentados, comparados e analisados nesta tese.

A CISS obteve resultados expressivos em relação às taxas de detecção de anomalias utilizando o modelo MHMM, onde se destacam: para ataques conhecidos obteve taxas acima de 96%; para ataques parciais por tempo, taxas acima de 80%; para ataques parciais por seqüência, taxas acima de 96% e para ataques desconhecidos, taxas acima de 54%.

As principais contribuições da CISS são a criação de uma estrutura de integração de serviços de segurança e a relação e análise de ocorrências anômalas para a diminuição de falsos positivos, detecção e classificação prematura de anormalidades e prevenção de sistemas computacionais. Contudo, soluções foram criadas para melhorar a detecção como o modelo seqüencial e recursos como o subMHMM, para o aprendizado em tempo real. Por fim, as implementações em SoC e Java permitiram a avaliação e utilização da CISS em ambientes reais.

Abstract

PEREIRA, F. D., **Proposal and Implementation of an Security Services Integration Layer (ISSL) in SoC and Multiplatform**, 2009, 149 f, Thesis (Doctoral) - Engenharia Elétrica, Universidade de São Paulo –USP, São Paulo, 2009.

Computer networks are increasingly complex environments and equipped with new services, users and infrastructure. The information safety and privacy become fundamental to the evolution of these environments. The anonymity, the weakness and other factors often encourage people to create malicious tools and techniques of attacks to information and computer systems. It can generate small inconveniences or even moral and financial damage. Thus, the detection of intrusion combined with other security tools can protect and prevent malicious attacks and anomalies in computer systems. Yet, considering the complexity and robustness of these systems, the security services are not always able to examine and audit the entire information flow, creating points of security failures that can be discovered and explored.

Therefore, this PhD thesis proposes, designs, implements and analyzes the performance of an Integrated Security Services Layer (ISSL). So several security services were implemented and integrated to the ISSL such as Firewall, IDS, Antivirus, authentication tools, proprietary tools and cryptography services. Furthermore, the main feature of our ISSL is the creation of a common structure for storing information about incidents in a computer system.

This information is considered to be the source of knowledge so that the system of anomaly detection, inserted in the ISSL, can act effectively in the prevention and protection of computer systems by detecting and classifying early anomalous situations. In this sense, behavioral models were created based on the concepts of the Hidden Markov Model (MHMM) and models for analysis of anomalous sequences. The ISSL was implemented in three versions: (i) System-on-Chip (SoC), (ii) JCISS software in Java and (iii) one simulator. Results such as the time performance, occupancy rates, the impact on the detection of anomalies and details of implementation are presented, compared and analyzed in this thesis.

The ISSL obtained significant results regarding the detection rates of anomalies using the model MHMM, which are: for known attacks, rates of over 96% were obtained; for partial attacks by a time, rates above 80%, for partial attacks by a sequence, rates were over 96% and for unknown attacks, rates were over 54%.

The main contributions of ISSL are the creation of a structure for the security services integration and the relationship and analysis of anomalous occurrences to reduce false positives, early detection and classification of abnormalities and prevention of computer systems. Furthermore, solutions were figured out in order to improve the detection as the sequential model, and features such as subMHMM for learning at real time. Finally, the SoC and Java implementations allowed the evaluation and use of the ISSL in real environments.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ADA	Anomaly Detection Algorithm
AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
BLP	Bell-LaPadula
CBS	Component-Based Servicing
CISS	Camada de Integração de Serviços de Segurança
CW	ClarkWilson
DDOS	Distributed Denial-of-Service
DES	Data Encryption Standard
DOS	Denial-of-Service
EAD	Ensino à Distância
EMAC	Ethernet Media Access Controller
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
GNU	General Public License
GPIO	General Purpose Input/Output
GSA	Grafo de Seqüências Anômalas
GST	Grafo de Seqüência Total
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
ICS	Integrated Component Security

IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IntC	Xilinx Interrupt Control
IP	Internet Protocol
ISO	International Organization for Standardization
KDDCup	Knowledge Discovery and Data Mining Cup
LUT	Look-Up-Table
MPMC	Multi Port Memory Controller
NIDS	Network Intrusion Detection System
NRU	No Read Up
NWD	No Write Down
OSI	Open Systems Interconnection
PLB	Processor Local Bus
RAM	Random-Access Memory
RFC	Request for Comments
RSA	R. Rivest - A. Shamir - L. Adleman
SoC	System-on-Chip
UART	Universal Asynchronous Receiver/Transmitter
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language

Sumário

CAPÍTULO 1

INTRODUÇÃO

A complexidade das redes de computadores e sistemas computacionais está aumentando consideravelmente, assim como o número de informações armazenadas ou em tráfego. A cada dia, novos serviços são criados, fazendo deste aglomerado uma ferramenta importante para diversas áreas como negócios, entretenimento, saúde, pesquisa, entre outras.

O anonimato, a fragilidade e outros fatores muitas vezes estimulam indivíduos mal intencionados a criar ferramentas e técnicas de ataques a informações e a sistemas computacionais. Isto pode gerar desde pequenas inconveniências até prejuízos financeiros e morais.

A detecção de intrusão aliada a outras ferramentas de segurança pode proteger e evitar ataques maliciosos e anomalias em sistemas computacionais. Porém, considerada a complexidade e robustez de tais sistemas, os serviços de segurança muitas vezes não são capazes de analisar e auditar todo fluxo de informações, gerando pontos falhos de segurança que podem ser descobertos e explorados (ANDROULIDAKIS, PAPAVASSILIOU, 2008).

É inevitável que indivíduos mal intencionados se organizem para criar ataques mais eficientes e inteligentes. Da mesma forma, devem ser criados sistemas integrados de segurança, principalmente conhecendo a importância da integração de serviços de segurança para a melhoria da previsão, detecção e atuação em situações anômalas (RASHEED, CHOW, 2007).

Geralmente, pesquisas sobre sistemas de segurança têm tipicamente o foco na criação de novos serviços ou melhoria do desempenho e confiabilidade de uma única técnica, algoritmo ou mecanismo.

As abordagens que almejam integrar uma ou mais ferramentas de segurança focalizam-se normalmente em uma determinada aplicação estratégica, uma vez que a abordagem sistemática para sistemas integrados de segurança requer a análise das relações entre os dados.

As publicações em relação à integração de sistemas de segurança podem ser anotadas desde a década de 70 (BELL, LAPADULA; 1974) (BIBA,1977) (BAKER,1989) (OKAMOTO, 1992). A preocupação em tornar um conjunto de serviços de segurança em um único sistema integrado é de suma importância, visto que a possível fragilidade de um serviço de segurança pode ser compensada por outros.

Comercialmente, empresas de equipamentos de rede e desenvolvimento de software têm como desafio conceber esta integração. A maior dificuldade é estabelecer uma estratégia comum para os inúmeros dispositivos, ferramentas e técnicas de segurança de informações.

Os modelos de integração de serviços de segurança atuais determinam qual a relação entre um conjunto específico de serviços de segurança para que esses possam integrar informações para prevenir ou tratar anomalias do sistema.

Contudo, os modelos atuais propõem soluções sob um conjunto específico de serviços, desprezando a existência de outros (ZILYS, et al, 2004) (JONSSON, 2006) (RASHEED, CHOW, 2007). O principal objetivo da integração é a possibilidade de melhorar a detecção de intrusão, atuação e prevenção de sistemas computacionais. Muitos modelos propostos não foram utilizados em situações reais, tornando-se complexo avaliá-los sob um sistema computacional completo (ver detalhes no Capítulo 2).

Apesar da dificuldade em definir uma estratégia para a criação de um modelo de integração de serviços de segurança, existem bons trabalhos que apresentam soluções

satisfatórias. Os trabalhos em destaque no capítulo 2 utilizam-se de técnicas de análise do comportamento de sistemas computacionais com o objetivo de distinguir situações anômalas daquelas consideradas normais e modelos de dados para formatar e organizar estas informações.

O trabalho descrito nesta tese apresenta uma Camada de Integração de Serviços de Segurança (CISS) que pode englobar desde serviços de segurança proprietários, *opensource,* até aplicativos que desejam prover segurança como necessidade ou diferencial.

A CISS possui uma estrutura comum capaz de agrupar os serviços de segurança pertencentes a um determinado sistema computacional. As informações armazenadas na CISS podem ser analisadas por modelos comportamentais (ver capítulo 3) e gerar grafos que representam o comportamento de diferentes anomalias.

Estes modelos comportamentais podem ser usados com eficiência para detectar prematuramente ataques, diminuir falsos positivos, classificar o poder do ataque, definir as técnicas utilizadas para defesa, entre outras vantagens que são descritas nesta tese.

1.1 Definições e Terminologias

Nesta seção são apresentadas algumas definições e terminologias que são utilizadas com freqüência no texto desta tese. Na seqüência, destacam-se as principais:

Atributos de segurança: o conceito de atributos de segurança abordado neste trabalho refere-se aos itens desejáveis que devem ser garantidos por aplicações, ferramentas, técnicas ou metodologia de segurança que têm o intuito de proteger, detectar e prevenir anomalias em sistemas computacionais. Os atributos de segurança podem ser classificados como (SHIREY, 2000):

- Confidencialidade: garantir que os conteúdos de informações sigilosas sejam acessados apenas por pessoas autorizadas;

- Autenticidade: garantir a validade e identificação de um remetente;
- Integridade: assegurar que a informação gerada pelo remetente é a mesma recebida pelo destinatário, ou seja, não sofreu nenhum tipo de alteração;
- Irretratabilidade: não repúdio por parte do destinatário quanto à autenticação e conteúdo de uma informação;
- Auditoria: verificar *logs e* detectar padrões com o intuito de perceber possíveis invasões ou uso incorreto do sistema;
- Disponibilidade: garantir que um serviço esteja disponível durante um período de tempo;
- Controle de Acesso: assegurar que apenas usuários autorizados terão acesso a informações sigilosas.

Uma técnica, metodologia, ferramenta ou aplicação que prima pela segurança de um sistema computacional pode abranger um ou mais atributos de segurança, conforme classificados anteriormente.

Ferramentas, aplicações ou serviços de segurança: Firewall, IDS, Antivírus, aplicações proprietárias e *opensource* são exemplos de ferramentas, aplicações ou serviços que promovem a segurança da informação.

Metodologias de segurança: Normas como ISO/IEC 27000:2009 são utilizadas nesta tese como metodologias que provêem a segurança.

Anomalias ou incidentes: para a CISS, são as ocorrências detectadas e geradas por aplicações de segurança. Portanto, a utilização e configuração inadequada destas aplicações podem gerar falsas ocorrências, ou simplesmente não ser capaz de tratar e/ou identificar uma anomalia, tendo impacto direto na metodologia aplicada e nos resultados contemplados.

1.2 Objetivos da Tese

O objetivo principal da tese é propor, projetar e implementar em software e hardware uma Camada de Integração de Serviços de Segurança – CISS, que além de integrar vários serviços de segurança, permita analisar a eficiência e o impacto dos modelos comportamentais da CISS na detecção de anomalias. O trabalho também propõe melhorias para detectar anomalias utilizando as técnicas de Wei Fan (2001) e subMHMM, sendo a última proposta na tese. É objetivo também ter um protótipo da CISS em hardware, especificamente em SoC - System On Chip, e em sistemas multiplataformas.

A CISS foi concebida com o intuito de criar um meio comum de integração de informações sobre incidentes de segurança, para que algoritmos e metodologias de proteção de sistemas computacionais tenham subsídios para detectar, atuar e prevenir situações anômalas prematuramente e com eficiência.

A integração de serviços de segurança pode proporcionar vários benefícios que estão mencionados nesta tese e citados na seqüência:

- **Integrar os serviços de segurança disponíveis**: informações concentradas de anomalias detectadas em um sistema computacional é uma fonte importante de dados que permite a criação de ferramentas de detecção, atuação e prevenção;
- **Permitir a identificação prematura de um ataque**: utilizando modelos comportamentais, permitir a identificação de situações anômalas antes mesmo que essas obtenham sucesso;
- **Diminuir a incidência de falsos positivos**: criando modelos comportamentais consistentes e confrontá-los para aumentar a precisão na detecção de situações anômalas;
- **Permitir a criação de regras de atuação mais precisas contra o ataque identificado**: se bem classificada e identificada, a anomalia pode ser inibida, prevenindo ações futuras;
- **Permitir uma visualização do comportamento de anomalias**: por meio de modelos

comportamentais classificar, identificar e visualizar as estratégias de ataques que caracterizam uma anomalia, possibilitando a criação de modelos e regras de atuação mais precisos e inteligentes;

- **Detectar anomalias não conhecidas**: este é um objetivo explicitamente importante, uma vez que pesquisas e trabalhos que investigam possíveis soluções para detectar atividades maliciosas não conhecidas ainda não encontraram soluções fortemente satisfatórias. Esta é uma característica forte da próxima geração de IDS (YANG, et al, 2007).
- **Implementar versões em hardware (CISS-SoC) e software (JCISS)**: e apresentar resultados de testes sob sistemas computacionais reais e analisar os resultados obtidos.
- **Implementar um Simulador para CISS**: permitindo a criação de cenários simulados para avaliar a eficiência das regras de detecção e atuação da CISS.

1.3 A Camada de Integração de Serviços de Segurança (CISS)

Nesta seção, tem-se uma caracterização sucinta da Camada de Integração de Serviços de Segurança (CISS). A CISS tem como característica principal a criação de uma estrutura comum para armazenar informações sobre incidentes ocorridos em um sistema computacional. Estas informações são consideradas como a fonte de conhecimento para um sistema de detecção de anomalias atuar com eficiência na prevenção e proteção de sistemas computacionais.

A CISS deve atender inicialmente a um conjunto de dispositivos, ferramentas e algoritmos que existem e possibilitar a inserção de outros que venham a existir. Isto é possível somente se esta camada possuir uma estrutura dinâmica que permita de forma flexível customizar suas características, mantendo a integridade das informações.

As informações são dados gerados por serviços de segurança presentes em um sistema computacional, como firewall, IDS, antivírus, antispyware, antipishing, mecanismos de

integridade, entre outros. Estas informações são analisadas pela CISS, que tem a capacidade de ponderar e atuar na proteção do sistema.

Análises das informações de segurança podem ser executadas para diversos objetivos. Nesta tese, alguns modelos comportamentais foram implementados, adaptados e integrados à CISS com o intuito de gerar grafos representativos de comportamento que podem ser comparados e analisados. A CISS enfatiza o uso de dois modelos comportamentais: o Modelo Oculto de Markov e o Modelo Seqüencial (ver capítulo 3).

O Modelo Oculto de Markov (*Hidden Markov Model* - HMM) foi adaptado para determinar o comportamento de situações anômalas. Este, aplicado à CISS, permitiu a implementação de um sistema eficiente de proteção que utiliza a integração dos serviços de segurança como fator predominante. O Modelo Seqüencial, de forma semelhante ao HMM, produz grafos de seqüências anômalas, porém não adota o fator tempo como métrica de detecção, diferente do HMM.

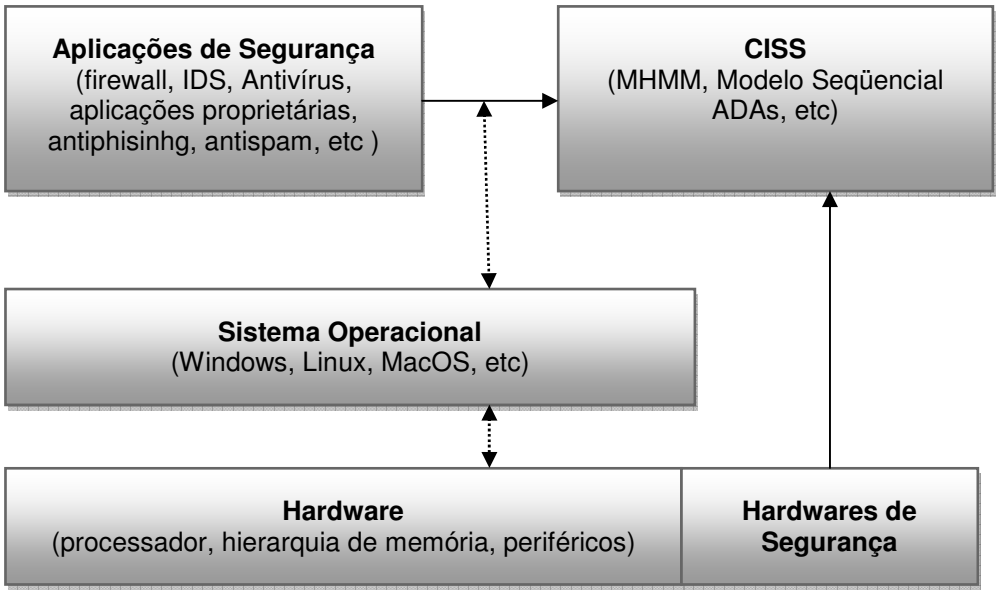

Figura 1.1. Escopo da CISS

A figura 1.1 representa o escopo de atuação da CISS. Nota-se que esta armazena informações de diversos serviços de segurança e possui um sistema de detecção de anomalias

(ADAs) para analisar o comportamento dos incidentes de segurança. A CISS armazena e analisa informações de ferramentas de segurança em hardware e software. Estas informações devem obedecer a um formato específico descrito pelo padrão IDMEF (DEBAR, et al, 2007), (ver detalhes no capítulo 3 desta tese).

1.4 Metodologia

O trabalho foi dividido em etapas que percorrem desde definições, conceitos básicos, revisão bibliográfica de trabalhos correlatos até a identificação da problemática, implementações e contribuições da tese. A seguir, a estrutura da metodologia adotada que levaram às contribuições deste trabalho.

1. Busca do tema: segurança de informações é uma área que se torna importante atualmente não apenas nas esferas acadêmicas e profissionais, mas também no dia a dia de pessoas comuns que utilizam sistemas computacionais em atividades de negócios, entretenimento, comunicação entre outras aplicações. Neste contexto, muitas soluções precisam ser criadas ou evoluídas, o que permite explorar e criar novos conceitos, aplicações e metodologias. O tema selecionado envolve a integração e a melhor utilização de soluções de segurança que já existem ou que ainda serão criadas.
2. Estado da Arte: foi uma revisão planejada para responder a um conjunto de perguntas específicas e que utiliza métodos explícitos e sistemáticos para identificar, selecionar e avaliar criticamente os estudos, e para coletar e analisar os dados destes estudos incluídos na revisão. A revisão bibliográfica feita neste trabalho (capítulo 2) ajudou a responder às questões em destaque na seqüência:
    - Existem metodologias para integração de serviços de segurança?
    - Existem ferramentas e aplicações em hardware ou software para este fim?
    - Existem aplicações em FPGA para esse fim?

- As metodologias e/ou ferramentas de integração são eficientes?
- Quais os principais desafios nesta área?
- Após a integração, como é feito o aproveitamento destas informações?
- Existem metodologias para detectar o comportamento de anomalias em sistemas computacionais?
- Os modelos comportamentais representam com precisão as situações de normalidade ou anormalidade?
- Quais as dificuldades na criação de modelos comportamentais?

3. Definição da Problemática: com base no estado da arte apresentado no capítulo 2, pode-se observar que muitas metodologias de integração de serviços de segurança abrangem um número limitado de aplicações e têm um fim específico como a detecção de anomalias, colaboração entre aplicações de segurança ou simplesmente apresentam um modelo para melhorar um serviço de segurança específico como, por exemplo, o serviço de controle de acesso.

   Neste contexto, a problemática que foi explorada nesta tese envolve o projeto e criação de uma estrutura comum para integrar aplicações de segurança que podem estar presente em um sistema computacional. Sendo que as informações acumuladas permitiram gerar e usar modelos comportamentais de anomalias, os quais são eficientes quando utilizados para a detecção, atuação e prevenção contra situações anômalas.
4. Solução Proposta: projetar, implementar e analisar uma Camada de Integração de Serviços de Segurança, chamada de CISS, com os objetivos e características citados anteriormente.
5. Implementação e Casos de Uso: a CISS proposta foi implementada em software (JCISS) e hardware (CISS-SoC). Foram criadas aplicações específicas para cada uma das versões. Testes e análises foram realizados (ver capítulo 4 e 5). Um Simulador para CISS foi

criado, possibilitando analisar a eficiência das regras de detecção e atuação definidas (ver capítulo 6).

6. Contribuições: as contribuições foram confrontadas com trabalhos correlatos, onde os itens que diferenciam este trabalho estão em destaque no capítulo de análise dos resultados alcançados (ver capítulo 7) e conclusão e trabalhos futuros (ver capítulo 8).

1.5 Organização da Tese

A tese está organizada em 8 capítulos que separam conceitos, descrição e implementação em hardware e software, assim como análise dos resultados. A seguir, a descrição sucinta de cada capítulo.

No capítulo 1, apresentam-se informações básicas sobre os pilares que fundamentam a tese como conceitos, definições e contexto. Em meio a estas informações são apontados alguns fatores importantes que são explorados na tese conforme os objetivos e metodologia descritos.

No capítulo 2, apresentam-se os trabalhos correlatos que contribuem com esta tese e fundamentam aspectos estruturais e metodológicos adotados. Os trabalhos relacionados são criticados e analisados, justificando, ao final, a importância desta tese.

O Capítulo 3 contém a descrição da Camada de Integração de Serviços de Segurança (CISS) e a fundamentação matemática e metodológica dos modelos comportamentais que representam as anomalias notificadas.

O capítulo 4 faz a caracterização da implementação em hardware. São apresentadas as ferramentas e aplicações de segurança implementadas e a integração com a CISS. Os resultados da versão System-on-Chip (SoC) da CISS são analisados e discutidos neste capítulo. O capítulo 5 apresenta a versão completa da CISS implementada em software. Esta versão possui integração com ferramentas e aplicativos de segurança reais. O capítulo 6

destaca o simulador da CISS, o qual permite a criação de cenários, regras de detecção e atuação, assim como um módulo gerador de anomalias que permite a criação de modelos comportamentais e analisar seu comportamento antes mesmo de implantá-lo em um ambiente real.

No capitulo 7 é apresentada uma análise dos resultados alcançados, comparando-os e validando a eficiência da camada de integração de serviços de segurança. A caracterização das versões da CISS, os dados referentes ao impacto na detecção de situações anômalas e as comparações com os trabalhos correlatos são os itens discutidos neste capitulo.

Finalmente, o capítulo 8 sumariza as principais contribuições e destaca os pontos fortes da tese e ressalta sua aplicabilidade em um sistema computacional. Como trabalhos futuros destacam-se melhorias para a CISS, assim como, pontos que devem ser explorados nas próximas etapas deste projeto.

CAPÍTULO 2

MODELOS DE INTEGRAÇÃO E COMPORTAMENTO DE SERVIÇOS DE SEGURANÇA

Neste capítulo, são apresentados alguns dos trabalhos que se destacaram na revisão bibliográfica como fortes correlatos a esta tese. Também estão descritos trabalhos pioneiros que inspiraram inúmeras pesquisas na área de integração e comportamento de serviços de segurança.

2.1 Integração de Serviços de Segurança

Modelos como Bell-LaPadula (BELL, LAPADULA, 1974), Biba (BIBA, 1977), Clark-Wilson (CLARK, WILSON, 1987) e outros são referenciados como modelos clássicos de políticas de segurança e controle de acesso, porém estes têm em sua concepção o conceito de integração colaborativa de atributos de segurança como: controle de acesso, integridade, confidencialidade e autenticidade. Desta forma, são tidos como pioneiros na integração de atributos de segurança em sistema computacionais. Em seguida, modelos atuais e mais sofisticados como o proposto por Rasheed e Chow (2007) também são descritos e analisados.

2.1.1 Modelo Bell-LaPadula

Os cientistas David Bell e Leonard LaPadula da MITRE Corporation, desenvolveram um modelo baseado nos procedimentos usuais de manipulação de informação em áreas ligadas à segurança nacional americana. Este modelo ficou conhecido como modelo Bell-LaPadula, ou modelo BLP. Existem diversas outras descrições do modelo Bell-LaPadula disponíveis na literatura, como (AMOROSO, 1994) (LANDWEHR, 1981) e (SANDHU, 1993), algumas delas apresentando pequenas variações em relação ao modelo original.

Na apresentação original do modelo (BELL, LAPADULA, 1974), um sistema é descrito por meio de uma máquina de estados finitos. As transições de estados no sistema obedecem a determinadas regras. Bell e LaPadula demonstraram indutivamente que a segurança do sistema é mantida se este parte de um estado seguro e as únicas transições de estado permitidas são as que conduzem o modelo a um outro estado seguro.

No modelo BLP, um sistema é descrito em termos de sujeitos que acessam objetos, onde cada sujeito possui uma habilitação e cada objeto possui uma classificação. A cada sujeito está associado também um rótulo corrente de segurança, que representa a classificação mais alta dentre as informações já consultadas pelo sujeito no sistema até um determinado instante, sendo, portanto, uma classificação flutuante (dinâmica). A habilitação de um sujeito deve sempre dominar o seu rótulo corrente de segurança.

A propriedade de segurança simples, também conhecida como *propriedade-ss* ou regra *no read up* (NRU)[1], diz que um sujeito só pode observar informações para as quais esteja habilitado; em outras palavras, a leitura de um sujeito *si* sobre um objeto *oj* é autorizada se, e somente se, *rótulo(si)* deve dominar *rótulo(oj)*. Por exemplo, uma informação classificada como SECRETO só pode ser lida por sujeitos com habilitação SECRETO ou ULTRA-SECRETO.

[1] NRU o sujeito não tem permissão para leitura de objetos com rótulo de segurança superior.

A *propriedade-ss* não é suficiente para garantir a segurança desejada do sistema: esta propriedade não evita que um sujeito malicioso coloque informações privilegiadas em um recipiente com classificação inferior à das informações, o que constitui claramente um fluxo não autorizado de informação. Assim, torna-se necessário adicionar outra propriedade a ser satisfeita pelo sistema.

A *propriedade-** (propriedade estrela), também chamada de regra *no write down* (NWD)[2], é satisfeita se, quando um sujeito tem simultaneamente um acesso de leitura sobre um objeto o_1 e um acesso de escrita sobre um objeto o_2. Sendo assim, o *rótulo*(o_1) deve dominar o *rótulo*(o_2), isto é, o acesso do sujeito *si* sobre um objeto *oj* é autorizado se (BELL, LAPADULA, 1976):

- *rótulo*(o_j) domina o *rótulo-corrente(si)* quando o acesso for de escrita;
- *rótulo(oj)* é dominado pelo *rótulo-corrente(si)* quando o acesso for de leitura.

O Modelo Bell-LaPadula possui limitações. A adoção do modelo BLP pode acarretar problemas se o sistema tiver que lidar também com ameaças de integridade. Quando um sujeito escreve em um objeto com uma classificação superior à sua habilitação (o que satisfaz a *propriedade-**), ele não pode observar os efeitos desta operação de escrita (o que violaria a *propriedade-ss*); por esse motivo, tal operação é chamada de escrita cega (AMOROSO, 1994) (SANDHU, 1993).

O cenário de escritas cegas torna-se uma preocupação na medida em que o mesmo sujeito considerado inadequado para ver o conteúdo de um objeto possui permissão para fazer modificações arbitrárias neste mesmo objeto. Isto pode causar problemas de integridade que só podem ser resolvidos através de alterações nas regras do modelo BLP. Por exemplo, escritas em objetos com níveis mais altos de segurança podem ser proibidas; um sujeito só poderia escrever em um objeto que tivesse o mesmo nível de segurança.

[2] NWD inibe o sujeito para escrita em objetos com rótulo de segurança inferior.

Entretanto, tal modificação restringe, de certa forma, o modelo BLP e muda o seu enfoque, que deixa de ser exclusivamente a ameaça de revelação não-autorizada e passa a ser uma combinação de revelação e integridade. Por outro lado, a adoção da *propriedade-** revisada é bastante comum em implementações de sistemas computacionais que seguem o modelo BLP.

Um dos principais problemas do modelo Bell-LaPadula reside no aspecto extremamente restritivo da *propriedade-**. Por exemplo, se um sujeito com rótulo corrente de segurança SECRETO deseja copiar um arquivo CONFIDENCIAL, a *propriedade-** impõe que a cópia tenha classificação SECRETO, mesmo que as informações ali contidas possuam classificação CONFIDENCIAL. Ao longo do tempo, isso faz com que as informações subam no reticulado de rótulos de segurança, recebendo classificações sucessivamente maiores. Este fenômeno é conhecido como superclassificação da informação (LANDWEHR, 1981). A superclassificação da informação provoca a necessidade de reclassificações periódicas dos objetos (através de sujeitos de confiança), apenas para garantir a usabilidade de sistemas baseados no modelo BLP.

O modelo Bell-LaPadula trata exclusivamente com a confidencialidade das informações no sistema. No entanto, está destacado nesta tese, pois a partir deste modelo inovações foram propostas incluindo outros atributos de segurança como autenticidade, integridade e outros.

2.1.2 Modelo Biba

O modelo Bell-LaPadula tem por objetivo conter ameaças de revelação não-autorizada; não obstante, os próprios criadores do modelo BLP discutiram como ele poderia ser adaptado para conter ameaças de integridade (BELL, LAPADULA, 1976). Embora as

idéias de Bell e LaPadula careçam de maior consistência, estas serviram de base para que Ken Biba desenvolvesse um modelo de segurança com o propósito de garantir a integridade da informação; conhecido como modelo de integridade Biba (1977).

O modelo Biba é definido como o dual do BLP. Suas regras são similares ao do modelo anterior, mas tem como objetivo a preservação da integridade das informações classificadas, evitando alterações não autorizadas. O modelo define níveis hierárquicos de integridade para os sujeitos (*si*'s) e para os objetos (*oj*'s) similares aos níveis de sensitividade definidos no BLP. A propriedade simples de integridade define que um sujeito só pode ler um objeto se o seu nível de integridade for dominado pelo do objeto.

Por ser o dual do modelo BLP, este modelo apresenta limitações similares às descritas no modelo anterior. No modelo Biba, ocorre uma degradação do nível de integridade, de maneira análoga a superclassificação da informação do modelo de BLP. Existe também a necessidade de sujeitos de confiança no modelo Biba, utilizados para alterar a integridade de sujeitos e objetos, mantendo o sistema viável.

Existem alguns outros modelos mandatórios além do Bell-Lapadula e do Biba citados em literatura. O modelo ClarkWilson (CW) é um exemplo; baseia-se na idéia de que a integridade é mais importante que a confidencialidade em operações comerciais (CLARK, WILSON, 1987). Porém, diferente dos modelos Bell-LaPadula e Biba, o CW assume transações bem formadas (todos os passos de uma seqüência de atividades são executados corretamente) e a separação de tarefas (cada sujeito desempenha um papel distinto na seqüência de atividades que formam uma transação) como essência de sua definição.

Apesar dos modelos de Bell-LaPadula e Biba serem pioneiros, muitas propostas atuais de integração de segurança aplicam seus conceitos fundamentais, como por exemplo, o modelo proposto por Nissanke (2007) que trata da integração de segurança para sistema baseados em componentes.

Segundo Nissanke (2007), maliciosamente códigos de componentes de terceiros são modificados acarretando em erros de codificação e falhas na concepção funcional que poderiam ser subvertidos por atacantes maliciosos, expondo potencialmente sistemas baseados em componentes para sérias ameaças à segurança.

Neste contexto, o artigo de Nissanke (2007) apresenta componentes integrados de segurança com base nos modelos de segurança Bell-LaPadula e Biba, para evitar violações de segurança voltadas à confidencialidade e integridade.

2.1.3 Modelo Rasheed

O modelo de informação para integração de serviços de segurança proposto por Rasshed e Chow (2007) explora três atributos de segurança: o controle de acesso, a detecção de intrusão e a resposta a intrusão. No modelo para cada ferramenta de segurança abordada, foram caracterizadas suas tarefas e responsabilidades. Os autores concluíram que muitas vezes ferramentas de segurança podem ser distintas ou até mutuamente exclusivas, porém é possível criar integração e colaboração em diferentes níveis de inserção.

Dois modelos de classificação e organização de dados foram evidenciados por Rasheed e Chow (2007): IDMEF (*Intrusion Detection Message Exchange Format*) (DEBAR; et al, 2007) e XACML (*eXtensible Access Control Markup Language*) (MOSES, 2005).

2.1.3.1 XACML

A XACML (*eXtensible Access Control Markup Language*) é um padrão OASIS, e foi publicado na versão 2.0 em fevereiro 2005. XACML especifica uma linguagem para definir

políticas de controle de acesso, bem como solicitações e repostas a acessos, em XML. A linguagem suporta políticas que são compostas pelos seguintes elementos (MOSES, 2005):

- **target**: o conjunto de fontes, objetos, sujeitos, eventos e ambientes das quais as políticas serão aplicadas;
- **rule-combining algorithm**: uma especificação de como diferentes regras devem ser compostas;
- **set of rules**: cada regra é uma combinação de target, efeito e uma condição;
- **obligations**: conjunto de operações que devem ser realizadas como resposta a uma possível intrusão.

A motivação por trás desta linguagem é proporcionar uma linguagem de política comum em relação às empresas/ferramentas para diminuir as dificuldades que vêm com a atual heterogeneidade nas representações de políticas de segurança de diferentes domínios.

Devido à sua gama descritiva, o modelo de relação de dados XACML inspirou o modelo utilizado nesta tese para auxiliar, principalmente, na organização e respostas a incidentes em um sistema computacional.

2.1.3.2 IDMEF

O *Internet Engineering Task Force's Intrusion Detection Work Group* compôs uma RFC experimental de uma formatação de mensagens de detecção de intrusão (*Intrusion Detection Message Exchange Format* -IDMEF). A parte do modelo de dados IDMEF mais relevante para esta tese de doutorado são as classes de alerta.

Mensagens contendo a especificação de uma classe de alerta são enviadas para um sistema de tomada de decisões, gerando um evento que reúne os critérios de resposta a uma

intrusão específica. Estas são enviadas de forma assíncrona e uma única mensagem pode corresponder a mais de um evento anômalo. Uma mensagem de alerta contém as seguintes informações agregadas (DEBAR, 2007):

1. **Analyzer**: Identificação de informações sobre a ferramenta/analisador que originou a ocorrência;
2. **Create Time**: quando o alerta foi criado;
3. **Classification**: O identificador do alerta;
4. **Detect Time**: o tempo que o evento foi detectado;
5. **Analyzer Time**: o relógio atual do Analisador, utilizado como base de tempo;
6. **Source**: a fonte do evento descrito no alerta;
7. **Target**: o alvo do evento descrito no alerta;
8. **Assessment:** a informação sobre o impacto do evento, as ações tomadas pela ferramenta em resposta a este, e a confiança da avaliação;
9. **Additional Data**: informações adicionais que não se encaixam no modelo de dados.

O modelo teórico de integração de serviços de segurança proposto por Rasheed e Chow (2007) pode ser definido utilizando uma estrutura de classes descrita na tabela 2.1.

Este modelo baseia-se em informações sobre as principais captações de eventos de segurança. Os três principais eventos são subclasses de solicitação de acesso, tentativa de intrusão e resposta à intrusão. Cada uma é representada separadamente no modelo, porém pode relacionar os serviços para proteger um sistema computacional.

Por fim, é importante salientar que o IDMEF foi utilizado para representar as diferentes classes de alertas de detecção de intrusão na CISS. Pode-se afirmar que as informações do modelo proposto incluem todos os grandes elementos relacionados com o controle de acesso, detecção, intrusão e resposta à intrusão.

Tabela 2.1 – Classes do modelo de Rasheed e Chow (2007)

Classe	**Evento de Segurança**
Atributos	1. Fonte: o iniciador do evento. Possíveis fontes são: dispositivo, usuário, processo ou serviço; 2. Alvo: o destino do objeto do evento. Possíveis alvos são: dispositivo, usuário, Processo, Serviço ou Arquivo.
Subclasses	1. Solicitação de acesso: uma tentativa por parte de uma fonte para a realização e exploração sobre um alvo. 2. Resposta à intrusão: a resposta a uma tentativa de intrusão.

Classe	**Solicitação de acesso**
Atributos	1. Avaliação Resultado: o resultado da avaliação do conjunto de requisitos aplicáveis a este pedido. Resultados possíveis são: Permitir ou Negar 2. Requisitos: conjunto de condições que devem ser satisfeitas para o pedido para ser concedido. Condições podem ser especificadas para o Ambiente, Assunto, ou Fonte. 3. Conseqüências: especificação das ações que devem ser tomadas para conclusão do pedido.
Subclasses	1. Tentativa de intrusão: um pedido de acesso que é considerada intrusiva.

Classe	**Tentativa de Intrusão**
Atributos	1. Significa: o que foi utilizado para perpetrar o ataque. Poderia ser em uma das três subcategorias: Ignorando Controle, Desvio Passivo ou Desvio Ativo. Neste caso o desvio refere-se ao mal uso ou uso malicioso. 2. Avaliação: informações dos sistemas de julgamento do evento sobre: seu impacto, a confiança da análise e a classificação da tentativa. 3. Resultados: o efeito líquido da tentativa de intrusão podendo ter três categorias: negação de serviço, exposição ou saída de erros.
Subclasses	1. Resposta à intrusão

Classe	**Resposta à intrusão**
Atributos	1. Limitações: qualquer fator do sistema que limita a resposta que pode ser tomada contra a tentativa de intrusão 2. Tentativa de intrusão: uma intrusão ou mais tentativas que serão respondidas
Subclasses	1. Resposta Durante o Ataque: pode ser ativa ou passiva 2. Resposta Após o Ataque: com o objetivo de avaliação, recuperação e prevenção.

2.2 Modelos Comportamentais

Os modelos comportamentais são utilizados para classificar ações normais e anormais em um sistema computacional. Estes podem ser utilizados para definir o comportamento do tráfego em redes de computadores para fins de detecção de anomalias (YASAMI, et al, 2007).

Entre as técnicas mais referenciadas, destaca-se o *Hidden Markov Model* (HMM) (JOSHI, 2005), *Support Vector Machine* (SVM) (LI, et al, 2003), *Fuzzy Data Mining* (FDM) (BRIDGES, 2000), entre outras.

2.2.1 Modelo Yasami

Yasami, et al (2007) apresentou um algoritmo para detecção de anomalias baseado na análise de requisições ARP, usando HMM. Este consegue analisar o tráfego ARP de uma rede de computadores, criando grafos de representação do comportamento normal de um sistema computacional. Para isso, exige-se um período de treinamento. Segundo o autor, quanto maior o período de treinamento, mais específico e representativo será o grafo do modelo.

Neste caso, não houve a integração de nenhum serviço de segurança, e o sistema de detecção trabalha em um nível baixo do modelo OSI, entre a camada física e de enlace. Contudo, o trabalho apresenta com clareza uma aplicação real utilizando o Modelo Oculto de Markov.

O HMM está sendo utilizado em pesquisas e trabalhos atuais (JOSHI, 2005) (YANG, 2007) para modelar o comportamento de sistemas computacionais, classificado seu modo de operação como normal ou anormal.

Nesta tese, o HMM foi utilizado para auxiliar na detecção de intrusão, ajudando na classificação de níveis de ataques, entre outras características. Na seqüência, tem-se uma descrição sucinta do trabalho de Yasami, et al (2007).

O período de treinamento é uma fase determinante para criação do modelo comportamental de Yasami (2007), pois é gerada uma representação de eventos normais de um determinado sistema computacional**.** O modelo normal construído durante o período de treinamento inclui estados e transições que são definidas como segue (YASAMI, et al, 2007):

- Estado: Estados neste modelo comportamental têm a função de identificar o nó correspondente ao IP destino solicitado.
- Transição: Qualquer solicitação ARP produz uma transição de um Estado, sendo que a transição pode ocorrer para um mesmo estado.

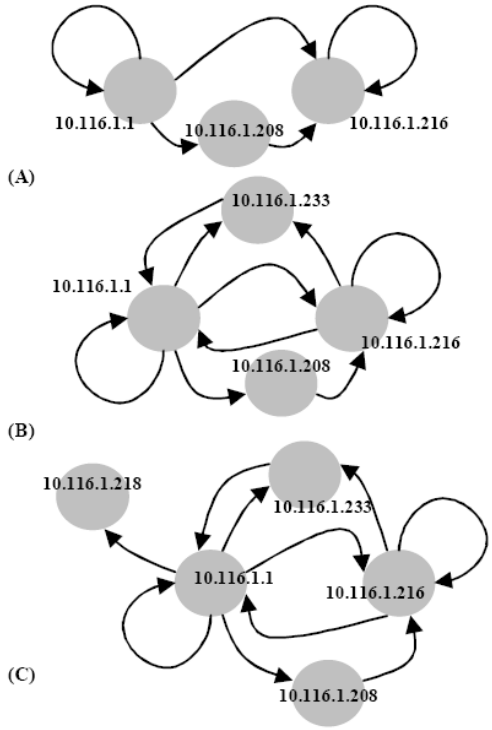

Figura 2.1. Criação do modelo comportamental HMM (YASAMI; et al, 2007)

Um exemplo de criação de estados e de transições entre estas é ilustrado na figura 2.1. O modelo ilustrado na figura 2.1A é construído no primeiro dia de treinamento do período. O mostrado na figura 2.1B é construído no segundo dia, onde novos estados e transições são somados ao modelo anterior. No 3º dia do período de treinamento, apenas um novo estado e transição são anexados ao modelo do dia anterior. Por meio desta figura, é possível compreender a formação do grafo comportamental que constitui o HMM. Segundo o autor, estes grafos tornam-se extremamente complexos e robustos conforme o ambiente de treinamento e regras de transição.

No período de treinamento, são calculados parâmetros de cada estado como: probabilidade de estar no estado S; Média do Tempo de Permanência no Estado S; Variância do Tempo de Permanência no Estado S. Estes parâmetros, assim como o grafo de transições, compõem o modelo comportamental do sistema computacional analisado.

Após o período de treinamento, este modelo comportamental é comparado com o modelo gerado em tempo real com o intuito de averiguar e distinguir ações diferentes das acumuladas durante o período de treinamento. Com base em um fator que indica a porcentagem de desvio de comportamento, o sistema indica a anormalidade ou não.

Para capturar o tráfego e recolher pacotes ARP, Yasami utilizou o Snort versão 2.3 em modo *sniffer* para capturar o tráfego total ARP. Outro importante fator que afeta os resultados do sistema é a duração do período de treinamento. A figura 2.2 ilustra o crescimento do número de estados, indicando o comportamento da maioria de nós dentro de 20 dias de período de treinamento.

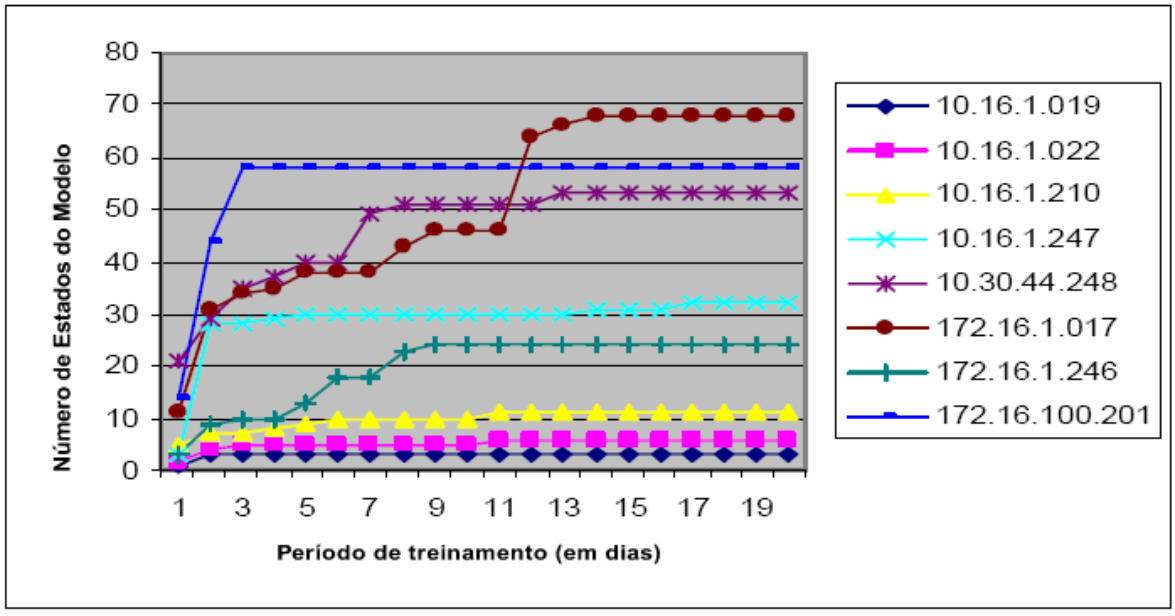

Figura 2.2. Evolução dos estados durante período de treinamento (YASAMI, et al, 2007, tradução nossa)

A figura mostra, por meio de um gráfico, que uma estabilização do número de estados criados pode ser uma métrica para avaliar se o período de treinamento foi suficiente, mesmo

antes de testar o modelo comportamental em uma situação real. No entanto, esta não deve ser a única métrica, principalmente sabendo que a estabilização do número de transições é tão importante quanto o número de estados.

Técnica equivalente foi apresentada por JOSHI (2005), no entanto, aplicado a um nível mais alto do modelo OSI, entre a camada de transporte e rede. Joshi (2005), em seu trabalho, tinha o objetivo principal de mostrar que o HMM é uma solução relevante para criação de sistema de detecção de intrusão.

Por meio de um conjunto de dados preparados para avaliar um sistema de detecção de intrusão, o KDDCup'99 (ACM, 1999), JOSHI avaliou sua solução obtendo um índice satisfatório de 79% de anomalias detectadas.

2.2.2 Modelo Yang

Segundo Yang (2007), pesquisas anteriores ao seu trabalho voltadas para sistemas de detecção de intrusão em rede (NIDS), utilizavam tipicamente técnicas para detecção de anomalias conhecidas.

Estas técnicas têm dificuldade em detectar novos tipos de ataques ou causa altas taxas de falsos positivos no ambiente de rede real. A detecção não supervisionada proposta por Yang (2007) superou de forma satisfatória estes inconvenientes.

Yang (2007) apresentou padrões de comportamento normal e anormal que são construídos sobre o tráfego da rede de dados que usa *subtractive clustering* (HAMMOUDA, KARRAY, 2004) e simultaneamente constrói um HMM correlacionando a observação de seqüências e transições para predizer o estado mais provável que identifica uma anomalia.

Os resultados amostrados por Yang (2007) em sua abordagem foram capazes de reduzir os falsos positivos e classificar intrusão em diferentes níveis de emergência. Os resultados experimentais foram também reportados utilizando o KDDCup'99 (ACM, 1999) e Matlab.

A proposta de Yang (2007) de aplicar HMM e *Subtractive Clustering* apresenta resultados parciais que comprovam a eficiência e aplicabilidade de HMM como uma solução satisfatória, destacando ainda uma solução para *Network IDS*, ou seja, para sistemas de detecção de intrusão distribuídos.

2.3 Geração de Ataques Artificiais

Sistemas de detecção de intrusão devem ser capazes de detectar ataques novos e desconhecidos. Wei Fan (2001) criou um algoritmo para gerar anomalias artificiais para coagir e induzir um IDS a descobrir uma rigorosa fronteira entre classes de anomalias conhecidas e desconhecidas. Estudos empíricos mostraram que o modelo proposto pelo autor aplicado a um ambiente real ou artificial é capaz de detectar mais de 77% de todas as classes de anomalias desconhecidas com mais de 50% de exatidão.

Classificação e detecção de anomalia são duas tarefas comuns na análise dos dados. A detecção de anomalia mapeia eventos que são inconsistentes, que se desviam de eventos que são conhecidos ou esperados em um modelo comportamental. Por outro lado, sistemas de classificação fazem uso de padrões de classes bem conhecidos para relacionar e identificar rótulos conhecidos e atribuir um identificador a um conjunto de informações que ainda não foram rotuladas.

Uma das maiores dificuldades na utilização de métodos de aprendizagem automática para a detecção de anomalia reside em fazer o modelo comportamental descobrir as fronteiras entre

classes de anomalias conhecidas e desconhecidas. Os modelos só podem gerar hipóteses de anomalias observando as informações organizadas no período de treinamento.

No trabalho desenvolvido por Wei Fan (2001), o modelo comportamental gerado no período de treinamento não é alterado, ou seja, o sistema em modo normal não possui autonomia para criar novos padrões de anomalia. No entanto, Wei Fan (2001) propôs um gerador artificial de anomalias como solução para melhorar a sensibilidade do modelo comportamental construído no período de treinamento. Assim, anomalias artificiais são injetadas no período de treinamento para ajudar a realçar as fronteiras e as variações de possíveis comportamentos anômalos.

Para isso, Wei Fan (2001) criou um algoritmo para gerar anomalias artificiais com base em anomalias conhecidas. Este foi aplicado ao modelo comportamental após o período de treinamento e pode enriquecê-lo e torná-lo mais eficiente e preciso para detectar anomalias conhecidas e desconhecidas. O algoritmo é relativamente simples como apresentado na seqüência.

```
Entrada: D (seqüência anômala); Saída: D' (seqüência anômala artificial)
   1. F = conjunto de todas as características de D.
   2. Vf = conjunto de valores únicos de alguma característica f ∈ F.
   3. D' = Ø.
   4. Para cada f ∈ F
      countVmax = número de ocorrências do valor mais freqüente em Vf
      Para cada v ∈ Vf
            countV = número de ocorrências de v em D.
            laço i: countV < i <= countVMax
                  d = escolha aleatória de d ∈ D
                  vf = valor da característica f para d.
                  substitui vf com um valor aleatório escolhido
                  v' → v'≠ v e v'≠ vf para criar d'.
                  D' ← D'U {d'}.
   5. Retorna D'
```

Este trabalho correlato foi importante para definir uma metodologia para analisar o comportamento da CISS mediante ataques conhecidos e desconhecidos. Os resultados serão apresentados e discutidos no capítulo 6.

2.4 Integração de Serviços de Segurança em Hardware

A arquitetura da versão em hardware da CISS, descrita no capítulo 4, mesclando software embarcado e hardware dedicado em um único chip, teve como base grupos de pesquisas e projetos acadêmicos e comerciais que almejam a implementação em hardware, por meio de sistema embarcados, de soluções integradas de segurança. Na seqüência destacam-se estes correlatos.

2.4.1 Grupos de Pesquisa

Existem muitos grupos de pesquisa que têm como foco a segurança de informações em hardware e que simultaneamente buscam metodologias para melhorar a relação de serviços de segurança para a proteção de um sistema computacional. Na seqüência, alguns grupos que merecem destaque como correlatos a esta tese.

O grupo de sistemas embarcados de segurança coordenado pelo professor Patrick Schaumont (SCHAUMONT, 2009), conhecido como *Secure Embedded Systems Group (SESG)*, pesquisa os aspectos de projeto e implementação de segurança em sistemas embarcados. Um dos principais objetivos do grupo é a integração entre hardware e software prototipados em FPGAs para aplicações que envolvam segurança.

Segundo os pesquisadores do SESG, os sistemas embarcados de confiança e seguros são essenciais para o mundo que faz uso de ambientes inteligentes e da computação pervasiva (ubíqua). Este grupo está focalizado em três projetos principais:

- RFID seguro com base na modulação Ultra-WideBand (UWB);
- Proteção da propriedade intelectual para hardware e software;
- Hardware/Software codesign para segurança de sistemas embarcados.

O COSY é um grupo de pesquisa dirigido pelo Prof. Christof Paar, sendo que este é um dos grupos que formam o Horst Görtz Institute for IT Security (PAAR, 2009). O foco principal está na segurança embarcada. O grupo COSY é reconhecido internacionalmente nas áreas de segurança embarcada tais como: criptografia em hardware, implementações seguras velozes e computadores cripto-analíticos. A seguir, algumas áreas de pesquisa deste grupo são:

- Hardware para algoritmos criptográficos.
- Algoritmos criptográficos em FPGAs.
- Algoritmos criptográficos em processadores embarcados.

O Grupo de pesquisa de segurança embarcada e distribuída da Universidade de Twente, o *Distributed and Embedded Security Research Group (DIES)* (DIES, 2009), desenvolve pesquisas e projetos como o Istrice e o Prosecco para integração de segurança e privacidade em redes de computadores. Estes projetos geraram contribuições importantes em diversas áreas da segurança de informação destacando trabalhos inclusive para a detecção de anomalias (BOLZONI, 2009).

O professor Dr. Cetin Kaya Koç (KOÇ, 2009) da Universidade de Oregon realiza pesquisa em diversas áreas ligadas a hardware criptográfico e sistemas embarcados de segurança, destacando-se principalmente nos tópicos:

- Hardware resistente a adulterações;
- Tecnologias em hardware para a proteção de conteúdos;
- Segurança para software e sistemas embarcados;
- Algoritmos de segurança eficientes para sistemas embarcados;
- Processadores específicos (coprocessadores) para criptografia;
- Hardware reconfigurável em FPGA para criptografia.

O grupo de segurança embarcada da universidade da Califórnia (UCLA Embedded Security Group) é conhecido pelos projetos desenvolvidos com FPGA e VLSI na área de segurança de informações (VERBAUWHEDE, 2009). O Grupo liderado pela professora Ingrid Verbauwhede, contribuem com pesquisas voltadas para processadores criptográficos e hardwares de autenticação.

O professor Ronald L. Rivest possui várias contribuições na área de segurança computacional, entre elas o algoritmo RSA (RSA, 1977). Desde 1995 colabora com grupo de segurança de informações e criptografia (*Cryptography and Information Security Group-CIS Group*) (CIS GROUP, 2009). O grupo está trabalhando atualmente na proposta do algoritmo SHA-3.

2.4.2 Plataformas Comerciais

Muitas empresas voltadas para a área de redes de computadores em seu catálogo de produtos possuem equipamentos para segurança em hardware. Sendo que alguns destes dispositivos conseguem relacionar dois ou mais serviços de segurança para identificar, atuar e prevenir situações anômalas.

Equipamentos mais sofisticados como a família Cisco ASA (CISCO, 2008), Juniper Networks ISG (JUNIPER, 2009), Sonicwall TotalSecure TZ (SONICWALL, 2009), WatchGuard Firebox(WATCHGUARD, 2009), 3Com® Unified Security Platforms (3COM,2009), entre outras, permitem a unificação de serviços de segurança como firewall, VPN, IPS, anti-vírus, anti-spam e anti-spyware.

No entanto, unificação não é sinônimo de integração. Estes dispositivos oferecem diversos serviços de segurança de forma modular, contudo poucos oferecem mecanismos de integração e colaboração dos serviços de segurança implementados.

As plataformas comerciais, muitas vezes, possuem características proprietárias que impedem a integração com outras ferramentas de segurança. Neste contexto, a CISS foi concebida para criar uma opção para integração de diferentes plataformas e serviços de segurança.

2.5 Considerações Finais do Capítulo

Os trabalhos correlatos apresentados neste capítulo foram importantes para o projeto e implementação da CISS. Os correlatos foram divididos em duas categorias: Integração de Serviços de Segurança e Modelos Comportamentais, sendo que estas compõem a estrutura principal da CISS.

Neste contexto, a CISS possui um modelo de dados para integrar as informações provenientes de serviços de segurança presentes em um sistema computacional e explora os benefícios destas informações de segurança concentradas e organizadas para atingir o objetivo descrito no primeiro capítulo desta tese.

Como ponto de partida, nota-se que existem inúmeros trabalhos que apresentam soluções para a integração de informações de serviços de segurança. O intuito principal destas soluções é criar um modelo de dados para organizar informações de segurança em classes, de forma que permitam identificar, classificar e principalmente relacionar ou integrar as mesmas.

A CISS, seguindo o mesmo raciocínio, possui um modelo de integração de dados que incorpora alguns padrões para relacionar informações de segurança inspirados em modelos já existentes, como o proposto por Rasheed (2007).

Rasheed propôs um modelo de integração com base em padrões como IDMEF (*Intrusion Detection Message Exchange Format*) e XACML (*eXtensible Access Control Modeling Language*). Na definição da CISS (ver capítulo 3), é ressaltada a importância da formalização do padrão de organização de dados referente a anomalias detectadas, pois diferentes serviços de segurança por meio de uma formatação comum podem gerar informações precisas sobre uma determinada anomalia.

O modelo IDMEF foi adotado na integra como padrão de organização de informações de segurança; no entanto, a CISS possui uma estrutura customizada e adaptada para auxiliar as etapas seguintes após a captura e registro de uma anomalia.

Uma vez definida a formatação e a estrutura de dados comum para armazenar informações relacionadas à segurança de um sistema computacional, muitas aplicações podem ser desenvolvidas para analisar, comparar e tomar decisões. A CISS possui módulos que permitem observar esse conjunto de informações e criar modelos comportamentais para identificar e classificar de forma prematura e com precisão situações anômalas.

Existem muitas pesquisas sobre modelos comportamentais para detecção de situações anômalas em sistemas computacionais. Estas foram adaptadas para o escopo da CISS e novos resultados foram alcançados e analisados nesta tese.

Yasami (2007) em seu trabalho modificou e adaptou o HMM original para atender as características do ambiente que desejava analisar. Como resultado, obteve grafos de comportamento mais representativos para que um sistema de tomada de decisões fosse mais preciso em relação à anomalia identificada.

Não obstante isto e inspirado nas modificações realizadas por Yasami (2007) no HMM foi proposta uma nova versão do HMM modificado e adaptado para representar e analisar as informações concentradas na CISS. Desta forma, este trabalho é um forte correlato quando o aspecto analisado são os modelos de comportamentais para detecção de anomalias em sistema computacionais.

Outros trabalhos como de Joshi (2005) aplicam técnicas semelhantes, porém enquanto Yasami aplica HMM na camada de enlace do modelo OSI especificamente sobre o protocolo ARP, Joshi trabalhou em um nível mais alto explorando a camada de Rede e analisando o comportamento do protocolo TCP/IP em busca de anomalias.

Neste contexto, estes trabalhos levaram a uma pergunta simples. Qual é o nível mais adequado no modelo OSI para instalar um sistema integrado de segurança que prima pela eficiência e eficácia na proteção e prevenção de sistemas computacionais?

O trabalho descrito nesta tese apresenta como solução uma Camada de Integração de Serviços de Segurança (CISS) que atua entre o sistema operacional e aplicações (camada de aplicação do modelo OSI), onde foi criada uma estrutura comum de integração.

A CISS é um modelo de integração que não despreza nenhum tipo de serviço de segurança, ou seja, não foi criada para substituir um serviço especifico de segurança, e sim para criar um modelo integrado e eficaz na proteção de sistemas computacionais (ver capitulo 3).

Por ser dependente dos serviços de segurança presentes no sistema, a CISS analisa os resultados de diversos serviços para a tomada de decisões. Neste contexto, são consideradas informações provenientes de diferentes níveis do Modelo OSI, conforme os serviços de segurança instalados em um determinado sistema computacional.

A primícias básica que justifica a CISS é que um ataque a um determinando sistema computacional passa por várias etapas e testes antes de obter sucesso. Esta trajetória de ataque pode ser mapeada, ou seja, os insucessos podem ser transformados em modelos comportamentais e futuramente poderão ser prevenidos. O insucesso neste caso refere-se as tentativas de ataques detectadas por serviços de segurança presentes no sistema e notificadas para a CISS.

Diferente da maioria dos trabalhos citados, nesta tese a CISS não cria modelos de comportamento normal para depois comparar e analisar com modelos criados em tempo real em busca de anomalias. A CISS cria modelos comportamentais de anormalidade que são comparados com modelos gerados em tempo real. Desta forma, pode identificar e criar modelos personalizados para um determinado sistema computacional ou importar banco de modelos de anomalias já identificadas.

A CISS tem como objetivo integrar serviços de segurança que existem ou que ainda serão criados. Como se pode observar, as técnicas de integração destacadas neste capítulo são específicas para um conjunto de serviços, sendo esta mais uma característica que justifica o trabalho proposto.

A revisão bibliográfica de integração de serviços de segurança em hardware apontou o uso de sistemas embarcados como uma boa alternativa na proteção de sistemas computacionais, utilizando software embarcado e hardware dedicado. Tecnologia esta adotada para a concepção da versão em hardware da CISS.

Plataformas comerciais possuem soluções unificadas de segurança, porém as características proprietárias e metodologias que abrangem um número limitado serviços de segurança específicos, dificultam a integração e colaboração real. No capítulo 4, apresenta-se a CISS implementada em hardware como uma solução.

CAPÍTULO 3

CISS - CAMADA DE INTEGRAÇÃO DE SERVIÇOS DE SEGURANÇA

Neste capítulo, destacam-se as principais características da Camada de Integração de Serviços de Segurança (CISS), destacando informações sobre a arquitetura, fundamentação matemática, modelo de informações e modelos comportamentais utilizados na detecção e proteção de sistemas computacionais.

3.1 Arquitetura Top-Level da CISS

A Camada de Integração de Serviços de Segurança (CISS) é composta por módulos que têm desde a função de armazenar informações sobre anomalias notificadas até módulos robustos de análise e detecção de situações anômalas.

As informações armazenadas na CISS podem ser utilizadas não apenas para o propósito geral destacado nesta tese, mas para outros fins, como integração de conjunto específico de serviços de segurança que necessitam de um meio e um formato comum para troca de informações sobre eventos anômalos.

Por meio da sua arquitetura top-level, destacada na figura 3.1, pode ser compreendida a dinâmica do tratamento dos eventos de segurança registrados. Quanto maior o número de ferramentas de segurança fornecendo informações sobre a "saúde" do sistema, mais poderosa pode se tornar essa arquitetura.

Com a CISS, pretende-se mostrar e fomentar que não apenas ferramentas específicas para segurança podem contribuir para a proteção de um sistema computacional, mas também software proprietário e *opensource,* que podem ser fundamentais para criar um roteiro de tentativas de ataques.

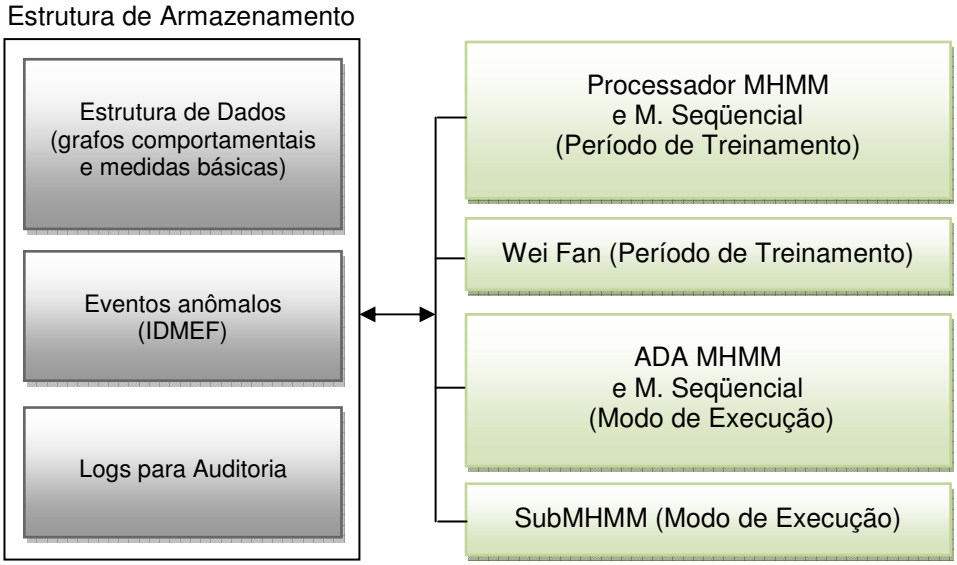

Figura 3.1. Arquitetura Top-Level da CISS

A figura 3.1 apresenta a arquitetura top-level da CISS. Como pode ser observado, a arquitetura pode ser dividida módulos:

- Estrutura de armazenamento: composta de tabelas relacionadas capazes de armazenar estrutura de dados dos grafos comportamentais, *logs* para auditorias e eventos de segurança obedecendo ao formato IDMEF, chamado também de modelo de informações;
- Processador de Modelos Comportamentais: com base no Modelo Oculto de Markov Modificado (MHMM) e no Modelo Seqüencial (ver seção 3.3), o sistema pode passar por um período de treinamento ou importar modelos de comportamento para alimentar a base de conhecimento de anomalias;

- Sistema de detecção de anomalia: com base nos modelos de comportamentos anômalos, o sistema identificará anomalias em modo normal de execução utilizando Algoritmos de Detecção de Anomalias (ADA);
- Módulos de Otimização de Detecção: Algoritmo de Wei Fan para o período de treinamento e a técnica de subMHMM para o período de execução em modo normal, ambos definidos neste capítulo.

É importante salientar que os resultados oriundos da CISS podem ser utilizados por um sistema de tomada de decisões que, por sua vez, pode atuar sobre os serviços de segurança para prevenir ocorrências futuras. O sistema de tomada de decisões não é o foco desta tese e não foi implementado. No entanto, a CISS foi preparada para fornecer subsídios suficientes para que seja classificada e identificada uma anomalia específica.

3.2 Modelo de Informações IDMEF-CISS

A organização de informações sobre anomalias detectadas em um sistema computacional constitui um dos pilares da CISS. Neste sentido, foi criada uma estrutura de dados capaz de organizar e relacionar estas informações com o intuito de prover uma fonte de dados que será utilizada por mecanismos de detecção, atuação e prevenção de situações anômalas.

Um evento de segurança, ocorrência ou notificação são sinônimos para uma anomalia identificada por alguma das ferramentas de segurança que atua em um sistema computacional. Todo evento de segurança deve ser registrado na CISS, seguindo o modelo de dados definido na seqüência.

O evento de segurança deve ser registrado no formato de alerta descrita no padrão IDEMF. Apesar deste modelo ter sido criado com o propósito de formatar um alerta de detecção de intrusões, na CISS, este tem a função de notificar não apenas o alerta de detecção de intrusão, mas também todas as anomalias detectadas em um sistema computacional. O formato IDMEF é abrangente e o uso do XML para representar os alertas facilitou a inclusão deste padrão na CISS.

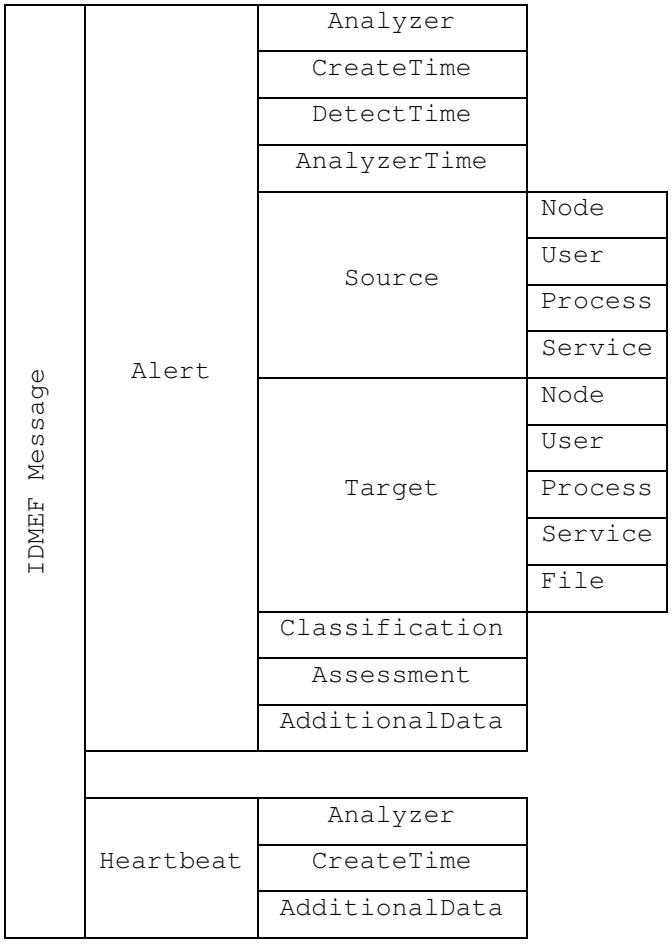

Figura 3.2. Padrão IDMEF adaptado a CISS (RFC 4765)

A estrutura de uma mensagem IDMEF, destacada na figura 3.2, permite armazenar todas as informações sobre a ocorrência de uma anomalia. A utilização desta formatação permite

que outras ferramentas possam fazer uso compartilhado da mesma estrutura. Na seqüência, uma descrição sucinta de cada campo da mensagem IDMEF.

Mensagens IDMEF : são possíveis dois tipos de mensagens ALERT e HEARTBEAT. A primeira é a notificação detalhada de uma anomalia; a segunda é utilizada como sincronismos entre ferramentas de segurança, onde periodicamente são enviadas mensagens do tipo HEARTBEAT para confirmar a atividade da comunicação.

ALERT

- Analyzer: Ferramenta que gerou o alerta. Deve possuir um identificador único. Neste campo, são armazenadas informações detalhadas da ferramenta que originou o alerta desde sua caracterização como versão, sistema operacional, desenvolvedor, até se é um hardware ou software;
- CreateTime: O tempo que o alerta foi registrado na CISS;
- DetectTime: tempo que o alerta foi identificado pela ferramenta de segurança;
- AnalyzerTime: tempo corrente do Analisador. Este tempo é importante caso exista diferença de tempo entre o analisador e a CISS;
- Source: informações detalhadas do nó que originou a ocorrência notificada através deste alerta. Entre as informações armazenadas destaca-se: endereço de rede, IP, usuário e processo;
- Target: informações sobre o nó alvo que está sobre impacto da anomalia detectada. Neste campo, são registradas todas as informações registradas no campo Source com a possibilidade adicional de descrever os arquivos afetados pela anomalia;
- Classification: identifica o tipo de anomalia gerada, como por exemplo: DOS, DDOS, “Ping da morte”, etc. A não classificação da anomalia por falta de

conhecimento das ferramentas de notificação pode ser suprida pelo módulo de análise de comportamento descrito na seção 3.3;

- Assessment: informações sobre o impacto do evento de segurança. O impacto pode ser classificado em quatro tipos: (i) apenas informativo, (ii) impacto baixo, (iii) médio e (iv) alto. Outras informações importantes são armazenadas neste campo como o sucesso ou não da ocorrência e ações tomadas em relação a mesma;
- AdditionalData: informações adicionais que não foram contempladas no modelo de informação IDMEF. Este campo é utilizado pela CISS para armazenar o tipo de atributo de segurança que notificou a anomalia. Esta informação é essencial para a criação dos modelos comportamentais descritos na seção 3.3.

HEARTBEAT : Este tipo de mensagem IDMEF não foi implementado nesta versão da CISS, simplesmente por não ter uma utilidade direta com o objetivo principal do projeto.

A seguir, tem-se um exemplo da formatação de uma notificação de uma anomalia classificada como “ping da morte”, que tem como objetivo indisponibilizar serviços em um sistema computacional.

```
<?xml version="1.0" encoding="UTF-8"?>

   <idmef:IDMEF-Message version="1.0"
                        xmlns:idmef="http://iana.org/idmef">
     <idmef:Alert messageid="abc123456789">
       <idmef:Analyzer analyzerid="bc-sensor01">
         <idmef:Node category="dns">
           <idmef:name>sensor.example.com</idmef:name>
         </idmef:Node>
       </idmef:Analyzer>
       <idmef:CreateTime ntpstamp="0xbc71f4f5.0xef449129">
         2000-03-09T10:01:25.93464Z
       </idmef:CreateTime>
       <idmef:Source ident="a1a2" spoofed="yes">
         <idmef:Node ident="a1a2-1">
           <idmef:Address ident="a1a2-2" category="ipv4-addr">
             <idmef:address>192.0.2.200</idmef:address>
           </idmef:Address>
         </idmef:Node>
```

```
    </idmef:Source>
    <idmef:Target ident="b3b4">
      <idmef:Node>
        <idmef:Address ident="b3b4-1" category="ipv4-addr">
          <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="c5c6">
      <idmef:Node ident="c5c6-1" category="nisplus">
        <idmef:name>lollipop</idmef:name>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="d7d8">
      <idmef:Node ident="d7d8-1">
        <idmef:location>Cabinet B10</idmef:location>
        <idmef:name>Cisco.router.b10</idmef:name>
      </idmef:Node>
    </idmef:Target>
     <idmef:Classification text="Ping-of-death detected">
      <idmef:Reference origin="cve">
        <idmef:name>CVE-1999-128</idmef:name>
        <idmef:url>http://www.cve.mitre.org/cgi-bin/
        cvename.cgi?name=CVE-1999-128</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>
</idmef:IDMEF-Message>
```

O uso do XML é fundamental para que diferentes sistemas de gerenciamento de banco de dados como MySql, Postgresql, SQL Server, entre outros, possam interpretar e armazenar informações de alertas de anomalias.

Através destas informações, é possível a criação de modelos de comportamento de ataques, regras de detecção e regras de atuação. Na seção 3.3, destaca-se a metodologia adotada para gerar os modelos comportamentais.

3.3 Modelos Comportamentais da CISS

Os modelos comportamentais são utilizados para classificar ações normais e anormais em um sistema computacional. Entre as técnicas mais referenciadas, destaca-se o Hidden Markov Model (RABINER,1989), Support Vector Machine (LI; et al; 2003) e Fuzzy Data Mining (BRIDGES, 2000).

Os modelos comportamentais podem ser aplicados para detectar seqüências de anomalias detectadas pela CISS. Entre as técnicas citadas, o Hidden Markov Model (HMM) foi adotada para formalizar as seqüências de anomalias que podem caracterizar um ataque a um sistema computacional. Na seqüência, a definição do HMM e a justificativa para a sua utilização.

3.3.1 HMM

Um processo estocástico é definido como uma coleção de variáveis aleatórias *X(t)*, sendo *t* geralmente a representação de uma variável de tempo. *X(t)* representa uma característica mensurável de interesse no tempo *t*. Exemplificando, *X(t)* pode representar o nível de estoque de um produto no fim do dia *t* (LEITE, 2008) (RABINER, 1989).

Processos estocásticos são utilizados para descrever o comportamento ao longo de um período de tempo de um sistema onde a incerteza é significativa. Em termos formais, a variável aleatória *X(t)* representa o estado do sistema no instante *t*. Os processos estocásticos podem ser classificados (LEITE, 2008):

Em relação a estado:

- Estado Discreto: *X(t)* é definido sobre um conjunto enumerável ou finito;
- Estado Contínuo: caso contrário.

Em relação ao tempo:

- Tempo discreto: *t* é finito ou enumerável;
- Tempo contínuo: caso contrário.

Um processo estocástico é dito markoviano se a probabilidade do processo estar em um dado estado, em um instante qualquer no futuro, depende apenas do estado presente e não dos

estados passados. Este tipo de processo estocástico pode ser classificado como *memoryless process,* uma vez que transições passadas são desprezadas (RABINER, 1989).

Um processo markoviano pode ser definido como um modelo de Markov, quando as variáveis aleatórias X(t) estão definidas em um espaço de estado discreto, ou seja, o número de estados é finito ou enumerável (RABINER, 1989).

Para instantes de tempo igualmente espaçados, o sistema passa por uma mudança de estado, podendo retornar ao mesmo estado, de acordo com um conjunto de probabilidades de transições. Cada instante de tempo em que ocorrem as transições de estados é descrito por t=1,2,..., e o estado no tempo t é denotado com o símbolo q_t, considerando um espaço de tempo discreto.

O estado inicial em um instante t de tempo que determina o primeiro evento pode ser qualquer um dos estados de um conjunto de N estados distintos ($S_1, S_2,..., S_N$) como ilustrado na figura 3.3.

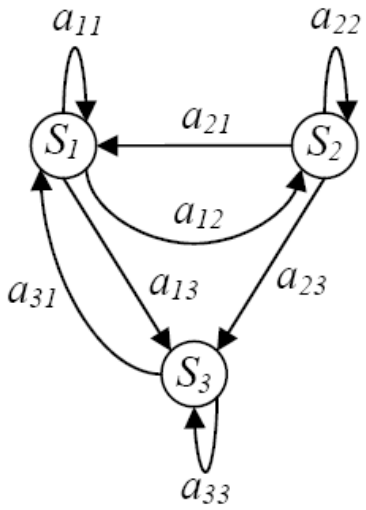

Figura 3.3. Exemplo de uma Modelo de Markov com três estados (S_1, S_2 *e* S_3).

Uma descrição probabilística completa do sistema requer a especificação do estado presente (no tempo t) e também os estados anteriores. Para o caso especial de um modelo de

markov de primeira ordem, essa descrição pode ser truncada apenas no estado presente e no seu antecessor (fórmula 3.1).

$$\mathrm{P}(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \ldots) = \mathrm{P}(q_t = S_j | q_{t-1} = S_i) \tag{3.1}$$

Considerando o processo em que o lado direito da fórmula 3.2 é independente do tempo, tem-se um conjunto de probabilidade de transições de estado a_{ij} :

$$a_{ij} = \mathrm{P}(q_t = S_j | q_{t-1} = S_i), \qquad 1 \le i, j \le N . \tag{3.2}$$

As probabilidades de transição de estado têm as seguintes propriedades descritas na fórmula 3.3:

$$a_{ij} \ge 0$$

$$\sum_{i=1}^{N} a_{ij} = 1 \tag{3.3}$$

Quando as probabilidades de transições não se alteram ao longo do tempo, diz-se que o processo é estacionário. Por meio das probabilidades de transições, é possível construir uma matriz de transições finita entre estados (fórmula 3.4), onde p_{ij} representa a probabilidade condicional de transição do estado S_i para S_j.

$$P = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \end{array} \tag{3.4}$$

O modelo de markov apresentada é um modelo observável uma vez que a saída do processo é um conjunto de estados em cada instante de tempo, onde cada estado representa um evento físico observável.

Nos modelos de markov descritos anteriormente, cada estado corresponde a um evento físico observável. Essa característica torna esses modelos muito restritos para a solução dos problemas de interesse nessa tese. Neste contexto, aplicou-se uma extensão do modelo de Markov chamado Modelo Oculto de Markov (*HMM-Hidden Markov Model*), onde o estado não é diretamente observável. Assim, uma observação é uma função probabilística do estado.

O HMM é um processo duplamente estocástico composto por um processo latente que não é observável (*hidden*), mas que se manifesta através de outro processo estocástico que produz uma seqüência de símbolos observáveis (LEITE, 2008).

Em um HMM, as observações (v_i) são símbolos emitidos por estados não observáveis (S_i) de acordo com determinadas funções probabilísticas, sendo que cada seqüência de estados é um modelo de markov de primeira ordem.

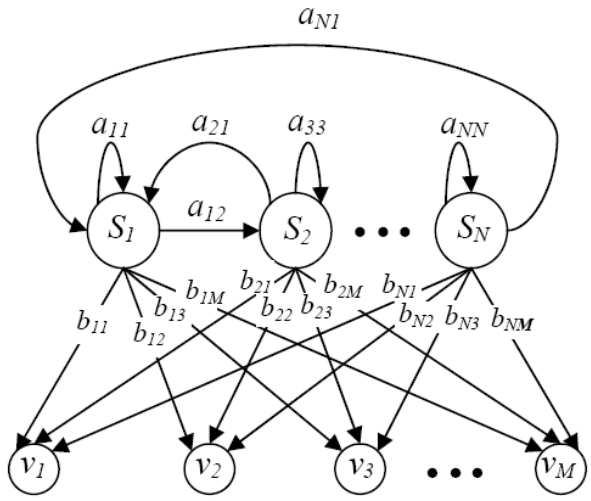

Figura 3.4. Exemplo de um Modelo Oculto de Markov.

A figura 3.4 ilustra um HMM, sendo que N é o número de estados do modelo (o conjunto de estados individuas é representado por $S=\{S_1..,S_N\}$ e o estado no tempo i ,pelo símbolo q_t) e M , o número de símbolos distintos observáveis (o conjunto dos símbolos individuais é representado por $V=\{v_1, ...,v_M\}$).

Um HMM é constituído por três conjuntos de parâmetros principais (RABINER, 1989):

1. As probabilidades de emissões de símbolo $B=\{b_{jk}\}$ – a probabilidade do símbolo v_k ser emitido pelo estado S_j , i.e.

$$b_{jk} = \mathrm{P}(v_k \text{ em } t | q_t = S_j), \quad 1 \le j \le N \text{ e } 1 \le k \le M \tag{3.5}$$

2. As probabilidades de transição de estados $A=\{a_{ij}\}$ – a probabilidade de estar no estado S_j no instante de tempo subseqüente, dado que o estado atual é S_i, i.e.

$$a_{ij} = \mathrm{P}(q_{t+1} = S_j | q_t = S_i), \quad 1 \le i,j \le N \tag{3.6}$$

3. A distribuição de probabilidade a priori $\Pi = \{\Pi_i\}$ do sistema estar em um dado estado S_i no instante inicial de tempo.

$$\pi_i = \mathrm{P}(q_1 = S_i), \quad 1 \le i \le N \tag{3.7}$$

Desta forma, pode-se concluir que um *Hidden Markov Model* (HMM) é um modelo estatístico, onde o sistema a ser modelado é assumido como sendo um processo Markoviano com um conjunto finito de estados e parâmetros desconhecido e cada um dos estados está associado a uma distribuição probabilística. O HMM gera uma seqüência interna e uma seqüência externa de símbolos e utilizando regras probabilísticas, o desafio consiste em determinar os parâmetros ocultos a partir dos parâmetros observáveis (RABINER, 1989).

As transições entre os estados são regidas por um conjunto de probabilidades chamadas probabilidades de transição. Em um estado particular, um resultado observável é gerado de acordo com distribuição probabilística associada. Este resultado, na verdade, corresponde aos estados ocultos (*hidden*) se manifestando para o exterior e podendo ser utilizados na análise comportamental do sistema.

Nesta tese, não se detalha o HMM, uma vez que podem ser encontradas muitas referências na literatura que abordam especificamente este tema (BUNKE, CAELLI, 2001) (VEERAVALLI, et al, 2005) (CAPPÉ, MOULINES, 2005). No entanto, é importante salientar que o HMM aplicado a CISS está detalhado na seqüência.

3.3.2 HMM Modificado

Segundo Yasami (2007), o modelo HMM pode ser modificado para atender características no que diz respeito a itens relevantes que se pretendem analisar. Yasami modificou o HMM e aplicou suas novas equações para a criação de um modelo comportamental para integrar um sistema de detecção de intrusão com base em requisições ARP.

Como destacado no capítulo 2, o trabalho de Yasami (2007) foi uma inspiração para esta tese e o HMM modificado (MHMM) foi incorporado à CISS. Porém, mudanças e melhorias foram implementadas, e este foi reestruturado para gerar modelos anormais de comportamento, proporcionando informações importantes para que um sistema de detecção de anomalias possa atuar com eficiência e eficácia.

A motivação para escolha do HMM como mecanismo de detecção de anomalia pode ser justificada pelas seguintes características encontradas na CISS.

- Número de estados finitos: os estados que compõem o HMM são únicos. Desta forma, quanto mais estados existirem mais complexo é o grafo de transição e posterior análise

de resultados. Para a CISS, os estados estão associados ao endereço IP de origem e características de um ataque. Sempre o modelo estará apontando para um único estado, e o número de estados é finito e enumerado;

- As transições entre estados ocorrem por eventos. Para a CISS, um evento significa uma nova ocorrência detectada pelos serviços de segurança presentes no sistema, ou seja, a transição de estado ocorre quando novas ocorrências de anomalias surgem;
- Construção de modelos para condições anormais: no projeto desenvolvido, o HMM constrói grafos de condições anômalas do sistema. Desta forma, podem ser geradas assinaturas de ataques robustas que podem ser distribuídas e utilizadas em outros sistemas computacionais;
- Adaptativo: o HMM permite modificações que não afetam diretamente no propósito do modelo. Em inúmeros trabalhos na literatura, não apenas na área de exatas, o HMM representa não apenas o modelo comportamental de sistemas computacionais, mas de qualquer objeto ou ser vivo que, quando estimulado por um evento, produz um comportamento ou uma reação. Para isso, o HMM deve ser adaptativo e flexível para modelar o comportamento desejado. Nesta tese, foi criada uma variação do HMM chamada MHMM;
- Relação tempo: O HMM utiliza o tempo para relacionar os eventos ocorridos para caracterizar um comportamento. Isto significa que a mesma seqüência de eventos distribuída de forma distinta durante um período de tempo fixo são computados como comportamentos diferentes. Esta característica permite distinguir anomalias de forma precisa e classificá-las.

3.4 MHMM aplicado à CISS

O MHMM pode ser representado por um grafo composto por Estados (vértices) e Transições (arestas). Os estados do MHMM são caracterizados sob a CISS como elementos únicos que possuem os seguintes atributos: IP origem da anomalia, aplicação de segurança e classificação do atributo segurança.

A classificação dos atributos de segurança é predefinida como: controle de acesso, confidencialidade, irretratabilidade, autenticação, integridade e disponibilidade. Outras classificações podem ser criadas e personalizadas na CISS.

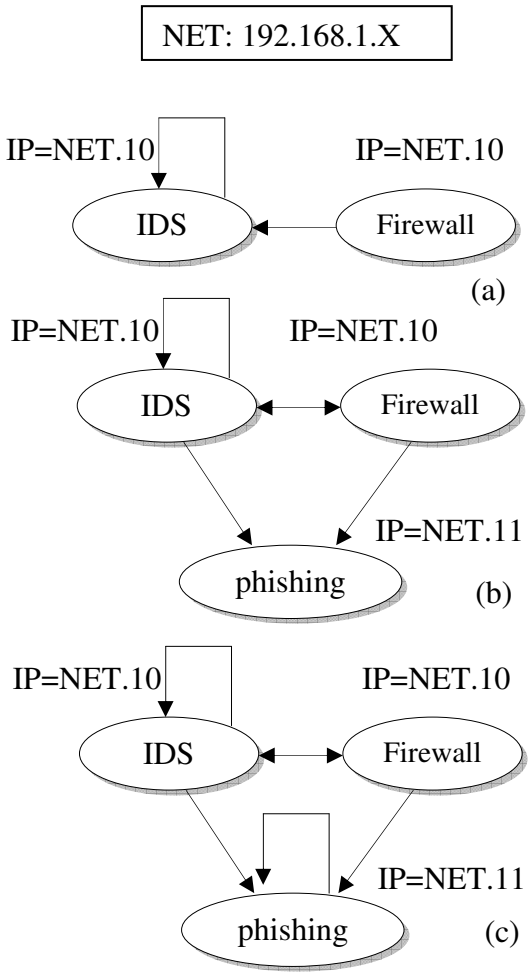

Figura 3.5. Criação do Modelo Comportamental

A criação do MHMM ocorre conforme a detecção de anomalias identificadas pelas aplicações de segurança presentes no sistema. A figura 3.5 apresenta os estágios do ciclo de criação do MHMM aplicado à CISS.

Nota-se que na figura 3.5a tem-se apenas o início da construção do modelo comportamental com a detecção de, no mínimo, três ocorrências. No decorrer do tempo, as ocorrências vão se sucedendo e as anotações no MHMM são acumulativas (fig 3.5b e 3.5c). Ao final de um período de treinamento, o grafo de ocorrências estará robusto o suficiente para identificar os ataques caracterizados.

A construção do modelo comportamental ocorre no início da configuração do sistema, chamado de período de treinamento. Desta forma, o grafo de comportamento é criado durante um período de treinamento, onde a CISS é estimulada sofrendo ataques que podem ser artificiais ou reais para a criação do modelo comportamental.

A CISS permite o treinamento contínuo, ou seja, o sistema em tempo real pode alimentar o MHMM com novos estados e transições. Com isso, o modelo pode se adaptar em tempo de execução aos novos ataques ou novas aplicações instaladas no sistema computacional.

A abordagem de treinamento contínuo, em um primeiro instante, parece ser interessante, uma vez que o sistema aprenderia em tempo real, ou seja, após o período de treinamento. No entanto, existem alguns problemas que desaconselham essa prática, sendo que os mais evidentes são a descaracterização do modelo comportamental base e inclusão de modelos anormais ou normais de forma equivocada. O efeito direto é a degradação da capacidade de caracterização e detecção de situações anômalas.

Neste contexto, se o modelo comportamental não expressa as características funcionais de normalidade ou anormalidade de um sistema computacional, isso significa que o período de treinamento foi insuficiente ou todos os eventos anômalos possíveis não ocorreram nesse período.

No entanto, a CISS aproveita eventos anômalos não previstos em período de treinamento para criar o *subMHMM* idealizado nesta tese e descrito na seção 3.4.3. Na seqüência, tem-se a fundamentação matemática e probabilística do MHMM proposto.

3.4.1 Medidas Básicas

As medidas básicas são elementos fundamentais para caracterizar o HMM construído. Neste sentido, qualquer alteração no modelo comportamental tem impacto em todas as medidas básicas calculadas. As medidas básicas são calculadas em período de treinamento para cada estado e estão descritas na seqüência:

A fórmula 3.8 determina a probabilidade de estar em um estado específico E (P_E), composto pelo IP origem, a aplicação e o atributo de segurança que registrou a ocorrência de uma anomalia, após um evento de segurança. O cálculo de P_E é dado pelo número de ocorrências no estado E dividido pela soma do número de ocorrências de todos os estados presentes no grafo.

E = representa um estado (vértice) no grafo de comportamento HMM.

$$P_E = \frac{N_E}{\sum_{i=1}^{k} N_i} \tag{3.8}$$

Onde,

N_E = Número de ocorrências de E

N_i = Número de ocorrências do Estado i

K = Número total de Estados

A fórmula 3.9 determina o tempo de duração média de permanência em um estado específico E. Esta fórmula, apesar de simples, tem um fator determinante: o tempo. Por meio desta, pode-se determinar o comportamento das ocorrências no decorrer do tempo. No escopo de implementação é necessária uma contagem de tempo entre a última ocorrência e a ocorrência atual. Em ataques de seqüência disparados por softwares maliciosos, este fator é fundamental para identificar o comportamento do ataque no decorrer do tempo.

$$\varepsilon_E = \frac{\sum_{i=1}^{N_E} T_{Ei}}{N_E} \tag{3.9}$$

A fórmula 3.10 determina a variância populacional do tempo de duração em um estado especifico E. A variância é importante, pois representa a dispersão estatística de um conjunto de ocorrências, indicando quão longe em geral os seus valores se encontram do valor esperado.

$$\sigma^2 = E[T_E^2] - E[T_E]^2 \tag{3.10a}$$

Onde,

T_E= Tempo decorrido entre detecções de uma ocorrência em um estado especifico E

A variância também pode ser calculada utilizando a formula 3.10b em destaque na seqüência.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2, \tag{3.10b}$$

Onde,

N = Número de termos da população

μ = Média da população

y_i = Termos da população

Por fim, é necessário o cálculo das probabilidades condicionais de transição de um estado i para um estado j. Uma matriz de transições auxilia no cálculo, e esta representada na fórmula 3.11.

$$P_{ij} = \begin{array}{c} \\ S1 \\ S2 \\ S3 \\ \dots \\ S_{m+1} \\ S_M \end{array} \begin{array}{c} \begin{array}{cccccc} S1 & S2 & S3 & S_{n+1} & \dots & S_N \end{array} \\ \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{1n+1} & \dots & P_{1N} \\ P_{21} & P_{22} & P_{23} & P_{2n+1} & \dots & P_{2N} \\ P_{31} & P_{32} & P_{33} & P_{3n+1} & \dots & P_{3N} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{m+11} & P_{m+12} & P_{m+13} & P_{m+1n+1} & \dots & P_{m+1N} \\ P_{M1} & P_{M2} & P_{M3} & P_{Mn+1} & \dots & P_{MN} \end{bmatrix} \end{array} \quad (3.11)$$

A ocorrência de um novo evento tem impacto direto no cálculo da probabilidade P_E, da média ε_E, da variância e da probabilidade condicional de todos os nós do grafo. Desta forma, os cálculos destas quatro mediadas básicas devem ser realizados a cada ocorrência para todos os nós do grafo.

Na fórmula 3.11, tem-se a matriz de transição que representa o grafo de comportamento HMM. A partir desta matriz, pode-se calcular as probabilidades condicionais de transições de um estado i para um estado j.

As medidas básicas apresentadas são calculadas em período de treinamento e utilizadas em tempo de execução normal. Na próxima seção, apresenta-se o algoritmo de detecção de anomalia (ADA) que utiliza estas medidas básicas para identificar uma situação anômala.

3.4.2 ADA – Algoritmo de Detecção de Anomalia

Os algoritmos de detecção de anomalia (*ADA- Anomaly Detection Algorithms*) são utilizados para interpretar o grafo de comportamento e gerar um índice que determina o nível de anomalia detectada. Para isso, os parâmetros básicos calculados durante o período de treinamento são comparados com os parâmetros gerados em tempo de execução.

O MHMM proposto por Yasami (2007) apresentou modificações simples em relação ao HMM que facilitaram a aplicação para a detecção de anomalias. As modificações propostas por Yasami foram incorporadas à CISS. As principais modificações estão ligadas a dois itens descritos na seqüência:

1. Não existe um estado inicial e final especial: diferente do HMM original, a modificação feita por Yasami (2007) não especifica um estado inicial e final no grafo de comportamento. Isso se fez necessário, uma vez que não é possível estabelecer de forma determinística qual serviço de segurança notificará a primeira ocorrência de anomalia. No entanto, isto pode ser previsto de forma probabilística como proposto no HMM original. O impacto dessa modificação tem como resultado a indiferença do estado inicial e final, porém agrega um tratamento especifico para estes casos, onde Yasami simplesmente adotou o mesmo critério para determinar o estado inicial no período de treinamento e no modo de execução normal.
2. Cálculo diferenciando de transição e permanência em um estado: YASAMI (2007), em seu modelo, adotou não apenas a métrica que destaca a probabilidade média de estar em um estado e a probabilidade de transição para outro. Yasami inseriu o cálculo da variância (fórmula 3.10) que é utilizada com medida de dispersão e que influencia na classificação de um ataque.

A seguir, o detalhamento do algoritmo de detecção de anomalia. Basicamente, o algoritmo confronta o modelo comportamental gerado em período de treinamento com o modelo criado em tempo de execução normal. Para isso, são criados *scores,* que quando têm seus valores compreendidos em uma mesma faixa numerada, caracterizam uma determinada anomalia. Inicialmente, são apresentadas as equações para o cálculo dos *scores* do período de treinamento *(ST)* (fórmula 3.12) e os *scores* do período execução normais (SE) (fórmula 3.13).

$$ST = \frac{PST_j}{P_{E_j}} + \sum_{k=2}^{N} \frac{PST_j}{P_{E_j} P_{ij}} \tag{3.12}$$

$$SE = \frac{PSE_j^k}{P_{E_j}} + \sum_{k=2}^{N} \frac{PSE_j^k}{P_{E_j} P_{ij}} \tag{3.13}$$

Como pode ser observado, na fórmula 3.13, a variável SE acumula os *PSE* (*Parciais SE*) que são resultados parciais do ADA. Da mesma forma, são calculados os PST (Parciais ST), que são explicados posteriormente. Nestas equações, tem-se *j* como o estado em que se encontra o modelo comportamental gerado no período de treinamento após *k* eventos anômalos. *N* é o número de eventos de segurança registrados na CISS durante o processo de detecção de anomalia.

PST_j/P_{Ej} e PSE_j^k/P_{Ej} são calculados para o primeiro evento anômalo tanto para o SE quanto para ST. Se for o primeiro evento anômalo $PSE_j^k = PSE_j^1$. Ainda nas fórmulas 3.12 e 3.13 encontra-se o $P_{ij.}$ Esta variável descreve a probabilidade condicional de ocorrer um evento do estado *i* para o estado *j* e pode ser descrita pela fórmula 3.14.

$$P_{ij} = P(T_{ij} \left| T_{I_1 I_2} T_{I_2 I_3} \ldots T_{I_{k-2} I_{k-1}} \right.) \tag{3.14}$$

Finalmente tem-se o PST (*Parcial ST*) e o *PSE* (*Parcial SE*), os quais são calculados a cada evento anômalo registrado na CISS utilizando as fórmulas 3.15 e 3.16 respectivamente.

$$PST_j = \frac{(\mathcal{E}_j)^2}{\sigma_j} 2 \tag{3.15}$$

$$PSE_j^k = \frac{(t_j^k)^2}{\sigma_j} 2 \tag{3.16}$$

O *PSE e o PST* utilizam as mesmas equações básicas do período de treinamento (ver seção 3.4.1). Onde t_{jk} encontrado na fórmula 3.16 é o intervalo de tempo entre a *k* e *(k+1)* ocorrência de anomalia.

Após calcular *ST* e *SE*, pode-se compará-los, e quanto mais próximos os valores, maior é a indicação de uma anomalia. Porém, deve-se estabelecer um limite para distinguir a anomalia ou normalidade do sistema. Este parâmetro é definido como *Threshold* (*Th*). Quando maior o *Th*, mais sensível é o ADA, podendo elevar o número de falsos positivos (YASAMI, et al; 2007). Para testes iniciais, foi fixado um valor de *15%* para este parâmetro. Isso significa que todos os eventos que têm 85% de proximidade com o modelo construído no período de treinamento notificam a identificação de uma anomalia.

$$\text{SE}(1\text{-}Th) \leq \text{ST} \leq \text{SE}(1\text{+}Th) \tag{3.17}$$

Desta forma, um sistema de tomada de decisão terá como identificar com uma porcentagem de acerto a ocorrência de uma anomalia. A definição de um valor para o *Threshold* pode ser feita por meio de simulação ou em testes reais, apurando o número de

falsos positivos gerados. Neste caso, o *Threshold* de 15% foi definido com base nas implementações e testes realizados e descritos nos capítulos posteriores. A definição do *Threshold* pode ser alcançada por meio da análise do número de falsos positivos (YASAMI; et al, 2007).

A figura 3.6 apresenta o ADA utilizando as equações citadas nesta seção. Ao final do processo, são gerados alertas no formato IDMEF que realimentam a CISS utilizando a técnica subMHMM, descrito na seção 3.4.3.

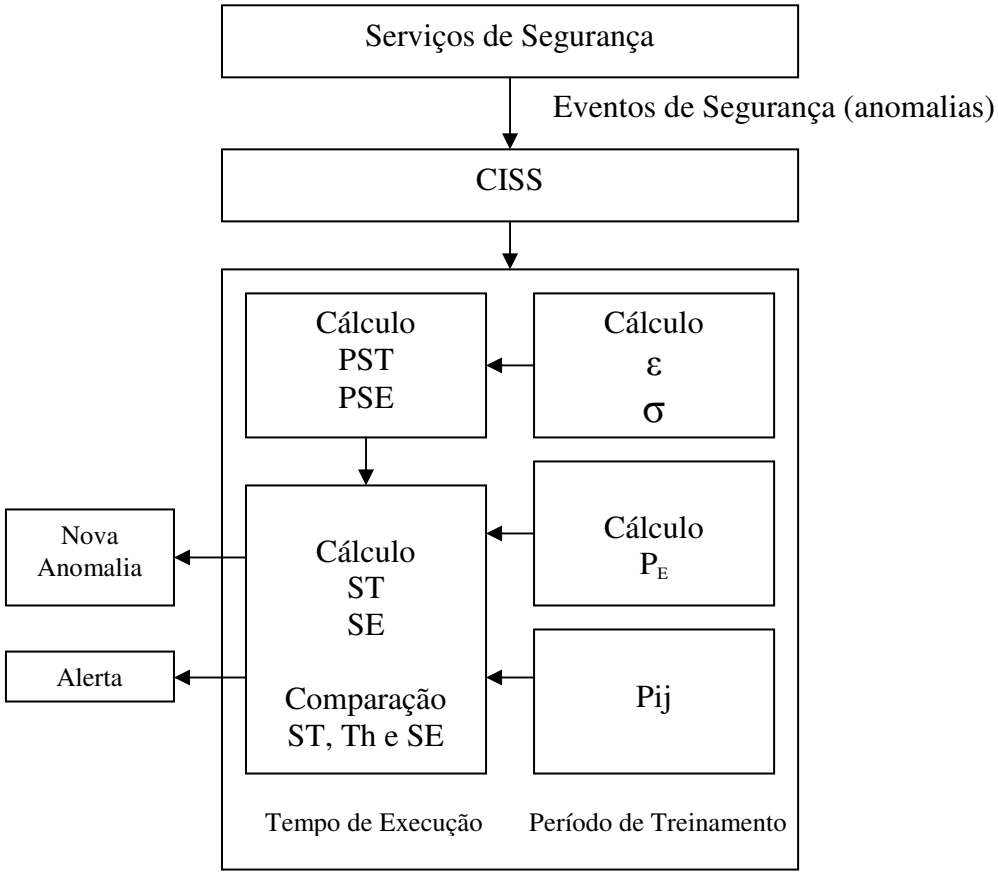

Figura 3.6. ADA – Algoritmo de Detecção de Anomalia da CISS

3.4.3 Problema do Estado X (subMHMM)

Uma situação que pode ocorrer durante o processo de detecção de anomalia é a ocorrência do *Estado X*. Este problema é dado durante a execução em modo normal quando um estado ou transição não prevista ocorre. Ou seja, um evento não é previsto durante o período de

treinamento, mas ocorre em tempo de execução, fazendo com que esse não esteja representado no modelo comportamental.

A ocorrência deste novo evento é possível, porém não deve ser freqüente, pois isso significaria que o modelo comportamental não está robusto o suficiente e, conseqüentemente, o período de treinamento foi insuficiente.

Algumas soluções podem ser encontradas: como atribuir taxas baixas de probabilidade para o *Estado X*, por exemplo, replicando a menor taxa de probabilidade dos estados existentes no modelo comportamental. Essa solução foi proposta por Yasami (2007). Porém, isso não resolve o problema, apenas mascara uma fragilidade do método.

Outra forma é simplesmente desconsiderar a análise, caso houver uma ocorrência para um *Estado X*, o que é novamente uma máscara para o problema. A figura 3.7 ilustra a ocorrência do *Estado X*, onde ocorre uma transição do estado *S2* conhecido, para um estado não definido no modelo comportamental construído durante o período de treinamento.

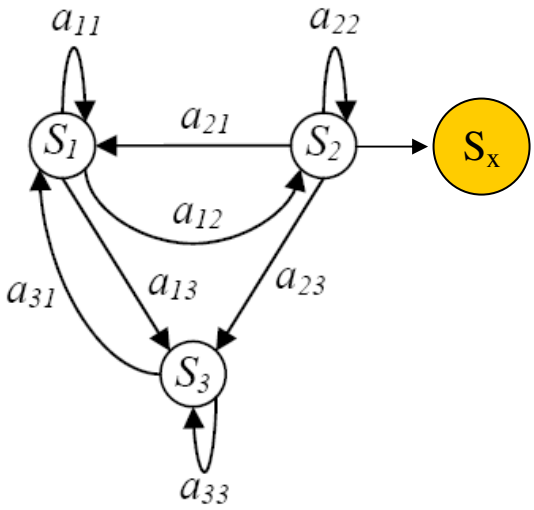

Figura 3.7. Problema do Estado X: Ocorrência em tempo de execução

A CISS utiliza estrategicamente a ocorrência do *Estado X*. Isso significa que o modelo pode ser modificado ou não em tempo de execução, caso regras para ocorrências de *Estado X*

tenham sido criadas e contempladas em tempo de execução. Essa técnica foi chamada de *subMHMM*.

Na CISS, quando uma nova seqüência é encontrada, os alertas são amortizados usando o método de atribuir ao *Estado X* a mais baixa probabilidade encontrada no modelo comportamental. Mas também é feito o registro e a classificação destas ocorrências não previstas em um *subMHMM*.

Caso reincidir uma determinada ocorrência, um novo peso é atribuído, reclassificando a nova seqüência detectada. Regras podem ser criadas para incorporar a nova seqüência anômala (*subMHMM)* ao modelo criado no período de treinamento, como por exemplo, após três reincidências, a nova seqüência é inserida no modelo comportamental MHMM. A seguir, tem-se os pesos atribuídos às medidas básicas e ao ADA para ocorrência do Estado X:

Se Pij, j não existe (chegada ao Estado X)

$Pij = MIN\{P_{ij}\}$

$P_{Ej} = MIN\{ P_{Ej} \}$

Se Pij, i não existe (partida do Estado X)

$Pij = MIN\{P_{ij}\}$

$P_{Ej} = P_{Ej}$ (mantém)

Ainda é possível um outro efeito, onde os estados i e j existem, porém a transição entre estes não existe. Desta forma:

$Pij = MIN\{P_{ij}\}$

$P_{Ej} = P_{Ej}$ (mantém)

Para todos os casos de Estado X ou transição inexistente, o calculo do PST e PSE é definido por:

$$MIN\{t_j^k\} \text{ e } MAX\ \{\sigma^2\}$$

3.5 Modelo Seqüencial

O uso do modelo MHMM na CISS foi uma forma eficaz para integrar informações importantes de segurança, com o intuito de detectar, atuar e prevenir situações anômalas. Este modelo permite alertar sobre possíveis situações anômalas diminuindo a ocorrência de falsos positivos.

No entanto, apesar dos bons resultados gerados com o uso do MHMM, apresentados nos próximos capítulos, o modelo de comportamento baseado em MHMM tem forte dependência do fator tempo. Ou seja, a seqüência de eventos de segurança em um determinado tempo classifica uma anomalia. Assim, uma mesma seqüência de eventos, porém distribuídas de forma diferente ao longo do tempo no MHMM é tratada como anomalias distintas. O que faz com que o período de treinamento tenha que ser exaustivo ou infindável para prever a maioria das variações possíveis de anomalias.

É certo que trabalhos como o descrito na tese de Wei Fan (2001) fazem com que o tempo de treinamento seja reduzido e o modelo comportamental seja enriquecido o suficiente para identificar as anomalias em um sistema computacional. Porém, isso não é um processo determinístico e leva a situações não satisfatórias.

Para contornar essa situação, e em complemento ao MHMM, foi criado o modelo seqüencial. Este modelo despreza o fator tempo e considera apenas a seqüência de eventos

anômalos. Sua estrutura é simples e permite gerar mais um parâmetro para melhor classificar o alerta de anomalias.

O modelo seqüencial pode ser estruturado através de um grafo, onde os estados são as características de uma ocorrência e as transições são os eventos anômalos detectados. O modelo possui duas etapas de execução, assim como o MHMM. O período de treinamento do modelo seqüencial deve ser combinado com o período de treinamento MHMM.

Toda seqüência de eventos anômalos é registrada em um grafo GSA (Grafo de Seqüência Anômalas). Durante o período de treinamento, muitos GSAs são gerados e novos podem ser incorporados no formato de assinaturas de seqüências de ataques. A figura 3.8 ilustra exemplos dos GSAs.

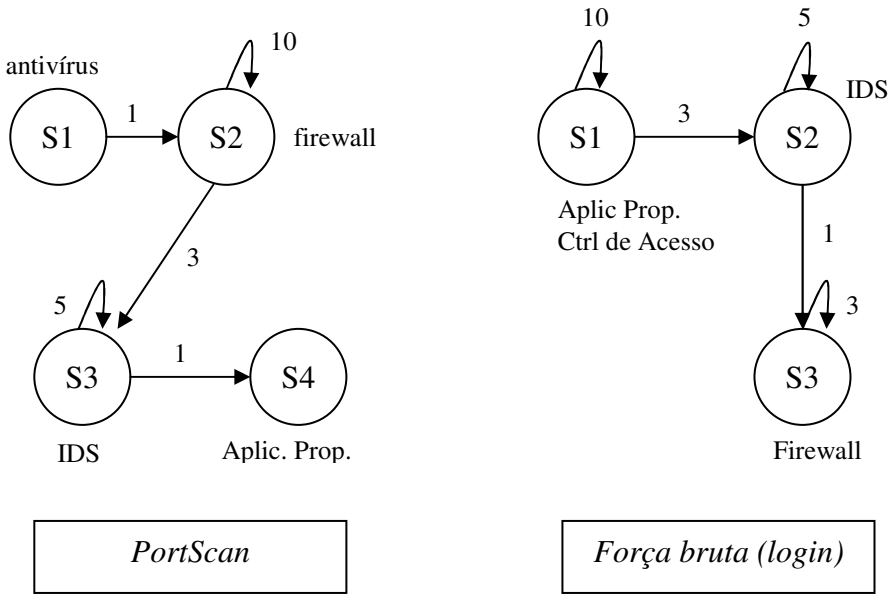

Figura 3.8. GSA: Grafo de Seqüência Anômalas.

As assinaturas são GSAs que abrangem ataques comuns e especificam as ferramentas de segurança, atributos de segurança e ao IP origem da ocorrência. Após o período de treinamento, tem-se não apenas o modelo comportamental MHMM, mas também os grafos de seqüências anômalas.

3.5.1 ADA para o modelo seqüencial

Durante o modo de execução normal, toda ocorrência anômala registrada em tempo de execução será definida como um novo estado e/ou transição, formando o chamado grafo de seqüência total (GST).

O ADA tem como objetivo encontrar um GSA em um GST. Caso isso ocorra, a anomalia é notificada. Essa busca pode ser identificada com uma busca de uma subgrafo em um grafo. A complexidade da busca em profundidade é proporcional ao número de vértices somados ao número de arestas dos grafos, tornando inviável computacionalmente a busca de subgrafos com um número *n* de vértices muito grandes.

Não é o foco desta tese encontrar ou criar o melhor algoritmo de busca. Assim, foi utilizada uma busca em profundidade para encontrar os subgrafos. Na prática, os subgrafos de ataques não são grandes e, com um esforço computacional aceitável, é possível identificar um subgrafo (ver Capítulo 6).

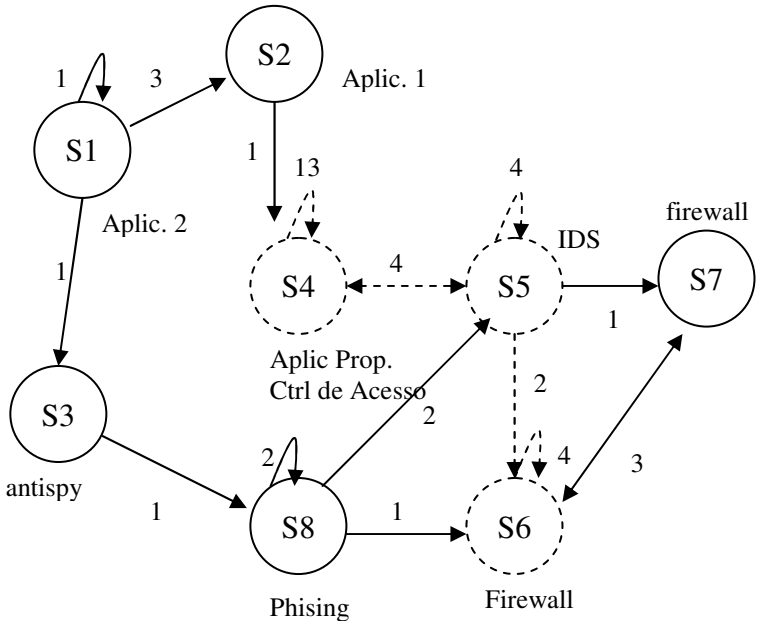

Figura 3.9. Identificação de um subgrafo em um grafo

A figura 3.9 mostra a identificação de um subgrafo orientado G(X) (tracejado) em um grafo G(Y). A anomalia identificada foi a armazenada no GSA exemplo (*portscan*) apresentado na figura 3.8.

3.5.2 Modelo Seqüencial Direto

O modelo direto permite que o administrador do sistema computacional construa suas regras de detecção e atuação conforme seus conhecimentos e experiência sobre detecção de anomalias.

Este modelo pode induzir a criação de regras inconsistentes que podem prejudicar o funcionamento normal do sistema e ainda não protegê-lo de situação de risco. Contudo, o modelo direto permite a personalização do sistema de segurança para especificidades de um determinado sistema computacional.

3.6 Considerações Finais do Capítulo

Nos capítulos 1 e 2 desta tese foram apresentados o propósito e a aplicação da Camada de Integração de Serviços de Segurança (CISS). No capítulo corrente, foram detalhados os aspectos matemáticos e estruturais que compõem a CISS. Os resultados obtidos por meio da implementação em hardware e software da CISS são apresentados e discutidos nos capítulos posteriores.

O modelo de informações para armazenar dados sobre a “saúde” de um sistema computacional foi o primeiro foco deste trabalho. Foram encontrados modelos que muitas vezes atendiam parcialmente a descrição de uma anomalia ou mesmo modelos específicos para alguns tipos de anomalias.

Uma vez definido o modelo de informações utilizado, o mesmo deve ser adotado por todas as ferramentas de segurança, forçando a existência de um padrão que não está em processo de normatização e dificilmente seria adotado por ferramentas comerciais.

Por meio da revisão bibliográfica de publicações na área, foi encontrada uma RFC proposta para normalizar a troca de informações de eventos de segurança em um sistema computacional, a RFC 4765 (IDMEF). Ao analisar o IDMEF, foi constatado que o mesmo poderia atender uma gama diversificada de serviços de segurança, uma vez que discrimina em seu modelo de informações dados sobre uma anomalia de forma robusta e organizada. Assim, este foi adotado como modelo de dados para CISS.

Algumas personalizações foram realizadas principalmente em campos do modelo IDMEF que prevê de forma flexível e adaptativa tais personalizações. Um exemplo foi o uso do campo *AdditionalData* para expressar o tipo de atributo de segurança que notificou a anomalia.

Neste contexto, uma vez organizadas as informações referentes a eventos de segurança, foi definida uma aplicabilidade para integrar e utilizar estas informações sobre a "saúde" de um sistema computacional.

Desta forma, foram criados modelos comportamentais para identificar anormalidades de forma prematura e precisa. O HMM modificado (MHMM) foi escolhido como o principal mecanismo para modelar o comportamento anormal e normal de uma seqüência de eventos de segurança.

O MHMM teve como base o modelo descrito por Yasami (2007). Este foi adaptado e modificado para atender os requisitos específicos de modelagem do comportamento de eventos de segurança ao longo do tempo.

Uma das dificuldades encontradas durante os testes iniciais do MHMM foi modelar durante o período de treinamento diferentes situações anômalas para que o sistema pudesse atingir resultados satisfatórios.

Wei Fan (2001) tratou exatamente de melhorar e enriquecer modelos comportamentais, utilizando a técnica de geração de ataques artificiais. O impacto foi a diminuição do período de treinamento e aumento da robustez do modelo comportamental gerado. Esta técnica foi aplicada apenas na implementação simulada da CISS (ver Capítulo 6).

Ainda neste foco, foi criado o modelo seqüencial. Este modelo é construído de forma simultânea ao modelo MHMM e tem como principal intuito melhorar a detecção de ataques não previstos no modelo MHMM. Os efeitos são satisfatórios, conforme apresentado nos próximos capítulos.

CAPÍTULO 4

IMPLEMENTAÇÃO EM HARDWARE (CISS-SOC)

Neste capítulo, apresenta-se a implementação da CISS em um System-On-Chip (SoC), chamado de CISS-SoC. Desde os modelos MHMM e Seqüencial até os algoritmos de detecção de anomalias e os serviços de segurança implementados estão embarcados no CISS-SoC. Assim, são apresentados detalhes da arquitetura implementada, a análise de desempenho e, por fim, eficiência na proteção do sistema.

4.1 Arquitetura Top-Level da CISS-SoC

Por meio de um *software* embarcado e componentes dedicados em *hardware*, foi possível criar um ambiente que contém serviços de segurança como Firewall, Antivírus, Criptografia, Autenticação e IDS, integrando-os com a CISS-SoC.

A CISS-SoC não foi projetada e implementada com o intuito de conceber um sistema de segurança completo. Neste caso, o objetivo principal foi a criação de uma versão SoC da CISS que pode ser utilizada em sistemas mais robustos de segurança. Para avaliar e testar a versão da CISS, foram criados serviços básicos de segurança, os quais são apresentados na seção 4.2.

A versão SoC da CISS foi implementada em um FPGA Virtex II Pro (XC2VP30) (XILINX, 2007) utilizando um processador embarcado PowerPC405 (XILINX, 2008a), software em linguagem C, módulos dedicados em VHDL e sistema operacional Linux 2.6.0 .

Esta é capaz de analisar o comportamento do SoC com base nos modelos MHMM e Seqüencial visto no Capítulo 3. Na seqüência tem-se a descrição da arquitetura e análise de desempenho e eficiência do sistema proposto.

A arquitetura do CISS-SoC é composta por módulos dedicados em hardware como criptografia e firewall, interface de comunicação UART, GPIO e Ethernet/EMAC 10/100. Os serviços de antivírus e IDS foram implementados em software embarcado e são executados pelo PowerPC 405 embarcado.

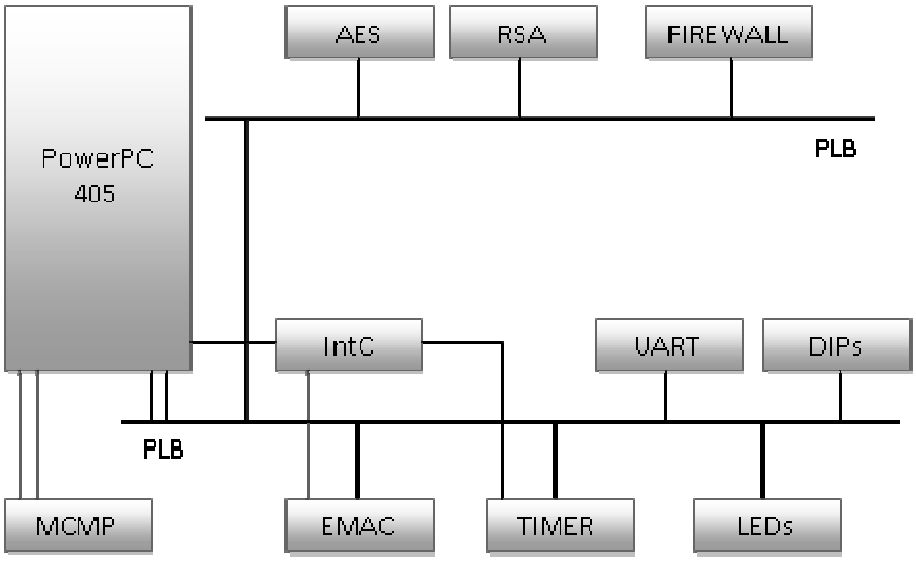

Figura 4.1. Arquitetura Top-level do CISS-SoC

O fluxo de execução é controlado pelo PowerPC que acessa a memória RAM de 512 Mbytes externa ao FPGA por meio do controlador MPMC (Multi Port Memory Controller) e o barramento de comunicação com os módulos dedicados PLB (*Processor Local Bus*), como pode ser observado na figura 4.1.

Os componentes em hardware dedicado do CISS-SoC podem ser visualizados na figura 4.1. É possível distinguí-los em duas categorias:

- Hardware dedicado de Segurança: são os módulos de Firewall, RSA e AES apresentados nas seções 4.2.1 e 4.2.4;
- Hardware de controle e interface: são os módulos MCMP, EMAC, TIMER, IntC, UART, DIPs e LEDs, detalhados na seção 4.2.6.

O software embarcado de controle executa sobre o *kernel* do sistema operacional Linux 2.6.0, herdando recursos como, semáforos, controle de interrupções, interfaces básicas, controle de acesso a memória e robustez de uma forma geral. Alguns serviços de segurança foram descritos em software embarcado, destacando os serviços de antivírus, autenticação por *login* e IDS que são apresentados na seqüência.

4.2 Módulos de Segurança

Os módulos de segurança presentes no CISS-SoC foram selecionados com o intuito de abranger funcionalidades básicas de serviços de segurança de um sistema computacional. No entanto, a implementação destes serviços não é o foco principal desta tese. Neste contexto, foram criadas versões leves, ou seja, simplificadas e funcionais de cada serviço proposto.

O objetivo é apenas ter disponíveis funcionalidades básicas de cada serviço de segurança. Sendo importante salientar que a função dos serviços de segurança neste trabalho é gerar ocorrências de anomalias para CISS-SoC analisar e notificar uma situação anômala. Na seqüência, uma descrição das características e funcionalidades de cada serviço de segurança.

4.2.1 Firewall

Firewall é o nome dado ao dispositivo de uma rede de computadores que tem por objetivo aplicar uma política de segurança a um determinado ponto de controle da rede. Sua função

consiste em regular o tráfego de dados entre redes distintas e impedir a transmissão e/ou recepção de acessos nocivos ou não autorizados de uma rede para outra (TANENBAUM, 2003).

Desta forma, o *firewall* provê diretamente o serviço de controle de acesso e indiretamente o serviço de disponibilidade, pois este poderá impedir ataques maliciosos com o intuito de prejudicar ou interromper os serviços de uma rede. Pode-se classificar os *firewalls* basicamente em dois tipos (TANENBAUM, 2003):

- Nível de pacotes: neste tipo de *firewall,* as decisões são baseadas nos parâmetros do pacote, como: porta/endereço de origem/destino, estado da conexão, protocolos e outros. Assim, pode-se bloquear, permitir e em alguns casos, redirecionar um pacote conforme as regras predefinidas; Este tipo foi o implementado nesta versão da CISS.
- Nível de aplicação: analisa o conteúdo do pacote para tomar suas decisões de filtragem. Isso o torna mais intrusivo (pois analisa o conteúdo de todas as aplicações) e permite um controle relacionado com o conteúdo do tráfego.

O módulo de firewall inibe uma comunicação de entrada e/ou saída do CISS-SoC conforme as regras implementadas. Existem inúmeros códigos maliciosos que utilizam portas e interfaces de comunicações disponíveis para transitar informações não autorizadas capturadas da máquina hospedeira. Uma das formas de bloquear esse tipo de invasão é a utilização de um firewall com regras bem definidas para um determinado ambiente e/ou políticas de segurança de uma organização.

O firewall implementado no CISS-SoC é composto basicamente por um buffer de entrada, um analisador de regras, um buffer de pacotes analisado, registradores de controle, além de fazer acesso à memória RAM e ao barramento de comunicação PLB (*Processor*

Local Bus). Na seqüência, tem-se a descrição do funcionamento básico das unidades de Firewall:

1. Firewall recebe do barramento de comunicação PLB um *frame* contendo IP, MAC e PORT armazenando estes dados no buffer de entrada;
2. Identifica o pacote como de saída ou entrada no sistema;
3. O módulo Analisador verifica o buffer de pacotes analisados. Se o pacote atual é reincidente, pode-se decidir o bloqueio ou permissão deste ainda neste estágio, senão segue para a próxima etapa.;
4. O Analisador busca na memória RAM as regras de restrição e permissões criadas pelo usuário;
5. O Analisador compara as regras com os dados armazenados no buffer de entrada e julga se o pacote será aceito ou bloqueado;
6. Finalmente é gerada a saída, bloqueando ou não o pacote. Retornar a etapa 1.

A reincidência de pacotes bloqueados é tratada para melhorar o desempenho. Uma vez analisado um pacote, este é enviado para o buffer de pacotes analisado. Caso haja a reincidência do pacote, este será bloqueado/permitido sem realizar a análise das regras novamente. Se o conjunto de regras for modificado, a lista de pacotes bloqueados é excluída.

As regras do Firewall são inseridas na memória RAM, onde a ordem de inserção é muito importante, pois esta seqüência irá determinar a prioridade de cada regra. A regra armazenada na última posição terá prioridades sobre as demais. A 4.2 apresenta o formato das regras:

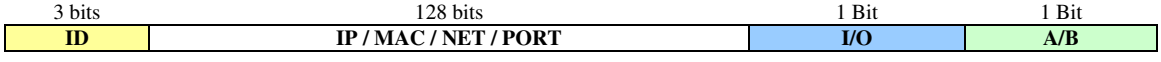

Figura 4.2. Formato das regras do Firewall

Onde,

ID: Identifica a Regra (1-IP, 2-MAC, 3-NET, 4-PORT, 5-AlowAll, 6-BlockAll).

IP / MAC / NET / PORT: Descreve o IP, MAC, NET ou PORT que será filtrado.

I/O: Define se a regra será para pacotes de entrada(1) ou de saída (0).

A/B (Allowed/Blocked) define se é uma regra de bloqueio(1) ou permissão(0).

O módulo de firewall foi implementado inicialmente em software embarcado e posteriormente convertido para hardware dedicado. Entre as versões deste módulo foram feitos testes e análise de reconfiguração parcial das regras de permissão. Estes resultados podem ser encontrados nos artigos publicados em (PEREIRA, ORDONEZ, 2008a), (PEREIRA, ORDONEZ, 2008b) e (PEREIRA, ORDONEZ, 2007).

As principais contribuições dos trabalhos citados anteriormente resumem-se em um firewall com capacidade de reconfiguração parcial, onde o módulo reconfigurado trata-se da memória responsável por armazenar as regras de bloqueio e permissão de pacotes.

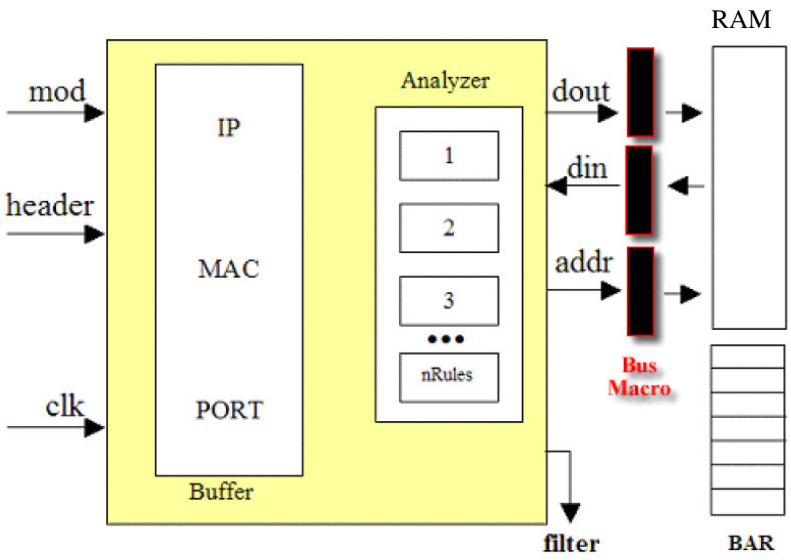

Figura 4.3. Arquitetura *top-level* do Firewall Reconfigurável (PEREIRA; ORDONEZ, 2008a)

A figura 4.3 apresenta a arquitetura *top-level* do firewall reconfigurável. Este é capaz de analisar e decidir sobre o bloqueio ou permissão de um pacote com base em regras armazenadas na memória RAM. O acesso à memória RAM é controlado por *bus macros* (LIM; PEATTIE, 2002), as quais são utilizadas para o isolamento da área reconfigurável quando é feita a reconfiguração parcial do dispositivo.

4.2.2 Antivírus

Os antivírus são programas que procuram detectar e, então, anular ou remover os vírus de computador (CERT, 2007). O módulo de antivírus analisa tráfego de pacotes enviado para o CISS-SoC por HTTP em busca de padrões de seqüência que identificam um possível vírus digital. Esses padrões são chamados de assinaturas.

Nesta tese, algumas assinaturas de possíveis vírus foram criadas artificialmente para configuração de uma base de assinaturas. O antivírus foi concebido plenamente em software embarcado e apenas identifica o vírus, ou seja, não elimina o mesmo.

O algoritmo de detecção de vírus está representado pelo código descrito na seqüência:

```
Captura pacote HTTP / UART
Separa campo de dados
Se HTTP então {
   Decifra campo de dados
}
nro_bytes = contaBytes(campo_dados)
nro_assinaturas = 0
se  nro_bytes = MAX_NRO_BYTES_PACOTE então {
    enquanto nro_assinaturas <= MAX_NRO_ASSIN então {
                    se assinatura <> campo_dados
                          IDMEF (ANTIVIRUS, ASSINATURA_DETECTADA)
                    Incrementa (nro_assinaturas)
     }
```

```
senão {
       IDMEF (ANTIVIRUS, PACOTE_INVALIDO)
}
```

4.2.3 Autenticação

O módulo de autenticação através de *login* permite que um conjunto de usuários faça acesso ao CISS-SoC. Tentativas inválidas de *login* são registradas na CISS-SoC. Uma vez autenticados, os usuários com permissão completa podem bloquear o acesso de outros usuários de menor prioridade. Este serviço de segurança foi implementado em linguagem C, tornando-se um software embarcado no CISS-SoC.

O controle de acesso por *login* é um método simples para inibir pessoas ou serviços não autorizados de fazer acesso a funcionalidades ou arquivos de um sistema computacional. No entanto, esta barreira de proteção sofre inúmeros tipos de ataques, pois uma vez autenticado o usuário, seja este verdadeiro ou não, terá acesso a uma área restrita do sistema.

Neste contexto, é importante que não apenas ferramentas especialistas em segurança notifiquem a ocorrência de uma anomalia, mas também qualquer software ou hardware que podem ser alvo de tentativas de ataques. Na seqüência, o pseudocódigo que representa o algoritmo de autenticação simplificado.

```
Captura pacote UART
Separa campo de dados
Identifica usuario e senha
nro_bytes_senha = contaBytes(senha)
nro_usuario = 0
se  nro_bytes_senha > MIN_NRO_BYTES_SENHAS então {
    enquanto nro_usuario <= MAX_NRO_USUARIO então {
                    se lista.usuario(nro_usuario) = usuário então {
                         se lista.senha(nro_usuario) <> senha então {
```

```
                        IDMEF (AUTENTICAÇAO, SENHA_INCORRETA)
                        break
                    }
                }
    }
     IDMEF (AUTENTICAÇAO, USUARIO_INCORRETO)
senão {
     IDMEF (AUTENTICACAO, SENHA_INVALIDA)
}
```

4.2.4 Criptografia

A criptografia é um dos serviços de segurança mais importante, pois pode garantir a confidencialidade, irretratabilidade, integridade, entre outros atributos de segurança. O módulo criptografia do CISS-SoC é constituído de dois dos principais algoritmos simétricos e assimétricos, o AES e RSA, respectivamente.

4.2.4.1 O Algoritmo AES

O AES (*Advanced Encryption Standard*) (FIPS-197, 2001) é o algoritmo de criptografia simétrica substituto do DES (*Data Encryption Standard*) (FIPSPUB-46, 1977). O AES possui características que promovem maior proteção a informações e um bom desempenho quando implementado em software ou hardware (DAEMEN, et al, 2002).

O AES foi implementado nesta tese como um módulo dedicado descrito em VHDL. Todo o frame recebido por HTTP é decifrado pelo módulo AES que possui uma chave simétrica já programada. A especificação completa do AES pode ser encontrada no FIPS-197 (*Federal Information Processing Standard*). O modo utilizado foi o CBC (Cipher-Block Chaining)

Para validar o frame cifrado, o primeiro bloco (128 bits), deve sempre ser incrementado e cifrado a cada transmissão. Caso o processo de decifragem não resultar no sucessor ao número anteriormente decifrado será gerado uma notificação ao CISS-SoC. A falha na contagem ocorrerá por perda de sincronismo, tentativa inválida de transmissão de informações ou ainda o uso de outra chave simétrica não autorizada.

4.2.4.2 O algoritmo RSA

O RSA (RIVEST, 1978) é um algoritmo de chave assimétrica criado por Ronald **R**ivest, Adi **S**hamir e Leonard **A**dleman. Este utiliza números primos para compor suas chaves públicas e privadas.

O RSA envolve um par de chaves, uma chave pública que pode ser conhecida por todos e uma chave privada que deve ser mantida em sigilo. Toda mensagem cifrada usando uma chave pública só pode ser decifrada usando a respectiva chave privada. A seguir o algoritmo de RSA para cifrar e decifrar mensagens.

Para transformar uma mensagem m em uma mensagem c cifrada usando a chave pública do destinatário n e e deve-se fazer uma potenciação modular apresentada na fórmula 4.1.

$$c = m^e \mod n \tag{4.1}$$

Para recuperar a mensagem m da mensagem cifrada c usando a respectiva chave privada do receptor ne d, deve-se fazer a potenciação modular descrita na fórmula 4.2.

$$m = c^d \mod n \tag{4.2}$$

Como pode ser observado, o algoritmo é relativamente simples, onde a principal dificuldade é executar a potência modular. Existem inúmeros algoritmos para executar com eficiência a potência modular utilizada no RSA. Nesta tese foi adotado o algoritmo de *Montgomery* para multiplicação modular. Este algoritmo pode ser encontrado com facilidade na literatura (MONTGOMERY, 1985). Os resultados da implementação do RSA pode ser visualizados nas seções 4.6.1 e 4.6.2.

4.2.5 IDS

Um sistema de detecção de intrusão (IDS) tem como um dos objetivos principais analisar as atividades do sistema e assim detectar caso haja alguma tentativa de invasão ou mal uso do mesmo. Geralmente o IDS executa constantemente em *background* e só notifica o usuário quando detectar algo suspeito ou ilegal. O esquema de se detectar uma anomalia no sistema varia amplamente, porém o objetivo final do IDS é capturar os infratores no momento antes de realmente danificarem os recursos do sistema (WHITMAN, MATTORD, 2009).

Um IDS protege um sistema de ataques, uso incorreto e danos, mas também pode monitorar as atividades da rede, auditar as configurações da rede e do sistema para detectar vulnerabilidades, analisar integridade de dados, entre outras funcionalidades. Dependendo dos métodos de detecção configurados, há diversos benefícios diretos e casuais em utilizar um IDS.

O módulo de identificação de intrusão (IDS) do CISS-SoC é um software embarcado implementado em linguagem C. Este foi concebido contra ataques conhecidos como negação de serviços e *portscan*.

O IDS proposto e implementado é muito simplificado, sendo capaz de identificar poucas situações anômalas, estando mais para um emulador simplificado de IDS do que um

sistema de identificação de intrusão real. É importante ressaltar que o módulo IDS é um serviço de segurança fundamental para a validação da CISS e que nas implementações em software (ver capítulos 5 e 6) este módulo tem uma participação de mais destaque.

4.2.6 Interfaces com o CISS-SoC

Para a interação do controle central de fluxo do CISS-SoC com os módulos dedicados e interfaces externas do chip, foram criados meios de comunicação como UART, interface EMAC 10/100, interface GPIO para entrada de switches de configuração, Leds de sinalização e interface com a memória RAM. Na tabela 4.1, tem-se a descrição sucinta das características destas interfaces e suas principais funcionalidades.

Tabela 4.1 - Módulo de interface e comunicação do CISS-SoC

Interface	Características	Funcionalidades
UART	9600 bps, sem paridade	A UART recebe tentativas de *login* na CISS-SoC para recuperação dos modelos comportamentais e outras informações geradas em tempo de execução
GPIO1	7 Switches	Estas chaves têm a função de inicializar o chip (*reset*) e ativar individualmente cada serviço de segurança (*enables*).
GPIO2	5 Leds	Tem a função de apontar o estado de cada serviço de segurança como ativado ou desativado e sinalizar que o chip está em atividade.
MCMP	Acesso a um espaço de memória de 512 Mbytes.	MCMP (*Multi-Controle Memory Port*) que faz a interface entre a memória RAM de 512 Mbytes e o processador PowerPC
EMAC 10/100	Velocidade de acesso limita-se a 10/100 MBits.	Define as camadas iniciais do protocolo Ethernet, onde se pode configurar um endereço MAC de acesso permitindo a comunicação através de uma rede de computadores convencional.
IntC	Utilizado por IPs que necessitam dessa função como o EMAC 10/100.	IntC (Xilinx Interrupt Control) é responsável pelo controle de interrupções permitindo que módulos distintos possam interromper o fluxo do programa

		principal do processador PowerPC, para o tratamento de um evento.
Timer	Criação de escalas de tempos	Temporizador utilizado por IPs e PowerPC para criar e definir escalas de tempo.
PLB	Barramento customizável	A integração dos PowerPC com os módulos dedicados é feito através de um barramento comum, chamado de PLB (*Processor Local Bus*). Para controlar o acesso ao barramento existe um árbitro de acesso, evitando conflitos e informações colisões. Embora exista um barramento comum, os módulos não são capazes de comunicar entre si. Assim, haverá sempre a intervenção de software que está sendo executado no PowerPC.

4.3 Bibliotecas Embarcadas no SoC

O software embarcado foi estruturado em conjunto de bibliotecas que têm funcionalidades específicas e implementadas para interagir com módulos dedicados de segurança e interfaces de comunicação. Na seqüência, uma breve descrição das funcionalidades de cada biblioteca:

- crypto.h: invoca as funções e parâmetros para a criptografia baseadas inicialmente nos algoritmos AES e RSA, onde é possível gerar chave, sub-chaves, definir o modo de funcionamento, cifrar, decifrar, entre outras funcionalidades;
- firewall.h: possui funções e parâmetros para inserir, excluir, alterar e buscar regras em uma tabela de regras na memória RAM;
- ids.h : permite operações para inserir, excluir, alterar e buscar padrões de anomalia em uma tabela na memória RAM.;
- audit.h: possui funções de criação de logs e depuração de falhas;
- table.h: aloca estrutura em formato de uma tabela em memória e implementa funções de manipulação e consulta. Esta é utilizada por outras bibliotecas para a criação de tabelas customizadas;

- xml.h: possui funções de um *parser* para que as informações geradas pelo sistema de segurança estejam no formato XML (IDMEF);
- ciss.h: organiza o fluxo de execução e tratamento de exceções do CISS-SoC;
- ada.h : algoritmos de detecção de anomalias do modelo MHMM e seqüencial;
- net.h: interface com o módulo EMAC para extração de informações dos pacotes de dados;
- uart.h: recebe comandos de autenticação, leitura e escrita de regras e modelos comportamentais.

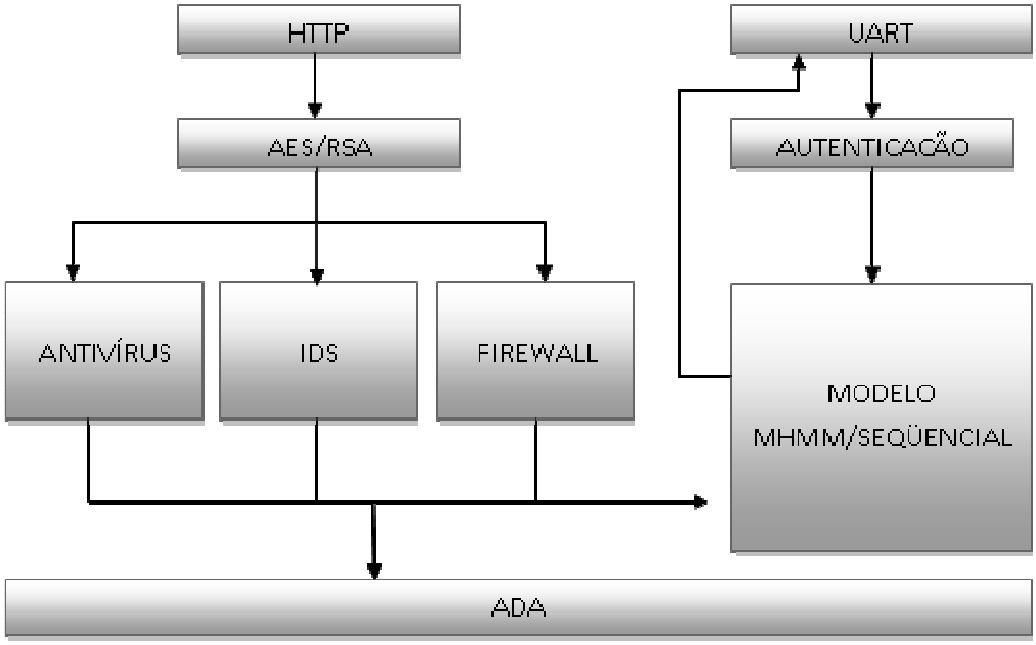

Figura 4.4. Diagrama de fluxo de dados do CISS-SoC.

Estas bibliotecas são utilizadas pela aplicação embarcada que executam sobre o sistema operacional Linux as funções de políticas de segurança do CISS-SoC. Na figura 4.4, tem-se o diagrama de fluxo de dados principal do CISS-SoC.

Todas as informações transmitidas para o CISS-SoC devem ser cifradas utilizando o algoritmo AES ou RSA, utilizando as chaves apropriadas. Desta forma, quando um pacote HTTP é recebido pelo CISS-SoC, este deve ser decifrado pelos módulos dedicados de

criptografia. Se o processo for completado com sucesso, as informações são transferidas simultaneamente para os módulos de firewall, antivírus e IDS.

Caso um ou mais módulos acusar alguma anomalia, esta será registrada no modelo comportamental MHMM e seqüencial, durante o período de treinamento ou será analisada pelos algoritmos de detecção de anomalias (ADA) em modo de execução normal.

Simultaneamente, dados do modelo comportamental ou dados de execução em modo normal podem ser lidos via UART mediante um *login* válido.

4.4. Módulo Gerador de Ocorrências.

O módulo gerador de ocorrência foi implementado em um segundo FPGA Spartan3 XC3S700AN(XILINX, 2008b) que transmite informações ao CISS-SoC por meio de HTTP ou UART. Os pacotes são montados por um software embarcado que faz uma leitura de rotinas de ataques que estão armazenadas na memória RAM e transmite via HTTP ou UART. Na figura 4.5, apresenta o CISS-SoC e a comunicação com o módulo gerador de ocorrências.

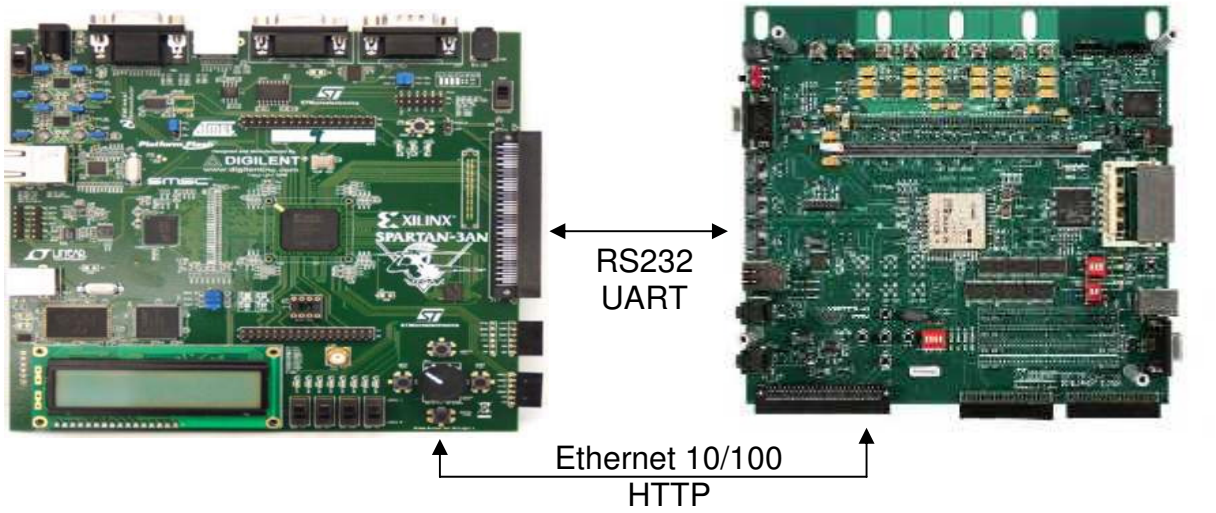

Figura 4.5. CISS-SoC e Módulo Gerador de Ocorrências

A interface UART é utilizada para fazer *login* no CISS-SoC para ter acesso ao modelo comportamental gerado em período de treinamento e informações sobre anomalias em modo de execução normal.

As rotinas de ataques foram descritas conforme a capacidade detecção de cada serviço de segurança descrito no CISS-SoC. Como os serviços são limitados, não é possível criar rotinas de ataques sofisticadas. A seguir, as principais categorias de ataques geradas:

- ataque de login: tentativa de *login* no CISS-SoC sem sucesso via UART.
- ataque por vírus: pacotes de dados com seqüências anômalas identificadas como vírus pelo CISS-SoC.
- IP bloqueados: tentar acessar o CISS-SoC através de um dispositivo que contém um IP bloqueado.
- Ataques conhecidos: gerar seqüências de ataques conhecidas para atuação do IDS como, por exemplo: “ping da morte” e *portscan*.
- Mensagens inválidas: gerar mensagens que não podem ser decifradas pelo módulo de criptografia instalado no CISS-SoC.

Estes foram os principais ataques gerados por este módulo. As rotinas de ataques consistem basicamente em gerar ataques aleatórios ou estruturados. Durante o período de treinamento, variações de rotinas de ataques ao longo do tempo foram criadas para construir um modelo comportamental satisfatório (ver seção 4.5).

4.5 Taxas de ocupação e propagação do CISS-SoC.

Nesta seção são apresentados dados de desempenho e ocupação física do CISS-SoC em relação à área total disponível no FPGA adotado. O FPGA adotado para a implementação do

CISS-SoC foi um XC2VP30, contendo dois processadores PowerPC *hardcore*s embarcados e uma área programável contendo 30816 células lógicas (XILINX, 2007).

4.5.1 Taxa de ocupação no FPGA

A taxa de ocupação em relação ao total de componentes disponíveis nesta família de FPGA pode ser visto na tabela 4.2. As métricas utilizadas são o número de Slices, LUTs e Flip-Flops consumidos.

Tabela 4.2 – Taxa de ocupação dos módulos dedicados de segurança do CISS-SoC

Módulo	**Slices**	**LUTs**	**Flip-Flops**
Firewall	859 (6%)	1612(6%)	1436(5%)
AES	1403(10%)	2815(10%)	4021(13%)
RSA	2871(20%)	5704(20%)	4267(13%)

Como pode ser observado na tabela 4.2, o módulo RSA possui a maior taxa de ocupação quando comparado com os demais módulos. Isto era esperado dado sua robustez e a necessidade de um multiplicador modular. O módulo de segurança AES consumiu 10% da área total e quando comparado a outras implementações, como as implementações de Zambreno (2004), com área minima de 2518 slices e de Chang (2007) com 122 slices, a implementação do AES do CISS-SoC pode ser considerada satisfatória.

As diferenças observadas em relação a área são motivadas pelo propósito da aplicação do AES que geralmente resume-se em ganho de desempenho ou área. A implementação de Chang (2007) realmente poupa área, porém tem a capacidade de cifrar apenas 2,18 Mbps, enquanto na versão de Zambreno (2004) a vazão atinge 12,18 Gbps.

A área total consumida pelo CISS-SoC é maior, pois foram adicionados os IP cores de comunicação e debug relacionados na tabela 4.1. A síntese, mapeamento e roteamento final apontam que o CISS-SoC ocupou cerca de 74% da área programável consumindo 10.112 Slices, 20.345 LUTs e 18.456 Flip-Flops. O resultado alcançado permite melhorar os serviços criados ou ainda implementar outros serviços de segurança para testes e validações da versão da CISS em SoC.

4.5.2 Taxa de Propagação

A taxa de propagação do circuito é uma das principais métricas para avaliar seu desempenho. Na tabela 4.3, tem-se a taxa de propagação total de cada módulo de segurança dedicado e a taxa total do CISS-SoC.

Tabela 4.3 – Taxa de propagação dos módulos dedicados de segurança do CISS-SoC.

Módulo	**Tempo de Propagação**	**Freqüência**
Firewall	11,214 ns	89,174 Mhz
AES	6,514 ns	153,515 MHz
RSA	25, 369 ns	39, 418 Mhz

Na tabela 4.3, o algoritmo AES obteve a maior taxa de propagação quando comparado com os demais blocos. Novamente esta taxa pode ser comparada com as implementações de Zambreno (2004), onde as taxas médias de propagação anotadas foram de 7,5 ns, demonstrando o bom desempenho da versão do AES do CISS-SoC.

No caso do Firewall e IDS, que fazem acesso a memória RAM para executar a procura de padrões ou regras, o desempenho está diretamente relacionado ao volume de regras e padrões

armazenados na memória. Para os testes realizados, os módulos citados foram capazes de analisar todo o fluxo de dados criados pelo módulo gerador de ocorrências.

4.5.3 Utilização dos Processadores Embarcados

O impacto na capacidade de processamento tem relação direta com os módulos e serviços de segurança ativados e o tipo de ataque que está sendo realizado. Ataques seqüenciais do tipo rajada podem estressar o sistema e elevar a taxa de utilização dos módulos de processamento dedicado e do processador embarcado.

A figura 4.5 apresenta a porcentagem de utilização dos módulos de processamento mais requisitados durante os testes do CISS-SoC em modo normal de funcionamento. É possível perceber que existe uma sobrecarga do processador embarcado que possui uma taxa de utilização de 87%.

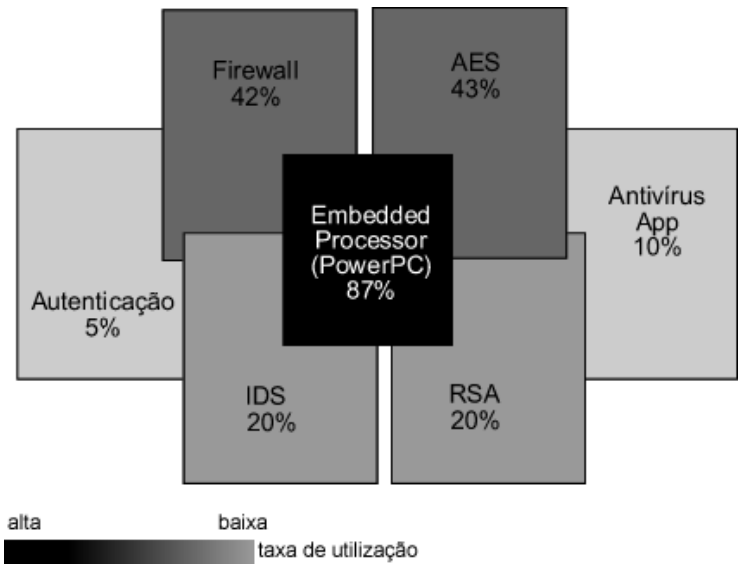

Figura 4.5. Taxa de utilização com um processador embarcado

Neste contexto, para melhorar o desempenho do sistema, o segundo processador embarcado presente no FPGA Virtex foi ativado, dividindo a carga de tarefas. Assim, o

modelo de arquitetura para essa versão possui dois processadores *hardcore* embarcados e ativos. O impacto desta modificação podem ser notadas na figura 4.6. Neste caso, foi possível o uso dos dois processadores, pois muitas tarefas são independentes, o que permitiu a execução em paralelo.

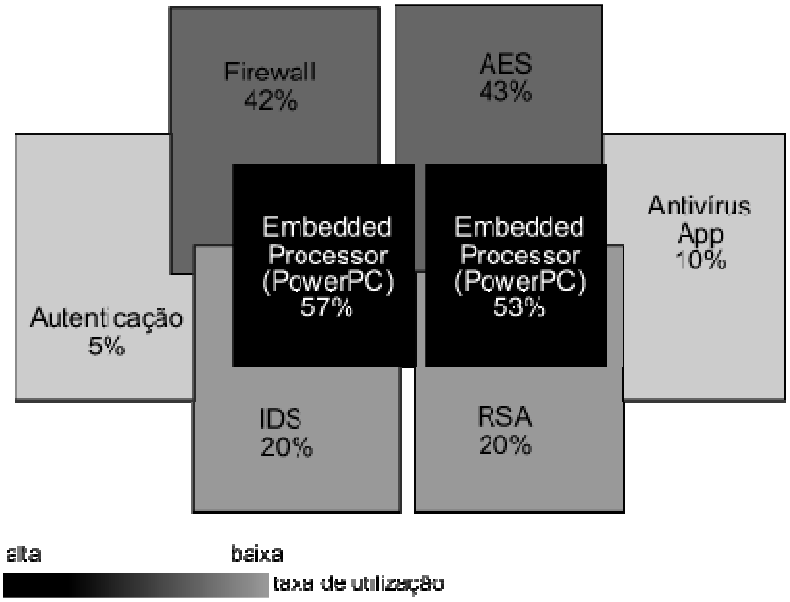

Figura 4.6. Taxa de utilização com dois processadores embarcados

A taxa média de utilização foi calculada utilizando o número de chamadas a um determinado módulo e o tempo de execução de cada rotina solicitada. No entanto, a média de utilização pode mascarar o tempo de sobrecarga do sistema. Com um único processador embarcado, este permanece com carga máxima de processamento durante 25% do tempo total analisado. Com utilização de dois processadores, essa taxa foi reduzida para 4%.

4.6 Modelo Comportamental em Hardware

O Modelo comportamental criado no período de treinamento foi relativamente simples, uma vez que o número de serviços de segurança disponível e sua simplicidade não permitem muitas variações de ataques e defesas.

No entanto, a estrutura de armazenamento e criação de modelos comportamentais baseados no MHMM e no modelo seqüencial, assim como os algoritmos de detecção de instrução (ADA) foram implementados, analisados e validados. Detalhes da metodologia de testes podem ser visualizados na seção 7.2.

4.6.1 Resultados do período de treinamento

Na tabela 4.4, têm-se dados dos modelos comportamental gerados no período de treinamento. O período de treinamento teve duração de 3 horas. Durante este período, o módulo gerador de ocorrências produziu variações de ataques ao longo do tempo.

Tabela 4.4 - Dados do modelo comportamental após período de treinamento.

Informações	Módulo	Quantidade
Número de Seqüências de Ataques	Gerador de ataques	150
Número de Estados	MHMM	30
Número de Transições	MHMM	49
Número de Seqüências	Modelo Seqüencial	129

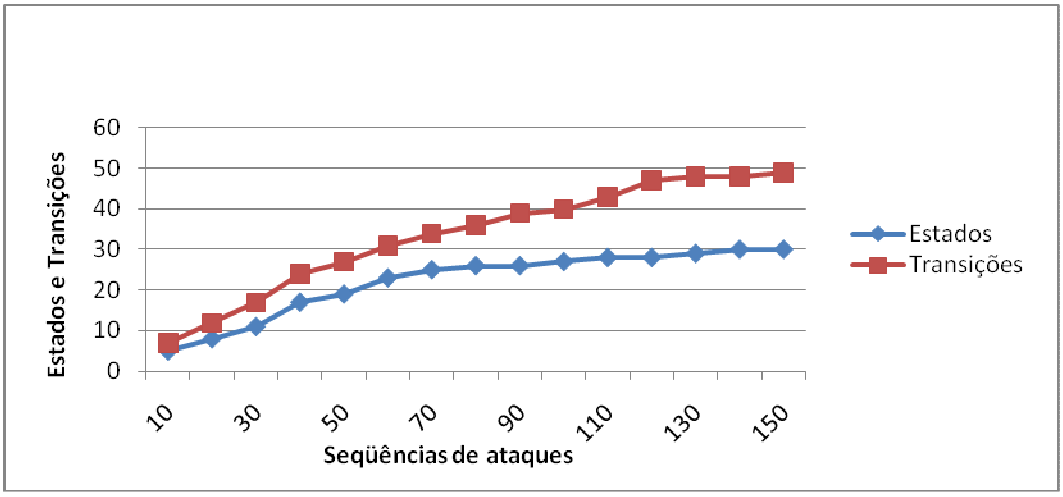

Figura 4.7. Gráfico da evolução de estados e transições (MHMM)

Como pode ser observado no gráfico da figura 4.7, o número de estados e transições se estabilizam, uma vez que as novas seqüências de ataques são similares a outras já incorporadas ao modelo.

É importante ressaltar que tanto a criação de estados como de transições possuem uma tendência de estabilização, conforme apresentado na figura 4.7. Isto é uma boa métrica para avaliar se o modelo comportamental está robusto o suficiente para distinguir anomalias.

O número de seqüências distinguidas pelo modelo seqüencial apresentado na tabela 4.4 representa no número de GSA (grafo de seqüência anômalas) criados; desta forma foram criados 129 GSA.

4.6.2 Dados da execução em modo normal

Na execução em modo real, os ataques foram reproduzidos de forma completa ou parcial, ou seja, os ataques completos representam as mesmas seqüências anômalas produzidas durante o período de treinamento. Os ataques realizados de forma parcial obedeceram três categorias distintas, as quais são destacadas a seguir.

- Por seqüência: neste ataque, o tempo entre as anomalias de uma rotina de ataque é mantido, porém as anomalias selecionadas são modificadas parcialmente ou totalmente;
- Por tempo: as seqüências de ocorrência de uma rotina de ataque são mantidas, mas os tempos entre estas são alterados parcialmente ou totalmente;
- Por seqüência e tempo: são criadas novas seqüências distribuídas de forma distintas ao longo do tempo.

Na tabela 4.5, têm-se os resultados obtidos após 100 seqüências de ataques disparadas pelo módulo gerador de ocorrência (ver seção 7.2). O *thresold* adotado foi de 15%, definindo assim a sensibilidade dos modelos comportamentais. As taxas de detecção apresentadas na tabela 4.5 são divididas em duas categorias:

- Taxa de identificação: porcentagem de quantos dos ataques disparados foram identificado após uma rotina anômala executada.
- Índice de certeza médio: para cada ataque detectado é gerada uma porcentagem relativa à comparação dos dados do modelo comportamental com as informações coletadas em tempo de execução normal. Ao final, é feita uma média dessa porcentagem para uma rotina anômala executada.

Tabela 4.5 – Dados do CISS-SoC após período de execução em modo normal.

Tipo de ataque	**Nro. De ataques**	**MHMM**	**Modelo Seqüencial**
Completo	25	100% identificados com índice de certeza médio de 95,4%	100% identificados com índice de certeza médio de 99,2%
Parcial por seqüência	25	96% identificados com índice de certeza médio de 93,3%	84% identificados com índice de certeza médio de 89,2%
Parcial por tempo	25	80% identificados com índice de certeza médio de 88,7%	94% identificados com índice de certeza médio de 97,4%
Distintos por tempo e seqüência.	25	54% identificados com índice de certeza médio de 91,1%	32% identificados com índice de certeza médio de 85,9%

A tabela 4.5 representa a eficiência dos modelos MHMM e seqüencial. Como pode ser observado, as fraquezas de um modelo podem ser parcialmente compensadas pelo outro. É importante salientar que os ataques distintos por tempo e seqüência significam novas rotinas anômalas que não foram definidas no período de treinamento, mesmo assim o CISS-SoC

obteve êxito em 54% de seqüências identificadas no caso do modelo MHMM e 32% para o modelo seqüencial. No capítulo 7, destacam-se comparativos dos resultados alcançados com outras implementações.

Em relação ao desempenho do CISS-SoC, é importante ressaltar que a versão em SoC atingiu um bom desempenho de processamento (ver seção 4.4.2), o que permitiu a execução e o tratamento de eventos anômalos sem atrasos de avaliação. Como o MHMM é fortemente acoplado ao parâmetro tempo, atrasos ou perda de pacotes teriam um impacto direto na detecção de anomalias. Não foi registrado nenhum prejuízo em relação aos ADAs durante o período de testes.

4.7 Considerações Finais do Capítulo

A versão em *system-on-chip* da CISS foi apresentada nesse capítulo, descrevendo sua arquitetura, taxas de ocupação e propagação e dados de testes reais do CISS-SoC implementado em um FPGA Virtex II-Pro.

Inicialmente, foi apresentada a arquitetura *top-level* do CISS-SoC, onde se pode observar os módulos dedicados em hardware e sua interação com o processador embarcado. Na seqüência, foram detalhados cada um dos módulos dedicados em hardware e as características do software e bibliotecas embarcadas.

Para testar o CISS-SoC, foi necessário um FPGA adicional contendo o módulo gerador de anomalias. Este possui pré-configuradas rotinas de ataques para o período de treinamento e o período de execução em modo normal.

Os resultados da implementação dos modelos MHMM e Seqüencial foram satisfatórios e são comparados com outras implementações e trabalhos correlatos no capítulo 7. Apesar dos resultados satisfatórios, o sistema implantado é muito simples, principalmente quando se trata

dos serviços de segurança implementados. No entanto, a CISS e os algoritmos de detecção de anomalias (ADA) são os mesmos para esta e outras versões com serviços de segurança mais robustos.

Foi constatado também que a versão em SoC permite a criação de serviços de segurança dedicados de alta velocidade, inibindo problemas em relação à capacidade de processamento dos modelos comportamentais.

CAPÍTULO 5

IMPLEMENTAÇÃO EM SOFTWARE (JCISS)

Neste capítulo, apresenta-se a implementação em software da CISS, chamada de JCISS. Descrita em linguagem Java, a API JCISS implementa a estrutura de armazenamento proposta pela CISS, os modelos comportamentais e os algoritmos de detecção de anomalias.

5.1 API JCISS

A API JCISS possui todas as funcionalidades da Camada de Integração de Serviços de Segurança (CISS), incluindo o recurso subMHMM (ver seção 3.4.3). A criação de uma API permite que outras aplicações possam instanciar a JCISS com o intuito de criar um ambiente personalizado de proteção. Na figura 5.1, tem-se o diagrama desta API, onde se destacam as seguintes classes:

- CISS: nesta classe são definidas as características do servidor ou *host* em que a instância da JCISS está em execução;
- Auditoria: está classe possui métodos para facilitar a geração de relatório sobre a ação da JCISS em período de treinamento ou modo de execução normal;
- IDMEF: controla a ocorrência de eventos de segurança e possui métodos para buscar informações específicas das anomalias registradas;
- MHMM: classe responsável por criar no período de treinamento o grafo de comportamento, cálculo das medidas básicas (ver seção 3.4.1) e probabilidade condicionais;

- MSeq: possui métodos para a criação do GSA(Grafos de Seqüências Anômalas) em período de treinamento (ver seção 3.5);
- subMHMM: possui métodos para o gerenciamento e tratamento do Estado X (ver seção 3.4.3) em tempo de execução;
- ADA-MHMM: implementa por meio de métodos os cálculos necessários para detectar situações anômalas utilizando MHMM (ver seção 3.4.2);
- ADA-MSeq: Criar o GST e implementa métodos de busca de subgrafos em grafo (ver 3.5.1).

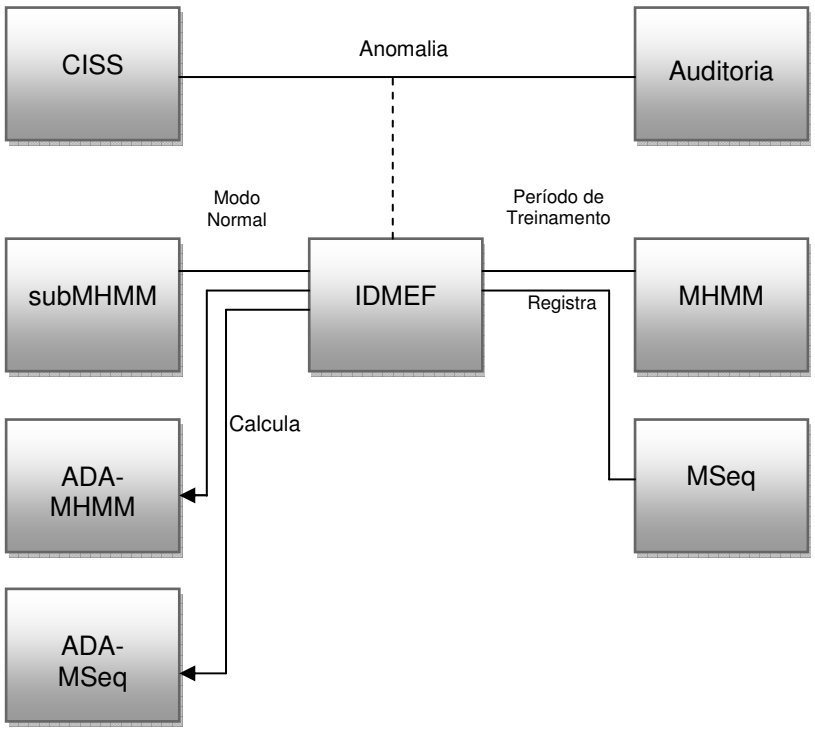

Figura 5.1. Diagrama da JCISS

5.2 Serviços de Segurança

Foram integrados alguns serviços de segurança reais na JCISS como antivírus *opensource* Clamwin (CLAMWIN, 2009), o IDS Snort (SNORT, 2009) e uma aplicação web proprietária.

Por ter acesso ao código fonte, foi possível criar pequenas modificações para que esses serviços de segurança pudessem relatar à JCISS as ocorrências de anomalias no formato IDMEF. Na seqüencia, a descrição dos serviços de segurança citados.

5.2.1 Antivírus Clamwin

O Clamwin é um antivírus *opensource* para Windows que utiliza a plataforma clamAV (CLAMAV,2009) sob uma interface gráfica. O Clamwin (CLAMWIN, 2009) foi descrito utilizando as linguagens C++ e Python. O programa está disponível sob a licença GNU pela Free Software Foundation e é de uso gratuito.

Por disponibilizar o código fonte, foi possível adaptá-lo para gerar ocorrências no formato IDMEF para a JCISS. As adaptações realizadas no código fonte do Clamwin estão apresentadas na seqüencia, onde estão em destaque as poucas linhas adicionadas.

```
...
#include<idmef.h>  /* adicionada */
...
if((ret = cl_scanfile("/tmp/test.exe", &virname, NULL, engine, &limits,
      CL_STDOPT)) == CL_VIRUS) {
      printf("Virus detected: %s\n", virname);

      insert_ciss(&virusname);  /* adicionada */

} else {
      printf("No virus detected.\n");
     if(ret != CL_CLEAN)
        printf("Error: %s\n", cl_strerror(ret));
    }
}
...
```

A biblioteca idmef.h contém as especificações do formato de alerta IDMEF e a função insert_ciss(char[] name), que se utiliza para inserir no banco de dados da JCISS a anomalia detectada. O código fonte do Clamwin, assim como todas as documentações podem ser encontradas no endereço: https://clamwin.svn.sourceforge.net/svnroot/clamwin/ (CLAMWIN SOURCE, 2009).

5.2.2 IDS Snort

O Snort, desenvolvido por Martin Roesch, é um sistema de detecção de intrusão para rede, capaz de realizar a análise de tráfego em tempo real e registro de pacotes em redes IP. O Snort analisa protocolos, busca e associa padrões de conteúdo e pode ser usado para detectar uma variedade de ataques (SNORT, 2009).

Para integrar o IDS Snort, à JCISS, não foi necessária a alteração do código fonte do mesmo. A alteração realizada foi apenas com a inserção de uma *trigger*[3] junto ao banco de dados Mysql do Snort, que ao detectar a inserção de uma nova anomalia faz uma réplica dos dados significativos para a base de dados da JCISS.

Ao instalar o Snort, pode-se utilizar o banco de dados MySql para armazenar informações de controle e detecção de anomalias. Quando isto é realizado, é criada uma base de dados com as seguintes tabelas:

```
mysql> SHOW TABLES;
------------------
| Tables_in_snort |
------------------
| acid_ag |
| acid_ag_alert |
| acid_event |
| acid_ip_cache |
| base_roles |
| base_users |
```

[3] Trigger: rotinas que podem ser executadas automaticamente pelo sistema de gerenciamento de banco de dados após um evento específico.

```
| data |
| detail |
| encoding |
| event |
| icmphdr |
| iphdr |
| opt |
| reference |
| reference_system |
| schema |
| sensor |
| sig_class |
| sig_reference |
| signature |
| tcphdr |
| udphdr |
+------------------+
```

Especificamente, a tabela *acid_event* permite armazenar informações sobre a anomalia detectada. Uma *trigger* pode ser criada com a função de copiar qualquer tupla inserida nesta tabela para o banco de dados da JCISS. A seguir, um exemplo de como criar esta *trigger*.

```
CREATE TRIGGER cpy_ciss BEFORE INSERT ON acid_event
  FOR EACH ROW BEGIN
     INSERT INTO ciss.occurs SET ip_dst = NEW.ip_dst;
     INSERT INTO ciss.occurs SET ip_src = NEW.ip_src;
     …
  END;
```

5.2.3 Aplicação Web proprietária

A aplicação web proprietária descrita nesta seção tem com principal função criar um sistema de ensino à distância (EAD) simplificado para as disciplinas especificas. Esta possui funções como simulados, desafios, lista de melhores alunos, etc. O sistema está descrito em PHP e pode ser acessado utilizando um browser.

Esta ferramenta é utilizada pelos professores do Instituto Superior de Tecnologia do Centro Universitário Eurípides de Marília (IST/UNIVEM) para disponibilizar materiais e atividades extracurriculares.

O usuário cadastrado, mediante o processo de *login,* adquire uma sessão para uso do sistema de EAD. O *login* incorreto, o acesso direto a links sem uma sessão estabelecida e o acesso a arquivos não permitidos geram notificações à JCISS.

5.3 Modelo Comportamental

A JCISS integrada com os serviços de segurança destacados anteriormente representa um sistema real de proteção. Ainda que simplificado, uma vez que possui apenas três serviços de segurança citados nas seções anteriores, foi possível criar um modelo comportamental de anomalias destacado na tabela 5.1 e na figura 5.2.

O ambiente de teste é composto por quatro microcomputadores, sendo que um destes é o servidor aonde está instanciada a JCISS. As configurações dos todos os microcomputadores são idênticas, compostas por: 2 GB de Memória RAM, processador Intel Core 2 Duo T5450 1,67Ghz, Sistema Operacional Windows Vista Home Premium, Service Pack 1.

Tabela 5.1 - Dados do modelo comportamental (período de treinamento)

Informações	Quantidade
Número de Seqüências de Ataques	100
Número de Estados	57
Número de Transições	79
Número de Seqüências	92

Durante o período de treinamento, foram criados ataques reais como infecção por vírus, ataques a rede que podem ser identificados pelo IDS Snort e ataques a aplicação web proprietária, totalizando 100 rotinas distintas de anomalias destacadas na tabela 5.1. Durante 30 dias, foram disparadas estas rotinas maliciosas contra o sistema protegido pela JCISS.

Desta forma, foi criado um modelo comportamental específico para os serviços de segurança descritos contendo 57 estados e 79 transições.

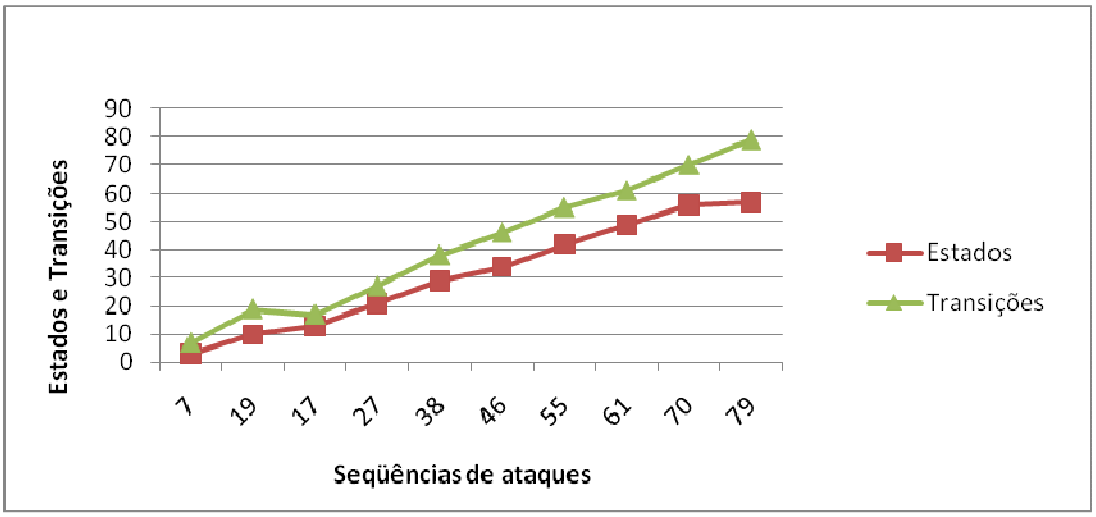

Figura 5.2. Evolução da criação de estados e transições (período de treinamento)

É importante ressaltar que, na figura 5.2, mesmo após 100 seqüências de ataques, o número de estados e transições ainda não se estabilizou. Isto significa que, em modo de execução normal, ataques não previstos durante o período de treinamento poderão ocorrer com uma freqüência acima da satisfatória.

Isto ocorre uma vez que a utilização de serviços reais de segurança como o IDS Snort, o antivírus Clamwin, e aplicação proprietária possibilita inúmeras variedades de ataques, o que pode gerar muitos estados e transições. Desta forma, torna-se difícil criar um modelo comportamental onde o número de estados e transições seja estável após o período de treinamento.

5.4 Resultados da Execução em Modo Normal

Nesta seção, destaca-se a análise do JCISS em modo de execução normal. Os algoritmos de detecção de anomalias dos modelos MHMM e Seqüencial acumularam os resultados apresentados na tabela 5.2 após 100 seqüências de ataques.

Tabela 5.2 - Dados da JCISS em modo normal de funcionamento

Tipo de ataque	Número de ataques	MHMM	Modelo Seqüencial
Completo	25	96% identificados com índice de certeza médio de 93,7%	100% identificados com índice de certeza médio de 98,9%
Parcial por seqüência	25	96% identificados com índice de certeza médio de 89,8%	80% identificados com índice de certeza médio de 90,1%
Parcial por tempo	25	84% identificados com índice de certeza médio de 85,5%	92% identificados com índice de certeza médio de 95,7%
Distintos por tempo e seqüência.	25	60% identificados com índice de certeza médio de 87,2%	36% identificados com índice de certeza médio de 88,8%

Como podem ser observados, os resultados foram semelhantes aos atingidos pela implementação em hardware (tabela 4.5). No entanto, uma nova seqüência de ataques foi incorporada a este teste, onde os ataques distintos por tempo e seqüência, ou seja, não realizados no período de treinamento, foram tratados pela subMHMM.

O critério para aceitação como novo padrão de anomalia incorporado ao modelo criado em período de treinamento foi:

- *Threshold* maior que 40% (sensibilidade ou índice de certeza).
- Número de recorrência do subMHMM igual ou maior a dois, ou seja, a mesma anomalia identificada mais de uma vez pela subMHMM e ainda não está presente no MHMM original.

Após a definição dos critérios para a subMHMM, os ataques distintos por tempo e seqüência foram repetidos por três vezes. Na terceira tentativa, a maior parte dos ataques foi identificada pela JCISS como descrito na tabela 5.3.

Tabela 5.3 – Resultados com o uso da subMHMM

Tipo de ataque	Número de ataques	MHMM	Modelo Seqüencial
Distintos por tempo e seqüência.	25	60% identificados com índice de certeza médio de 87,2%	36% identificados com índice de certeza médio de 88,8%
Distintos por tempo e seqüência.	25	68% identificados com índice de certeza médio de 89,2%	36% identificados com índice de certeza médio de 88,8%
Distintos por tempo e seqüência.	25	89% identificados com índice de certeza médio de 91,2%	36% identificados com índice de certeza médio de 88,8%

5.5 Considerações Finais do Capítulo

Os resultados obtidos pela JCISS refletem de forma satisfatória sobre um sistema computacional real utilizando serviços de segurança conhecidos e aplicações proprietárias, uma vez que os índices detecção para as rotinas anômalas executadas foram satisfatórios. A API JCISS criada possibilita o reuso da mesma estrutura para construção de outros sistemas de proteção com mais recursos e serviços de segurança. Partindo deste principio, foi criado um simulador para CISS contendo funcionalidades mais robustas para avaliar a CISS. O CISS-SIM está descrito no capítulo 6.

Como um objetivo paralelo foi possível mostrar que a implementação em software da CISS tornou possível que ferramentas de segurança e aplicativos que desejam prover a segurança por necessidade ou diferencial podem utilizar a API JCISS como uma estrutura comum de proteção integrada.

CAPÍTULO 6

SIMULADOR PARA CISS (CISS-SIM)

Neste capítulo, apresenta-se o simulador para a camada de integração de serviços de segurança, chamado de CISS-SIM, destacando suas principais funcionalidades e a possibilidade de construção de cenários mais robustos para avaliar os modelos comportamentais e os algoritmos de detecção de anomalias propostos.

6.1 CISS-SIM

Após realizar os testes com o CISS-SoC e JCISS, pode-se observar que uma análise mais completa da eficiência da CISS poderia ser demorada e complexa, uma vez que muitos serviços de segurança e rotinas de ataques deveriam ser construídos e adaptados.

Neste contexto, foi criado um simulador para facilitar a criação de cenários e a geração de rotinas de ocorrências mais robustas em um menor tempo. As principais funcionalidades do CISS-SIM estão organizadas na tabela 6.1.

Tabela 6.1 Principais Funcionalidades do CISS-SIM

Escopo	Funcionalidade	Descrição
Cenários	Criar aplicações de Segurança	Configurar aplicações de segurança fornecendo dados de descrição e distinção de serviços de segurança que são oferecidos. Deve ser compatível com o padrão IDMEF.

	Criar Rede de Computadores	Cria a relação entre aplicações e computadores, ou seja, são definidas quais as aplicações de segurança de cada computador e qual a função de cada máquina. Deve-se definir qual computador é o portador da CISS.
Rotinas de Ocorrências	Linha do tempo	Configura a escala da linha do tempo de ocorrências, onde a escala mínima é o milissegundo; Associar e organizar periodicamente as rotinas de ocorrências ou ocorrências individuais.
	Ocorrências	Configuração completa de uma ocorrência (padrão IDMEF).
	Rotinas	Associar seqüências de ocorrências caracterizando um ataque específico.
	Relação com Aplicação	Criar relação da ocorrência com a aplicação e computador que será a fonte geradora.
Modelos Comportamentais	MHMM	Permite limitar o número de transições e estados; estatísticas do modelo MHMM;
	Seqüencial	Permite limitar número de transições e estados; estatísticas do modelo MHMM;
	Direto	Importar modelos seqüenciais.
subMHMM	Configurar Regras	Permite configurar as regras de criação, classificação e maturação dos modelos subMHMM.
ADA	Configurar Threshold	Configurar sensibilidade da detecção de anomalias.
Simulação	Tempo Real	Simulação associada à escala de tempo real, ou seja, se uma rotina está programada para durar uma hora, as ocorrências serão executadas obedecendo este intervalo de tempo real.

	Funcional	Simulação funcional; as ocorrências serão executadas obedecendo à seqüência pré-estabelecida, no entanto, a escala de tempo não será considerada, porém serão registrados os tempos definidos nestas. O efeito é uma simulação rápida que gera os mesmos resultados da simulação em tempo real.
	Período de Treinamento	Criação dos modelos comportamentais;
	Execução em Modo Normal	Período de detecção de anomalias (ADA)
Ocorrências artificiais	Gerar Ocorrências artificiais	Cria variações de ocorrências com base nas ocorrências pré-configuradas;
Relatórios	Geração de relatórios	Relatórios do MHMM e Seqüencial; Relatórios de anomalias identificadas e índices médios de certeza;
E/S	Exportar	Modelos MHMM e Seqüencial; Cenários; Rotinas de Ocorrências; Inicialmente, apenas uma cópia do Banco de Dados;
	Importar	Modelos MHMM e Seqüencial; Cenários; rotinas de ocorrências.

Neste capítulo, não se pretende criar um manual do CISS-SIM, mas apenas apresentar suas principais funcionalidades e os requisitos para a criação de um cenário e análise dos resultados de simulações.

Na figura 6.1, tem-se a janela principal do CISS-SIM. O Simulador foi implementado em Java utilizando o IDE Netbeans. Foi utilizado o banco de dados MySQL e os relatórios foram gerados por meio da API JasperReports e a ferramenta iReport.

É importante destacar que foi utilizada a técnica de persistência (HIBERNATE, 2009), transformando o banco de dados relacional em classes. Estas classes podem ser instanciadas

como objetos e associadas a componentes gráficos como tabelas, caixa de texto, etc. Neste sentido, é possível realizar consultas, inserções, atualizações e exclusões através de métodos, sem a necessidade do uso da linguagem SQL diretamente.

Figura 6.1. Desktop CISS-SIM

Na seqüência, tem-se a construção dos cenários de testes e validações e os resultados alcançados utilizando o CISS-SIM.

6.2 Cenário de Testes (CISS-SIM)

O cenário de simulação e testes deve ser construído em etapas. Inicialmente, devem ser criadas as aplicações e rotinas de ocorrências. Este é um trabalho árduo, mas quando

comparado à criação das mesmas funcionalidades em ambientes reais, torna-se mais rápido, flexível e independente.

Foram criadas várias rotinas de ocorrências associadas a diferentes aplicações de segurança. Muitas vezes, apenas conhecendo o comportamento de algumas aplicações de segurança, pode-se descrever sua atuação de forma precisa através do CISS-SIM. Quanto mais próximo do comportamento da aplicação real, mais representativo será o modelo comportamental gerado.

Nos testes realizados, foram criadas aplicações de segurança que abrangem todos os atributos de segurança. Entre as aplicações descritas, tem-se: firewall, IDS, antivírus, antispam, antiphishing, serviços de criptografia, serviços de integridade, aplicações proprietárias e serviços de controle de acesso. A figura 6.2 apresenta uma das etapas de configuração e criação das aplicações de segurança, onde se destaca a criação das ferramentas de segurança utilizadas no capitulo 5. As informações sobre a aplicação devem obedecer a descrição da RFC 4765 (IDMEF).

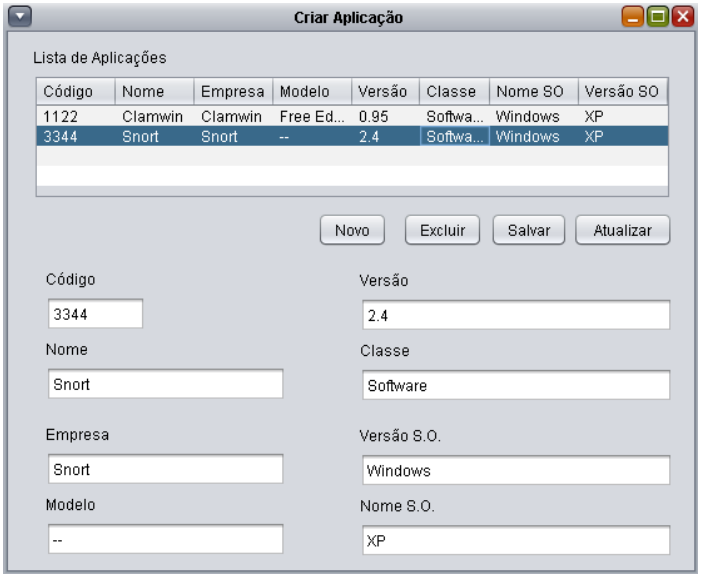

Figura 6.2. Criação e configuração de uma aplicação no CISS-SIM.

Variações de rotinas de ocorrência para a construção do modelo seqüencial e testes em modo de execução foram descritas especificamente para as aplicações de segurança citadas anteriormente. A figura 6.3 mostra a criação e configuração de uma ocorrência; novamente é importante ressaltar que a descrição tem como base a RFC 4765.

O reuso das ocorrências criadas é uma funcionalidade prevista no CISS-SIM. Na criação da ocorrência, são inseridos apenas dados básicos como: código da ocorrência, nome, classificação e atributo de segurança. Os demais atributos são preenchidos quando esta ocorrência é associada à determinada aplicação e computador, durante a construção de rotinas de eventos de segurança.

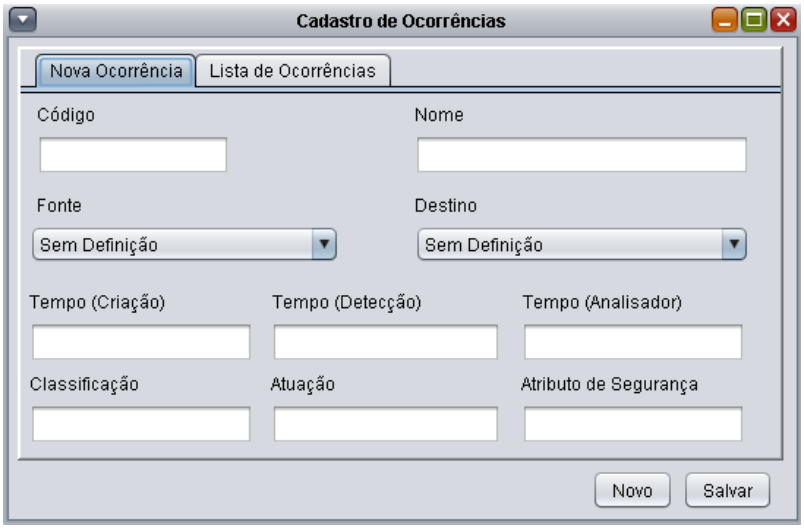

Figura 6.3. Criação e configuração de uma ocorrência no CISS-SIM.

Na figura 6.4, mostra-se a associação de ocorrências para concepção de rotinas de ocorrências. Nesta etapa é definida uma seqüência de eventos considerada anômala. As ocorrências são distribuídas por ordem e tempo.

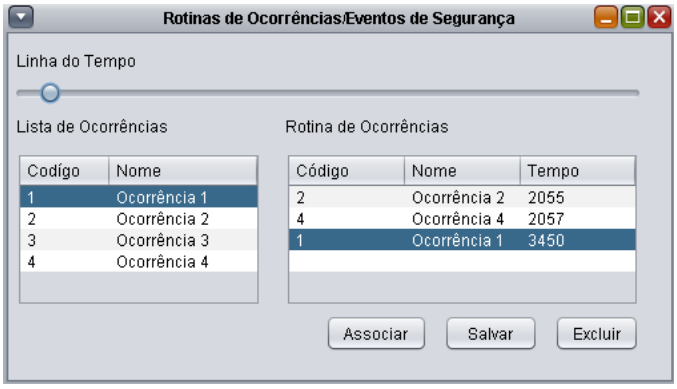

Figura 6.4. Associação de ocorrências para formação de rotinas

Uma vez construídas as rotinas de ocorrências, deve-se relacionar estas com as aplicações maliciosas, criadas de forma semelhante às aplicações de segurança (figura 6.2). Uma rotina de ocorrências pode ser atribuída a mais de uma aplicação maliciosa, caso seja conveniente.

Por fim, a criação da rede de computadores. Nesta etapa, as aplicações de segurança maliciosas devem ser associadas aos computadores de uma rede. Neste processo, deve-se definir qual dispositivo será o portador da CISS. Além disso, cada dispositivo da rede deve ser configurado conforme as características básicas descritas na RFC 4765. Ao concluir esse processo, o cenário está estabelecido e os testes podem ser executados.

6.3 Período de Treinamento

O passo inicial do período de treinamento é a associação das rotinas de ocorrências e a linha de tempo principal. Uma vez criadas as associações, o ambiente pode ser simulado, gerando automaticamente as ocorrências anômalas para construção de modelos comportamentais MHMM e Seqüencial. A simulação pode ser em tempo real ou funcional.

Em tempo real, a simulação pode ser pausada para analisar parcialmente a criação dos modelos comportamentais. A simulação funcional tem como características a simulação sem obedecer à escala de tempo, no entanto os resultados finais são os mesmos.

Durante o período de treinamento, foram utilizadas 10 aplicações de segurança, 170 ocorrências e 320 variações de rotinas de ocorrências. Em conseqüência, foi criado um MHMM e um Modelo Seqüencial mais robusto, quando comparado com testes reais apresentados nos capítulos 4 e 5. Na tabela 6.2, têm-se os dados dos modelos construídos.

Tabela 6.2 - Resultados após o Período de Treinamento

Informações	Quantidade
Número de Seqüências de Ataques	320
Número de Estados	541
Número de Transições	758
Número de Seqüências	290

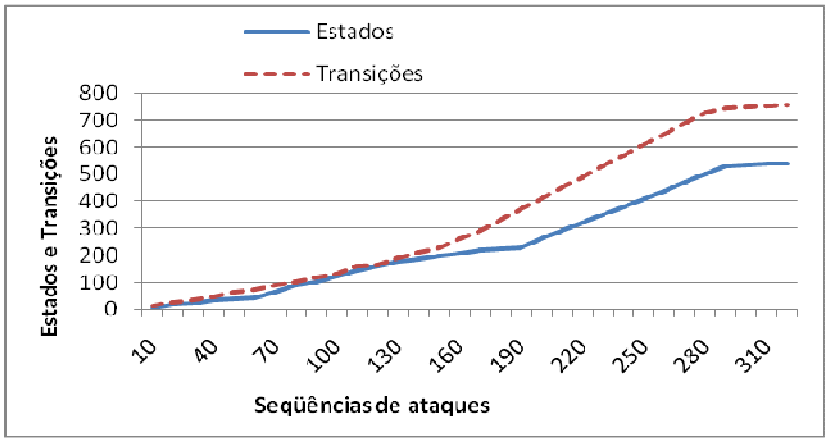

Figura 6.5. Evolução da criação dos estados e transições

A figura 6.5 apresenta a evolução dos modelos conforme a execução das rotinas de ocorrências. Para os dados considerados, observa-se que o número de estados e transições estabilizou após 280 ocorrências efetuadas. O que significa que o modelo pode gerar bons resultados na detecção de anomalias.

6.4 Detecção de Anomalias

Após a construção dos modelos comportamentais, as rotinas de ocorrências utilizadas no período de treinamento, algumas variações destas e novas rotinas foram executadas em modo normal, com o intuito de avaliar a eficiência da CISS.

Os resultados obtidos foram próximos aos computados para as implementações reais em software (capítulo 5) e SoC (capítulo 4). No entanto, os modelos comportamentais gerados são bem mais complexos e os testes utilizando o *subMHMM* puderam ser avaliados com mais detalhes.

6.4.1 Detecção de Anomalias sem Recursos Adicionais

Na seqüência, a discussão dos resultados alcançados para ocorrências completas, ou seja, as mesmas disparadas em período de treinamento. Para todos os testes foram selecionadas 100 rotinas de ocorrências. Neste caso, o MHMM é capaz de identificar uma seqüência anômala com mais de 95% de certeza em 96% das ocorrências.

Com isso, quatro em cem seqüências anômalas não foram identificadas pelo MHMM. Isso ocorre, pois a ordem de criação dos estados e transições no período de treinamento não é a mesma que no período de execução normal. Neste sentido, pode-se levar ao não reconhecimento de um padrão, uma vez que a distribuição probabilística dos estados e transições não são exatamente as mesmas.

Outro fator determinante é a diferença de tempo entre a computação executada no período de treinamento e a execução em modo normal. Como o algoritmo MHMM é fortemente acoplado ao fator tempo, isto pode contribuir para o erro de 4% encontrado na detecção de rotinas de ocorrências completas.

No entanto, essas ocorrências não identificadas foram marcadas pelo ADA-MHMM. Ao observar o Modelo Seqüencial, foi constatado que 98% das rotinas anômalas foram detectadas com 97% de índice médio de certeza. Assim, duas rotinas não foram identificadas, porém não são as mesmas do modelo comportamental MHMM. Neste contexto, quando o MHMM não acusou a anomalia, o Modelo Seqüencial foi capaz de identificá-la.

A mesma rotina de testes foi realizada por dez vezes durante a execução em modo normal e dados semelhantes foram anotados. Uma importante observação notada é que o MHMM tem dificuldade de identificar rotinas de ocorrências muito curtas. Já o Modelo Seqüencial não identificou as rotinas com maior número de ocorrências. Na tabela 6.3, apresentam-se os resultados alcançados após as seqüências de ocorrências parcialmente e totalmente modificadas.

Tabela 6.3 - Resultados após de rotinas de ocorrências parcialmente modificadas.

Tipo de ataque	**Número de ataques**	**MHMM**	**Modelo Seqüencial**
Parcial por seqüência	100	96% identificados com índice de certeza médio de 89,8%	80% identificados com índice de certeza médio de 90,1%
Parcial por tempo	100	84% identificados com índice de certeza médio de 85,5%	92% identificados com índice de certeza médio de 95,7%
Distintos por tempo e seqüência.	100	56% identificados com índice de certeza médio de 87,2%	36% identificados com índice de certeza médio de 88,8%

Nas rotinas de ocorrências parcialmente modificadas por tempo ou por seqüência, os resultados alcançados foram similares aos atingidos nas implementações CISS-SoC e JCISS, como pode ser visto na tabela 6.3.

Rotinas diferenciadas por tempo e seqüência foram criadas para avaliar o comportamento da CISS para ataques não conhecidos. No entanto, foram mantidos os mesmos serviços de segurança e não foi gerado o efeito do *Estado X* (ver seção 3.4.2).

É importante salientar que esta porcentagem está diretamente ligada ao fator de sensibilidade (*threshold*) da CISS, que está configurado para 15%. O índice de identificação variou bastante, dependendo das rotinas de ocorrências não conhecidas. Sendo que a média apresentada foi de 56% de detecção utilizando o MHMM e 36% para o modelo Seqüencial. Estes resultados são comparados e discutidos no capitulo 7.

6.4.2 Detecção de Anomalias utilizando o algoritmo de Wei Fan (2001)

Porcentagens semelhantes às encontradas nos testes de detecção de rotinas distintas por tempo e seqüência descritas na seção anterior foram descritas por Wei Fan (2001), o qual criou ataques artificiais para deixar mais robusto os modelos comportamentais gerados em período de treinamento com o intuito de aumentar a probabilidade de detecção de ataques não conhecidos.

O algoritmo de Wei Fan (2001) para geração de rotinas de ocorrências artificiais é simples e eficiente e foi implementado na CISS-SIM para verificar o seu impacto. Neste sentido, os modelos comportamentais foram enriquecidos utilizando o algoritmo Wei Fan.

Foram criadas cinco variações para cada rotina de ocorrências executadas durante o período de treinamento. O efeito do enriquecimento está descrito através dos números apresentados na tabela 6.4. Como se pode observar, os modelos comportamentais são enriquecidos e o número de estados e transições aumentam.

Tabela 6.4 - Resultados após o Período de Treinamento enriquecido (Wei Fan)

Informações	Quantidade
Número de Seqüências de Ataques	1600
Número de Estados	810
Número de Transições	1044
Número de Seqüências	1211

Em princípio, uma alternativa a esta modificação seria o aumento da taxa de sensibilidade (*threshold*) de 15% para um valor maior, assim variações maiores de rotinas de ocorrências seriam consideradas como situações anômalas.

No entanto, essa não seria uma alternativa válida, uma vez que aumentar o *threshold* aumentaria a probabilidade de rotinas de ocorrências falsas. Ou seja, poderiam ser consideradas ocorrências que não estão no escopo de um ataque previsto no período de treinamento.

Aplicando o algoritmo de Wei Fan (2001), essa probabilidade é minimizada, dado que os ataques artificiais são compostos por variações mínimas das rotinas de ocorrências concebidas durante o período de treinamento, mantendo assim o mesmo escopo de atuação.

O efeito maior alcançado com a aplicação do algoritmo de Wei Fan (2001) foi na detecção de rotinas de ocorrências não conhecidas. Em modo de execução normal, foi possível detectar em média 66% dos ataques desconhecidos utilizando o MHMM e 42% por meio do Modelo Seqüencial.

6.4.3 Detecção de Anomalias utilizando subMHMM proposto

Em modo de execução, foi ativado o módulo *subMHMM* e os resultados atingidos foram satisfatórios, pois após dez repetições das rotinas de ataques a taxa de identificação de ataques não conhecidos passou de 56% para 91% na média. A tabela 6.5 mostra essa evolução.

Os resultados alcançados são semelhantes à implementação JCISS. As regras para incorporar um subMHMM ao MHMM original foram:

- *Threshold* maior igual a 40%
- Número de recorrência do subMHMM maior igual a 7.

Tabela 6.5 - Resultados aplicando subMHMM

Tentativa	Número de ataques	MHMM
1º a 6º	100	56% identificados com índice de certeza médio de 87,2%
7º	100	62% identificados com índice de certeza médio de 89,1%
8º	100	74% identificados com índice de certeza médio de 89,5%
9º	100	86% identificados com índice de certeza médio de 91,2%
10º	100	91% identificados com índice de certeza médio de 93,7%

6.5 Considerações Finais do Capítulo

O CISS-SIM foi implementado utilizando a API JCISS. Foram implementadas funções para a criação de cenários, rotinas de ataques e visualização de resultados. O simulador foi fundamental para criar cenários mais complexos e analisar com mais propriedade o impacto da CISS na detecção de anomalias. Os resultados da versão simulada, como os modelos comportamentais gerados, podem ser utilizados em sistemas reais, onde a CISS esteja implantada.

Neste sentido, cenários reais podem ser construídos, testados e analisados pelo CISS-SIM, antes mesmo de serem implantados realmente.

CAPÍTULO 7

ANÁLISE DOS RESULTADOS ALCANÇADOS

Neste capítulo é apresentada uma análise dos resultados alcançados, comparando-os e validando a eficiência da camada de integração de serviços de segurança. A caracterização das versões da CISS, os dados referentes ao impacto na detecção de situações anômalas e as comparações com os trabalhos correlatos são apresentados e discutidos neste capítulo.

7.1 Versões da CISS

A camada de integração de serviços de segurança foi especificada (ver capitulo 3) e implementada em três ambientes diferentes, hardware (CISS-SoC), software (JCISS) e simulada (CISS-SIM). Na tabela 7.1, resumem-se as características de cada implementação, com o intuito de fazer análise final da CISS proposta e implementada.

Tabela 7.1 – Versões da CISS

	CISS-SoC	JCISS	CISS-SIM
Nro. de Serviços de Segurança	5	3	10
Sistema Operacional	Linux	Multiplataforma	Multiplataforma
Plataforma	FPGA	PC	PC
Linguagem	C e VHDL	Java	Java
Banco de Dados	Hashtable	Mysql	Mysql
Acesso aos dados	Endereçamento de Memória	Persistência (*hibernate*)	Persistência (*hibernate*)
MHMM	Sim	Sim	Sim
Modelo Seqüencial	Sim	Sim	Sim
subMHMM	Não	Sim	Sim
Algoritmo de Wei Fan	Não	Não	Sim

Serviços de Segurança	IDS, Firewall, Criptografia, antivírus e autenticação	IDS Snort, Clamwin Antivírus, Aplicação WEB EAD.	Firewall, IDS, Antivírus, antispam, antiphishing, criptografia, integridade, aplicações proprietárias e serviços de controle de acesso

7.2 Vetores de Testes

Nesta seção, apresenta-se como foram concebidos os vetores de anomalias para testes e validações da CISS. Inicialmente, são apresentados os tipos de testes realizados e posteriormente os métodos de construção dos vetores testes.

7.2.1 Tipos de Testes

Tanto o MHMM como o Modelo Seqüencial geram grafos de comportamento anormal das ocorrências de segurança registradas na CISS. Após o período de treinamento, muitas rotinas anômalas foram registradas, construindo um modelo representativo das anomalias comuns para o sistema computacional em foco.

As rotinas de testes para validar a robustez e eficiência dos modelos gerados são baseadas em variações das rotinas de anômalas executadas em período de treinamento. Estas variações são classificadas em quatro tipos:

- **Completa**: neste caso, as mesmas rotinas de testes executadas em período de treinamento foram executadas em modo normal de execução. Foi notado que atrasos referentes à execução das rotinas de testes completa, tem impacto na detecção de

anomalias, isso ocorre pela forte dependência do fator tempo principalmente no MHMM;

- **Por seqüência**: Neste caso variações entre 10% até 50% das seqüências originais são geradas para validar a detecção de anomalias parcialmente conhecidas. O algoritmo para criar estas variações é apresentado na seção seguinte;
- **Por tempo**: a variação afeta o fator tempo entre 10% até 50%. O fator tempo é determinante para o MHMM. Nos resultados apresentados na tese é possível ver a maior dificuldade em detectar rotinas modificadas neste contexto;
- **Por tempo e seqüência**: neste caso são rotinas modificadas por tempo e seqüência, com taxa de 10% a 50%. Estas rotinas são consideradas nesta tese, como rotinas não conhecidas pelos modelos comportamentais. Como podem ser observadas na tese, as taxas de identificação são menores que as anotadas com os outros tipos de variações.

O parâmetro descrito como *threshold,* tem impacto direto na sensibilidade da detecção. Quanto maior o *threshold,* menos sensível é o algoritmo de detecção de anomalias (ADA), gerando taxas de identificação maiores. No entanto, pode ser mais vulnerável a identificações equivocadas de rotinas que não estão sendo tratadas no modelo comportamental.

A determinação do *threshold* foi definida com base nos testes reais realizados, principalmente com a execução de rotinas do tipo completa, parcial por tempo e parcial por seqüência. Neste caso, foi adotado um *threshold* de 15%.

7.2.2 Método para Construção dos Vetores de Testes

Cada rotina de ataque realizada no período de treinamento é constituída de duas ou mais ocorrências anômalas (eventos). As ocorrências anômalas de uma rotina são organizadas em

uma ordem fixa e separadas por espaços de tempo predefinidos entre ocorrências. Na figura 7.1, tem-se a representação deste conceito.

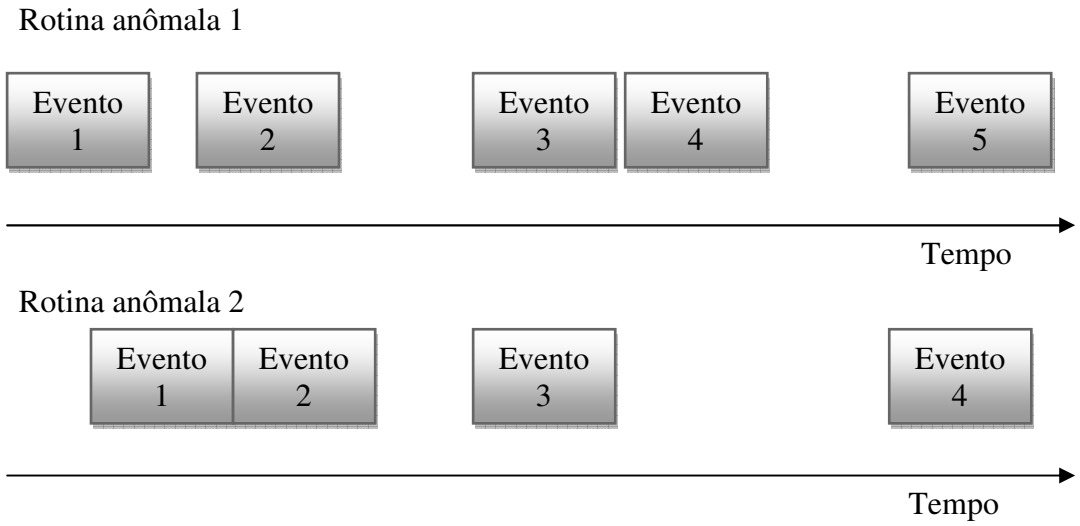

Figura 7.1. Exemplo de organização de uma rotina de ataque

Para a geração dos tipos de testes descritos anteriormente, devem-se manter as mesmas rotinas para os testes completos; variar tempo, para as rotinas modificadas parcialmente por tempo; variar a seqüência, para as rotinas modificadas parcialmente por seqüência; variar tempo e seqüência, para a criação de uma nova rotina. O algoritmo para a geração dos vetores de testes é apresentado a seguir:

Ordena lista de rotinas anômalas do período de treinamento de forma aleatória

Para todas as rotinas anômalas *i* do período de treinamento faça

nroModTempo = 0

nroModSeq = 0

nroModSeqTempo = 0

nroMaxMod <= [(10+random()%40)/100]*j // taxa de mod. da rotina 10% a 50%

Enquanto j <= i.tamanho

Se teste = completo então

Rotina(ij).prox <= Rotina(ij).prox // *mantém tempo e seq.*

Rotina(ij).tempo <= Rotina(ij).tempo

```
            Senão Se teste = parcial por tempo e nroModTempo <= nroMaxMod então
                Rotina(ij).prox <= Rotina(ij).prox // mantém seq.
                novoTempo = random() % maiorTempo
                Rotina(ij).tempo <= novoTempo // muda tempo.
                Incrementa nroModTempo
            Senão Se teste = parcial por seqüência e nroModSeq <= nroMaxMod então
                Rotina(ij).tempo <= Rotina(ij).tempo // mantém tempo.
                novoProx = random() % Rotina(ij).prox
                Rotina(ij).prox <= novoProx // muda seq.
                Incrementa nroModSeq
            Senão Se teste = parcial por seqüência e tempo e nroModSeq <= nroMaxMod então
                novoTempo = random() % maiorTempo
                Rotina(ij).tempo <= novoTempo // muda tempo.
                novoProx = random() % Rotina(ij).prox
                Rotina(ij).prox <= novoProx // muda seq.
                Incrementa nroModSeqTempo
            Senão
                Rotina(ij).prox <= Rotina(ij).prox // mantém tempo e seq.
                Rotina(ij).tempo <= Rotina(ij).tempo
        Incrementa j        // próxima ocorrência de uma rotina anômala
Fim
```

As variações de rotinas anômalas são disparadas em modo de execução normal e os resultados foram apresentados e analisados no corpo desta tese.

7.3 Impacto da Detecção de Anomalia

As tabelas 7.2 e 7.3 apresentam os dados referentes aos modelos comportamentais gerados para cada uma das versões da CISS, após o período de treinamento e em execução em modo normal.

Tabela 7.2 - MHMM e Seqüencial após período de treinamento

	CISS-SoC	JCISS	CISS-SIM	CISS-SIM + Wei Fan
Nro. de Seqüências de Ataques	150	100	320	1600
Número de Estados	30	57	541	810
Número de Transições	49	79	758	1044
Número de Seqüências	129	92	290	1211

Na tabela 7.2, tem-se o resumo de cada implementação da CISS após o período de treinamento, onde foram geradas seqüências de ataques para a construção dos modelos comportamentais. Destaca-se também o número de estados e transições criadas no MHMM para cada versão da CISS, assim como o número de seqüências (GSA) registradas no modelo seqüencial.

Por meio da versão simulada da CISS, foi possível construir modelos comportamentais mais robustos, onde o número de estados criados após o período de treinamento (CISS-SIM, 541 estados) foi cerca de dez vezes maior que as implementações reais em software (JCISS, 57 estados) e hardware (CISS-SoC, 30 estados), como pode ser notado na tabela 7.2.

O algoritmo de Wei Fan (2001) tem o intuito de aumentar a robustez do modelo comportamental. O impacto da versão simulada após a aplicação do algoritmo de Wei Fan (2001) pode ser observada pelo número de estados criados, que aumentou de 541 para 810 estados.

Na tabela 7.3 apresentam todos os resultados referentes ao impacto na detecção de anomalias utilizando as versões em hardware (CISS-SoC) e software (JCISS e JCISS-SIM) da CISS. Mostram-se também os resultados obtidos por meio da aplicação de melhorias propostas e implementadas como o algoritmo de Wei Fan e o recurso subMHMM, sob os modelos comportamentais MHMM e Seqüencial.

Tabela 7.3 - Impacto na detecção em modo de execução normal para cada versão da CISS.

Tipo de ataque	SoC MHMM	SoC Seq.	JCISS MHMM	JCISS Seq	JCISS subMHMM	SIM MHMM	SIM Seq.	SIM subMHMM	SIM Wei fan
Completo	**100%*** **95,4%****	100% 99,2%	96% 93,7%	100% 98,9%	--	98% 92,3%	97% 95,4%	--	--
Parcial por seqüência	96% 93,3%	84% 89,2%	96% 89,8%	80% 90,1%	--	96% 89,8%	80% 90,1%	--	--
Parcial por tempo	80% 88,7%	94% 97,4%	84% 85,5%	92% 95,7%	--	84% 85,5%	92% 95,7%	--	--
tempo e seqüência.	54% 91,1%	32% 85,9%	60% 87,2%	36% 88,8%	89% 91,2%	56% 87,2%	36% 88,8%	91% 93,7%	66% 93,2%

* porcentagem de anomalias identificadas
** índice médio de certeza

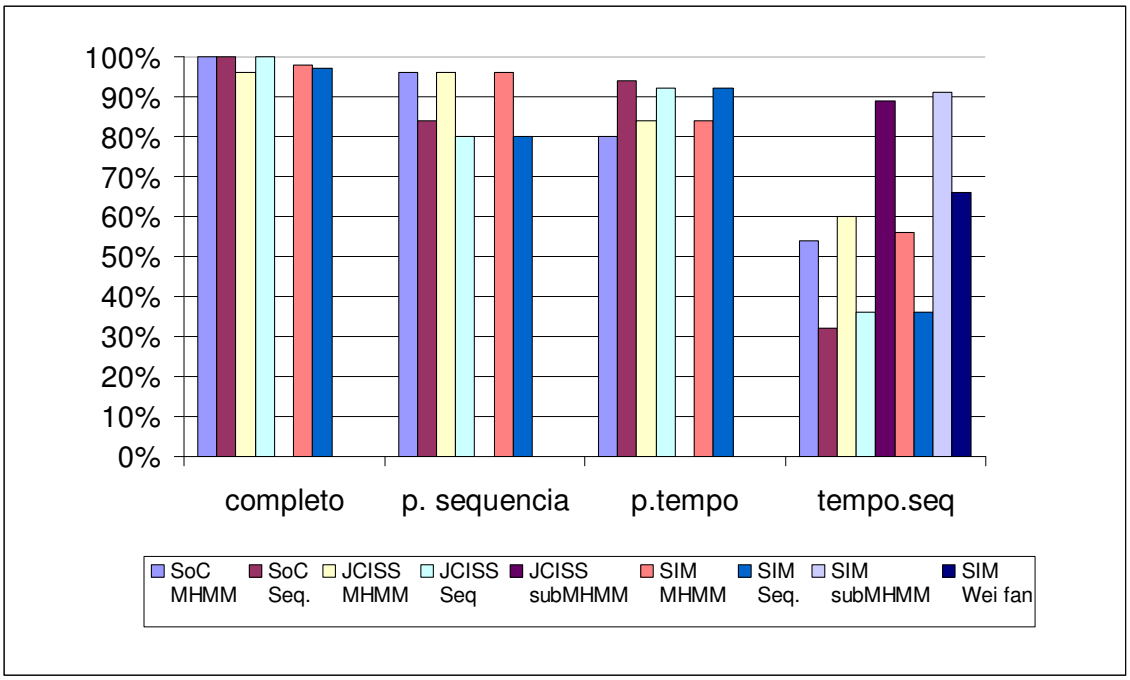

Figura 7.2. Gráfico da porcentagem de anomalias detectada por cada modelo

A figura 7.2 apresenta o impacto da detecção de anomalias utilizando os modelos comportamentais, onde se observa que os ataques de tipo completo atingem maiores taxas de detecção, acima de 96% e os ataques parciais por tempo e seqüência (*tempo.seq*), não obtiveram o mesmo desempenho. Estas e outras observações são discutidas na seqüência.

Observando a figura 7.2, nota-se que o algoritmo MHMM continua sendo eficiente para ataques parciais por seqüência (*p.sequencia*), mas não apresenta a mesma eficiência para ataques parciais por tempo (*p.tempo*), uma vez que este é fortemente acoplado ao parâmetro tempo e qualquer variação neste parâmetro tem impacto direto nos resultados.

O Modelo Seqüencial foi criado para suprir esta dificuldade e o fez com sucesso, dado que, quando o modelo MHMM não atinge bons resultados, o Modelo Seqüencial o complementa. Como pode ser observado, nas rotinas de ocorrências parciais por tempo (*p.tempo*), onde o modelo seqüencial obteve maior êxito que o MHMM.

Para ataques completos, ou seja, as mesmas rotinas de ocorrências executadas no período de treinamento, ambos os modelos obtiveram taxas satisfatórias de reconhecimento, de no mínimo 96%. Isto significa que, se o modelo comportamental criado no período treinamento for suficientemente robusto, a taxa de detecção de ataques conhecidos será alta e a CISS notificará relações entre incidentes de segurança com índices de certeza maiores.

Para rotinas distintas por tempo e seqüência (*tempo.seq*) merece destaque os modelos dotados do recurso de subMHMM que, ao longo do tempo, aprende novas seqüências anômalas, com base em regras e parâmetros como *threshold* e número de recorrências. As implementações que contemplam o subMHMM obtiveram maior impacto na detecção de anomalias, tanto na versão em simulação (*SIM-subMHMM) quanto na versão em software (JCISS-subMHMM).*

Os resultados encontrados nesta tese podem ser comparados com trabalhos correlatos no que diz respeito à capacidade do modelo comportamental representar os eventos de segurança de um sistema e sua capacidade de detectar e distinguir situações normais de anômalas.

Neste sentido, é possível comparar trabalhos mesmo que seus propósitos sejam diferentes. Nesta seção, é feita a comparação com os trabalhos de Yasami (2007), Joshi

(2005) e Wei Fan (2001). Os quesitos em comum são a utilização do modelo comportamental HMM e técnicas diferenciadas para melhorar a detecção de anomalias.

Tabela 7.4 – Comparação com Correlatos

	Anomalias detectadas	**Índice médio de certeza**
CISS-SoC	54	91,1%
JCISS	60	87,2%
JCISS + subMHMM	89	91,2%
CISS-SIM	56%	87,2%
CISS-SIM + Wei Fan	66%	93,2%
CISS-SIM + subMHMM	91%	93,7%
JOSHI (2005) - HMM	79%*	--
YASAMI (2007) HMM	90 a 99% *	--
WEI FAN (2001) - HMM	77%	--

* mescla de rotinas de ataques conhecidas e desconhecidas

A tabela 7.4 apresenta os melhores resultados encontrados em cada um dos trabalhos citados anteriormente para ataques distintos por tempo e seqüência, ou seja, anomalias não previstas no período de treinamento.

Como pode ser observado na tabela 7.4, os trabalhos correlatos não apontam um parâmetro importante, no que diz respeito ao índice de certeza da identificação de uma anomalia. Este parâmetro pode ser utilizado por outras ferramentas de segurança para tomar decisões em relação à prevenção e inibição de um ataque específico.

O trabalho proposto por Wei Fan (2001) obteve bons resultados na detecção de anomalias, atingindo taxas de até 77% de detecção de anomalias não conhecidas, mas quando inserido na CISS não obteve o mesmo êxito, porém ainda apresenta uma boa taxa de 66% de detecção. Essa diferença observada era esperada uma vez que as rotinas de ocorrências utilizadas por Wei Fan (2001) são diferentes das utilizadas na CISS.

Os trabalhos de Joshi (2005) e Yasami (2007) utilizam o HMM para distinguir

situações anômalas analisando o comportamento do tráfego TCP/IP e ARP, respectivamente. Os resultados alcançados foram sob uma mescla de seqüências anômalas conhecidas e desconhecidas. Quando comparados com os resultados obtidos pela CISS, pode-se afirmar que os resultados atingidos pela CISS são satisfatórios, uma vez que as taxas de detecção são próximas e até maiores que as encontradas nos trabalhos correlatos.

Existe uma dificuldade para realizar análises mais profundas, uma vez que os algoritmos de detecção deveriam ser testados em um mesmo ambiente preparado. No entanto, é possível observar a eficiência do MHMM e de técnicas para melhorar os modelos comportamentais como o recurso subMHMM e a criação do modelos seqüencial.

Apesar dessas limitações, é possível verificar que os objetivos traçados inicialmente foram atingidos, pois os dados obtidos permitem afirmar que com as implementações e testes realizados nesta tese, e em especial, com as três implementações da CISS (hardware, software e via simulador) foi possível cumprir as seguintes metas:

- **Integrar os serviços de segurança disponíveis:** usando de formato IDMEF diferentes serviços de segurança puderam gerar informações sobre eventos anômalos e organizá-los para fornecer informações fundamentais para a construção dos modelos comportamentais.
- **Permitir a identificação prematura de um ataque:** com o uso da CISS foi possível integrar e analisar comportamentos anômalos antes nunca relacionados para combater ataques sofisticados a sistemas computacionais de forma prematura;
- **Diminuir a incidência de falsos positivos:** caso algum serviço de segurança indicar uma situação anômala, estas informações podem ser relacionadas a outras ocorrências já mapeadas, caracterizando um ataque real, apenas uma situação de alerta ou um falso positivo;
- **Permitir a criação de regras de atuação mais precisas contra o ataque identificado:** um sistema de tomada de decisões, além de informações sobre o ataque, tem como

parâmetro o índice de certeza sobre a anomalia detectada, facilitando a ponderação na atitude de combate ser executada;

- **Permitir uma visualização do comportamento de anomalias:** os modelos comportamentais permitem visualizar, por meio de grafos ou matrizes de transição, o comportamento de ataques e situações anômalas em um sistema computacional. Diferentes análises e algoritmos podem ser aplicados a essa base de conhecimento;
- **Detectar anomalias não conhecidas:** a utilização dos modelos MHMM, Seqüencial e técnicas como a de Wei Fan (2001) e subMHMM apresentaram bons resultados para detecção de ataques não conhecidos;
- **Implementação em diferentes plataformas:** a CISS foi implementada em hardware, software e uma versão simulada.

CAPÍTULO 8

CONCLUSÕES E TRABALHOS FUTUROS

8.1 Conclusões

Atualmente, existe uma forte preocupação em tornar um conjunto de serviços de segurança em um único sistema integrado, uma vez que a possível fragilidade de um serviço pode ser compensada por outros. Apesar dessa necessidade, a maioria das abordagens encontradas na literatura almeja a integração de uma ou várias ferramentas de segurança, mas enfatizam geralmente, em uma única estratégia, pois além da complexidade inerente em oferecer uma possível integração é necessário que exista relação entre os dados que são usados e gerados por diferentes mecanismos de segurança.

Nesse sentido, esta tese de doutorado criou e implementou uma Camada de Integração de Serviços de Segurança, chamada de CISS. A CISS pode ser definida como uma aplicação responsável por armazenar, organizar e relacionar eventos anômalos, notificados por serviços de segurança presentes em um sistema computacional. Para isto, estes eventos devem ser notificados no formato IDMEF e analisados pelos modelos comportamentais MHMM e Modelo Seqüencial, com o objetivo de detectar anomalias de forma eficiente, prematura e preventiva.

A CISS foi implementada em SoC, utilizando um FPGA Virtex II Pro, e em software multiplataforma, descrito em linguagem Java. Na CISS, foram implementados e integrados serviços de segurança como Firewall, IDS, Antivírus, ferramentas de autenticação, ferramentas proprietárias e serviços de criptografia.

As implementações da CISS, destacadas anteriormente, foram testadas e validadas variando as formas de ataques executadas durante o período de treinamento e construção dos modelos comportamentais. Estas variações foram concebidas aplicando modificações nos fatores tempo e seqüência dos eventos anômalos executados (ataques).

Partindo deste principio, quatro variações foram implementadas e executadas para avaliar a CISS: o modelo completo, sem variar tempo e seqüência; modelo parcial por tempo; parcial por seqüência; e parcial por tempo e seqüência; Por meio destes vetores de testes, foi possível gerar os dados referentes ao impacto na detecção de anomalias destacados nesta tese.

Melhorias foram implementadas para atingir um maior nível de detecção para ataques conhecidos e desconhecidos, destacando a criação do método subMHMM (ver seção 3.4.3) que atua em modo de execução normal e a aplicação do algoritmo de WEI FAN(2001) no período de treinamento para aumentar a robustez do modelo comportamental MHMM.

A análise do impacto da detecção de anomalias mostrou que a CISS é uma ferramenta de integração importante para sistemas computacionais, uma vez que por meio desta, é possível classificar seqüências de eventos anômalos e julgar em modo de execução normal rotinas de ataques.

Na seqüência, algumas contribuições da tese:

- **Integração de serviços de segurança:** integração utilizando uma estrutura comum com base no padrão IDMEF, adaptando as características necessárias para a CISS. Por meio desta estrutura comum, foi possível relacionar por tempo e por seqüência eventos de segurança de ferramentas distintas, integrando-as e caracterizando situações anômalas em um sistema computacional;
- **Criação do CISS-SoC:** muitos serviços de segurança são desenvolvidos em hardware, uma vez que o volume de dados que devem ser analisados é grande e os algoritmos de segurança podem ser complexos. Neste contexto, a versão da CISS-SoC vem de

encontro a estas necessidades, provendo um ambiente de melhor desempenho para a detecção de situações anômalas em hardware. No CISS-SoC, foram implementados todos os recursos propostos para CISS (ver capítulo 3) e serviços de segurança como firewall, antivírus, IDS e Criptografia;

- **Criação do Simulador:** o simulador é uma ferramenta importante para teste de maior complexidade em um menor tempo. Informações geradas pelo simulador podem ser incorporadas a sistemas reais o que diminui o tempo de treinamento e torna o modelo comportamental usado seja mais robusto;
- **Melhoria do MHMM proposto por Yasami:** o modelo MHMM de Yasami foi incorporado na CISS, porém foram adicionados tratamentos a situações não previstas e adaptado para a geração de modelos de comportamentos de anormalidade, diferente do proposto por Yasami (2007). Neste caso, pode-se destacar o tratamento para ocorrência do Estado X;
- **Inclusão da técnica de subMHMM:** técnica utilizada para o aprendizado em tempo real, com regras de precisão e qualidade para a geração de novos modelos comportamentais baseada no *threshold* e número de recorrência do SubMHMM.

Em relação ao impacto da aplicação dos modelos comportamentais para detecção de anomalias, pode ser visualizado nos resultados alcançados, que dependendo do tipo da seqüência de ataque, a CISS tem precisões diferentes de reconhecimento.

Neste sentido, pode-se a afirmar que para os testes realizados, o MHMM obteve os seguintes resultados: para ataques completos obteve taxas acima de 96%; para ataques parciais por tempo taxas acima de 80%; para ataques parciais por seqüência taxas acima de 96% e para ataques parciais por tempo e seqüência taxas acima de 54%.

Quando o ADA analisado é o modelo seqüencial, observa-se que os testes criados

obtiveram as seguintes porcentagens de detecção: para ataque completo, taxas acima de 80%; para ataques parciais por tempo, taxas acima de 92%; para ataques parciais por seqüência, taxas acima de 96% e para ataques parciais por tempo e seqüência, taxas acima de 32%.

No entanto, é importante ressaltar que quando aplicada a técnica de subMHMM e o algoritmo de WEI FAN (2001), as taxas para ataques parciais por tempo e seqüência obtiveram taxas de detecção de 91% e 66%, respectivamente.

8.2 Aplicabilidade da CISS

Após os testes e análises, percebe-se que a CISS deve atuar em dispositivos centrais de uma rede de computadores, como servidores. Estes ambientes são alvos de ataques por conter informações e serviços que despertam interesses alheios.

Servidores são administrados por pessoas capacitadas, que podem fazer da CISS um aliado na proteção e prevenção de anomalias. Uma ferramenta de segurança mal configurada ou mesmo ações em um sistema sem técnica ou metodologia, podem fazer das próprias ferramentas de segurança uma dificuldade para a gestão do sistema computacional.

O uso correto da CISS permitirá ao administrador entender mais sobre o próprio sistema computacional e seu comportamento em situações anômalas, podendo facilitar a tomada decisões precisas para erradicação ou diminuição de incidentes. Como já salientado, a CISS não substitui nenhuma ferramenta de segurança. Esta foi criada para potencializar o uso de recursos disponíveis em um sistema computacional.

Nada impede o uso da CISS como um *personal* CISS, ou seja, a instalação em um dispositivo comum da rede, onde o usuário convencional pode fazer uso dos benefícios da integração de serviços de segurança. Apenas com a ressalva que a CISS, deveria ser transformada em uma aplicação específica para este fim, sendo que muitas funções deveriam

ser automatizadas.

A versão CISS-SOC permitiu criar em um único chip uma estrutura robusta e eficiente de integração de serviços de segurança. Por meio desta implementação é possível integrar serviços de segurança incluídos no próprio SoC, como foi apresentado neste trabalho, assim como serviços de segurança externos ao chip. A capacidade de processamento do CISS-SoC depende exclusivamente do desempenho dos processadores embarcados, dos módulos dedicados de processamento e interface e da eficiência do software embarcado. Neste sentido, melhorias podem ser realizadas no CISS-SoC, para aumentar sua capacidade de processamento (ver seção 8.3).

A versão em software da CISS (JCISS) destaca-se por sua portabilidade e facilidade de integração de serviços de segurança. Algumas ferramentas de segurança em software como o IDS Snort, já possuem *plug-ins* para gerar informações no formato IDMEF, o que evidencia a possível adoção deste padrão por outras ferramentas de segurança. As limitações da JCISS são em relação ao desempenho e capacidade de análise de eventos anômalos. No entanto, algumas soluções são destacadas na seção 8.3, com a criação de uma versão distribuída ou de uma versão em código nativo.

8.3 Trabalhos Futuros

Como ações futuras para evolução da CISS, destacam-se itens imediatos que estão relacionadas à integração de novos serviços de segurança e a criação de uma versão distribuída para CISS.

Para a inserção de novos serviços de segurança, faz-se necessária a modificação ou criação de *plugins* para a geração de notificações de anomalias no padrão IDMEF e tratamento na CISS.

Apesar de não ser uma tarefa difícil, existem inúmeras ferramentas e serviços de

segurança, e muitas delas são proprietárias, o que torna não trivial adaptar um número grande e suficiente de serviços.

Neste contexto, para promover o padrão IDMEF e a CISS, a última pode ser preparada para ser uma aplicação *opensource* e distribuída através de sites de acesso público na Internet. No mesmo sentido, espera-se que o padrão IDMEF seja homologado e adotado por mais ferramentas comercias e proprietárias.

Atualmente, a CISS prevê ataques a um dispositivo especifico da rede, porém futuramente poderá ser criada uma versão distribuída para análise e comportamento de anomalias em uma rede de computadores completa.

Na versão JCISS pode-se melhorar consideravelmente o desempenho transformando a versão atual que é completamente interpretada pela máquina virtual Java (JVM), para uma versão em código nativo. Para isso, o próprio Java disponibiliza o compilador JIT (*just-in-time*) que transforma *bytecodes* em código de máquina. O impacto direto desta técnica é a perda da portabilidade da aplicação compilada. No entanto, é possível compilar a aplicação para diferentes plataformas sem nenhum esforço adicional ou modificação em código.

Em relação à versão em SoC, sugere-se criar módulos em hardware dedicado para o processamento dos algoritmos de detecção de anomalias (ADA), uma vez que estes são fatores determinantes e limitantes para a capacidade de processamento do CISS.

Para o simulador seria interessar adicionar melhorias em relação a usabilidade e automação de tarefas, que visem melhorar o tempo e criação de ambientes de simulação e uma melhor visualização dos dados alcançados. Para isso deve ser criada também uma documentação formal (*javadoc*) da API JCISS para que outros projetos possam utilizá-la.

Com relação aos modelos comportamentais, espera-se que a criação dessa primeira proposta e implementação da CISS, seja utilizada como escopo para outros projetos e pesquisas que almejam aperfeiçoar e até aplicar novos algoritmos de detecção de anomalias.

REFERÊNCIAS

3COM, Datasheet USP, http://www.3com.com/other/pdfs/products/en_US/3com_401013.pdf, acesso em 15/02/2009.

AMOROSO, E. G. Fundamentals of Computer Security Technology, Prentice Hall PTR, Upper Saddle River, NJ, 1994.

ANDROULIDAKIS, G. PAPAVASSILIOU, S., Improving network anomaly detection via selective flow-based sampling, Institution of Engineering and Technology (IET), Vol. 2, No. 3, pp. 399–409, 2008.

ATTIG, M. LOCKWOOD, J., SIFT: snort intrusion filter for TCP, IEEE High Performance Interconnects, 2005.

BAKER, M.P, Integrated security system, Proceedings International Carnahan Conference on Security Technology, 1989.

BELL, D. E.; LAPADULA, L. J. Secure computer systems: Mathematical foundations and model, Technical Report ESD–TR–278, 1974.

BELL, D. E LAPADULA L. "Secure Computer System: Unified Exposition and Multics Interpretation". Technical Report MTR-2997 Rev. 1, MITRE Corporation, Bedford, MA, 1976.

BIBA, K. J. Integrity Considerations for Secure Computer Systems,Technical Report TR–3153, 1977.

BOLZONI, D., Revisiting Anomaly-based Network Intrusion Detection Systems. PhD thesis, University of Twente. CTIT Ph.D.-thesis series No. 09-147 ISBN 978-90-365-2853-5, 2009.

BRIDGES, S. M., VAUGHN, R. B., Intrusion Detection Via Fuzzy Data Mining. Proc. of 12th Annual Canadian Information Technology Security Symposium, June 19-23, Ottawa, Canada, pages 109-122, 2000.

BUNKE H., CAELLI, T., Hidden Markov Models: Applications in Computer Vision ISBN: 9810245645, World Scientific Publishing, 2001.

CAPPÉ, O., MOULINES, E., Inference in Hidden Markov Models, Editora: Springer, ISBN: 0387402640, 2005.

CHANG, C., HUANG, C., TAI, H., LIN, M., 8-bit AES Implementation in FPGA by Multiplexing 32-bit AES Operation, ISDPE, pp.505-507, The First International Symposium on Data, Privacy, and E-Commerce, 2007

CLAMAV, Site Oficial clamAv, http://www.clamav.net/, acesso em 29/04/2009.

CLAMWIN, Site oficial, http://www.clamwin.com/, acesso em 29/04/2009.

CLAMWIN SOURCES, Diretórios com ultimas atualizações do código fonte https://clamwin.svn.sourceforge.net/svnroot/clamwin/, acesso em 29/04/2009.

CLARK, D. D., WILSON, D. R.; A Comparison of Commercial and Military Computer Security Policies; in Proceedings of the IEEE Symposium on Research in Security and Privacy (SP'87), Oakland, CA; IEEE Press, pp. 184–193, 1987.

CERT, Cartilha de segurança para internet, http://cartilha.cert.br/, revisão 3.1, 2007.

CIS GROUP, Cryptography and Information Security Group (CIS Group) , http://groups.csail.mit.edu/cis/, acesso em 20/04/2009.
CISCO, Datasheets Cisco ASA http://www.cisco.com/en/US/products/ps6120/ products_data_sheets_list.html, acesso em 15/02/2009.

DEBAR, H., CURRY D., FEINSTEIN B., The intrusion detection message exchange format, disponível em http://www.rfc-editor.org/rfc/rfc4765.txt, 2007.

DAEMEN, J., BORG, S., RIJMEN V., The Design of Rijndael: AES - The Advanced Encryption Standard. Springer-Verlag, ISBN 3-540-42580-2, 2002.

DIES, Distributed and Embedded Security Research Group, http://dies.cs.utwente.nl/, acesso em 29/05/2009.

FERRAIOLO, D. F., SANDHU, R., GAVRILA, S., KUHN, D. R., CHANDRAMOULI, R., Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security, v. 4, n. 3, p. 224-274, ago. 2001.

FIPS197, Federal Information Processing Standards Publication 97, ADVANCED ENCRYPTION STANDARD (AES), Novembro, 2001.

HAMMOUDA, K., KARRAY, F., A Comparative Study of Data Clustering Techniques, University of Waterloo, Academic Report, 2004.

HIBERNATE, Site official Hibernate Persistence, https://www.hibernate.org/, acesso em 02/05/2009.

JONSSON, E., Towards an integrated conceptual model of security and dependability, Availability, Reliability and Security, ARES, 2006.

JOSHI, S. S. and PHOHA V. V., Investigating hidden Markov models capabilities in anomaly detection, ACM Southeast Regional Conference Proceedings of the 43rd annual Southeast regional conference, 2005.

JUNIPER, Datasheet ISG, www.juniper.net/us/en/local/pdf/datasheets/*1100036-en.pdf* , acesso em 15/02/2009.

KOC, C.K., Pesquisas na área de segurança de informações http://islab.oregonstate.edu/ koc/index.html, acesso em 17/02/2009.

LANDWEHR, C. E. Formal Models for Computer Security, ACM Computing Surveys, 13(3): p. 247–278, 1981.

LEITE, P.B.C., Identificação de Tipos de Culturas Agrícolas a partir de Seqüências de Imagens Multitemporais Utilizando Modelos de Markov Ocultos, Dissertação de Mestrado, PUC-RIO, 2008.

LI, H., GUAN, X.H., ZAN, X., et al, Network intrusion detection based on support vector machine, Journal of Computer Research and Development, vol.40, no.6, pp.799-807, 2003.

LANGHAMMER, F. J., RSA & Public Key Cryptography in FPGAs, Altera document, 2005.

MICHALSKI, A., BUELL, D., A Scalable Architecture for RSA Cryptography on Large FPGAs, Field-Programmable Custom Computing Machines, FCCM 14th Annual IEEE Symposium, 2006.

MONTA VISTA, Monta Vista Embedded Linux Software, disponível em http://www.mvista.com/, 2009.

MONTGOMERY, P.L., Modular multiplication without trial division, Math. Computation, 44:519-521, 1985.

MOSES, T., eXtensible Access Control Markup Language(XACML) Version 2.0. OASIS, February 2005.

NISSANKE, N., An integrated security model for component-based systems, IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2007, pp. 638-645, 2007.

OKAMOTO, E, Proposal for integrated security systems, Proceedings of the Second International Conference on Systems Integration ICSI '92, 1992.

PAAR, C., COSY - COmmunication SecuritY, http://www.crypto.ruhr-uni-bochum.de/en_portrait.html, acesso em 29/05/2009.

PEREIRA, F. D. ; ORDONEZ, E. D. M., SSDR - FIREWALL RECONFIGURÁVEL: ARQUITETURA E DESEMPENHO EM FPGA. In: XIII-Workshop Iberchip, 2007, Lima/Perú. XIII-Workshop Iberchip, 2007.

PEREIRA, F. D. ; ORDONEZ, E. D. M., SSDR - Reconfigurable Firewall: Reconfiguration Model Impact. In: ISA2008 - Information Security and Assurance, Busan, Korea. IEEE proceeding: International Conference on Information Security and Assurance, 2008a.

PEREIRA, F. D. ; ORDONEZ, E. D. M., SSDR-Reconfigurable Firewall: Reconfiguration Model Performance. In: SPL'2008- IV Southern Programmable Logic Conference, Bariloche, IEEE proceeding IV Southern Programmable Logic Conference, 2008b.

PEREIRA, F. D. ; ORDONEZ, E. D. M., A Hardware Architecture for Integrated-Security Services. Lecture Notes in Computer Science, v. 1, p. 100-114, 2009.

PPC405 XILINX, PPC405 (Wrapper), Product Overview DS422, 2004

RABINER, L.R.. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proc. IEEE, vol. 77, No.2, 1989.

RASHEED H., CHOW Y.C.R., An Information Model for Security Integration, 11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'07), pp.41-47 2007

RIVEST, R., SHAMIR, A., ADLEMAN L., A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Vol. 21 (2), 1978.

SANDHU, R. S. Lattice Based Access Control Models, IEEE Computer, 26(11):p.9–19, 1993

SCHAUMONT, P., Secure Embedded Systems Group, http://www.ece.vt.edu/schaum/ research.html, acesso em 29/05/2009.

SHIREY, R.t W. RFC 2828: Internet Security Glossary, 2000. Disponível em: < http://www.ietf.org/rfc/rfc2828.txt>.

SNORT, Documentação Snort IDS, http://www.snort.org/start/documentation, acesso em 02/05/2009.

SONICWALL, Especificação familia TZ, http://www.sonicwall.com/us/products/ TZ_180.html, acesso em 15/02/2009

TANENBAUM, Andrew S. Redes de Computadores. Editora Campus, 4º Edição. 2003
LIM, D., PEATTIE, M.: Two Flows for Partial Reconfiguration: Module Based or Small Bit Manipulations. Xilinx XAPP290 (2002)

VEERAVALLI, A.G. PAN, W.D. ADHAMI, R. Cox, P.G., A tutorial on using hidden Markov models for phoneme recognition, 2005, SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory.

VERBAUWHEDE, I., UCLA EMBEDDED SECURITY GROUP (EmSec) http://www.emsec.ee. ucla.edu/index.html, acesso em 17/02/2009.

XILINX, Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, Product Specification DS083, 2007.

XILINX, PowerPC 405 Processor Block Reference Guide, Embedded Development Kit, document: ug018, 2008a.

XILINX, Spartan-3AN FPGA Family Data Sheet DS557, 2008b

ZAMBRENO, J., GUYEN, D., CHOUDHARY, A., Exploring Area/Delay Tradeoffs in an AES FPGA Implementation, LNCS 3203, Springer-Verlag, pp. 575–585, 2004.

ZILYS, M.; VALINEVICIUS, A.; EIDUKAS, D., Optimizing strategic control of integrated security systems, 26th International Conference on Information Technology Interfaces, 2004.

YANG, C., DENG F., HAIDONG, Y., An Unsupervised Anomaly Detection Approach using Subtractive Clustering and Hidden Markov Model IEEE International Conference on Communications and Networking in China, CHINACOM, 2007.

YASAMI Y., FARAHMAND, M., ZARGARI, V., An ARP-based Anomaly Detection Algorithm Using Hidden Markov Model in Enterprise Networks, IEEE Second International Conference on Systems and Networks Communications (ICSNC 2007), 2007.

WATCHGUARD, Datasheet Firebox, http://www.watchguard.com/docs/datasheet/ edge_ds.asp, acesso em 15/02/2009.

WEI FAN, MILLER, M., STOLFO, S. J., LEE, W., CHAN P. K., Using Artificial Anomalies to Detect Unknown and Known Network Intrusions, First IEEE International Conference on Data Mining (ICDM'01), 2001.

WHITMAN, M., MATTORD, H., Principles of Information Secuirty. Thomson, 2009.