Marcos Pirmez

# *Prometheus*

# *Um Serviço de Segurança Adaptativa*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, IM/NCE, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Orientadores: Luiz Fernando Rust da Costa Carmo, D.SC.
Luci Pirmez, D.Sc.

.

Rio de Janeiro

2009

P669 Pirmez, Marcos.
Prometheus: um serviço de segurança adaptativa / Marcos Pirmez. – 2009.
114 f. il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, 2009.

Orientadores: Luiz Fernando Rust da Costa Carmo; Luci Pirmez

1. Computação Ubíqua –Teses. 2. Segurança em Redes - Teses.
3. Ciência de Contexto - Teses. 4. Segurança Adaptativa – Teses
I. Luiz Fernando Rust da Costa Carmo (Orient.). II. Luci Pirmez (Orient.).
III. Título

CDD

Marcos Pirmez

# *Prometheus:*

# *Um Serviço de Segurança Adaptativa*

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática do Instituto de Matemática e do Núcleo de Computação Eletrônica, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do grau de Mestre em Informática.

Aprovada em: Rio de Janeiro, 14 de agosto de 2009.

_____

Prof. Luiz Fernando Rust da Costa Carmo, Docteur, NCE/UFRJ (Orientador)

_____

Prof ª. Luci Pirmez, D.Sc., NCE/UFRJ (Orientadora)

_____

Prof ª. Flavia Coimbra Delicato, D.Sc., UFRN

_____

Prof. Artur Ziviani, Ph.D., LNCC

_____

Prof. Adriano Joaquim de Oliveira Cruz, Ph.D., NCE/UFRJ

À minha amada

que ilumina meu caminho todos os dias da minha vida.

# Agradecimentos

Agradeço a meus professores pela oportunidade que me deram ao compartilhar comigo sua experiência e conhecimento, e a meus orientadores, Luiz Fernando Rust da Costa Carmo e Luci Pirmez, pela excepcional dedicação a esse trabalho.

Agradeço especialmente à professora Luci Pirmez pelas muitas horas de conversa, pelas críticas, pelos ensinamentos sobre redes, sobre segurança, pela ajuda inestimável na organização do texto da dissertação e, principalmente, por sua enorme paciência comigo.

Agradeço aos professores Paulo Pires e Maria Luiza Machado que me mostraram um mundo de tecnologias que eu desconhecia completamente. Agradeço à professora Flavia Delicato pela parceria na elaboração dos artigos e pelas críticas, sempre construtivas, a meu trabalho.

Agradeço a meus colegas pelas agradáveis tardes (e noites!) que passamos juntos estudando, principalmente ao Marcelo Pitanga, nos primeiros passos em MDA, ao Carlos Alves e ao Mario Nadai nas tarefas de Engenharia de Software e Análise de Risco.

Agradeço aos funcionários do IM e do NCE pela presteza e competência na ajuda que me deram no tratamento dos detalhes, tão importantes, para concluir o curso. Em particular, agradeço a Selma Regina Mendes Martins pela revisão nas Referências Bibliográficas e na formatação final da dissertação.

Agradeço a meus pais que me despertaram a vontade de conhecer sempre mais.

Agradeço a meus filhos, Rodrigo e Carolina, que compreenderam a importância desse momento para mim, e não me cobraram as muitas horas que não dediquei a eles.

Agradeço finalmente à minha esposa querida que tanto me apoiou nessa jornada.

# Resumo

PIRMEZ, Marcos. **Prometheus**: um serviço de segurança adaptativa. 2009. 114 f. Dissertação (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

A ampla disponibilidade de redes sem fio e a proliferação de dispositivos portáteis (celulares, laptops, PDAs, etc) aliados à computação móvel levaram ao surgimento de aplicações e ambientes de computação ubíquos. Essas aplicações podem trazer uma importante vantagem competitiva para as empresas. O ambiente ubíquo, entretanto, impõe uma série de desafios ligados ao desenvolvimento e à execução de aplicações em dispositivos móveis. Um dos principais desafios é o de garantir o cumprimento da Política de Segurança da empresa num ambiente em que os Requisitos de Segurança da aplicação mudam conforme o contexto. Este trabalho propõe um Serviço de Segurança ciente de contexto. Este serviço é capaz de acionar Controles de Segurança dinamicamente conforme o contexto em que as aplicações estão executando. Esses Controles de Segurança são acionados num processo adaptativo, efetuado de forma transparente para a aplicação e para o usuário.

# Abstract

PIRMEZ, Marcos. **Prometheus**: um serviço de segurança adaptativa. 2009. 114 f. Dissertação (Mestrado em Informática) - Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

The wide availability of wireless networks and the proliferation of portable devices (cell phones, laptops, PDAs, etc) allied to mobile computing leads to the emergence of ubiquitous applications and computing environments. These applications can provide an important competitive advantage to corporations. The ubiquitous environment, however, imposes many challenges related to the development and to the execution of such applications. One of the main challenges is to fulfill the company Security Policy in an environment where the applications Security Requirements change according to the context. This work proposes a context aware security service. This service is able to trigger Security Controls dynamically according to the context the applications are executing in. These Security Controls are triggered in an adaptive environment, in a transparent way to the application and to the user.

# Lista de Figuras

# Lista de Tabelas

# Sumário

# 1 Introdução

O avanço tecnológico das redes sem fio e a recente proliferação de dispositivos portáteis (celulares, laptops, PDAs, etc.), aliados ao aumento de popularidade da computação móvel, levaram ao surgimento de aplicações e ambientes de computação ubíquos [Weiser 1991]. Esses ambientes possuem características que originam uma série de desafios ligados ao desenvolvimento e à execução de aplicações em dispositivos móveis.

Um importante desafio está relacionado ao desenvolvimento de aplicações ubíquas. Aplicações ubíquas devem levar em conta as constantes mudanças dos requisitos dos usuários, da aplicação e da própria rede em seu estado de execução a fim de contornar ou amenizar o impacto dessas alterações em sua operação. Essas mudanças fazem com que as aplicações ubíquas necessitem realizar alterações em seu comportamento para se adaptar às novas condições de execução. Para tal, essas aplicações precisam ser cientes de seu contexto de execução. Um sistema é *ciente de contexto* [Hoh 2006] se ele usar o contexto para prover informações ou serviços ao usuário. *Contexto*, segundo [Springer 2006], é qualquer tipo de informação acerca do ambiente de execução de uma aplicação que possa ser usada para aprimorar seu comportamento fazendo-a operar de maneira adaptativa, personalizada, autônoma e flexível. Como não é desejável que o funcionamento de aplicações ubíquas seja interrompido para realizar adaptações nas próprias aplicações, é necessário que tais adaptações sejam feitas de forma dinâmica e transparente. Dessa necessidade, surge o conceito de adaptação dinâmica, que consiste em alterar o comportamento da aplicação, isto é, do código que implementa as funcionalidades providas para os usuários, durante sua execução e de forma transparente ao usuário ([Zhang 2004], [Zhang 2005], e [Zhang 2006]).

## 1.1 Motivação

Além das aplicações operarem em ambientes de Computação Ubíqua, caracterizada por constantes mudanças em seu estado de execução, causadas, dentre outros fatores, pela mobilidade de seus usuários; por suas diferentes preferências; pelas características de seus dispositivos; e pela variabilidade dos recursos disponíveis tanto na rede quanto nos próprios dispositivos, é necessário garantir um ambiente seguro para essas aplicações.

Assim, um outro importante desafio relacionado ao dinamismo e a heterogeneidade de ambiente de execução, relaciona-se à execução propriamente de aplicações ubíquas, particularmente no tocante a questões de segurança. Trata-se de como gerenciar os requisitos de disponibilidade, integridade e confidencialidade da aplicação em presença de um ambiente altamente dinâmico e heterogêneo que faz com que controles de segurança sejam diferentes

conforme o contexto corrente. As propriedades dinâmicas e heterogêneas dos ambientes relacionadas ao contexto de segurança se referem aos requisitos da aplicação, da rede e dos dispositivos. Observa-se na literatura que a importância de prover requisitos de segurança adaptativos não é exatamente uma novidade ([Hinton 1999]; [Izquierdo 2007]). Entretanto, a proliferação de redes sem fio e seu uso por dispositivos portáteis impõem um novo patamar no tratamento da segurança. Uma mudança no ambiente de execução que acarrete um aumento do risco de incidentes de segurança (ex: aumento de vulnerabilidades), é razão suficiente para implicar um acréscimo de controles de segurança no próprio dispositivo (comportamento pró-ativo) de forma a garantir um ambiente seguro para as aplicações ubíquas. Um outro desafio relacionado à execução de aplicações ubíquas consiste em tratar eventos que não foram originalmente considerados durante o projeto de um serviço de segurança. Não é possível, em tempo de projeto, antecipar todas as respostas que um serviço de segurança ciente de contexto pode dar frente à ocorrência de eventos ainda não previstos, uma vez que novas vulnerabilidades são descobertas todos os dias.

Para superar esses desafios, todo o alicerce de sustentação dos requisitos de segurança das aplicações deve ser adaptado segundo o ambiente de execução, levando-se em conta propriedades dinâmicas e heterogêneas que reflitam os diferentes ambientes computacionais envolvidos. Isso envolve uma manipulação em tempo real de: (i) Controles de Segurança (mecanismos utilizados para mitigar os riscos encontrados em um contexto); e (ii) as próprias Políticas de Adaptação usadas para determinar quais são os controles de segurança adequados para cada contexto. O presente trabalho enfoca especificamente questões relativas à automatização das adaptações necessárias à sustentação dos requisitos de segurança das aplicações ubíquas.

## 1.2 Objetivos

Neste cenário de grandes e acelerados avanços tecnológicos, constata-se que os problemas de segurança passam a ter dimensões maiores quando redes sem fio estão sendo usadas concomitantemente com a mobilidade. Em face do dinamismo do contexto de execução associado principalmente à mobilidade dos dispositivos, questões e tarefas relacionadas à adaptação de políticas e controles de segurança também se tornam mais complexas.

Assim, o objetivo desse trabalho é apresentar um Serviço de Segurança Adaptativa, denominado Prometheus. O Prometheus busca adaptar em tempo de execução o conjunto de ***controles de segurança*** e as ***próprias políticas de adaptação*** durante a execução de

aplicações nos dispositivos móveis, de forma a assegurar a disponibilidade, a confidencialidade e a integridade dessas aplicações em ambientes de computação ubíqua. Assim, o Prometheus é um serviço de segurança adaptativo, cujas principais funcionalidades são:

(i) adaptação de políticas de segurança dinamicamente de acordo com o contexto de execução e segundo um conjunto de políticas de adaptação previamente definidas, e

(ii) adaptação dinâmica dos controles de segurança utilizados para mitigar as vulnerabilidades de acordo com o contexto de execução;

As adaptações são disparadas por mudanças de contexto e norteadas por políticas de adaptação previamente definidas. Assim, controles de segurança são definidos para cada contexto de execução e políticas de adaptação são determinadas para definir quais controles de segurança devem ser usados para um dado contexto. Uma forma de adaptar políticas de adaptação e controles de segurança em tempo de execução é permitir que tais políticas e controles sejam trocados devido a mudanças no contexto de execução. Vale ressaltar que a adaptação das próprias políticas de adaptação permite ao Prometheus tratar a imprevisibilidade em tempo de projeto de mudanças ambientais, por exemplo, ameaças que não existiam no momento da construção da aplicação.

É importante destacar que adaptação dinâmica propriamente dita introduz alguns problemas em virtude de permitir mudanças não antecipadas tanto na aplicação como nas políticas e nos controles de segurança, ou seja, alterações não previstas pelos desenvolvedores durante a fase inicial de projeto. É necessário que as adaptações em virtude de alterações não antecipadas sejam introduzidas de forma organizada e modularizada, com o intuito de minimizar os impactos da alteração na própria aplicação como um todo. Em particular, é necessário definir abstrações e modelos que permitam administrar a complexidade introduzida pela possibilidade de alterar dinamicamente as aplicações e as políticas e controles de segurança.

No Prometheus foi adotada uma abordagem baseada em componentes uma vez que o modelo de componentes de software define abstrações que o tornam bastante adequado para a construção de sistemas dinamicamente adaptáveis [Canal 2006], minimizando os impactos da alteração na própria aplicação e nos controles de segurança e políticas de adaptação. Em suma, essas abstrações foram definidas nos modelos de componentes com o intuito de administrar a complexidade dos sistemas e permitir uma rápida construção de aplicações

através da composição de componentes, além de facilitar a manutenção desses sistemas devido às dependências bem definidas e o desacoplamento dos componentes. Essas características contribuem para que o modelo de componentes de software seja bastante adequado para a construção de sistemas dinamicamente adaptáveis. Diversos trabalhos propõem mecanismos de adaptação dinâmica baseados no modelo de componentes [Maia 2007], [Moura 2002], [Silva 2003] e [Robertson 2001].

## 1.3 Relevância

A concentração nessa frente de investigação, explorando a segurança num ambiente de computação ubíqua, foi seguramente proposital, e visa alavancar o domínio sobre essas tecnologias emergentes (segurança e adaptação dinâmica) e gerar contribuições significativas dada a atual demanda das comunidades científica e empresarial.

Esse trabalho se enquadra no quinto desafio apontado no relatório que sintetiza os resultados do Seminário "Grandes Desafios da Pesquisa em Computação no Brasil – 2006 – 2016" da SBC [SBC 2006]. O objetivo do seminário de Grandes Desafios de Pesquisa é definir questões de pesquisa que serão importantes para a ciência e para o país no longo prazo.

O quinto desafio é denominado de “Desenvolvimento Tecnológico de Qualidade: sistemas disponíveis, corretos, seguros, escaláveis, persistentes e ubíquos” e trata dos seguintes pontos principais, descritos a seguir:

- "A Tecnologia da Informação está cada vez mais presente em nosso quotidiano. ... Se esta ubiqüidade traz conforto, também acarreta problemas. Como dependemos desses sistemas, eles precisam estar sempre disponíveis e não apresentarem falhas; devem funcionar da forma prevista e ser escaláveis e seguros. Este desafio visa, desta forma, a pesquisa em ambientes, métodos, técnicas, modelos, dispositivos e padrões de arquitetura e de projeto capazes de auxiliar os projetistas e desenvolvedores de grandes sistemas de software e hardware a atingirem esses objetivos."
- "As várias propriedades abrangidas por este desafio foram agrupadas, ..., em um novo termo – computação onivalente – cunhado para designar a ubiqüidade associada à segurança, fidedignidade e evolução de sistemas computacionais, em especial software. Cada uma dessas propriedades apresenta desafios de pesquisa; sua combinação, portanto, é um grande desafio."
- "Há inúmeros benefícios advindos da pesquisa neste desafio. Vários deles são decorrentes das vantagens resultantes do desenvolvimento de sistemas confiáveis e

seguros, contribuindo para a melhoria da qualidade e a redução dos custos de desenvolvimento e uso de software e sistemas computacionais em geral".

- "Falhas na segurança de dados, software ou sistemas acarretam também prejuízos financeiros e sociais, como aqueles causados por ataques a redes e vírus em todo o mundo. A violação da privacidade decorrente deste tipo de falha é outra conseqüência grave para o cidadão, com custos incalculáveis".

"A construção e manutenção de softwares robustos exige profissionais qualificados e compromissos de longo prazo, que permitam manter, expandir e gerenciar o software. Praticar engenharia de software fidedigno é bem mais amplo do que incorporar tolerância a falhas e assegurar corretude, entre outras propriedades".

## 1.4 Estrutura do Documento

Esta dissertação está estruturada em seis capítulos, na seguinte disposição. O Capítulo 2 apresenta os conceitos básicos com o intuito de facilitar a compreensão dos assuntos os quais foram utilizados no desenvolvimento do trabalho como: Provisão de contexto adaptação de Software, segurança da informação e segurança adaptativa. Ainda nesse Capitulo 2 são apresentados os trabalhos relacionados, descrevendo as suas semelhanças e diferenças com o Prometheus.

O Capítulo 3 apresenta o Prometheus através da descrição da arquitetura e dos componentes que compõem esta arquitetura e dos detalhes de sua implementação.

O Capítulo 4 descreve o estudo de caso desenvolvido para a validação da proposta do Prometheus, onde são expostas as características do cenário proposto bem como são descritos todos os passos realizados pelo Prometheus para tornar o ambiente seguro.

No Capítulo 5 são descritos os experimentos realizados, bem como a avaliação do Prometheus através da apresentação dos resultados obtidos nesses experimentos.

Por fim, o Capítulo 6 apresenta as conclusões e os trabalhos futuros.

# 2 Conceitos Básicos

Este capítulo apresenta os conceitos básicos que envolvem diversas áreas de pesquisa, incluindo Tecnologia da Informação, Segurança da Informação, Computação Ubíqua, Contexto, Adaptação Dinâmica, Segurança Adaptativa, e trabalhos relacionados que fundamentaram a pesquisa do Prometheus e o desenvolvimento do protótipo para a execução de testes e simulação. A seguir tais áreas serão brevemente descritas.

## 2.1 Segurança da Informação

A Informação pode ser considerada como um ativo essencial para indivíduos e para os negócios de uma organização. Como qualquer ativo importante, precisa ser preservada e protegida.

A Segurança da Informação está relacionada com a proteção de um conjunto de dados a fim de preservar seu valor para um indivíduo ou uma organização. Ou seja, toda informação possui um valor para seus usuários e precisa ser protegida contra acidentes e ataques. Exemplos de acidentes são: falha nos discos, cafezinho no laptop e terremoto. Invasão do sistema, roubo do laptop e falsificação são exemplos de ataques.

### 2.1.1 Propriedades

As três propriedades básicas que caracterizam a segurança da informação são resumidas pela sigla em inglês *CIA* (*Confidentiality*, *Integrity* e *Availability*), descritas em seguida.

- A **Confidencialidade** (ou Privacidade) é a proteção contra a leitura ou cópia de informação por quem não está explicitamente autorizado pelo seu proprietário.
- A **Integridade** é a proteção da informação ou sistema contra remoção ou modificação de qualquer espécie sem a permissão de seu proprietário.
- A **Disponibilidade** é a proteção de um serviço ou sistema contra ações que impossibilitem ou limitem o acesso à informação por pessoas devidamente autorizadas.

Outras propriedades freqüentemente ligadas à Segurança da Informação são: a Autenticidade, o Não Repúdio, a Consistência, o Controle de Acesso e a Auditoria:

- A **Autenticidade** é a certeza de que a informação fornecida provém das fontes anunciadas e que foi recebida na íntegra.
- O **Não Repúdio** ou Irretratabilidade é a impossibilidade de quem fornece uma informação, ou que executa uma ação, de negar tê-lo feito.

- A **Consistência** é a certificação de que um sistema se comporta como esperado por seus usuários autorizados. Geralmente essa propriedade é considerada um conceito complementar à Integridade.
- O **Controle de Acesso** consiste em regular o acesso a um sistema ou informação, ou seja, é a aplicação de um conjunto de regras para permitir ou não o acesso de usuários a uma informação ou a um sistema.
- A **Auditoria** é a possibilidade de rastrear as ações efetuadas pelos usuários. Da mesma forma que deve ser feito um controle para prevenir o acesso não autorizado a um sistema, deve também ser feito um controle dos usuários autorizados. Controles de auditoria devem manter históricos de acessos válidos para, eventualmente, verificar atividades irregulares executadas por usuários devidamente autorizados.

### 2.1.2 Regulamentação

O valor da Informação e a importância em protegê-la fica evidenciado pelo enorme esforço feito por instituições privadas e governamentais para orientar e regulamentar a Segurança da Informação. O Departamento de Defesa americano (*DoD*) patrocinou um estudo abrangente sobre Segurança da Informação. O resultado desse estudo foi publicado num documento chamado *Trusted Computer System Evaluation Criteria*, também conhecido como *Orange Book*. Embora tenha sido publicado pela primeira vez em 1983, esse estudo ainda hoje é uma referência importante para boa parte dos trabalhos desenvolvidos na área. A ISO (International Standards Office) desenvolveu a série de normas [ISO 27000] especialmente para tratar de segurança da informação.

Nos EUA, algumas indústrias, incluindo saúde (aplicações médicas) e de cartão de crédito, vêm-se com os seus próprios regulamentos e normas que regem privacidade e segurança. Por exemplo, o *Payment Card Industry* (PCI) *Data Security Standard* [PCI] foi um trabalho desenvolvido pela Visa e Master Card para melhorar a segurança em transações com cartões de crédito. O *Health Information Portability and Accountability Act* [HIPAA] é uma lei que regula a proteção dos prontuários médicos. O *Gramm-Leach-Bliley Act* [GLBA] (consolidação de bancos comerciais e de investimentos), que explicitamente estabelece regras quanto a privacidade das informações financeiras, planos de salvaguarda, sendo ainda o único a abordar a questão da precaução quanto a possíveis ataques de engenharia social.

O [FISMA] (*Federal Information Security Management Act of 2002*) é uma lei aprovada pelo congresso americano que dispõe sobre um conjunto de procedimentos

obrigatórios a serem seguidos pelos sistemas de informação utilizados ou operados por qualquer agência federal dos EUA.

Há uma quase infinidade de outros exemplos de regulamentação. Embora a maioria desses exemplos tenha origem nos Estados Unidos e na Europa, devido ao aspecto global dos negócios, seu impacto se faz sentir em toda parte.

### 2.1.3 Vulnerabilidade, Ameaças e Ataques

A vulnerabilidade é o grau em que as pessoas estão suscetíveis a perdas, danos, sofrimento ou morte em caso de um ataque. Em suma, a vulnerabilidade é a possibilidade de violação de alguma propriedade de segurança.

Uma ameaça é qualquer situação ou evento com algum potencial para causar dano a um sistema. Na medida em que os sistemas vão ficando mais complexos, vão ficando também mais sujeitos a ameaças. Manter os sistemas livres dessas ameaças constitui um enorme desafio. Não é possível para projetistas de sistemas ou administradores de se manter informado sobre o surgimento a cada momento de novas ameaças. Entretanto, conhecer os tipos de ataque já existentes pode ajudar a determinar o nível de proteção necessário a ser estabelecido no sistema.

A [RFC2828] e a norma [X.800] classificam os ataques em duas categorias, descritas em seguida.

- Ataques **Passivos** são aqueles que tentam aprender ou usar a informação de sistemas, sem afetar os recursos do sistema.
- Ataques **Ativos** são aqueles que tentam modificar os recursos ou afetar o seu funcionamento. As informações obtidas a partir de um ataque passivo podem ser usadas para executar um ataque ativo mais agressivo.

As ameaças podem ser classificadas quanto à origem do atacante em:

- **Externas**, quando o atacante está fora das fronteiras da instituição alvo;
- **Internas**, quando o ataque é originado dentro do perímetro de segurança da instituição, geralmente por pessoas que possuem privilégio de acesso, tais como funcionários mal intencionados, ou pessoas infiltradas que, através de técnicas de engenharia social, obtiveram acesso.

Segundo [Stallings 2003], os ataques podem ser divididos em quatro tipos básicos:

- ***Interrupção:*** o atacante impede a informação de chegar ao destinatário. Nesse ataque a propriedade de disponibilidade é violada (Figura 2.1b).

- ***Interceptação:*** o atacante acessa informações que não tem direito de acessar. Nesse caso, apenas a confidencialidade é violada (Figura 2.1c).
- ***Modificação:*** o atacante não só ganha acesso como também interfere ou modifica o conteúdo das mensagens que estão sendo transmitidas a fim de obter benefício ilícito. A integridade e a confidencialidade são violadas nesse tipo de ataque (Figura 2.1d).
- ***Personificação:*** o atacante emite uma mensagem como se fosse outra entidade. Fabricação é quando o atacante transmite uma nova e forjada mensagem. Repetição é quando o atacante emite uma mensagem anteriormente interceptada por ele mesmo. Nesse tipo de ataque é violada a propriedade de autenticidade (Figura 2.1e).

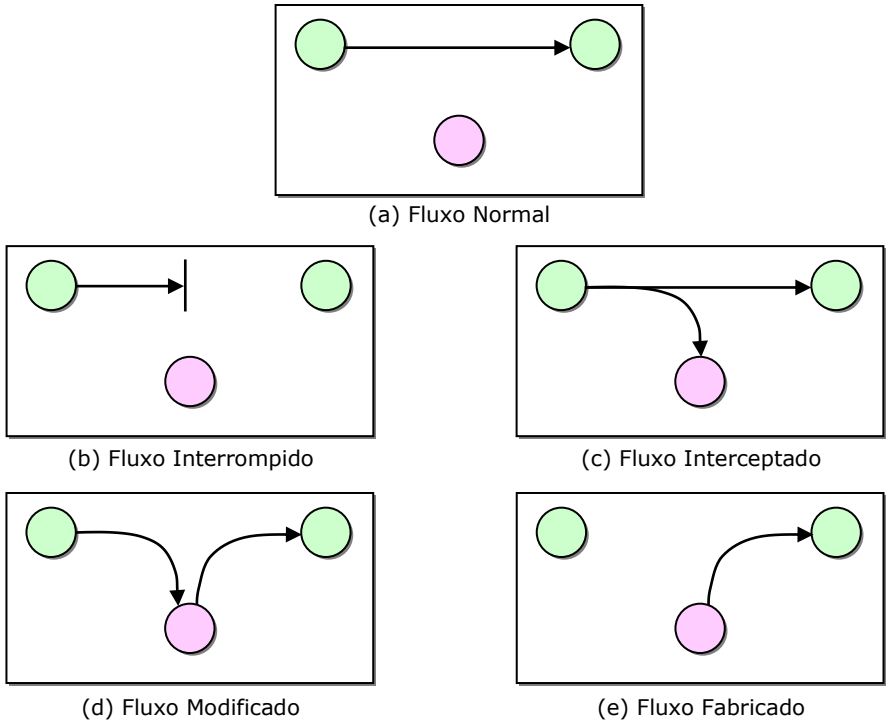


**Figura 2.1 Tipos de Ataque**

A Tabela 2.1 mostra algumas ameaças e possíveis contra medidas. A tabela não pretende de modo algum ser exaustiva, mas apenas ilustrar a diversidade das ameaças que hoje pairam sobre os sistemas de informação.

**Tabela 2.1. Ameaças e Contra Medidas**

| Tipo de Ameaça | Descrição | Contra Medidas |
|---|---|---|
| Vírus, Bombas, Vermes e Cavalos de Tróia | Programas maliciosos que podem causar vários tipos de dano ou executar operações indesejadas. | Antivírus e Firewall |

| | | |
|---|---|---|
| *Trap Doors* | Mecanismos construídos em aplicações por seus programadores para permitir acesso não autorizado. | Especificação rígida das interfaces e execução de testes; Firewall; e Auditoria. |
| Negação de Serviço / *Denial of Service* (DoS) | Pode ocorrer de várias formas, desde a destruição física de recursos até a inundação do servidor com tantas requisições que ele não é mais capaz de atender usuários válidos. | Proteção física dos recursos.<br><br>Mecanismos para detectar a inundação de requisições e bloquear ou ignorar esses pedidos. |
| *Snooping* / *Eavesdropping* (ataque passivo) | É a interceptação não autorizada de informações. Escuta de tráfego na rede. | Proteção física do meio de comunicação.<br>Uso de criptografia. |
| Personificação / *Spoofing* (ataque ativo) | É o envio informação na rede com o remetente forjado, na tentativa de enganar o destinatário ou de executar funções de administrador. | Proteção física do meio de comunicação.<br>Uso de criptografia. |
| Acesso não autorizado ao equipamento | Equipamento roubado ou "emprestado" por algum tempo. | Proteção física do equipamento.<br>Autenticação freqüente do usuário.<br>Arquivos criptografados.<br>Autodestruição dos dados. |
| Ataques à senha | O invasor tenta descobrir uma senha por força bruta | Bloquear usuário após algumas tentativas mal sucedidas.<br>Inibir o uso de senhas fáceis. |
| Simuladores de *Login* | Programas maliciosos que exibem uma tela semelhante à de Login para roubar senhas. | Antivírus e controle rígido dos programas instalados. |
| Falhas no sistema operacional | Usuários mal intencionados podem explorar falhas conhecidas do sistema operacional para obter acesso não autorizado. | Firewall.<br>Manter em dia as atualizações do sistema operacional. |

### 2.1.4 Políticas de Segurança

Política de Segurança da Informação é um conjunto de diretrizes que deve expressar o pensamento dos dirigentes de uma organização a respeito do uso da Informação por todos que têm acesso a ela. Essas diretrizes especificam como um sistema deve prover os seus serviços, limitar as operações dos usuários e determinar como as informações e os recursos devem ser administrados, protegidos e distribuídos. Segundo [Stallings 2003] a Política de Segurança; a conscientização dos usuários; e o uso eficiente dos mecanismos de segurança são os três pilares que sustentam a Segurança da Informação.

A política de segurança atribui os direitos e responsabilidades às pessoas que lidam com os recursos computacionais de uma instituição e com as informações neles armazenados. Ela também define as atribuições de cada um em relação à segurança dos recursos com os quais trabalham.

A adoção de uma Política de Segurança é fundamental para uma organização. A Política de Segurança serve como ponto de referência para adquirir, configurar e auditar sistemas de informação e redes de computadores para que sejam adequados aos requisitos propostos.

### 2.1.5 Mecanismos de segurança

Os mecanismos de segurança (ou Controles de Segurança) são responsáveis pelo efetivo cumprimento das políticas de segurança. Os principais mecanismos de segurança se dividem em: Controles administrativos (políticas); Controles físicos e Controles lógicos. Os ***Controles físicos*** são barreiras, como portas, trancas, paredes, blindagem, guardas que limitam o contato ou o acesso direto à informação ou à infra-estrutura (que garante a existência da informação) que a suporta. Já os ***Controles lógicos*** são barreiras que impedem ou limitam o acesso à informação, que está em ambiente controlado, geralmente eletrônico. Mecanismos de criptografia, assinatura digital, mecanismos de garantia da integridade da informação, mecanismos de controle de acesso e mecanismos de certificação são exemplos de controles lógicos.

Os ***mecanismos de criptografia*** são responsáveis por garantir a confidencialidade, permitem a transformação reversível da informação de forma a torná-la ininteligível a terceiros. Algoritmos e chaves secretas são utilizados para, a partir de um texto em claro (conjunto de dados não criptografados) produzir uma seqüência de dados criptografados.

A ***assinatura digital*** faz uso de um código para verificar a integridade de um texto ou mensagem. A assinatura digital também pode ser utilizada para verificar se o remetente de uma mensagem é mesmo quem diz ser, garantindo a autenticação do remetente.

Os ***mecanismos de garantia da integridade da informação*** são processos que permitem verificar se uma mensagem (texto, código, imagem, etc) foi alterada, intencional ou acidentalmente, durante sua transmissão ou ao longo de sua existência. A técnica consiste em anexar a uma mensagem um resumo de tamanho relativamente pequeno, como 128 bits, através do qual pode-se verificar a integridade da mesma.

***Mecanismo de controle de acesso*** é termo geral usado para um grupo de técnicas de segurança, como o uso de senhas; sistemas biométricos; *firewalls* ou de cartões inteligentes, para limitar o acesso a um computador ou a uma rede somente a usuários autorizados.

Um ***certificado digital*** é um objeto digital que garante que uma chave pública é de uma certa pessoa; Este certificado é considerado válido através da confiança em uma terceira entidade.

## 2.2 Computação Ubíqua

A proliferação de dispositivos portáteis com grande poder de processamento e armazenamento a baixo custo, aliada à ampla disponibilidade das redes sem fio, fez surgir um cenário no qual usuários acessam a Internet a partir de dispositivos portáteis, a qualquer hora e em qualquer lugar, tornando a computação ubíqua uma realidade. A expressão Computação Ubíqua (*Ubiquitous Computing*) foi idealizada por [Weiser 1991] para definir um novo paradigma da ciência da informação.

Segundo [Weiser 1991], a computação ubíqua é um paradigma de interação usuário-computador em que a tecnologia é integrada de forma transparente a ambientes físicos para auxiliar pessoas na realização de suas tarefas diárias de forma contínua e onipresente. Os primeiros computadores eram enormes e seu uso era compartilhado por vários usuários. A partir da década de 1980 se popularizaram os computadores pessoais, dedicados a apenas um usuário. Weiser vislumbrou um ambiente em que vários computadores estariam a serviço de cada usuário. Os computadores passariam a fazer parte do ambiente, e seu uso aconteceria sem que as pessoas precisassem pensar neles, de forma imperceptível, invisível, nas palavras de Weiser.

Atualmente, o usuário pode transportar consigo vários dispositivos que se comunicam e trocam informações com outros dispositivos presentes no ambiente pelo qual o usuário evolui. Esses equipamentos interagem com o usuário e com o ambiente, prestando serviços ao usuário sem que seja necessário solicitá-los. A essência da Computação Ubíqua é que a interface entre o usuário e as aplicações seja natural, transparente. Não é necessária uma ordem explícita do usuário para que sejam acionadas funcionalidades nas aplicações.

## 2.3 Contexto

A maneira como pessoas se conectam à Internet vem mudando nos últimos tempos. Novas tecnologias e infra-estruturas de comunicação sem fio estão permitindo uma área de alcance muito maior do que a oferecida pelas redes cabeadas tradicionais. A disseminação de

dispositivos portáteis, que agregam cada vez mais funções e suportam múltiplas formas de conexão, aumenta a cada ano. Tudo isto está convergindo para o surgimento de um ambiente de comunicação altamente integrado e ubíquo, tornando possível que usuários acessem serviços de Internet a qualquer momento e de qualquer lugar.

Para que se possa fazer um melhor proveito de toda esta mudança na forma de comunicação, é preciso que os serviços de Internet também evoluam. Essa nova classe de serviços deve levar em consideração que seus usuários não precisam realizar o acesso de um local fixo, mas podem mover-se pelo ambiente. Esta mobilidade faz com que o usuário migre de uma rede para outra, alterando, entre outros fatores, a disponibilidade de recursos e, portanto, suas necessidades dentro do serviço. Então, os serviços/aplicativos para este ambiente de comunicação ubíquo devem levar em consideração qual é o contexto de execução dos seus usuários. Aplicações que percebem mudanças no ambiente em que estão executando, e usam essas informações para ajustar seu funcionamento, são definidas como cientes de contexto. O uso do contexto é um importante campo de investigação no âmbito do paradigma da Computação Ubíqua.

Segundo [Springer 2006], contexto é qualquer tipo de informação acerca do ambiente de execução de uma aplicação, que pode ser usada para aprimorar seu comportamento, fazendo-a operar de maneira adaptativa, personalizada, autônoma e flexível. Devido a sua natureza, o contexto nem sempre está explicitamente disponível e deve ser extraído de fontes heterogêneas. As fontes de contexto podem ser sensores, componentes de software, bases de dados, interações do usuário ou aplicações, que podem nem estar cientes de sua função como fontes implícitas de dados no sistema. Então, serviços sensíveis ao contexto precisam lidar com essa disparidade de fontes, que podem estar amplamente distribuídas na infra-estrutura de execução da aplicação.

A heterogeneidade das fontes de contexto compreende os seguintes aspectos: tecnologia, acesso, tipo de contexto, estrutura e semântica dos dados, granularidade e usuário. Além do mais, o contexto é fornecido em diferentes níveis de abstração. Um sensor provê informação de baixo nível sobre o ambiente (p.e., velocidade do usuário), enquanto que a informação provida por uma aplicação é mais abstrata (p.e., atividade do usuário). Para que sejam úteis em aplicações, as informações de contexto colhidas de fontes diferentes precisam ser processadas, gerando um contexto de mais alto nível.

As fontes de contexto podem estar centralizadas (uma base de dados central), particionadas (divididas em várias bases de dados) ou ser específicas (informação da interface

de rede do dispositivo). Conseqüentemente, cada usuário pode ter um conjunto diferente de fontes de contexto, que podem ser de diferentes tipos, e também conter fontes alternativas.

Informação sobre contexto é normalmente usada em múltiplos domínios de aplicação. Por exemplo, informação sobre uma pessoa pode ser interpretada como informação de usuário numa aplicação de comunicação móvel, mas também pode ser informação de um paciente numa aplicação médica.

No âmbito do paradigma de computação ubíqua, o contexto do usuário, da aplicação, bem como o do dispositivo e o da rede sem fio devem ser considerados nos projetos do sistema, com o objetivo de aumentar a capacidade de adaptação sempre que houver mudanças no ambiente de execução. O ***Contexto do usuário*** se refere às informações relativas ao usuário como nome do usuário, *login*, senha, localização endereço, perfil, velocidade e questões de segurança relacionadas à autenticação e autorização. O ***Contexto do dispositivo*** se refere às informações relativas ao dispositivo como, por exemplo: tipo, modelo, configurações de tela, interfaces de comunicação, capacidade de memória e fontes de energia. O ***Contexto da aplicação*** se refere às informações relativas à aplicação. O ***Contexto de rede*** se refere às informações relativas a redes sem fio como se a rede é segura, tipo de protocolo, tipo de criptografia além de outras como largura de banda, atraso máximo e taxa de erros.

## 2.4 Adaptação dinâmica

As aplicações para a computação ubíqua executam em um ambiente altamente dinâmico e devem estar preparadas para reagir a diferentes situações e contextos. Um dos desafios consiste justamente em tratar dos eventos ambientais não previstos, isto é, tratar das mudanças que não foram originalmente consideradas no projeto da aplicação. Nos ambientes de computação ubíqua não é possível antecipar em tempo de projeto todas as respostas que uma aplicação ciente de contexto pode dar frente à ocorrência de eventos imprevisíveis.

Segundo [Santos 2008], a imprevisibilidade de mudanças ambientais e o fato de decisões de reconfiguração afetarem diferentes partes de uma aplicação impõem novos desafios para a construção das aplicações. Na área de Desenvolvimento Baseado em Componentes (DBC), a Adaptação de Software [Canal 2006] é uma disciplina que estuda a modificação ou extensão de comportamento de componentes através do uso de componentes especiais chamados adaptadores (adaptors), os quais são capazes de possibilitar que componentes com interfaces desiguais interoperem. Tais adaptadores são automaticamente construídos a partir de uma descrição abstrata de como as desigualdades existentes podem ser

resolvidas (ou seja, através de mapeamento de adaptação), tendo como base a descrição das interfaces dos componentes.

Segundo [Canal 2006], a adaptação de software obtida através da construção de adaptadores pode ser classificada em diferentes tipos de adaptação com relação ao momento em que a mesma ocorre, considerando as diversas fases do ciclo de vida do software. A adaptação de software em [Bulcão 2005] é classificada em dois tipos: Adaptação Estática e Adaptação Dinâmica.

A ***Adaptação Estática*** é realizada antes de o sistema estar rodando, ou seja, em Tempo de Projeto. Esse tipo de adaptação engloba a adaptação dos requisitos, permitindo que a especificação de um sistema possa ser estendida para atender novos requisitos, ou para modificar os requisitos antigos, bem como de partes de código já existentes, possibilitando o reuso de código. Na adaptação estática, os procedimentos referentes à adaptação são conhecidos – foram planejados – antes do momento em que a adaptação de fato ocorre.

A ***Adaptação Dinâmica*** é realizada após o sistema estar em execução, e sem que seja necessário interromper o seu funcionamento. Ou seja, partes de software de um sistema/serviço em execução precisam ser alteradas em virtude de mudanças no modo com o qual um serviço é fornecido. Nesse tipo de adaptação, os componentes a adaptar, assim como os passos para gerenciar a adaptação, são desconhecidos antes do instante em que a adaptação ocorre.

## 2.5 Segurança em Computação Ubíqua

Além das aplicações em si, ou seja, do código que implementa as funcionalidades providas para os usuários, é imprescindível que políticas e controles de segurança também possam ser capazes de se adaptar devido às propriedades dinâmicas dos ambientes. A adaptação de tais políticas e controles visa assegurar o controle de acesso, a disponibilidade e a confiabilidade da aplicação. As propriedades dinâmicas dos ambientes referem-se ao contexto de execução (como, por exemplo, flutuação de banda, localização, preferência e perfil do usuário, segurança, mudanças na taxa de erro, ruído ambiental), bem como aos recursos limitados dos dispositivos (memória disponível e energia residual do dispositivo). Portanto, tais propriedades dinâmicas podem afetar tanto o resultado da própria aplicação como a segurança e, por conseqüência, a satisfação do usuário. Quanto à especificação das políticas, elas podem ser associadas a qualquer entidade do sistema (usuário, dispositivo ou recurso) ou mesmo a um grupo de tais entidades (por exemplo, que compartilhem um mesmo

contexto de execução). Já os controles de segurança são utilizados para mitigar as vulnerabilidades encontradas em um contexto.

Por exemplo, um controle de segurança pode ser ativado ao se identificar que a rede sem fio corrente é insegura (sem criptografia). Ao ativar esse controle, um sistema de túneis seguro baseado em protocolos semelhantes ao SSH (*secure shell*) pode ser estabelecido, por exemplo, para garantir os requisitos de integridade e confidencialidade da aplicação. Para garantir disponibilidade, um controle pode ser acionado ao se identificar um possível ataque de *jamming*, o qual se caracteriza por um intruso causar interferência nas redes de forma a atrapalhar a comunicação entre os nós. Outros controles podem ser ativados quando o dispositivo possuir pouca energia residual ou pouca memória disponível.

Assim, diante do dinamismo e heterogeneidade do contexto de execução, as políticas e os controles de segurança, bem como as aplicações ubíquas executadas nos dispositivos móveis, em especial, aquelas aplicações consideradas críticas (entre outras, aplicações militares, controle de processos industriais), onde um mau funcionamento (falha) ou ataque pode ter conseqüências catastróficas, necessitam se adaptar para um contexto de execução corrente de acordo com os objetivos e políticas de adaptação especificadas pelo usuário. Ou seja, do ponto de vista da aplicação, esta pode ter diferentes comportamentos, um para cada contexto de execução, e as políticas de adaptação são responsáveis por definir qual o comportamento (configuração) a ser usado para um dado contexto de execução. Da mesma forma, do ponto de vista da segurança, políticas e controles são definidos para cada contexto de execução e políticas de adaptação são determinadas para definir quais são as políticas e os controles de segurança a serem usados para um dado contexto. Assim, graças à capacidade de se adaptar, as aplicações, as políticas e os controles de segurança, além de reagir a simples mudanças contextuais, passam também a poder compensar ou lidar com falhas e ataques em tempo de execução. A abordagem de segurança adaptativa para infra-estruturas de TI tenta conter ameaças ativas e neutralizar vetores de ataque potenciais. Segundo [Weise 2008] a segurança adaptativa, como outras arquiteturas de segurança, se empenha em:

- reduzir a amplificação da ameaça;
- diminuir a superfície atacada (abrangência do ataque);
- diminuir a velocidade do ataque;
- reduzir o tempo de correção;
- facilitar a disponibilidade de dados e recursos de processamento;
- promover a integridade dos dados e a confiabilidade dos recursos.

Um dos principais objetivos da segurança adaptativa é a sobrevivência. De fato, a sobrevivência é na realidade a principal meta da segurança adaptativa. Segundo [Taylor 2003], sobrevivência é a "capacidade de um sistema de cumprir sua missão de forma adequada na presença de ataques, de falhas ou de acidentes".

Quanto a abordagens adaptativas, em [Elkodary 2007] é apresentada uma interessante taxonomia para classificação das abordagens adaptativas. As abordagens são categorizadas em seis dimensões: (i) autenticação; (ii) autorização; (iii) tolerância a falhas; (iv) o paradigma computacional empregado para a transformação (parametrizado, composição baseado em aspectos, orientado a aspecto, reflexão, etc.); (v) a escala de reconfiguração, e (vi) tratamento de conflitos.

A ***Autenticação*** é o processo de verificar uma credencial provida pelo usuário contra outra armazenada no sistema. Autenticação atende a dois objetivos de segurança: identificação e não repúdio. O primeiro verifica se o usuário é realmente quem ele diz ser. O segundo evita que o usuário negue o envolvimento em atividades que ele tenha participado. Autenticação adaptativa permite que o sistema escolha dentre vários métodos de autenticação, dependendo das condições do contexto.

A ***Autorização*** é o processo de controlar o acesso do usuário a recursos do sistema. Envolve a atribuição de privilégios e direitos de acesso de usuários sobre recursos. Autorização atende aos objetivos de segurança de confidencialidade e integridade. O primeiro garante que a informação só pode ser lida por usuários apropriados, enquanto a segunda garante que somente modificações consistentes da informação serão efetuadas no sistema. A autorização adaptativa permite que políticas de controle de acesso possam mudar baseadas em condições do contexto.

A ***Tolerância a Falhas*** é o processo de mascarar as falhas no sistema eliminando componentes com falha (*faulty components*), limitando as conseqüências da falha e alocando substitutos válidos. Um exemplo seria a capacidade de um banco de dados distribuído de redirecionar as requisições a réplicas em caso de falha do nó primário. A tolerância a falhas atende ao objetivo de segurança de disponibilidade, que visa garantir a máxima continuidade do serviço e o aceite do nível de serviço (*acceptance of service-level*) mesmo na ocorrência de falhas.

As outras três dimensões, ***Paradigma Computacional***, ***Escala de Reconfiguração e Tratamento de Conflito*** estão relacionadas com o método de adaptação.

O ***Paradigma Computacional*** é o mecanismo pelo qual unidades de reconfiguração são iniciadas ou compostas em tempo de execução para levar o sistema de uma configuração

válida para outra. O trabalho de [Elkodary 2007] define quatro principais paradigmas para reconfiguração:

- **Parametrização** – é a capacidade de mudar o comportamento de uma unidade de configuração em tempo de execução baseado no valor de um parâmetro do contexto ou selecionado pelo usuário. Essa abordagem é apropriada quando o número de unidades de configuração é pequeno e todas as configurações possíveis são conhecidas em tempo de projeto.
- **Composição baseada em Componentes** – é a capacidade de reconectar (*rewire*) diferentes implementações de um componente com interface conhecida em tempo de execução, sem a necessidade de recompilação. Essa abordagem suporta qualquer configuração de um sistema desde que seja respeitado o contrato especificado pelas interfaces.
- **Reflexão (*reflection*)** – é a capacidade do programa de observar e modificar seu próprio comportamento ao revelar alguns detalhes de sua implementação em nível meta (informações descritivas da estrutura do código, como nome de variáveis e assinatura de procedimentos). A mudança da implementação interna de um componente é possível com esse paradigma.
- **Orientação a Aspectos** – permite a separação de características funcionais de características transversais. Em tempo de execução todas as características podem ser compostas baseadas num modelo de costura de pontos de junção (*weaving join point model*). Permite uma ampla gama de reconfigurações e simplifica significativamente o desenvolvimento e a manutenção do sistema.

A ***Escala de Reconfiguração*** se refere à escala em que a configuração é efetuada (unidade individual; intra-unidades; sistema). O limite inferior dessa dimensão é a unidade individual que permite a adaptação do comportamento ou de constantes (*invariants*) num único componente ou serviço. O limite superior é a reconfiguração em nível de arquitetura, que envolve a reestruturação da instalação (*setup*) dos componentes do sistema bem como a mudança das propriedades externas da arquitetura do sistema.

O ***Tratamento de Conflitos*** se refere à capacidade do sistema de **detectar** e **resolver** conflitos surgidos devido às incompatibilidades entre unidades de configuração ou a conflitos devidos à natureza dos objetivos das configurações. Um requisito essencial para reconfiguração dinâmica é a detecção de conflitos. A solução de conflitos pode seguir três abordagens:

- **resolução de conflitos dirigida pelo usuário (*user driven*)** – dá aos usuários total controle do processo de resolução. O sistema não toma nenhuma decisão em nome dos usuários;
- **resolução de conflitos autônoma** – permite que o sistema tome decisões em nome dos usuários;
- **solução de conflitos interativa** – funde as vantagens de cada uma das abordagens anteriores. Resoluções triviais são tratadas automaticamente e resoluções complexas são tratadas de forma interativa com os usuários.

## 2.6 Trabalhos Relacionados

Vários trabalhos propõem abordagens para realizar adaptação dinâmica no âmbito de computação pervasiva como [Carton 2007] e [Preuveneers 2006]. Por exemplo, em [Preuveneers 2006] os autores argumentam que um ambiente de computação pervasiva assume que as aplicações: (i) são instaladas em dispositivos que variam de computadores *desktop* a *handhelds*; (ii) são cientes da informação contextual do usuário, do próprio dispositivo e do seu ambiente; e (iii) devem poder se adaptar a esse contexto dinâmico. Eles então propõem uma infraestrutura ciente ao contexto que satisfaz esses três requisitos provendo suporte em tempo de execução para adaptação dirigida ao contexto de serviços móveis baseados em componentes. A infraestrutura proposta foi implementada em Java e os testes comprovaram que ela realmente provê adaptação dinâmica para as aplicações, incluindo descoberta de serviços "*on-the-fly*". Entretanto, nenhum dos trabalhos levantados trata a questão específica de comportamento adaptativo relacionado aos requisitos de segurança da aplicação.

No tocante à segurança, observa-se nos diversos trabalhos encontrados na literatura que a importância do provimento de requisitos de segurança adaptativos não é exatamente uma novidade. A proposta de [Hinton 1999] já apontava para a necessidade de se estabelecer um compromisso entre segurança e desempenho. Em 1999 era relativamente natural supor que um sistema privilegiasse mais o desempenho em troca de uma menor segurança. A estratégia proposta por [Hinton 1999] consiste de acrescentar controles de segurança gradativamente ao sistema toda vez que o mesmo estivesse sob algum tipo de ataque. Tal proposta é caracterizada por uma postura reativa face à ocorrência de falhas, i.e, somente após um ataque é que ocorre efetivamente a adaptação de segurança. Hoje com a proliferação de redes sem fio, a importância da segurança junto aos negócios passa a ter um papel mais importante. Na abordagem empregada no Prometheus, uma mudança de contexto com

acréscimo do risco de incidentes de segurança (ex: aumento de vulnerabilidades), já seria suficiente para implicar em um acréscimo de controles de segurança.

Segundo [Elkodary 2007], um importante desafio é lidar com o aumento do número e da complexidade de ataques a sistemas de software. As primeiras tentativas de obter segurança dos sistemas usavam métodos estáticos com medidas fixas de segurança. Os resultados obtidos reduziam as vulnerabilidades dos sistemas e eram aceitáveis. Entretanto, enquanto os ataques cresciam em quantidade e complexidade, as medidas de segurança precisavam ser reforçadas. Com essa expansão, o custo (*overhead*) de processamento e o consumo de recursos de infra-estrutura de TI alcançaram níveis alarmantes, requisitando uma infra-estrutura mais cara e maiores custos de manutenção para suportar os mecanismos de segurança necessários. Essa infra-estrutura é inacessível para muitas pequenas e médias organizações. Uma possível solução para a crescente complexidade da infra-estrutura de segurança de TI é a segurança adaptativa. Soluções adaptativas podem ser adotadas em nível de hardware, de sistema operacional, de rede e de aplicação. Em [Elkodary 2007], é realizada uma análise (*survey*) de quatro tipos de abordagem para serviços de segurança adaptativos, porém todas são restritas ao nível de aplicação, diferentemente da abordagem empregada no Prometheus. Também em [Elkodary 2007] é apresentada uma interessante taxonomia para classificação das abordagens adaptativas, que foram detalhadas na seção anterior.

A **classificação** da versão do Prometheus nesse trabalho de acordo com a taxonomia proposta por [Elkodary 2007] é descrita em seguida. O Prometheus não trata diretamente da ***autenticação*** de usuários (ausente). Está prevista em versões futuras, a criação de uma Política de Adaptação que só permita a execução de aplicações ou a disponibilização de certos recursos para usuários autenticados. Adicionalmente, o Prometheus não trata diretamente da ***autorização*** ao acesso de recursos pelo usuário ou pela aplicação (ausente). Está prevista em versões futuras a criação de uma Política de Adaptação que só permita o acesso a recursos por usuários autorizados. No que se refere à ***Tolerância a Falhas***, falhas no ambiente podem ser tratadas pelo processo de adaptação proposto no Prometheus.

Quanto ao **Paradigma computacional,** o funcionamento do Prometheus pode ser classificado como "**Parametrização**", já que Informações de Contexto são usadas para influir no funcionamento de componentes; mas também pode ser classificada como "**Composição baseada em Componentes**", já que o processo de adaptação por mudança de versão, por exemplo, permite a carga dinâmica de novos componentes e do estabelecimento de um novo relacionamento entre os componentes (Adaptadores e Controles de Segurança). No que se refere à escala de reconfiguração, o Prometheus pode ser classificado como "**intra-unidades**",

já que a mudança de uma única Informação de Contexto pode provocar mudanças em vários componentes e na maneira como eles se relacionam.

No tocante ao **Tratamento de Conflitos,** a versão atual do Prometheus é capaz de efetuar a detecção de conflitos mas não a resolução automática de conflitos. Nessa versão o usuário é informado da ocorrência de um conflito, o que impede a execução de uma aplicação. Desse modo, quanto ao Tratamento de Conflitos, o Prometheus pode ser classificado como "**Dirigida pelo Usuário**".

A abordagem fundamental para o processamento da segurança adaptativa envolve a detecção de ameaças; análise e resposta. O autor do trabalho em [Weise 2008], por exemplo, propõe os passos descritos em seguida para o processamento da segurança adaptativa: (i) Uso de *Telemetria* na etapa de obtenção e monitoramento da informação; (ii) A *Correlação* é aplicada para a avaliação dos dados da telemetria; (iii) Por fim, a *Resposta* é obtida através de uma ou mais ações tomadas para reagir a uma ameaça detectada.

O trabalho de [Marcus 2004] apresenta o conceito de Infra-estrutura de Segurança Adaptativa (ASI - *Adaptive Security Infrastructure*). O autor afirma que a formalização dos mecanismos de segurança providos como uma unidade numa Infra-estrutura de Segurança Adaptativa é o primeiro passo para efetuar uma eventual validação desses mecanismos. São discutidas algumas abordagens para a formalização. O autor faz várias considerações sobre Políticas de Segurança, tais como hierarquia de políticas de segurança; união, consistência e mudança de políticas de segurança. Segundo o autor Política de Segurança é "o que é permitido", ou "a especificação do que é permitido". Ele divide o problema em questões espaciais e temporais. Quanto a questões espaciais, o autor aponta que pesquisas devem ser realizadas em: especificação de estruturas hierárquicas; coordenação central e distribuída de detecção, análise e resposta; transição suave entre hierarquias; capacidade de testar a satisfação (atendimento) da política; e a capacidade de executar (fazer cumprir) a resposta. Duração da resposta; sincronização; velocidades relativas do ambiente mutante, detecção, análise e resposta; incorporação do tempo na política; e confirmações, avisos de sucesso são questões para pesquisas temporais.

Em [Izquierdo 2007] é proposto um serviço de criptografia adaptativo específico para aplicações em ecossistemas digitais, levando em conta as limitações, capacidades de processamento e requisitos de segurança de cada um desses ecossistemas. A proposta se concentra no uso de um modo de criptografia orientado a bloco (OCB) não encadeado, de forma a permitir que apenas blocos específicos com informações confidenciais sejam codificados, i.e. que não seja necessário encriptar toda a informação trocada; e traça um perfil

do compromisso entre porcentagem de informação encriptada versus tempo de processamento. A aplicação de Ecossistema digital se adéqua perfeitamente como um caso de uso do serviço Prometheus, onde a abordagem orientada a contexto adotada no presente trabalho permitiria indexar diferentes níveis de criptografia.

No domínio das redes Ad hoc existe uma preocupação constante quanto aos recursos demandados para o fornecimento de serviços de segurança, principalmente quanto ao respectivo consumo de energia.

Em [Chigan 2005] é proposto um *framework* para provisionamento de serviços de segurança auto-adaptativo em função da disponibilidade de recursos composto por dois módulos principais: (i) módulo *offline* de seleção do conjunto otimizado de protocolos seguros entre camadas (evitando redundância funcional) e (ii) módulo *online* de adaptação dos controles de segurança em função do desempenho da rede. Esse framework é capaz de empregar diferentes combinações de protocolos seguros para satisfazer diferentes necessidades de segurança de diferentes aplicações. Já o trabalho proposto por [Doomun 2007] faz uma análise específica dos principais protocolos seguros (para autenticação, confidencialidade e integridade) passíveis de serem usados em uma rede IEEE 802.11i usando modelos rigorosos de consumo de energia. Com base nesses resultados é criada uma tabela de referência relacionando os limiares de energia disponíveis com as pilhas de protocolos recomendadas. Esses dois trabalhos anteriores fazem uso de monitores de recursos para inferir quanto à necessidade de adaptação da segurança, sendo que o primeiro é orientado a desempenho e o segundo é orientado a energia. Entretanto, ambos relaxam a segurança quando o desempenho (primeiro trabalho) ou a energia (segundo trabalho) está em queda. A abordagem proposta no presente Prometheus também faz uso de monitores, mas sugere o seu uso em conjunto de técnicas de caracterização do contexto para uma melhor avaliação da segurança necessária.

[Xiao 2009] propõe uma abordagem que combina MDA (*Model Driven Architecture*) e Agentes de software para tratar a adaptação dinâmica da segurança de aplicações. Na proposta o modelo é modificado e transformado continuamente. As modificações no modelo são introduzidas na aplicação por meio de adaptações em tempo de execução. Nesse modelo a segurança é tratada por agentes de software. Nessa abordagem, entretanto, as adaptações ocorrem em conseqüência de alterações no modelo efetuadas por especialistas, e não pelo tratamento de informações de contexto.

Em relação a os trabalhos relacionados de segurança, o Prometheus apresenta ainda os seguintes diferenciais: (i) é sensível ao contexto; (ii) adapta dinamicamente as políticas e os

controles de segurança; (iii) adapta em tempo de execução as próprias políticas de adaptação; e (iv) utiliza a abordagem de componentes, permitindo a atualização dinâmica a partir de um repositório remoto.

# 3 O Prometheus

O objetivo do ***Serviço de Segurança Adaptativa***, proposto nesse trabalho, denominado Prometheus, é o de prover um ambiente seguro segundo a ***Política de Segurança*** criada por uma organização para a execução de seus aplicativos em dispositivos móveis.

A mobilidade dos dispositivos introduz uma série de restrições e dificuldades. A primeira dificuldade é que os recursos em dispositivos móveis são limitados, restringindo fortemente as capacidades de processamento, de armazenamento, e de comunicação. Estas restrições limitam os mecanismos de segurança a utilizar. Mecanismos muito sofisticados podem consumir muita memória, processador e/ou energia, ou ainda, provocar atrasos no funcionamento das aplicações, possivelmente causando impacto nos requisitos de tempo de resposta das aplicações.

Uma outra dificuldade está relacionada ao fato de que os requisitos de segurança podem variar dependendo da aplicação que está sendo executada, da rede utilizada, do usuário e da localização do dispositivo. Por exemplo, aplicações de entretenimento não exigem grandes esforços para proteger a informação. Por outro lado, aplicações de negócio, em que dados confidenciais são exibidos ao usuário ou transmitidos pela rede podem exigir procedimentos específicos para garantir que os requisitos de segurança sejam atendidos. A execução de aplicativos em locais controlados pode exigir uma segurança menos rígida que a execução do mesmo aplicativo num local público, como uma rua ou o escritório de uma empresa concorrente. Assim, o Prometheus procura prover o nível adequado de segurança para as aplicações em execução utilizando o mínimo de recursos. Para isso, o Prometheus deve levar em consideração o **contexto** em que cada aplicação está sendo executada.

Segundo [Dey 2000], "*Contexto é qualquer informação que possa ser usada para caracterizar a situação de entidades (seja uma pessoa, local ou objeto) consideradas relevantes para a interação entre um usuário e uma aplicação, incluindo os próprios usuário e aplicação. Contexto é, tipicamente, o local, a identidade e o estado de pessoas, grupos, e objetos computacionais e físicos*".

Se um sistema usar o contexto para prover informações ou serviços ao usuário, ele é dito ser *ciente de contexto* [Hoh 2006].

O Prometheus é ciente de seu contexto de execução uma vez que ele leva em conta as constantes mudanças dos requisitos de segurança dos usuários, da aplicação e da própria rede em seu estado de execução, a fim de contornar ou amenizar o impacto dessas alterações na operação das aplicações ubíquas. Essas mudanças fazem com que o Prometheus necessite realizar alterações em seu comportamento para se adaptar às novas condições de execução.

Uma mudança no ambiente de execução que acarrete um aumento do risco de incidentes de segurança (ex: aumento de vulnerabilidades), é razão suficiente para implicar em adaptações que gerem um acréscimo de controles de segurança no próprio dispositivo (comportamento pró-ativo) de forma a garantir um ambiente seguro para as aplicações ubíquas. As informações de contexto no âmbito do Prometheus são detalhadas na seção 3.1. Assim, o ambiente seguro criado pelo Prometheus agrega às aplicações benefícios da ciência de contexto com adaptações dinâmicas sem que para isso seja necessário modificar as aplicações.

No tocante à Política de Segurança, conforme já mencionado, esta é o conjunto de regras em forma textual que devem ser seguidas pelos usuários de recursos de uma organização [RFC2196]. Embora não trate diretamente com a Política de Segurança, todo o funcionamento do Prometheus está estreitamente ligado a ela. A Política de Segurança é observada para determinar a direção a tomar nos ajustes efetuados pelo Prometheus para adequar seu funcionamento ao contexto de execução. Ajustes nada mais são que configurações de segurança realizadas em nível do dispositivo, de rede e da aplicação. Em suma, o Prometheus é um serviço de segurança adaptativa que garante que as políticas e os controles de segurança sejam sensíveis ao contexto, assegurando em tempo de execução o controle de acesso, a disponibilidade e a confiabilidade de aplicações críticas executadas em dispositivos.

Este Capítulo está estruturado em cinco seções, na seguinte disposição. A Seção 3.1 apresenta uma descrição da ciência do contexto no Prometheus. A Seção 3.2 trata da Informação de Contexto no âmbito do Prometheus e descreve as operações para manipulação de Informações de Contexto. A Seção 3.3 tece considerações a respeito da Política de Segurança, Regras de Segurança e Requisitos de Segurança A Seção 3.4 descreve a arquitetura do Prometheus. As Seções 3.5 e 3.6 descrevem detalhadamente os componentes do núcleo e os componentes de segurança do ambiente de segurança do Prometheus. Por fim, a Seção 3.7 tece as considerações finais desse capitulo.

## 3.1 Ciência de Contexto no Prometheus

A abordagem utilizada para tratar a ciência de contexto no Prometheus é composta por três atividades principais: detecção, análise e resposta. A detecção consiste da obtenção das Informações de Contexto relevantes. A análise consiste de processar as Informações de Contexto obtidas e decidir qual será a reação. A resposta consiste na execução da reação decidida pela análise.

Por exemplo, um controle de segurança pode ser acionado ao se identificar um ataque de *jamming*, o qual se caracteriza por um intruso causar interferência nas redes de forma a atrapalhar a comunicação entre os nós. Nesse exemplo, a **detecção** obtém Informações de Contexto indicando uma mudança no nível do sinal na rede sem fio e no volume de mensagens recebidas. A **análise** indica que o nível do sinal está anormalmente alto (uma potência maior que um limiar considerado normal) impedindo ou mascarando a recepção do sinal de outros nós, e que o volume de mensagens recebidas está muito acima do esperado (número de mensagens recebidas num intervalo de tempo maior que um determinado limiar), provocando o consumo acelerado da energia residual do dispositivo. A **resposta** consiste de acionar uma configuração alternativa que utiliza outra rede, e dessa forma livrar-se do ataque.

## 3.2 Informações de Contexto no Âmbito do Prometheus

O Contexto é formado por uma coleção de informações de contexto. Cada informação de contexto tem um conjunto próprio de características. Por exemplo, a temperatura ambiente é representada por um número real; o nome do usuário é uma cadeia de caracteres; a coordenada geográfica do dispositivo é um par de números. Algumas informações de contexto são multivaloradas. Exemplos disso são a relação de aplicativos em execução ou a lista de redes disponíveis para conexão.

Na implementação do Prometheus, as características de cada Informação de Contexto são mapeadas numa especialização da classe ***InfoContexto***, conforme ilustrado na Figura 3.1. Cada instância desta classe carrega um valor que representa a situação (ou o estado) de uma determinada Informação de Contexto num dado momento.

No diagrama de classes da Figura 3.1 são mostradas algumas das classes que representam Informações de Contexto. ***InfoUsuario, InfoPosicao*** e ***InfoAplicacao*** se referem, respectivamente, ao nome do usuário atual do dispositivo, à posição geográfica do dispositivo e ao nome das aplicações em execução. ***InfoCriptografia*** é a Informação de Contexto que informa o nível de criptografia usado para garantir a confidencialidade das informações trafegadas pela rede. Possíveis valores para *InfoCriptografia* são "*nenhuma*", "*fraca*", "*media*" e "*forte*". Por fim, a Informação de Contexto ***InfoTunelSsh*** identifica os túneis SSH usados para garantir a privacidade de conexões TCP.

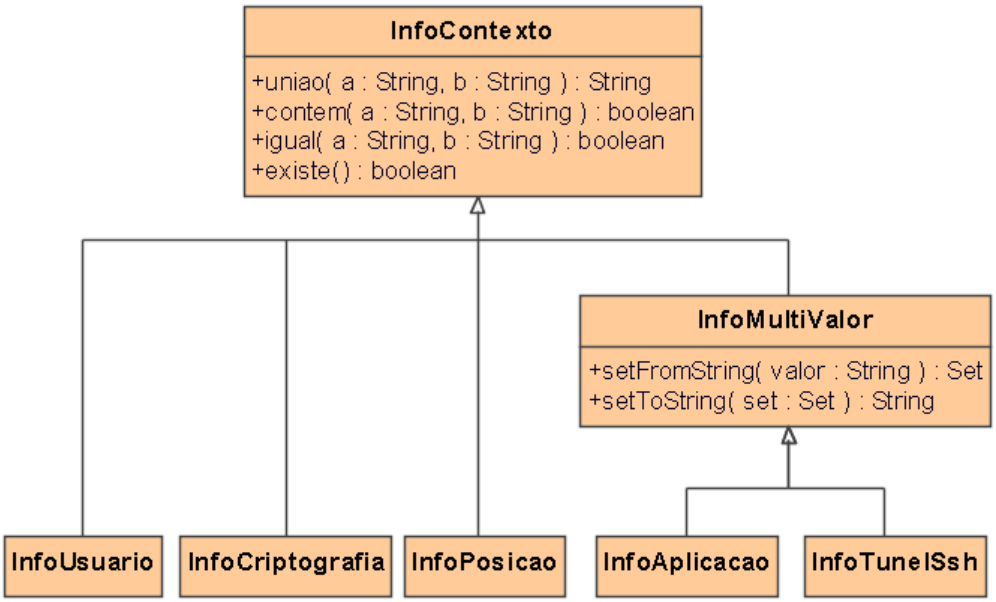


**Figura 3.1. Fragmento do Diagrama de Classes para Informações de Contexto**

As principais operações definidas para *InfoContexto* são *contem*, *uniao, igual* e *existe*. Essas operações permitem relacionar e combinar o valor das Informações de Contexto. Para permitir as operações funcionem de forma independente do tipo de dado que cada Informação de Contexto representa, os valores são sempre armazenados como cadeias de caracteres. Assim, os parâmetros das operações são sempre cadeias de caracteres.

A operação ***contem*** é usada para verificar se o valor da Informação de Contexto atende a um dado requisito (se o contexto *contem* o valor requisitado, então o requisito está atendido).

Aplicações diferentes podem ter como requisito, cada uma, valores diferentes para uma mesma Informação de Contexto. A operação ***uniao*** é responsável por combinar valores de uma mesma Informação de Contexto de forma a obter um valor que represente todos os requisitos.

A operação ***igual*** é responsável por *informar* se dois valores de uma dada informação de contexto são equivalentes. É usada para comparar os valor requisitado com o valor do contexto, para então determinar se é necessário fazer alguma adaptação.

A operação ***existe*** informa se há no contexto algum valor para a Informação de Contexto.

O uso dessas operações será visto nas seções 3.5.3, 3.5.4 e 3.5.5. O funcionamento dessas operações está detalhado nas próximas seções.

### 3.2.1 Operação *contem*

A operação ***contem*** recebe dois parâmetros, e verifica se o valor do primeiro parâmetro contém o valor do segundo parâmetro. Os dois parâmetros são cadeias de caracteres que possuem valores relativos a uma Informação de Contexto. O primeiro parâmetro é o valor corrente de uma informação de contexto e o segundo o valor desejado para essa mesma informação de contexto. Assim, essa operação verifica se o valor corrente da Informação de Contexto "contém" o valor desejado para essa informação. O resultado da operação, verdadeiro ou falso, indica se o valor do primeiro parâmetro é tal que contenha o valor do segundo parâmetro. O significado de "conter" depende de cada Informação de Contexto.

Por exemplo, seja a Informação de Contexto multivalorada *InfoAplicacao*, que indica o nome das aplicações em execução. Um possível valor corrente dessa Informação de Contexto seria "calc;explorer;notepad". Caso a operação *contem,* recebesse o como segundo parâmetro o valor desejado "explorer" ou "calc;notepad", o resultado da operação seria verdadeiro. Se o valor desejado para o segundo parâmetro fosse "sndrec32" ou "calc;netscape", o resultado seria falso.

Para Informações de Contexto simples, ou seja, com um único valor, a operação *contem* funciona de modo análogo. Por exemplo, *InfoCriptografia*, informa o nível de criptografia usado para garantir a confidencialidade das informações trafegadas pela rede. Possíveis valores para *InfoCriptografia* são "*nenhuma*", "*fraca*", "*media*" e "*forte*". A operação *contem("media", "fraca")* retorna verdadeiro, porque o nível de criptografia "*media*", disponível no contexto, corrente é suficiente para suprir o nível desejado, "*fraca*".

### 3.2.2 Operação *uniao*

A operação ***uniao*** é responsável por combinar valores de uma mesma Informação de Contexto. A operação recebe dois valores, *valA* e *valB* e o seu resultado, *valC*, é tal que *contem(valC, valA)* e *contem(valC, valB)* são verdadeiros. Por exemplo, a Informação de Contexto *InfoPortaTcpAberta* indica os números das portas TCP abertas no *Firewall* local do dispositivo. Assim, o resultado da operação *uniao("443;80", "21;443")* é *"21;443;80"*.

A operação *uniao* pode não se aplicar a algumas Informações de Contexto. Nesses casos, a operação *uniao(valA, valB)* deve retornar *valB*. Por exemplo, *InfoPosicao* informa a posição geográfica do dispositivo (a *uniao* da *"posição anterior"* com a *"posição atual"* é igual a *"posição atual"*).

### 3.2.3 Operação *igual*

A operação ***igual*** é responsável por *informar* se dois valores de uma dada Informação de Contexto são equivalentes. Por exemplo, o resultado da operação *igual* para os valores *"joão"* e *"maria"* de *InfoUsuario* (Informação de Contexto que informa o nome do usuário atual do dispositivo) é falso. Já o resultado da operação *igual* para os valores "calc;netscape" e "netscape;calc" de *InfoAplicacao* é verdadeiro, pois os dois valores, embora diferentes, são equivalentes.

### 3.2.4 Operação *existe*

A operação existe é responsável por verificar se há no contexto algum valor para uma dada Informação de Contexto. Se nunca foi atribuído um valor a uma determinada Informação de Contexto, ou se o último valor atribuído foi NULO então essa operação retorna o valor falso. Em qualquer outra situação retorna verdadeiro.

## 3.3 Política de Segurança, Regras de Segurança e Requisitos de Segurança

A Política de Segurança é um documento que estabelece regras e padrões para a proteção da informação. O formalismo desse documento, entretanto, é insuficiente para que possa ser processado automaticamente. Para permitir o processamento pelo Prometheus, a Política de Segurança é mapeada em um conjunto de Regras de Segurança as quais são definidas para cada um dos Requisitos de Segurança das aplicações, da rede e do dispositivo. Através da Regras de Segurança, são definidos os valores desejados para cada um dos ***Requisitos de Segurança***. Portanto, o Prometheus deve assegurar que os Requisitos de Segurança, definidos por meio das Regras de Segurança, sejam atendidos para que a Política de Segurança seja respeitada.

As Regras de Segurança são fornecidas ao Prometheus sob a forma de um arquivo XML, o **Repositório de Regras de Segurança.** A sintaxe e semântica das regras são explicadas em detalhes no item 3.5.10.

Contexto e Requisitos são informações da mesma natureza, entretanto, o Contexto representa o valor corrente de informações ambientais, enquanto os Requisitos de Segurança representam valores desejados, do ponto de vista da segurança, para essas mesmas informações. Se o valor de cada um dos Requisitos de Segurança estiver satisfeito num determinado contexto, então podemos afirmar que a Política de Segurança está atendida para esse contexto.

No que concerne aos requisitos, os Requisitos de Segurança contêm os valores referenciais para suas respectivas Informações de Contexto. Por exemplo, o Requisito de Segurança *InfoCriptografia* sugere qual algoritmo de criptografia deve ser usado para garantir a confidencialidade dos dados trafegados na rede. Possíveis valores para *InfoCriptografia* são *Fraca*, *Media* e *Forte*, sendo que *Fraca* sugere o uso de um algoritmo menos seguro porém mais rápido (ex: DES), enquanto *Forte* sugere um algoritmo mais seguro porém mais lento (AES-256). Se a Política de Segurança através do seu conjunto de Regras de Segurança, impõe que deve ser usada no mínimo criptografia *media* nas comunicações com o servidor corporativo, então ao requisito de segurança *InfoCriptografia* é atribuído o valor *media* que é o valor referencial para a Informação de Contexto *InfoCriptografia*.

No contexto de computação ubíqua, uma das dificuldades do desenvolvimento de aplicações é que os requisitos das aplicações mudam conforme o contexto. Assim como mudam os requisitos das aplicações, mudam também os requisitos de segurança relativos à aplicação, à rede e ao dispositivo.

Por exemplo, suponha que a Política de Segurança de uma instituição permita que uma aplicação acesse diretamente um servidor corporativo, ou seja, sem a necessidade de criptografia, desde que esse acesso seja feito a partir da rede interna da empresa, e que acessos feitos a partir de qualquer outra rede devam ser feitos através de um canal seguro, criptografado, como uma VPN. Assim, uma dada aplicação executando em um dispositivo móvel localizado dentro da empresa pode acessar os servidores corporativos através da rede WI-FI interna sem fazer uso de um canal seguro. Entretanto, se o usuário se mover para fora da empresa, o acesso à rede passará a ser feito através de um provedor de acesso. Nesse caso, o Requisito de Segurança indica que um canal seguro deve ser estabelecido entre o dispositivo e o servidor corporativo. Nesse exemplo o Requisito de Segurança relativo à necessidade de estabelecer ou não um canal seguro depende, portanto, da localização do dispositivo móvel, podendo este estar dentro ou fora da empresa, ou melhor, dentro ou fora do ambiente corporativo.

Em outras palavras, o valor desejado dos Requisitos de Segurança (estabelecer canal seguro) é dependente do contexto, ou, mais especificamente, do valor de Informações de Contexto (localização).

Para determinar se os Requisitos de Segurança estão atendidos num determinado contexto, é preciso primeiro determinar quais são os Requisitos de Segurança relacionados com esse contexto. A solução proposta pelo Prometheus para representar os Requisitos de Segurança em função do Contexto é de formalizar essa dependência nas Regras de Segurança.

## 3.4 Arquitetura

A arquitetura do Prometheus, conforme ilustrado na Figura 3.2, consiste de um conjunto de componentes que provêem o serviço de segurança adaptativa ciente de contexto.O estágio atual da arquitetura é uma evolução da primeira versão, apresentada em [Pirmez 2008]. A arquitetura é composta por dois subsistemas principais: Núcleo e Ambiente de Segurança.

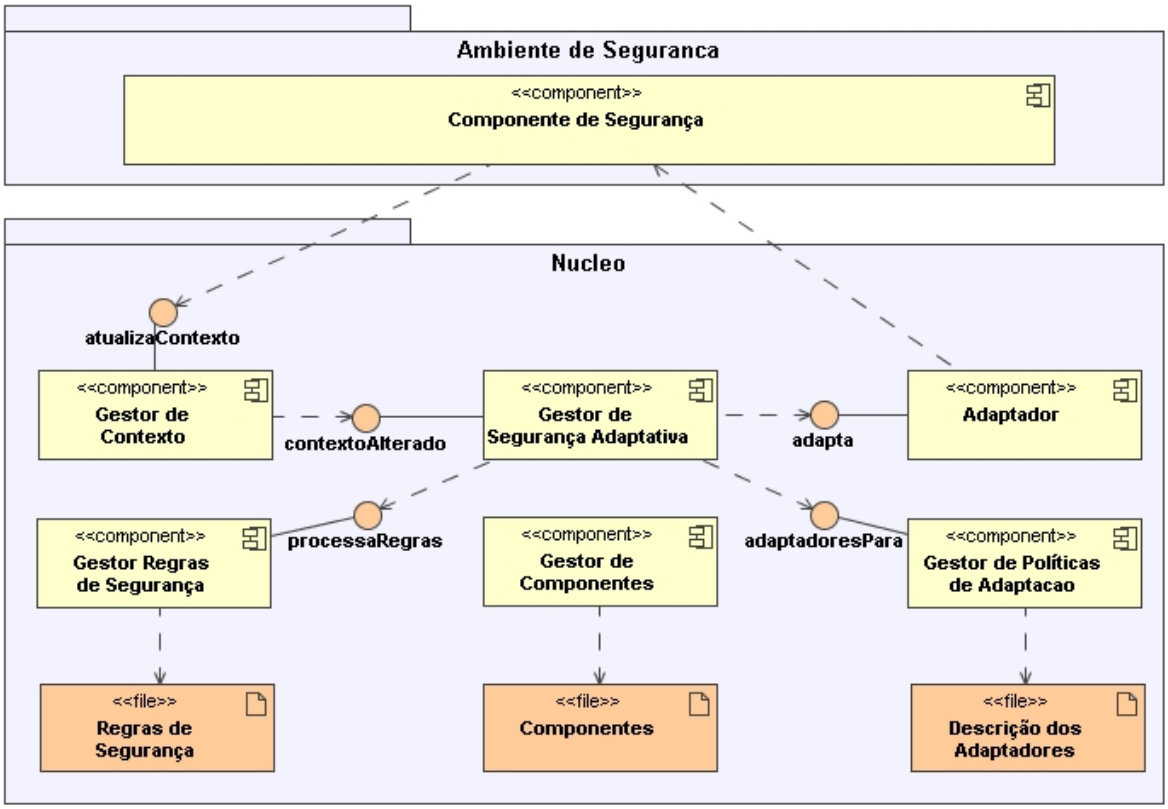


**Figura 3.2. Componentes do Prometheus**

O subsistema ***Ambiente de Segurança*** do Prometheus é formado por um conjunto de componentes especializados, os Componentes de Segurança, os quais são configurados dinamicamente de acordo com o contexto de execução e com os requisitos de segurança das aplicações.

Os ***Componentes de Segurança*** são classes que interagem com o contexto. Esses componentes são responsáveis por alimentar o núcleo do Prometheus com Informações de Contexto, e por executar Controles de Segurança que disparam ações que resultam em modificações das Informações de Contexto de forma a atender a Política de Segurança. Para tanto podem interagir diretamente com o ambiente de execução ou fazê-lo através de um *Middleware de Contexto* como o MoCA [Sacramento 2004] ou o Infraware [Pereira 2006]. As Informações de Contexto enviadas ao Núcleo, bem como os valores de Informações de

Contexo recebidos do Núcleo são mapeados em valores para instancias de classes *InfoContexto*.

É importante ressaltar que a arquitetura do Prometheus foi concebida de modo que não haja dependência entre os componentes do Núcleo do Prometheus em relação à implementação dos componentes do Ambiente de Segurança, ou seja, em relação aos Componentes de Segurança. Graças a isso, Componentes de Segurança podem ser adicionados ou retirados dinamicamente.

O subsistema ***Núcleo*** é responsável pela manutenção das Informações de Contexto e pela lógica de adaptação. Os componentes do Núcleo são o Gestor de Contexto, o Gestor de Segurança Adaptativa, o Gestor de Regras de Segurança, os Adaptadores, o Gestor de Políticas de Adaptação e o Gestor de Componentes.

O **Gestor de Contexto** é responsável por manter uma estrutura de dados com o valor corrente das Informações de Contexto. Essa estrutura de dados é a Tabela de Informação de Contexto, ou TIC.

O **Gestor de Segurança Adaptativa**, componente central na arquitetura, é responsável por comandar a análise do contexto e pelo disparo de Controles de Segurança para satisfazer aos Requisitos de Segurança em função do contexto.

O **Gestor de Regras de Segurança** é responsável por analisar o contexto e determinar quais são os Requisitos de Segurança para o contexto corrente e, ainda, por determinar quais desses requisitos não estão satisfeitos, ou seja, por determinar o Conjunto de Requisitos Não Atendidos (CRNA). O ***CRNA*** é obtido como resultado do processamento das Regras de Segurança localizadas no repositório ***Regras de Segurança***.

Os **Adaptadores** são classes que implementam a interface *Adaptador*. Os adaptadores têm o método *adapta*, que, quando chamado, interage com os Componentes de Segurança de forma a acionar Controles de Segurança específicos. O repositório ***Descrição dos Adaptadores*** contém informações sobre cada um dos adaptadores do acervo do Prometheus, inclusive o nome da classe que implementa o código do adaptador e quais Informações de Contexto podem ser modificadas pelos Controles de Segurança acionados pelo adaptador. Na Figura 3.3 pode ser visto o diagrama de classes com alguns adaptadores.

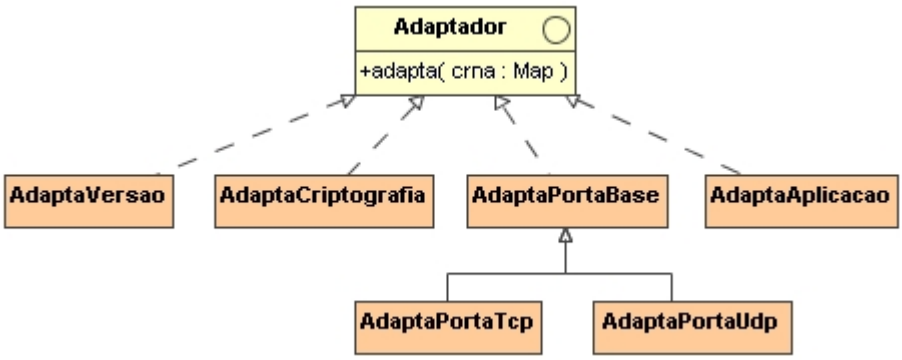


**Figura 3.3 Diagrama de Classes dos Adaptadores (parcial)**

O **Gestor de Políticas de Adaptação** é responsável por, dado um CRNA, determinar a estratégia para modificar o valor das Informações de Contexto no CRNA para valores que atendam aos Requisitos de Segurança, e, portanto, à Política de Segurança. Para isso, o Gestor de Políticas de Adaptação examina o repositório *Descrição dos Adaptadores* e seleciona um conjunto de adaptadores para interagir com os Componentes de Segurança. A ação desses adaptadores será de comandar junto aos Componentes de Segurança a execução de Controles de Segurança que devem ser acionados em resposta de uma ameaça. Essa execução deve provocar mudanças no contexto. Essas mudanças serão refletidas em novos valores para Informações de Contexto. Esses novos valores devem ser tais que a Política de Segurança seja atendida.

O **Gestor de Componentes** é responsável pelo ciclo de vida dos demais componentes do sistema. Na inicialização do Prometheus, o Gestor de Componentes carrega os demais componentes do núcleo e os Componentes de Segurança. Todas as referências a componentes são obtidas através do Gestor de Componentes, permitindo que os componentes possam ser carregados e descarregados dinamicamente. A relação de componentes que devem ser carregados pelo Gestor de Componentes é obtida a partir do repositório *Componentes*. O Gestor de Componentes também é descrito em maiores detalhes nessa seção.

Os principais componentes do Prometheus são descritos em detalhes nas subseções referente ao Ambiente de Segurança e ao Núcleo.

## 3.5 Componentes do Núcleo do Prometheus

Os componentes do subsistema Núcleo, conforme já mencionado, são: o Gestor de Contexto, o Gestor de Segurança Adaptativa, o Gestor de Regras de Segurança, o Gestor de Políticas de Adaptação, os Adaptadores e o Gestor de Componentes. A seguir será feita a descrição de cada um desses componentes.

### 3.5.1 Gestor de Contexto

O Gestor de Contexto é o componente responsável por manter uma estrutura de dados com o valor mais recente das Informações de Contexto. Esses valores ficam armazenados na Tabela de Informações de Contexto, ou TIC. O valor mais recente para cada uma dessas Informações de Contexto é fornecido por chamadas ao método *atualizaContexto* efetuadas pelos Componentes de Segurança. O método atualiza imediatamente o valor da Informação de Contexto na TIC; notifica o Gestor de Segurança Adaptativa de que houve mudança no contexto; e retorna.

A execução de *atualizaContexto* é simples e rápida. Por outro lado, o processamento relativo às modificações no contexto pelo Gestor de Segurança Adaptativa é uma tarefa complexa e de duração indeterminada. Sua execução é feita de forma assíncrona e pode ocasionar novas modificações no contexto.

É importante notar que todo o conteúdo da TIC é proveniente dos Controles de Segurança. Os valores são introduzidos na TIC através de uma chamada explícita pelo Controle de Segurança ao método *atualizaContexto*.

### 3.5.2 Gestor de Segurança Adaptativa

O Gestor de Segurança Adaptativa (GSA), componente central do Prometheus, é responsável por garantir que a Política de Segurança seja respeitada. Para isso, quando ocorrem modificações no contexto, o GSA, fazendo uso das Regras de Segurança, verifica se alguma dessas modificações comprometeu a Política de Segurança, isto é, se alguma informação de contexto passou a possuir valores que ferem a Política de Segurança. Se houver algo em desacordo, o GSA, fazendo uso da Política de Adaptação, seleciona adaptadores que invocarão Componentes de Segurança para acionar Controles de Segurança de modo a corrigir o problema.

O processo de verificação do atendimento da Política de Segurança através das Regras de Segurança, e os conseqüentes ajustes, pode ser demorado. Por outro lado, modificações no contexto podem ocorrer dezenas de vezes por segundo, ou mais. O processamento síncrono e seqüencial da verificação do atendimento da Política de Segurança através das Regras de Segurança e dos ajustes no contexto a cada modificação de uma Informação de Contexto poderia acarretar a perda de informações (*overrun*), além de comprometer o desempenho do equipamento. Para contornar esses problemas, a execução do GSA é feita numa linha de processamento (*thread*) separada. O Gestor de Contexto, ao registrar uma mudança no contexto, envia uma notificação ao GSA. Ao receber a notificação, o GSA obtém do Gestor

de Contexto uma cópia da estrutura de dados que armazena as informações de contexto (TIC – Tabela de Informações de Contexto). A cópia da TIC representa a situação corrente do contexto num determinado momento. O GSA trabalha sobre essa cópia, prevenindo assim os problemas decorrentes do acesso concorrente a dados compartilhados por mais de um processo. O GSA solicita ao Gestor de Regras de Segurança que calcule o Conjunto de Requisitos Não Atendidos (CRNA) para o contexto retratado na cópia da TIC. O CRNA é composto pelas Informações de Contexto que estão em desacordo com os Requisitos de Segurança para o contexto analisado. O GSA solicita então ao Gestor de Políticas de Adaptação quais adaptadores devem ser chamados para corrigir as violações dos Requisitos de Segurança encontradas no contexto. Os adaptadores interagem com Componentes de Segurança para acionar Controles de Segurança de modo a corrigir as violações. Cada adaptador pode interagir com um ou mais Componentes de Segurança. A atualização do contexto faz com que o Gestor de Contexto notifique novamente o GSA, que reinicia o procedimento. Esse laço se repete até que nenhuma modificação seja efetuada no contexto pelas Políticas de Adaptação, indicando que o contexto atende à Política de Segurança.

A interação entre o núcleo do Prometheus e o contexto propriamente dito é feita através dos Componentes de Segurança. Os Componentes de Segurança são especializados, cada um tratando de aspectos específicos do contexto. As mudanças significativas no contexto percebidas por esses Componentes de Segurança são mapeadas em valores para objetos *InfoContexto* e então repassadas ao Gestor de Contexto. Cada Componente de Segurança pode interagir diretamente com o contexto ou faze-lo com o auxílio de um *Middleware* de Contexto.

A Figura 3.4 apresenta mais detalhadamente a seqüência de procedimentos disparada em conseqüência da atualização de uma Informação de Contexto.

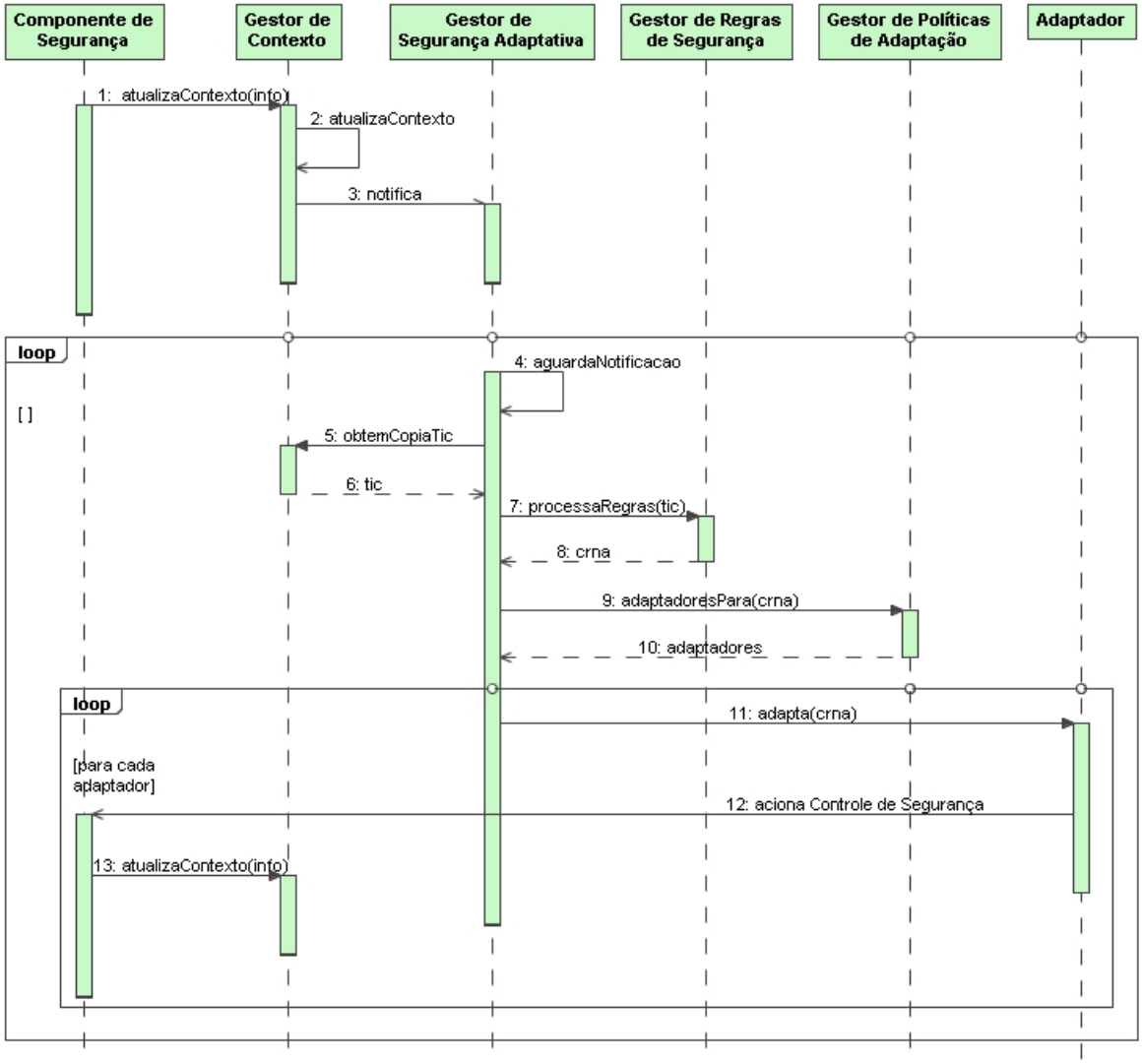


**Figura 3.4. Diagrama de Seqüência para a atualização de Informação de Contexto**

No passo (1) um Componente de Segurança informa ao Gestor de Contexto que houve uma modificação no contexto. Para isso chama o método *atualizaContexto* com uma Informação de Contexto e seu novo valor. Em (2) o Gestor de Contexto atualiza a Tabela de Informações de Contexto (TIC) com o novo valor da Informação de Contexto. Em (3) o Gestor de Contexto notifica o Gestor de Segurança Adaptativa de que houve modificação no contexto. O controle retorna imediatamente ao Gestor de Contexto. O processamento do Gestor de Segurança Adaptativa ocorre numa linha de execução (*thread*) separada, iniciada junto com o Prometheus. Em (4) o Gestor de Segurança Adaptativa aguarda até que ocorra uma modificação no contexto e, em (5) solicita uma cópia atualizada do contexto. Em (6) o Gestor de Contexto retorna ao GSA a cópia da estrutura de dados com as Informações de Contexto (TIC). Em (7) o GSA solicita ao Gestor de Regras de Segurança que verifique quais Informações de Contexto estão em desacordo com a Política de Segurança. O resultado desse

processamento (8) é o CRNA, ou Conjunto de Requisitos Não Atendidos. O CRNA é o conjunto de Informações de Contexto que devem ser modificadas para que o contexto volte a respeitar a Política de Segurança. Em (9) o Gestor de Segurança Adaptativa solicita ao Gestor de Políticas de Adaptação quais adaptadores (10) devem ser chamados a fim de modificar as Informações de Contexto no CRNA para os valores desejados. Em (11) cada um dos adaptadores é executado. Em (12) cada adaptador interage com um ou mais Componentes de Segurança, modificando sua configuração ou acionando Controles de Segurança. A ação dos adaptadores sobre os Controles de Segurança provoca mudanças no contexto. Os Controles de Segurança mapeiam essas mudanças em valores para objetos *InfoContexto* e (13) repassam esses valores para o Gestor de Contexto. Em seguida a execução retorna ao ponto (4) onde o Gestor de Políticas de Adaptação aguarda por modificações no contexto. Observe que a execução do laço continua até que não seja necessário realizar nenhuma adaptação, ou seja, que o CRNA retornado no passo 8 seja um conjunto vazio.

### 3.5.3 Gestor de Regras de Segurança

O Gestor de Regras de Segurança é responsável por analisar a cópia da Tabela de Informação de Contexto (cópia da TIC) e verificar se há alguma combinação de valores para as Informações de Contexto que represente uma ameaça. Ou seja, o Gestor de Regras de segurança busca determinar para um dado contexto quais Informações de Contexto devem ser modificadas. O resultado desse procedimento de análise é o Conjunto de Requisitos Não Atendidos (CRNA).

O CRNA é a coleção de Informações de Contexto cujo valor no contexto analisado está em desacordo com as Regras de Segurança, isto é, com a Política de Segurança. O CRNA contém também o valor que cada uma das Informações de Contexto deveria ter para que as Regras de Segurança sejam atendidas. O procedimento de obtenção do CRNA é composto por dois passos, a saber: (i) determinar Requisitos de Segurança para um dado contexto e (ii) verificar quais Informações de Contexto não satisfazem aos Requisitos de Segurança.

### 3.5.4 Determinar Requisitos de Segurança

O primeiro passo consiste em determinar quais são os Requisitos de Segurança para o contexto fornecido pelo Gestor de Segurança Adaptativa. Os Requisitos de Segurança, como já mencionado, são representados no Prometheus como uma coleção de valores referenciais para Informações de Contexto. Os Requisitos de Segurança são obtidos **executando** as Regras da Segurança. Ou seja, dado o contexto e fazendo uso da Regras de Segurança obtém-se os

Requisitos de Segurança. As Regras de Segurança são compostas por condições e ações. As condições são expressões lógicas que relacionam o valor de Informações de Contexto com valores referenciais. Para cada regra em que as condições foram verdadeiras são executadas as ações. Cada ação atribui um valor para um Requisito de Segurança. Todas as regras são executadas em seqüência, e os valores das ações para as regras em que as condições foram verdadeiras são adicionados a uma coleção. Se uma ação atribuir um valor para um Requisito de Segurança para o qual outra regra já tenha determinado um valor, então é usado um valor tal que satisfaça às duas regras. Esse valor é obtido combinando o novo valor com o valor anterior através da operação *uniao* da Informação de Contexto correspondente. A coleção resultante é o conjunto de Requisitos de Segurança para o contexto analisado (Requisito Representativo).

### 3.5.5 Verificar Informações de Contexto que não satisfazem aos Requisitos de Segurança

O segundo passo consiste de verificar junto à cópia da TIC quais Informações de Contexto não satisfazem os Requisitos de Segurança obtidos no primeiro passo. Essas Informações de Contexto não satisfeitas vão integrar Conjunto de Requisitos Não Atendidos (CRNA). O procedimento para construir esse conjunto é de comparar os valores dos Requisitos de Segurança com os valores das Informações de Contexto correspondentes na cópia da TIC. Essa comparação verifica, para cada Informação de Contexto, se o valor disponível no contexto atende ao valor requisitado. As Informações de Contexto cujo valor na cópia da TIC não atendem ao valor requisitado são adicionadas ao CRNA junto com o valor requisitado. Para determinar se o Requisito de Segurança está atendido é usado o método *contem* do objeto *InfoContexto* relativo ao Requisito de Segurança. Assim, se o valor da Informação de Contexto **não *contem*** o valor do Requisito de Segurança, então o Requisito de Segurança **não está satisfeito**. O CRNA será formado, no final desse passo, pelo conjunto de Informações de Contexto cujo valor viola a Política de Segurança no contexto examinado, e o valor que essas Informações de Contexto deveriam conter para que não ocorresse a violação, ou seja, o valor requisitado.

### 3.5.6 Adaptadores

Adaptadores são implementações da interface ***Adaptador*** capazes de interagir com os Componentes de Segurança. Cada implementação de Adaptador interage com um ou mais Componentes de Segurança para que estes acionem Controles de Segurança específicos. Para

acionar os Controles de Segurança os Componentes de Segurança atuam sobre o ambiente operacional. Podem fazê-lo por meio de um *middleware* de contexto ou diretamente. O acionamento desses Controles de Segurança provoca modificações no contexto. Essas modificações são registradas pelos Controles de Segurança junto ao Gestor de Contexto. Em resumo, os adaptadores são usados para fazer com que determinadas Informações de Contexto atinjam valores desejados.

Cada *Adaptador* tem um método *adapta*. Esse método recebe o conjunto de Informações de Contexto que se quer modificar e os valores desejados. O método *adapta* seleciona, dentre as Informações de Contexto recebidas, os valores daquelas sobre as quais é capaz de atuar. Esses valores são usados pela lógica do adaptador na sua interface com os Controles de Segurança. Os Controles de Segurança por sua vez atuam sobre o ambiente de operacional. A conseqüência dessa atuação é refletida no contexto.

Há uma dependência estreita entre a implementação dos Adaptadores e a dos Componentes de Segurança. Essa dependência, entretanto, é transparente para o núcleo do Prometheus que conhece apenas a interface *Adaptador*.

Assim como no caso dos Controles de Segurança, não há dependência do núcleo do Prometheus em relação à implementação de qualquer adaptador. Isto permite que, também os adaptadores, possam ser carregados dinamicamente.

### 3.5.7 Gestor de Políticas de Adaptação

O Prometheus é um serviço ciente de contexto (*context aware*), ou seja, ele percebe as mudanças no ambiente onde está sendo executado e faz uso das informações de contexto para ajustar o seu funcionamento. Assim, as modificações no ambiente onde as aplicações estão sendo executadas se refletem no contexto. Essas modificações afetam o valor de uma ou mais Informações de Contexto. Ao perceber uma modificação, o Prometheus examina as Informações de Contexto a fim de verificar se a Política de Segurança está sendo respeitada.

Por exemplo, a Política de Segurança pode determinar que se a energia residual do dispositivo estiver acabando, devem ser tomadas providências para economizar energia de modo a preservar a disponibilidade da informação, mesmo que a qualidade do serviço prestado fique um pouco prejudicada. Essa determinação da Política de Segurança deve ser formalizada em Regras de Segurança.

Uma das Regras de Segurança poderia definir, por exemplo, que, se a Informação de Contexto para a Energia Residual do dispositivo tiver o valor corrente igual a "*Pouca*", então o valor desejado para "*Iluminação da Tela*" deve ser igual a "*Fraca*". Assim, caso a energia

residual do dispositivo passe a "*Pouca*", o Requisito de Segurança "*Iluminação da Tela*" passa a ter o valor "*Fraca*". Se o valor de "Iluminação da Tela" no contexto corrente não tiver o valor "Fraca" será disparado um procedimento cuja ação será a de modificar a "*Iluminação da Tela*" para o valor desejado, "*Fraca*". Esse procedimento é uma adaptação.

Assim, o Gestor de Políticas de Adaptação (***GPA***) é responsável por escolher, dado um conjunto de Informações de Contexto cujo valor não atende aos Requisitos de Segurança, quais adaptadores devem ser utilizados para tornar o ambiente seguro. Para isso o GPA utiliza o Mecanismo de Seleção de Adaptadores.

Os critérios utilizados na escolha dos adaptadores por parte do Mecanismo de Seleção dos Adaptadores dizem respeito aos Requisitos de Segurança das aplicações em execução e englobam, dentre outros: (i) a restrição no uso de serviços específicos (portas) e no uso simultâneo de serviços; (ii) a necessidade de acesso a determinados sítios; (iii) o uso de protocolos seguros de transporte (SSL), de rede (IPSEC) ou de enlace (IEEE 802.11i); (iv) o uso de um tipo especifico de rede (pública ou corporativa); e (v) a prioridade de um aplicativo específico em relação a outros aplicativos em uso no dispositivo. Além disso, são considerados ainda: (vi) os requisitos do dispositivo - englobam o consumo de energia e o gasto de memória, ao serem aplicadas adaptações; e (vii) os requisitos de rede - englobam informações sobre os tipos de redes sem fio disponíveis no dispositivo e qual é o protocolo de segurança no nível de enlace adotado em cada uma dessas redes.

O Mecanismo de Seleção de Adaptadores faz uso de um conjunto I de requisitos de segurança; do acervo J de adaptadores; e do conjunto A de aplicações controladas pelo Prometheus. Cada dispositivo móvel possui (i) um subconjunto de aplicações de A em uso, e (ii) um subconjunto dos requisitos I de segurança, os quais podem, ou não, ser atendidos pelo contexto corrente (TIC). Esse mecanismo utiliza o CRNA (Conjunto de Requisitos Não Atendidos) produzido pelo Gestor de Regras; e o conjunto J de adaptadores do Prometheus.

A seleção dos adaptadores não é uma tarefa trivial, pois não há uma relação biunívoca entre adaptador e requisito. Um adaptador pode tratar de mais de uma Informação de Contexto. Da mesma forma, uma Informação de Contexto pode ser tratada por mais de um adaptador.

Os procedimentos do ***Mecanismo de Seleção de Adaptadores*** são agrupados em etapas detalhadas a seguir: (1) escolher adaptadores potenciais; (2) calcular a função utilidade; e (3) selecionar adaptadores.

1. *Escolher Adaptadores Potenciais*

O objetivo dessa etapa é obter, para cada Informação de Contexto no CRNA, o conjunto de adaptadores capazes de tratá-la. Esse conjunto é obtido a partir de uma busca no acervo de adaptadores do Prometheus (repositório *Descrição dos Adaptadores)*. A ação a tomar depende do número de adaptadores encontrados. Se apenas um adaptador for encontrado, então ele é o escolhido. Se mais de um adaptador foi encontrado, então a decisão é deixada para a próxima etapa.

O repositório ***Descrição de Adaptadores*** usado nesta etapa está detalhado na seção 3.5.11.

2. *Calcular a Função Utilidade*

O cálculo da função utilidade é necessário para ajudar na seleção da melhor alternativa apenas quando há mais de um adaptador capaz de atender a um dado requisito. A função retorna um valor que é tanto maior quanto mais "útil" for o adaptador. O cálculo dessa função pode ser visto em detalhes na seção 3.5.8 .

3. *Selecionar Adaptadores*

A meta dessa etapa do mecanismo é de selecionar os melhores adaptadores fazendo uso da ***Função Utilidade*** $\boldsymbol{F_j}$. Busca-se uma solução que maximize $\boldsymbol{F_j}$ de forma a escolher o melhor adaptador possível dentre o conjunto de adaptadores obtidos na primeira etapa, garantindo um ambiente seguro e, ao mesmo tempo, o uso eficiente dos recursos do dispositivo.

Devido à existência de adaptadores que atendem a mais de uma Informação de Contexto e, ao mesmo tempo, para evitar escolher adaptações redundantes, a segunda e a terceira etapa são executadas repetidamente até que não exista nenhum requisito no conjunto de requisitos não atendidos (CRNA). Assim, para cada rodada, após escolher o adaptador que maximize a função utilidade, são retirados do CRNA os requisitos para os quais já foi selecionado um adaptador. Adicionalmente, para cada rodada, são eliminados do conjunto de adaptadores obtidos na etapa 1 os adaptadores que não satisfazem a nenhum dos requisitos dentre os restantes no CRNA.

### 3.5.8 Cálculo da Função Utilidade

O cálculo da função utilidade é necessário apenas quando há mais de um adaptador capaz de satisfazer a um dado requisito. Para o cálculo da função utilidade $\boldsymbol{F_j}$ de um adaptador

*j*, adotou-se uma abordagem orientada à segurança: é dada prioridade aos adaptadores que tenham *maior relevância* do ponto de vista de segurança e, ao mesmo tempo, tenham um *menor impacto* para o dispositivo, ou seja, que gerem o menor consumo de energia e gastem o menor tempo para efetuar a adaptação.

A relevância de um adaptador é tanto maior quanto maior for o número de requisitos que ele é capaz de tratar. Assim, a ***relevância*** $\boldsymbol{I_j}$ de um adaptador *j* é estimada da seguinte forma: (i) calcula-se a diferença entre o número de elementos na CopiaCRNA e o número de Informações de Contexto tratadas por esse adaptador; essa diferença é dividida pelo número de elementos na CopiaCRNA.

Atribui-se uma ***penalidade*** $\boldsymbol{P_j}$ a um adaptador *j* para cada Informação de Contexto tratada pelo adaptador e não requisitada, estimada como segue: (i) calcula-se a soma do número de elementos na CopiaCRNA com o número de Informações de Contexto tratadas pelo adaptador e não requisitados; e (ii) obtém–se $P_j$ através da divisão do número de Informações de Contexto tratadas pelo adaptador e não requisitadas, pela soma calculada em (i).

Com ***relação ao impacto*** do uso do adaptador para o dispositivo, busca-se por um adaptador com menor consumo de energia e maior rapidez no processo de adaptação. A execução de um adaptador é um dos fatores que afetam o consumo de energia $W_j$ do dispositivo. A ***energia residual*** *U* de um dispositivo é definida como sendo a quantidade de energia atual que o dispositivo possui em um dado instante de tempo. Assim, o termo $(U – W_j)$ denota a energia residual do dispositivo após o uso do adaptador *j* (energia inicial *U* menos o consumo de energia do dispositivo no processo de adaptação $W_j$). Em suma, busca-se escolher um adaptador *j* para a qual o termo $(U – W_j)$ seja maximizado.

No que se refere à ***Duração Relativa da Adaptação (DRA)***, busca-se escolher um adaptador *j* no qual o tempo gasto para efetuar a adaptação, $D_j$, seja o menor. Para tal, define-se a ***Duração Máxima de Adaptação (DMA)*** como sendo o maior tempo gasto para efetuar uma adaptação dentre os adaptadores do conjunto ***J***. Assim, a duração *DRA* é definida como sendo $(DMA – D_j) / DMA$. Em suma, busca-se escolher um adaptador j para a qual o termo $(DMA – Dj) / DMA$ seja maximizado. Logo, a ***Função Utilidade*** $\boldsymbol{F_j}$ (Fórmula 3-1) pode ser representada por:

$$F_j = \alpha\left(I_j - P_j\right) + \beta\left(\frac{U - W_j}{U}\right) + \gamma\left(\frac{DMA - D_j}{DMA}\right) \text{ onde } \alpha + \beta + \gamma = 1$$

**Fórmula 3-1. Função Utilidade**

O primeiro termo, $\alpha(I_j - P_j)$, dá a utilidade ao adaptador $j$ de acordo com sua relevância. Quanto maior o número de Informações de Contexto tratadas pelo adaptador que estejam no CRNA, maior sua relevância. O segundo termo, $\beta(U - W_j)/U$, dá a utilidade de $j$ de acordo com a estimativa de energia consumida para fazer a adaptação. Quanto menor a energia consumida, maior a utilidade. O terceiro termo, $\gamma(DMA - D_j)/DMA$, dá a utilidade de j de acordo com o tempo estimado para fazer a adaptação. Quanto menor o tempo, maior a utilidade. As constantes α, β e γ equilibram o peso dos termos na fórmula. Os valores (0,1; 0,1 e 0,8) por exemplo, podem ser usados num perfil em que é dada maior importância ao tempo de adaptação. Já num perfil em que é dada importância exclusivamente à energia consumida devem ser usados os valores (0, 1, 0).

A falta de ferramental adequado impediu que fossem efetuadas medidas de consumo de energia. Nos testes efetuados foi dada igual importância para a relevância dos adaptadores e para a duração estimada das adaptações. Assim, foram usados α = ½, β = 0 e γ = ½. O valor ideal para essas constantes será objeto de estudos futuros.

### 3.5.9 Gestor de Componentes

O Gestor de Componentes é responsável pelo ciclo de vida dos Componentes de Segurança. Cabe ao Gestor de Componentes obter do repositório *Componentes* as informações sobre os componentes, carregar e descarregar dinamicamente os componentes da memória. A Figura 3.5 mostra um trecho do repositório *Componentes*.

```
<componente nome="ControleConexoes" versao="1.03">
  <impl>prometheus.ssh.ganymed.ControleConexoesImpl</impl>
</componente>

<componente nome="Firewall" versao="3.05">
  <impl>prometheus.firewall.windows.FirewallWindows</impl>
  <init>"C:\Documents and Settings\user\FwUtil\FwUtil.exe"</init>
</componente>

<componente nome="ControleExecucao" versao="3.05">
  <impl>prometheus.controle.ControleExecucao</impl>
</componente>
```

**Figura 3.5. Trecho do repositório *Componentes***

Na Figura 3.5 são descritos três Componentes de Segurança. O atributo "nome" é usado para obter uma referência à instância do componente. O atributo "versao" é usado pelo procedimento de atualização dos componentes. O valor "impl" informa o nome da classe que

implementa o componente. O valor "init" é um parâmetro, opcional, para a rotina de inicialização do componente.

Ao ser carregado, na inicialização do Prometheus, o Gestor de Componentes é responsável por ler as informações do repositório Componentes, instanciar os componentes e construir um mapa relacionando as instâncias dos componentes com seus respectivos nomes.

As principais operações do Gestor de Componentes podem ser vistas na Figura 3.6.


**Figura 3.6. Gestor de Componentes**

O método *instanciaDe* retorna uma referência à instância do componente para o nome fornecido. Para permitir a carga dinâmica de componentes, os usuários desses componentes não armazenam uma referência a esses componentes. Em vez disso, obtêm uma referência ao componente chamando *instanciaDe*.

O método *atualizaComponentes* é chamado como parte do procedimento de atualização do Prometheus descrito no item 3.6.2. A atualização dos componentes é feita em duas etapas: (i) são descarregados os componentes que não constam do repositório *Componentes*; (ii) são carregados os componentes que constam do repositório *Componentes* e cuja versão mudou; e (iii) é atualizada a referência à instância dos componentes carregados no mapa que relaciona os componentes com seus nomes.

Componentes de Segurança podem manter estruturas de dados com seu estado de execução. Essas estruturas de dados podem ser complexas e depender fortemente da implementação, tornando muito complexa a substituição de uma nova versão do componente em tempo de execução. A solução utilizada para essas situações é que quando a nova versão do componente é instalada, ela mantém uma referência à versão anterior, e usa o código e as estruturas de dados da versão anterior sempre que for preciso. A referência à versão anterior do componente é mantida pela nova versão até que ela não seja mais necessária. O momento exato em que ela não é mais necessária é altamente dependente da implementação do componente.

Por exemplo, o Componente de Conexões cria túneis dentro de sessões SSH para garantir a privacidade e integridade dos dados trafegados entre o dispositivo e servidores de dados e faz uso de estruturas de dados para refletir o estado de sua execução. A instância da versão anterior do Componente de Conexões é mantida até que sejam fechadas todas as

conexões que já estavam estabelecidas no momento da troca de versão. Em outros casos, em que as estruturas de dados são mais simples, a descarga da instância da versão anterior pode ser imediata.

### 3.5.10 Repositório Regras de Segurança

Conforme já mencionado, as Regras de Segurança do Prometheus são usadas para obter os Requisitos de Segurança para um dado contexto. Esse repositório consiste de um conjunto de regras expressas na forma Condição – Ação. Cada regra pode ter zero ou mais condições, e uma ou mais ações. Se todas as condições de uma regra da Política de Adaptação forem satisfeitas, então as ações relacionadas devem ser executadas. "Executar" uma ação significa atribuir um valor ao Requisito de Segurança para a Informação de Contexto indicada.

```
<regra nome="Cadastro">
  <condicao contexto="InfoAplicacao" op="contem">Cadastro</condicao>
  <acao requisito="InfoSaidaTcp">srv.glostora.com:22</acao>
  <acao requisito="InfoTunelSsh">
      srv:8080:www.empresa.com:80
  </acao>
</regra>
```

**Figura 3.7. Exemplo de Regra de Segurança**

Um exemplo de Regra de Segurança é ilustrado na Figura 3.7, onde o nome da regra é "Cadastro". O nome existe apenas para efeito de documentação. A parte da regra relativa à condição é verdadeira se o valor da Informação de Contexto *InfoAplicacao* contiver "Cadastro". Essa regra tem duas ações. As ações indicam os valores que os Requisitos de Segurança para as Informações de Contexto *InfoTunelTcp* e *InfoSaidaTcp* devem conter. "Conter" neste caso se refere à implementação da operação *contem* para a Informação de Contexto correspondente. Essa regra estabelece que, se a aplicação "Cadastro" estiver em execução, então deve haver um túnel SSH entre o dispositivo e o servidor definido pela referência "srv" ligando conexões à porta local 8080 com o *site* em www.empresa.com na porta 80; e que o *Firewall* local do dispositivo deve que sejam efetuadas conexões TCP a partir do dispositivo com o endereço "srv.glostora.com" na porta 22.

A sintaxe da parte da regra relativa à condição permite avaliar:

- uma relação entre o valor de uma Informação de Contexto e uma constante;
- a relação entre os valores de duas Informações de Contexto; ou ainda;
- testar a presença de algum valor para uma Informação de Contexto.

Por exemplo, a primeira regra, "Inicializacao", na Figura 3.8 não possui condições, há apenas ações e, conseqüentemente, as ações devem ser executadas sempre, incondicionalmente. Esse tipo de regra é utilizado para estabelecer valores mínimos para Requisitos de Segurança.

```
<regra nome="Inicializacao">
  <acao requisito="InfoCriptografia">FRACA</acao>
</regra>

<regra nome="Versao">
  <condicao contexto="InfoVersaoDisponivel" op="existe"/>
  <condicao contexto="InfoVersaoDisponivel" op="naoIgual"
            valorDe="InfoVersaoEmUso"/>
  <acao requisito="InfoVersaoEmUso"
          valorDe="InfoVersaoDisponivel"/>
</regra>
```

**Figura 3.8. Regra incondicional e relação entre Informações de Contexto**

A segunda regra da Figura 3.8, "Versao", tem duas condições. As ações de regras com mais de uma condição são executadas apenas se todas as condições forem verdadeiras. A primeira condição faz uso da operação *existe* para verificar se existe algum valor para a Informação de Contexto indicada. A condição é verdadeira se o resultado da operação for verdadeiro. A segunda condição faz uso do operador *naoIgual* para comparar o valor de duas Informações de Contexto entre si. A condição é verdadeira se o resultado da operação for verdadeiro. Se ambas as condições forem verdadeiras, a ação desta regra consiste em indicar que o valor desejado para *InfoVersaoEmUso* deve ser igual ao valor corrente de *InfoVersaoDisponivel*.

A Tabela 3.1 resume as possíveis operações com Informações de Contexto usadas na parte relativa às condições das regras de Política de Adaptação do Prometheus bem como seus operandos. O Prometheus não é responsável por nenhum tipo de validação sobre a compatibilidade entre os valores de diferentes Informações de Contexto em relação às operações. Essa validação é de responsabilidade de quem elabora as regras da Política de Adaptação.

**Tabela 3.1. Operações usadas nas condições das Regras de Segurança**

| Operação | Operando 1 | Operando 2 | Descrição |
|---|---|---|---|
| *contem* | *contexto* | constante / *valorDe* | resultado da operação *contem* † |
| *naoContem* | *contexto* | constante / *valorDe* | negativa da operação *contem* |
| *existe* | *contexto* | - | se existe valor não nulo |
| *naoExiste* | *contexto* | - | negativa da operação *existe* |
| *igual* | *contexto* | constante / *valorDe* | resultado da operação *igual* |
| *naoIgual* | *contexto* | constante / *valorDe* | negativa da operação *igual* |

† A operação utilizada no teste é sempre a da classe indicada pelo *Operando 1*

Os Requisitos de Segurança das aplicações podem incluir proibições. Exemplos disso são a proibição de executar duas aplicações específicas simultaneamente, ou a proibição de executar uma aplicação com um determinado tipo de conexão à rede. Essas proibições são representadas como "conflitos", um tipo especial de regra no repositório Regras de Segurança. A Figura 3.9 mostra alguns exemplos dessas regras.

```
<conflito>
  <condicao contexto="InfoAplicacao" op="contem">Cadastro</condicao>
  <condicao contexto="InfoAplicacao" op="contem">Bate-Papo</condicao>
</conflito>

<conflito>
  <condicao contexto="InfoAplicacao" op="contem">Financeiro</condicao>
  <condicao contexto="InfoPosicao" op="naoContem">Matriz</condicao>
  <condicao contexto="InfoPosicao" op="naoContem">Filial</condicao>
</conflito>
```

**Figura 3.9. Regras para Requisitos de Segurança conflitantes**

A condição de conflito é verdadeira quando todas as condições são verdadeiras. A primeira regra da Figura 3.9 indica que há um conflito quando as aplicações "Cadastro" e "Bate-Papo" estão em execução simultaneamente. A segunda regra indica um conflito se a aplicação "Financeiro" estiver em execução e a Informação de Contexto *InfoPosicao*, que indica a posição geográfica do dispositivo, não contiver nem "Matriz" e nem "Filial", ou seja, se a aplicação "Financeiro" estiver sendo executada fora das dependências da Matriz e da Filial da empresa.

### 3.5.11 Repositório Descrição de Adaptadores

O Prometheus dispõe, para efetuar as adaptações, de um acervo de adaptadores. Cada adaptador tem capacidades e características diferentes. Essas características estão descritas no repositório **Descrição dos Adaptadores**. Esse repositório contém valores para os atributos de cada um dos adaptadores. Um trecho desse repositório pode ser visto na Figura 3.10.

```
<adaptadores>
  ...
  <adaptador
          classe="prometheus.adaptador.AdaptaPortaTcpLocalAberta"
          duracao="14" tamanho="1287" energia="5000" versao="1.01">
    <infoContexto>InfoPortaTcpAberta</infoContexto>
  </adaptador>
  <adaptador
          classe="prometheus.adaptador.AdaptaPortaUdpLocalAberta"
          duracao="14" tamanho="1314" energia="5000" versao="1.03">
    <infoContexto>InfoPortaUdpAberta</infoContexto>
  </adaptador>
  ...
</adaptadores>
```

**Figura 3.10. Repositório Descrição dos Adaptadores**

Para cada adaptador no repositório está descrito o nome da classe que implementa o adaptador (*classe*); o tempo máximo esperado da execução de uma adaptação (*duracao*); o tamanho do código desse adaptador (*tamanho*); uma estimativa da quantidade de energia consumida para executar uma adaptação (*energia*); e a versão do código que implementa o adaptador (*versao*). Adicionalmente, é fornecida uma relação de Informações de Contexto sobre a qual esse adaptador é capaz de atuar.

## 3.6 Ambiente de Segurança

O subsistema ***Ambiente de Segurança***, conforme já mencionado, é formado por um conjunto de componentes especializados, os ***Componentes de Segurança***, os quais são configurados dinamicamente pelos adaptadores. Exemplos de Componentes de Segurança são, dentre outros: Componente de Energia, Componente de Execução, Componente de Posição, Componente de Redes, Componente de Conexões, Componente de Firewall e Componente de Versão. A Figura 3.11 mostra um diagrama de classes parcial com alguns Componentes de Segurança.

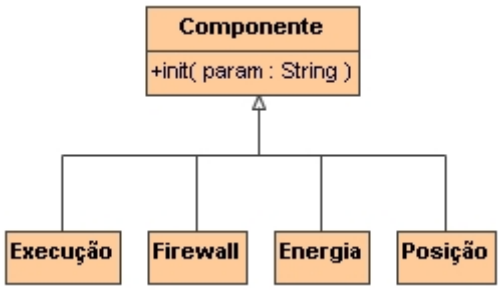


**Figura 3.11. Diagrama de classes parcial dos Componentes de Segurança**

O **Componente de Energia** provê informações sobre a energia residual do dispositivo. Ele produz a Informação de Contexto *InfoEnergiaResidual*, que pode ter os valores "muita", "media", "pouca" ou "critica". Ao ser carregado, o Componente de Energia solicita ao sistema operacional que informe sobre a energia residual do dispositivo como parte da estratégia para garantir a disponibilidade da informação. Assim, o sistema operacional informa ao Componente de Energia quando o nível de energia do dispositivo mudar de um limiar pré-determinado para outro. Quando esse limiar for ultrapassado, o Componente de Energia informa ao Gestor de Contexto o novo valor de InfoEnergiaResidual. Essa mudança no contexto pode, por exemplo, ocasionar o disparo de Controles de Segurança por meio de adaptadores para prolongar a disponibilidade da informação reduzindo o consumo de energia. Possíveis estratégias usadas por adaptadores são: reduzir a iluminação na tela, diminuir o volume do som ou usar configurações alternativas de redes que consumam menos energia.

O **Componente de Memória** provê informações sobre a memória disponível no dispositivo. Ele produz a Informação de Contexto *InfoMemoriaDisponivel*, medido em KB (1KB = 1024 bytes). Ao ser carregado, o Componente de Memória obtém do sistema operacional a quantidade de memória disponível e informa esse valo ao Gestor de Contexto. Periodicamente a quantidade de memória disponível é obtida novamente do sistema operacional. Se a diferença entre a quantidade de memória disponível atual e o último valor informado ao Gestor de Contexto for maior que um certo patamar, então o novo valor é informado ao Gestor de Contexto. Caso essa mudança no contexto indique que há "pouca" memória disponível pode, por exemplo, ser disparado um Controle de Segurança, por meio de um adaptador, de forma a liberar estruturas de dados que não estejam mais em uso.

O **Componente de Execução** é usado para disparar aplicações e garantir os requisitos de segurança do aplicativo. O mecanismo usado para garantir esses requisitos está descrito em detalhes no item 3.6.1.

O **Componente de Posição** provê informações sobre a posição geográfica de um dispositivo. O Componente de Posição é configurado com coordenadas que delimitam áreas específicas, de interesse da Política de Segurança, e informa quando o dispositivo entra ou sai de uma dessas áreas. Essa informação é registrada pelo Componente de Posição junto ao Gestor de Contexto como um valor para *InfoPosicao*. Possíveis valores para *InfoPosicao* são "*EscritorioCentral*", "*Fabrica*", ou "*LocalPublico*".

O **Componente de Rede** provê informações sobre as redes disponíveis para a conexão e os mecanismos usados para sua conexão. Ao ser carregado, o Componente de Rede solicita ao sistema operacional que informações sobre o nível do sinal recebido no dispositivo e o volume de dados recebidos por unidade de tempo (volume de tráfego). Assim, o sisema operacional informa ao Componente de Rede quando o nível do sinal recebido pelo dispositivo ou o volume de dados recebidos por unidade de tempo cruzarem os respectivos limiares pré-determinados. Quando algum desses limiares for ultrapassado, o Componente de Rede informa ao Gestor de Contexto o novo valor de *InfoNivelSinalRede ou de InfoTrafegoRede*.

Determinadas combinações de valores de *InfoNivelSinalRede* e de *InfoTrafegoRede* podem ser interpretadas como ataque de *Jamming* (*InfoNivelSinalRede = MuitoAlt*o e *InfoTrafegoRede = MuitoAlto*). Ataques desse tipo fazem o dispositivo consumir energia inutilmente. Um exemplo simples é a ocorrência de Jamming quando a rede não está sendo utilizada por nenhum aplicativo que esteja em execução no dispositivo. Assim, caso a mudança no contexto indique que os valores MuitoAlto para *InfoNivelSinalRede e MuitoAlto para InfoTrafegoRede* pode, por exemplo, ocasionar o disparo de Controles de Segurança por meio de adaptadores de forma a desativar temporariamente a interface de rede para, dessa forma, preservar a energia residual do dispositivo.

O **Componente de Firewall** permite abrir ou fechar portas TCP ou UDP dinamicamente no *firewall* local do dispositivo para, dessa forma, permitir ou não efetuar/receber conexões TCP, e enviar/receber pacotes UDP.

Durante sua inicialização o Componente de Firewall registra junto ao Gestor de Contexto o valor das informações de contexto *InfoPortaTcpAberta, InfoSaidaTcp, InfoPortaUdpAberta* e *InfoSaidaUdp* com a situação corrente das permissões no *firewall* local do dispositivo. Em seguida, o Componente de Firewall monitora o estado de abertura das portas TCP e UDP no *firewall* e registra qualquer mudança junto ao Gestor de Contexto, informando um novo valor para *InfoPortaTcpAberta, InfoSaidaTcp, InfoPortaUdpAberta* ou para *InfoSaidaUdp*. O adaptador *AdaptaFirewall* atua sobre o Componente de Firewall

solicitando ajustes nas permissões do *firewall* conforme os valores requisitados para *InfoPortaTcpAberta* e *InfoPortaUdpAberta*.

O **Componente de Conexões** busca proteger as informações em trânsito contra a modificação/divulgação por uma entidade não autorizada, ou seja, garantir a integridade e a confidencialidade da informação. Para garantir tais requisitos, adaptadores podem ser acionados dependendo do nível de segurança oferecido pelo ambiente onde o dispositivo está localizado (hostil: rede pública - ou não hostil: rede corporativa) e do nível de segurança oferecida pela rede (segura – em termos de presença de algoritmo de criptografia - ou insegura). Os adaptadores acionados criam um sistema de túneis providos através de uma sessão SSH (*secure shell*) [RFC4251] até o servidor, provendo a criptografia mínima para garantir os requisitos de segurança da aplicação e estabelecendo a melhor relação entre criptografia e consumo de recursos, tais como memória e energia. Quanto ao funcionamento do Componente de Conexões, este aguarda que as aplicações se conectem aos servidores. A cada estabelecimento de conexão entre uma aplicação e seu servidor, o Componente de Conexões intercepta a conexão e cria um sistema de túneis até o servidor usando o algoritmo de criptografia adequado. A descrição detalhada do Componente de Conexões será vista na seção 3.6.3.

Por fim, o **Controle de Versão** verifica periodicamente junto a um servidor específico se deve ser feita a atualização dos componentes e dos repositórios. Os procedimentos para efetuar essa atualização estão descritos em detalhes na seção 3.6.2.

A relação de Componentes de Segurança descrita acima não é exaustiva. Como já mencionado, o Ambiente de Segurança do Prometheus não está restrito a estes Componentes de Segurança. A arquitetura foi concebida de forma que novos Componentes de Segurança possam ser acrescentados, retirados ou substituídos a qualquer momento, inclusive, com o Prometheus em execução.

### 3.6.1 Descrição do Componente de Execução

Aplicações têm requisitos de segurança específicos que devem estar satisfeitos desde o momento em que iniciam a execução até o seu final. Para garantir que esses requisitos estejam satisfeitos as aplicações devem ser executadas através do Componente de Execução. Quando o Componente de Execução recebe a solicitação de executar uma aplicação, este verifica se já há outra instância dessa mesma aplicação em execução. Se não houver, informa ao Gestor de Contexto que o valor da Informação de Contexto relativa a aplicações em execução, *InfoAplicacao*, mudou de valor.

Ao informar o novo valor de *InfoAplicacao* é disparado um processo em que o Gestor de Contexto notifica o Gestor de Segurança Adaptativa da mudança no contexto. O Gestor de Segurança Adaptativa por sua vez fará uma série de procedimentos que, possivelmente, resultarão em adaptações. A conseqüência dessas adaptações é que os Requisitos de Segurança para a execução da aplicação ficarão atendidos. Ocorre que o procedimento de adaptação regido pelo Gestor de Segurança Adaptativa é feito de forma assíncrona. Assim, após informar ao Gestor de Contexto o novo valor de *InfoAplicacao*, será iniciado o processo de adaptação, mas a aplicação só deve de fato iniciar a execução após a conclusão das adaptações, sob pena de ferir os Requisitos de Segurança da aplicação. Para contornar esse problema, o Componente de Execução usa o método *atualizaContextoSinc* do Gestor de Contexto para informar o novo valor de *InfoAplicacao*. Esse método chama *atualizaContexto* e só retorna quando o processamento do Gestor de Segurança Adaptativa relativo à mudança no valor de *InfoAplicacao* for encerrado.

Uma vez satisfeitos os Requisitos de Segurança da aplicação, o Controle de Execução abre uma linha de execução separada (*thread*) que dispara a aplicação. Essa linha de execução aguarda o término da aplicação, quando então verifica se a instância da aplicação que terminou era a última em execução. Se for este o caso, retira do contexto o nome da aplicação, ou seja, informa ao Gestor de Contexto do novo valor de *InfoAplicacao*, agora sem conter a aplicação que terminou. A retirada do nome da aplicação pode provocar adaptações para desalocar recursos ou tornar mais restrito o ambiente de segurança (fechar portas não utilizadas, desalocar túneis SSH, etc). É algo comum que aplicações sejam executadas repetidamente. Se a cada início/final de execução for necessário efetuar adaptações, pode haver um custo grande em termos de energia, de processador e de outros recursos. Para atenuar esse problema, o nome da aplicação não é retirado imediatamente do contexto, mas o Controle de Execução aguarda por um determinado tempo. Passado esse tempo verifica se há alguma instância da aplicação em execução. Se houver, nenhuma modificação é feita no contexto e o processo encerra. Senão, o nome da aplicação é retirado do contexto, possivelmente provocando adaptações. O tempo que o Controle de Execução aguarda é definido pelo parâmetro "atraso.fimExec". O melhor valor para esse parâmetro será objeto de trabalhos futuros. A seqüência de eventos ocorridos no início e final de execução de uma aplicação está ilustrada na Figura 3.12.

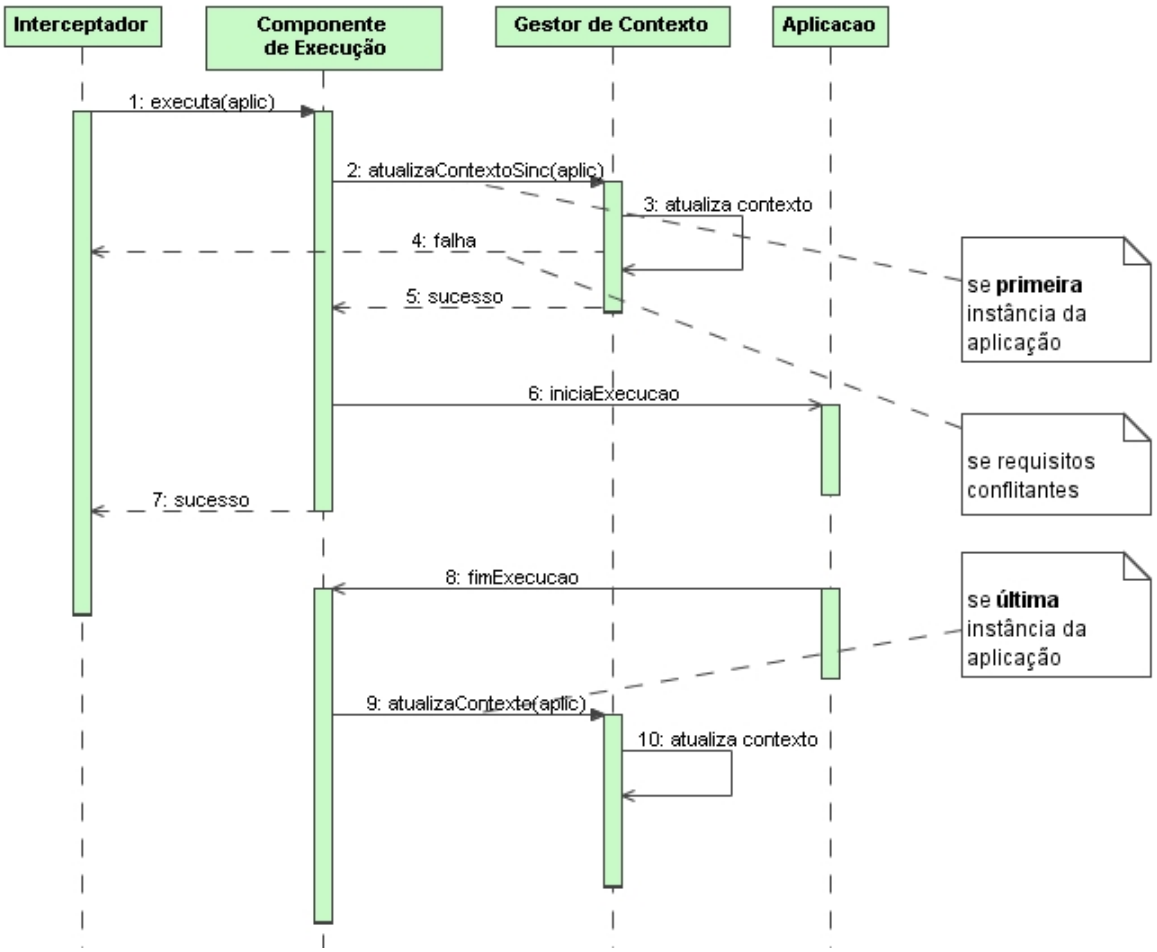


**Figura 3.12. Início e final de execução de uma aplicação**

Observe que a o método usado pelo Controle de Execução para modificar o contexto é *atualizaContextoSinc*, e não *atualizaContexto*. Este método chama *atualizaContexto* e, antes de retornar, aguarda que as mudanças solicitadas tenham sido processadas pelo GSA, e que, portanto, as adaptações decorrentes dessas modificações tenham sido efetuadas.

O Gestor de Regras, como já mencionado, é responsável por montar os requisitos de Segurança para o novo Contexto. Nesse ponto, é importante destacar como se constrói uma nova Aplicação Representativa (AR). Define-se uma Aplicação Representativa (AR), como sendo uma aplicação que representa, em termos de Requisitos de Segurança, todas as outras que estão em execução no dispositivo. Assim, uma AR é representada por um conjunto de requisitos de segurança representativos (RSRs). Um RSR é calculado de forma a atender todos os valores de um dado Requisito de Segurança das aplicações em uso. Ou seja, para conseguir atender simultaneamente a cada um dos Requisitos de Segurança das aplicações em uso, os valores de cada Requisito de Segurança das diferentes aplicações são combinados de forma a obter no final um valor do RSR em questão que satisfaça todas as aplicações.

Em suma, a construção de cada RSR deve refletir os anseios sob o prisma de segurança de todas as aplicações em uso para o requisito em questão. Para entender melhor como esse valor é obtido, examina-se o que ocorre com o requisito *InfoPortaTcpAberta* com valores diferentes para três aplicações. Esse requisito especifica quais portas TCP devem estar abertas no *Firewall* local do dispositivo para receber conexões. A Tabela 3.2 relaciona os valores requisitados para cada aplicação.

**Tabela 3.2. Valor do requisito *InfoPortaTcpAberta* para cada aplicação**

| Aplicação | Requisito *InfoPortaTcpAberta* |
|---|---|
| A | 80 |
| B | 21 |
| C | 80;443 |

Caso apenas uma das três aplicações esteja em execução, o valor para o RSR *InfoPortaTcpAberta* deve ser o indicado na tabela. Se mais de uma aplicação estiver em execução, A e B, por exemplo, o valor do RSR deve ser tal que represente o valor requisitado pela aplicação A e também o valor requisitado por B. Os requisitos das aplicações em execução são combinados pelo Gestor de Regras como parte do processo para obter o CRNA (Conjunto de Requisitos Não Atendidos). Durante esse procedimento, os valores de cada um dos Requisitos de Segurança das aplicações são combinados através da operação *uniao* da classe que representa a Informação de Contexto relativa ao requisito. Os possíveis valores do RSR *InfoPortaTcpAberta* para os valores requisitados no exemplo acima estão relacionados na Tabela 3.3.

**Tabela 3.3. Possíveis combinações de valores RSR *InfoPortaTcpAberta***

| Aplicações em execução | RSR *InfoPortaTcpAberta* |
|---|---|
| A+B | 21;80 |
| A+C | 80;443 |
| A+B+C | 21;80;443 |

Informações de Contexto têm características próprias. Essas características precisam ser observadas para combinar os valores. Cada Informação de Contexto é representada por uma implementação diferente de *InfoContexto*. A operação *uniao*, usada para combinar os valores dos Requisitos de Segurança, tem, dessa forma, uma semântica adequada à Informação de Contexto relativa a cada Requisito de Segurança.

Examinemos agora a Informação de Contexto *InfoCriptografia* que indica o nível de criptografia utilizada pelos túneis SSH providos pelo Gestor de Conexões. Possíveis valores

para InfoCriptografia são *Fraca*, *Media* e *Forte*. A Tabela 3.4 indica os valores requisitados pelas aplicações.

**Tabela 3.4. Valores do Requisito *InfoCriptografia* para cada Aplicação**

| Aplicação | Requisito *InfoCriptografia* |
|---|---|
| A | *Fraca* |
| B | *Media* |
| C | *Forte* |

Os possíveis valores do RSR *InfoCriptografia* para os valores requisitados no exemplo acima estão relacionados na Tabela 3.5.

**Tabela 3.5. Possíveis combinações de Valores RSR *InfoCriptografia***

| Aplicações em execução | RSR *InfoCriptografia* |
|---|---|
| A+B | *Media* |
| A+C | *Forte* |
| A+B+C | *Forte* |

Como regra geral o registro no Gestor de Contexto do valor das Informações de Contexto detectadas no ambiente é feito de forma passiva, ou seja, os Componentes de Segurança tendem a apenas detectar e registrar fielmente fatos já ocorridos. O Componente de Execução é uma exceção a esta regra, pois não apenas registra no contexto as aplicações em execução, mas participa ativamente dos procedimentos para executar essas aplicações. Isso dá a este Componente de Segurança a oportunidade de negar a modificação da Informação de Contexto. Essa oportunidade é usada para o tratamento de requisitos conflitantes.

Como parte do procedimento para iniciar a execução do aplicativo, o Gestor de Execução prepara o contexto chamando o método *atualizaContextoSinc* do Gestor de Contexto. Esse método, antes de notificar o Gestor de Segurança Adaptativa, verifica se o novo valor da Informação de Contexto ocasiona algum conflito. Se ocasionar, então o valor original da Informação de Contexto é restaurado e é disparada uma exceção, indicando que a mudança no contexto não pode ser efetuada, ou seja, a aplicação não é executada.

### 3.6.2 Descrição do Componente de Versão

O Componente de Versão provê ao Prometheus a capacidade de se atualizar dinamicamente. Podem ser atualizados os repositórios *Regras de Segurança*, *Descrição dos*

*Adaptadores*, e *Componentes*. Além desses repositórios, pode ser também atualizado o código dos Componentes de Segurança e dos Adaptadores, e, desse modo, a própria Política de Adaptação. A atualização desses componentes, como se verá, é feita como conseqüência de uma adaptação.

A atualização dos componentes e da Política de Adaptação é feita a partir de um servidor, denominado Servidor de Atualização. O Servidor de Atualização contém um banco de dados mantido pela equipe de segurança da organização. Na medida em que a equipe de segurança elabora soluções e contornos para vulnerabilidades, são feitos ajustes nas Regras de Segurança e nos Componentes de Segurança. O produto desses ajustes é registrado no banco de dados com um novo número de versão. Os dados dessa versão ficam disponíveis para o Prometheus no Servidor de Atualização. A comunicação entre o Prometheus e o Servidor de Atualização é feita através de uma conexão HTTPS, evitando dessa forma problemas com a autenticidade do servidor e a integridade dos dados.

O Controle de Versão periodicamente obtém do Servidor de Atualização o número da versão disponível. Se o valor obtido for diferente do valor da Informação de Contexto *VersaoDisponivel*, o Controle de Versão obtém a nova versão do servidor através de um procedimento de *download*. Os dados assim obtidos ficam armazenados numa área temporária. Após o *download,* o Controle de Versão chama o método *atualizaContexto* do Gestor de Contexto com esse valor informando o novo valor de *VersaoDisponivel*. Como conseqüência deve ocorrer uma adaptação. O adaptador invocado fará a atualização dos repositórios e dos componentes. Para que esse mecanismo funcione corretamente, há nas Regras de Segurança uma que trata do valor de *VersaoDisponivel*. Essa regra pode ser vista na Figura 3.13.

```
<regra nome="Versao">
  <condicao contexto="VersaoDisponivel" op="existe"/>
  <condicao contexto="VersaoEmUso"
                    op="naoIgual"
              valorDe="VersaoDisponivel"/>
  <acao    requisito="VersaoEmUso"
              valorDe="VersaoDisponivel"/>
</regra>
```

**Figura 3.13. Regra para teste de versão**

A regra tem duas condições e uma ação. A primeira condição verifica se a Informação de Contexto *VersaoDisponivel* existe no contexto. Este teste é necessário, pois é possível que, por algum motivo, o Controle de Versão ainda não tenha conseguido entrar em contato com o Servidor de Atualização para obter o valor de *VersaoDisponivel*. A segunda condição, que só

é executada caso a primeira seja verdadeira, compara o valor de duas Informações de Contexto entre si: *VersaoEmUso* e *VersaoDisponivel*. Se o valor de *VersaoDisponivel* for diferente do valor de *VersaoEmUso* então o novo valor do Requisito de Segurança *VersaoEmUso* será o da Informação de Contexto *VersaoDisponivel*.

A seqüência de eventos ocorridos no processo de atualização pode ser vista no diagrama seqüência mostrado na Figura 3.14.

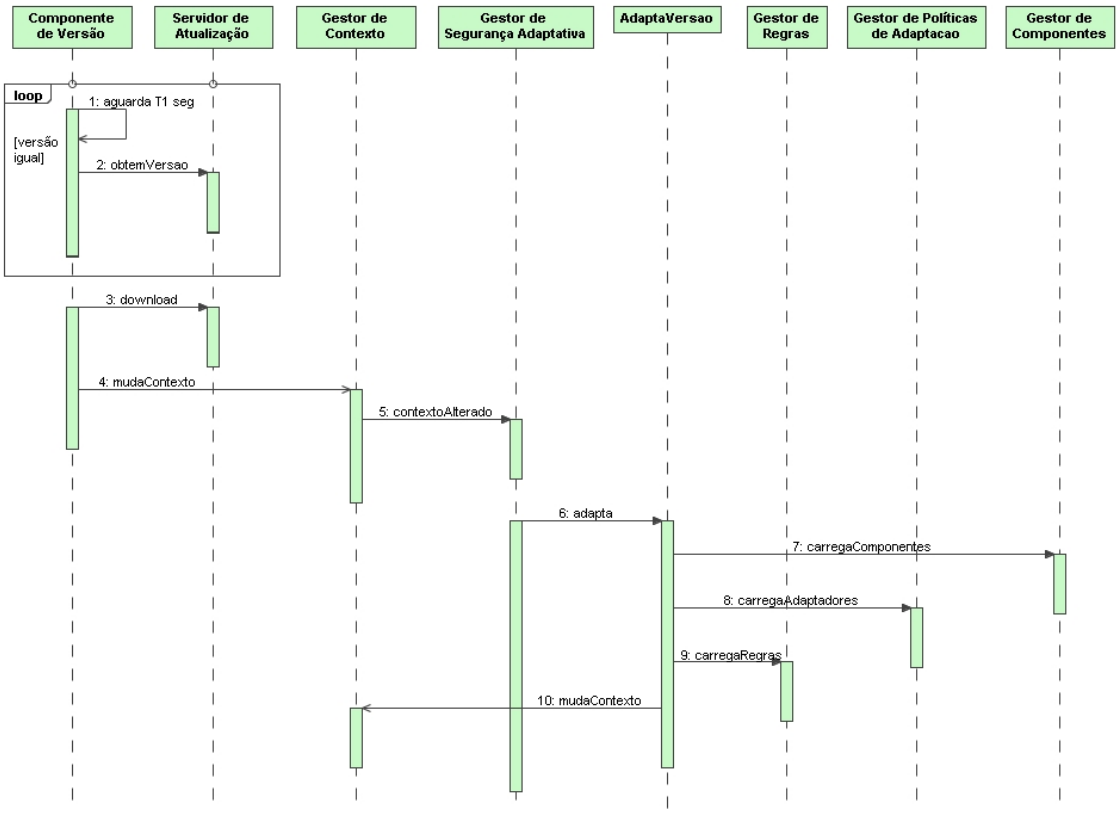


**Figura 3.14. Diagrama de seqüência para atualização de versão**

Em (1) a execução do Controle de Versão é suspensa por um determinado tempo. O tempo a aguardar é definido pelo parâmetro T1. Em (2) o Controle de Versão obtém do Servidor de Atualização o valor de versão disponível. Se o valor retornado pelo servidor for igual ao valor de *VersaoDisponivel* no contexto o controle retorna ao passo (1), caso contrário a execução continua em (3). Neste passo é feito o *download* dos dados da nova versão para um repositório temporário. Como os componentes do Prometheus têm, cada um, o número de versão, é possível efetuar uma atualização incremental, transferindo-se apenas o código dos componentes cuja versão mudou, e, dessa forma, economizando recursos. Em (4) o Controle de Versão informa o novo valor de *VersaoDisponivel* chamando o método *atualizaContexto* do Gestor de Contexto. A mudança dessa Informação de Contexto muda o valor do Requisito de Segurança *VersaoEmUso*, o que provoca uma adaptação (5). O adaptador utilizado para

tratar o novo valor de *VersaoEmUso* é *AdaptaVersao* (6). O procedimento do método *adapta* de *AdaptaVersao* consiste de (7) chamar o método *carregaComponentes* do Gestor de Componentes para que sejam carregados os novos componentes e descarregados aqueles que não são mais utilizados; (8) chamar o método *carregaAdaptadores* do Gestor de Política de Adaptação para informar que mudou o conteúdo do repositório *Descrição dos Adaptadores*; e (9) chamar o método *carregaRegras* do Gestor de Regras para informar que mudou o conteúdo do repositório *Regras de Segurança*. Finalmente, em (10) o adaptador informa ao Gestor de Contexto o novo valor de *VersaoEmUso*.

### 3.6.3 Descrição do Componente de Conexões

O Componente de Conexões provê mecanismos para garantir a integridade e a confidencialidade das informações trafegadas entre o dispositivo e os servidores com os quais a aplicação se comunica.

O mecanismo utilizado na implementação de referência do Prometheus intercepta as conexões TCP entre o dispositivo e os servidores, e as redireciona para que os dados trafeguem através de túneis SSH [RFC4251]. Duas Informações de Contexto são tratadas pelo Componente de Conexões: *InfoTunelSsh* e *InfoCriptografia*. A primeira especifica os parâmetros dos túneis, tais como identificação do servidor; porta local; endereço do servidor remoto; e porta remota. A segunda Informação de Contexto, *InfoCriptografia*, é usada pelo Componente de Conexões para selecionar os algoritmos de criptografia utilizados para encriptar as informações trafegadas pelos túneis. O algoritmo a utilizar é negociado entre o cliente SSH e o servidor. Durante a negociação o cliente propõe uma lista de algoritmos em ordem de preferência. O servidor aceita o primeiro algoritmo suportado. Como regra geral, algoritmos de criptografia mais complexos provêem uma maior confidencialidade, porém, com um consumo maior de recursos. A Tabela 3.6 mostra os algoritmos propostos pelo Componente de Conexões na negociação com o servidor. A montagem da tabela, entretanto, foi arbitrária. A configuração ideal da tabela será objeto de estudos futuros.

**Tabela 3.6. Algoritmos propostos ao servidor x *InfoCriptografia***

| *InfoCriptografia* | 1ª Opção | 2ª Opção | 3ª Opção | 4ª Opção |
|---|---|---|---|---|
| NENHUMA | none | | | |
| FRACA | 3des-cbc | 3des-ctr | | |
| MEDIA | blowfish-cbc | blowfish-ctr | aes128-cbc | aes128-ctr |
| FORTE | aes256-cbc | aes256-ctr | aes192-cbc | aes192-ctr |

## 3.7 Considerações Finais

Esse capítulo apresentou a arquitetura e a descrição detalhada do funcionamento dos principais componentes do Prometheus, bem como a descrição de alguns Componentes de Segurança. No próximo capítulo será apresentado um estudo de caso de forma a ilustrar o uso do Prometheus numa situação real.

# 4 Estudo de Caso: Aplicações do Grupo Capivara

Nesse capitulo é descrito o estudo de caso do Grupo Capivara com o intuito de ilustrar a importância do Prometheus para prover um ambiente seguro para a execução de aplicações ubíquas e validar a presente proposta.

O presente estudo de caso foi idealizado com o intuito de ilustrar a utilização do Prometheus frente a mudanças no ambiente de execução que acarretam um aumento do risco de incidentes de segurança para as aplicações do Grupo Capivara. Nesse capítulo são descritas situações em que ocorrem acréscimos e decréscimos de Controles de Segurança conforme o contexto em que as aplicações estão sendo executadas de forma a garantir o cumprimento da Política de Segurança da empresa. Assim, fazendo uso do Prometheus, todo o alicerce de sustentação dos Requisitos de Segurança das aplicações do Grupo Capivara será adaptado segundo o ambiente de execução, levando-se em conta propriedades dinâmicas e heterogêneas que reflitam os diferentes ambientes computacionais envolvidos.

Este capítulo está estruturado em quatro Seções. As Seções 4.1 e 4.2 apresentam, respectivamente, a justificativa do uso do Prometheus para garantir um ambiente seguro para as Aplicações do Grupo Capivara e a descrição do estudo de caso do Grupo Capivara propriamente dito. A Seção 4.3 descreve o uso do Prometheus frente a mudanças no ambiente de execução quando as aplicações do Grupo Capivara são executadas em um dispositivo móvel em movimento. Por fim, a Seção 4.4 apresenta as considerações finais deste capítulo.

## 4.1 Justificativa do uso do Prometheus

Os telefones celulares evoluíram de simples aparelhos de áudio para os atuais *smartphones*, dispositivos portáteis com capacidade de processamento e armazenamento significativa, capazes de suportar conectividade pela Internet e dar acesso remoto a aplicativos cruciais para a linha de negócios. As vantagens de oferecer acesso móvel a e-mail e a aplicativos corporativos não se obtêm sem implicações de risco ou de custo. Do ponto de vista do dispositivo, deve haver suporte para o acesso do usuário a fontes de dados, opções de customização pessoal para facilitar o uso, e funções administrativas para coordenar políticas corporativas do uso dos dispositivos. Enquanto as capacidades de administração devem proporcionar recursos de TI para suportar e controlar dispositivos móveis, as capacidades de segurança do ambiente móvel devem proteger as informações corporativas confidenciais que serão armazenadas nos dispositivos móveis ou transmitidas por eles, justificando a necessidade de serviços como o provido pelo Prometheus.

Com o intuito de ilustrar a importância do Prometheus, nesse trabalho é descrito o uso de uma aplicação corporativa que exige segurança, cujo acesso à base central localizada no servidor corporativo pode ser efetuado por dispositivos móveis. Através do dispositivo móvel, os funcionários da empresa (vendedores) podem consultar a lista de determinados tipos de produtos, sua quantidade em estoque e preços, bem como registrar a venda de produtos, atualizando informações na base central. No tocante à venda, são transmitidas para o servidor, entre outras, informações relacionadas à cobrança e o endereço de entrega. Informações estratégicas do negócio como preço de custo dos produtos e quantidade em estoque, bem como informações a respeito do cliente (endereço e dados de cobrança) devem ser transferidas até o servidor de forma segura.

O uso da rede corporativa para transferir informações é razoavelmente seguro, uma vez que a utilização de criptografia no nível de enlace é obrigatória para a rede sem fio e, na rede cabeada, o acesso ao meio físico até os servidores é controlado. O dispositivo, entretanto, pode precisar transferir informações através de uma rede pública ou através da rede do cliente. Nesse caso, o uso de criptografia é essencial para garantir a confidencialidade e a integridade da informação. O consumo de energia é um ponto crítico para dispositivos portáteis, e, portanto, a criptografia deve ser usada apenas quando necessário, uma vez que impõe um maior processamento e a transmissão de mensagens e de controles adicionais. Esse uso adicional de processador e o acréscimo no tamanho e no número de mensagens representam um consumo adicional de energia. Um dos diferenciais desse trabalho é justamente prover o nível de proteção à informação adequado ao contexto de execução, permitindo, por exemplo, utilizar criptografia apenas quando o contexto de execução não for seguro. Ainda sobre a criptografia, o trabalho de [Doomun 2007] mostra que a utilização de diferentes algoritmos com diferentes tamanhos de chave e número de rodadas resulta em diferentes consumos de energia (Tabela 4.1). O mesmo trabalho mostra que 10% do consumo de energia é destinado ao processamento dos algoritmos de criptografia e 90% com a transmissão. Pode-se observar na Figura 4.1 que o consumo de energia varia de acordo com o algoritmo e tamanho da chave utilizada.

**Tabela 4.1. Consumo de Energia AES**

| Chave (bits) | Rodadas | Energia (mJ) | |
|---|---|---|---|
| | | Encriptação | Decriptação |
| 128 | 8 | 0.27 | 0.30 |
| 128 | 10 | 0.31 | 0.36 |
| 192 | 10 | 0.35 | 0.55 |
| 192 | 12 | 0.37 | 0.62 |
| 256 | 12 | 0.39 | 0.82 |
| 256 | 14 | 0.42 | 1.01 |

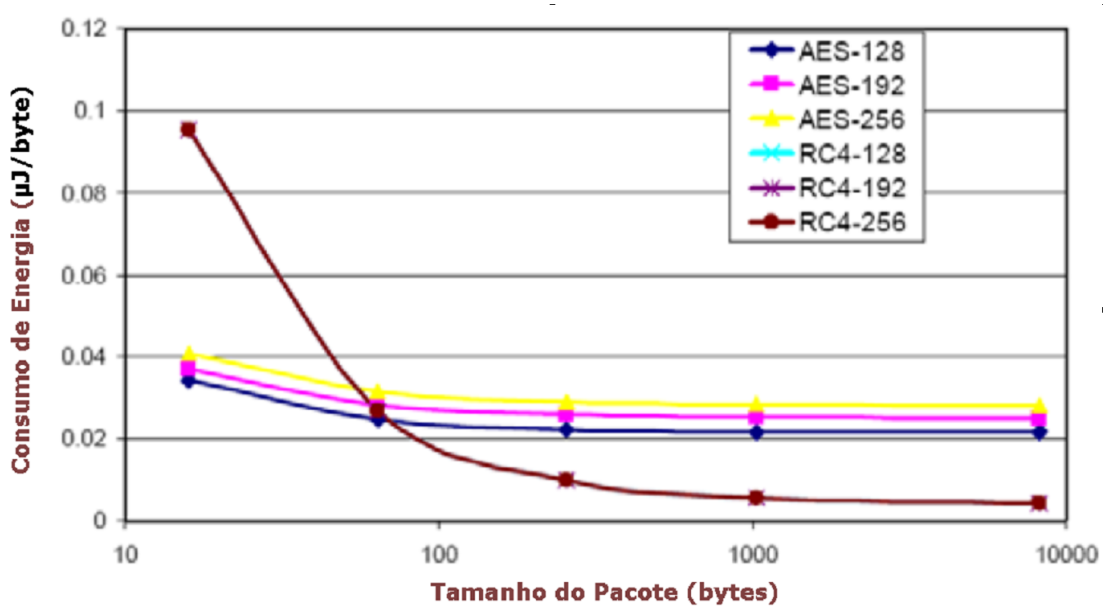


**Figura 4.1. Consumo de Energia x Tamanho do Pacote**

Um dos principais requisitos de uma aplicação corporativa é proteger as informações em trânsito contra a modificação/divulgação por uma entidade não autorizada, ou seja, garantir a integridade e a confidencialidade da informação. Para garantir tais requisitos, Controles de Segurança podem ser acionados dependendo do nível de segurança oferecido pelo ambiente onde o dispositivo está localizado (dependências da empresa ou num local público) ou do nível de segurança oferecido pela rede (segura ou insegura).

Quanto ao tipo de ambiente, o Prometheus pode perceber se o ambiente é seguro (ou hostil) se detectar que a rede sem fio em uso é a rede corporativa (ou outra rede qualquer) ou se as coordenadas geográficas do dispositivo apontarem para um local considerado seguro (ou para outro local). Caso a rede em uso não seja a da própria empresa ou se o local do dispositivo não for considerado um local seguro, então serão executados Controles de Segurança para garantir a segurança da informação, possivelmente acionando configurações para utilizar algoritmos de criptografia mais seguros (e caros em termos de consumo de

recursos). Da mesma forma, a identificação de uma rede segura ou de um local seguro permite o relaxamento da segurança de modo a consumir menos recursos do dispositivo.

Um outro requisito básico da segurança diz respeito à garantia de que determinado recurso ou o próprio sistema (aplicação corporativa no presente trabalho), esteja sempre "disponível" para as entidades autorizadas, ou seja, ***garantir a disponibilidade*** da aplicação corporativa. Para garantir tal requisito, diversos controles podem ser acionados dependendo do tipo da Informação de Contexto que indicará, entre outros, a existência de ataques, ou limitação de recursos do dispositivo (memória, energia). É importante também que novos controles e políticas de segurança possam ser inseridos no servidor corporativo de forma que o dispositivo possa carregá-los dinamicamente e que esteja sempre apto a detectar novos ataques, garantindo assim a disponibilidade da aplicação.

## 4.2 Descrição do Estudo de Caso

Nessa seção é apresentado o estudo de caso em que aplicações do Grupo Capivara, executando em dispositivos móveis, acessam uma base de dados num servidor corporativo.

O Grupo Capivara equipou seus melhores vendedores com *smartphones* para agilizar os processos internos da empresa. Com esses dispositivos os vendedores podem ter acesso aos bancos de dados da empresa de qualquer local. Para garantir a segurança o Prometheus foi instalado nesses *smartphones*.

O Grupo definiu os requisitos de segurança para cada um dos seus aplicativos. Os aplicativos referenciados nesse estudo de caso são: CLIENTES, VENDAS e BATE-PAPO. O aplicativo CLIENTES fornece, entre outros, as visitas que um vendedor deve realizar no dia. Já o aplicativo VENDAS permite que um vendedor registre suas vendas, dentre outras funcionalidades. Um outro aplicativo usado é o BATE-PAPO para troca de mensagens instantâneas.

Na Tabela 4.2 são relacionados os requisitos de segurança dos aplicativos CLIENTES, VENDAS E BATE-PAPO além dos Requisitos de Segurança mínimos, que devem ser atendidos mesmo que não haja nenhuma aplicação em execução (aplicação *DEFAULT*).

Os Requisitos de Segurança da aplicação *DEFAULT* são:

- não há permissão para receber Conexões TCP por nenhuma porta (*InfoPortaTcpAberta*);
- não há permissão para enviar/receber pacotes UDP para/de nenhum endereço/porta (*InfoSaidaUdp*);

- não há permissão para fazer conexões TCP a qualquer endereço/porta (*InfoSaidaTcp*); e
- necessidade de usar criptografia *Fraca.*

Os requisitos de segurança da aplicação ***CLIENTES*** são:

- necessidade de efetuar conexões TCP a "srv.capivara.com" na porta 22 (*InfoSaidaTcp*); e
- necessidade de usar um túnel SSH até "srv.capivara.com" ligando conexões locais à porta 3306 ao servidor de banco de dados em "db1.intra" na porta 3306 (*InfoTunelSsh*).

Os requisitos de segurança da aplicação ***VENDAS*** (AP2) são:

- necessidade de efetuar conexões TCP a "srv.capivara.com" na porta 22 (*InfoSaidaTcp*);
- necessidade de usar um túnel SSH até "srv.capivara.com" ligando conexões locais à porta 3307 ao servidor de banco de dados em "db2.intra" na porta 3306 (*InfoTunelSsh*);
- proibição de executar simultaneamente com o aplicativo BATE-PAPO.

Por fim, o requisito de segurança da aplicação ***BATE-PAPO*** (AP3) é:

- receber/enviar pacotes UDP no endereço bp.capivara.com pela porta 4321 do serviço de BATE-PAPO (*InfoSaidaUdp*).

**Tabela 4.2. Requisitos de Segurança das Aplicações do Grupo Capivara**

| Requisito de Segurança | | CLIENTES | VENDAS | BATE-PAPO |
|---|---|---|---|---|
| *InfoAplicacao* | R[1] | | | |
| | **P[2]** | | ***BATE-PAPO*** | |
| *InfoSaidaTcp* | R | srv.capivara.com:22 | srv.capivara.com:22 | |
| | **P** | | | |
| *InfoSaidaUdp* | R | | | bp.capivara.com:4321 |
| | **P** | | | |
| *InfoTunelSsh* | R | srv:3306:db1.intra:3306 | srv:3307:db2.intra:3306 | |
| | **P** | | | |

[1] R = Requisito de Segurança
[2] P = Proibição

**Tabela 4.3. Outros Requisitos de Segurança**

| Requisito de Segurança | Requisitos Mínimos (aplicação *DEFAULT*) | Se rede em uso não é WI-FI Capivara |
|---|---|---|
| *InfoCriptografia* | *Fraca* | *Forte* |

No que diz respeito a esse estudo de caso, os dispositivos podem utilizar: (i) a rede WI-FI da empresa (ambiente corporativo); (ii) a rede WI-FI do Cliente; ou (iii) a rede pública de dados da operadora de telefonia. A conexão aos servidores da empresa pela rede WI-FI corporativa é considerado "segura". Entretanto, adota-se no ambiente corporativo por *default* criptografia do tipo *Fraca* para evitar a exposição, ainda que acidental, da informação em claro (não encriptada), por exemplo, a técnicos que estejam fazendo manutenção na rede. Conexões ao servidor através de outras redes são consideradas “inseguras”, isto é, considera-se que a rede é pública requerendo a adoção de criptografia do tipo *Forte* para as conexões usadas para a troca de dados entre o dispositivo e a base de dados do servidor corporativo.

No início e no término de execução de uma aplicação e a cada mudança no ambiente de execução das aplicações tais como energia residual, memória disponível, tipo de criptografia usada na interface de rede e ataques, o Prometheus deve realizar adaptações de forma a garantir a segurança. Assim, os ***comportamentos adaptativos*** considerados neste estudo de caso são baseados nas características contextuais descritas em seguida.

- **Inicio de Execução das Aplicações (CLIENTE, VENDAS, BATE-PAPO)**

O Prometheus, através de Componente de Execução, verifica se já há outra instância da aplicação em execução. Se não houver, verifica se a execução da aplicação causa algum conflito. Em outras palavras, é verificado se há alguma Informação de Contexto cujo valor proíba a execução da aplicação, ou se a modificação no contexto causada pela execução da aplicação torna proibida a execução de alguma outra aplicação já em execução. Em caso de conflito, o Prometheus não executa a aplicação e retorna um erro. Se não houver conflito, o Componente de Execução modifica o contexto para que *Infoaplicacao* contenha o nome da aplicação. Essa modificação no contexto realizada pelo Prometheus é feita de maneira síncrona, ou seja, só retorna após a concluir todo o processamento relacionado a essa modificação, que pode incluir a execução de várias adaptações. Uma vez que o contexto tenha sido modificado, o Prometheus (Componente de Execução) dispara a aplicação. Em suma, o Prometheus efetua adaptações necessárias em tempo de execução de forma que os Requisitos de Segurança da aplicação em questão sejam satisfeitos, garantindo assim que o aplicativo possa ser executado respeitando a Política de Segurança.

- **Término de Execução de Aplicação (CLIENTE, VENDAS, BATE-PAPO)**

O Controle de Execução, ao detectar que uma aplicação encerrou a execução, verifica se há mais alguma instância da aplicação em execução. Se não houver mais nenhuma instância em execução, o Controle de Execução informa ao Gestor de Contexto o novo valor de *InfoAplicacao*, agora sem conter o nome da aplicação que encerrou. Essa modificação no contexto pode provocar adaptações para desalocar recursos ou tornar mais restrito o ambiente de segurança (fechar portas não utilizadas, desalocar túneis TCP, etc).

## 4.3 Execução das Aplicações do Grupo Capivara

O vendedor sênior João foi um dos contemplados com um *smartphone*. Num dia típico de um vendedor, João, ao chegar à empresa, aciona o aplicativo CLIENTES para verificar quais são as visitas do dia. Usando o aplicativo, João consulta a base de dados de clientes através da conexão à rede WI-FI da empresa, e verifica que precisa visitar a empresa Glostora, seu melhor cliente. João sai do prédio com seu *smartphone* e pega um táxi. Ao se afastar do prédio o *smartphone* perde a conexão com a rede WI-FI e se conecta à Internet através de uma rede pública. Ainda no táxi, João aproveita o tempo gasto no trânsito para, usando o aplicativo CLIENTES, verificar outras visitas a fazer no dia. Ao chegar à Glostora, o dispositivo se conecta à rede WI-FI do cliente para realizar a venda de um lote importante de mercadorias. Para registrar a venda, ele aciona em seu *smartphone* o aplicativo VENDAS. Após efetuar o

registro da venda, João aciona o aplicativo BATE-PAPO, mas recebe uma mensagem de erro informando que deve encerrar o aplicativo VENDAS se deseja usar o BATE-PAPO. João encerra o aplicativo VENDAS, e tenta novamente executar o BATE-PAPO, desta vez com sucesso.

O Prometheus é iniciado no tempo T0, tão logo o *smartphone* é ligado. Nesse momento, os Requisitos de Segurança relativos à aplicação *DEFAULT* são inicializados. Em conseqüência, são disparados os Controles de Segurança necessários para atender a esses requisitos. Em seguida, no tempo T1, o dispositivo se conecta à rede WI-FI da Capivara, na sede da empresa. Nenhuma adaptação é necessária.

Antes de ser iniciada a execução do aplicativo CLIENTES acionado por João, o Prometheus:

i. obteve os requisitos de segurança dessa aplicação;
ii. verificou se havia algum conflito entre os requisitos da aplicação CLIENTE e os requisitos das aplicações já em execução;
iii. verificou quais requisitos da aplicação não estavam atendidos pelo contexto;
iv. selecionou os adaptadores;
v. efetuou as adaptações necessárias no contexto; e
vi. executou a aplicação.

Os requisitos da aplicação CLIENTES, conforme já mencionado, são: necessidade de conexões TCP ao endereço "srv.capivara.com" na porta 22 (*InfoSaidaTcp*); e uso de um túnel SSH para transportar informações entre a porta local 3306 e a porta 3306 do servidor "db1.intra" na intranet da empresa. Esses requisitos são obtidos do repositório de *Regras de Segurança* do Prometheus. O trecho de do repositório relativo a essas regras pode ser visto na Figura 4.2.

```
<regra nome="DEFAULT">
  <acao requisito="InfoCriptografia">Fraca</acao>
</regra>

<regra nome="CLIENTES">
  <condicao contexto="InfoAplicacao" op="contem">CLIENTES</condicao>
  <acao requisito="InfoSaidaTcp">srv.capivara.com:22</acao>
  <acao requisito="InfoTunelSsh">srv:3306:db1.intra:3306</acao>
</regra>
```

**Figura 4.2. Requisitos das aplicações DEFAULT e CLIENTES**

A regra "DEFAULT", incondicional, é usada para indicar os Requisitos de Segurança mínimos, a satisfazer independente do contexto. A regra "CLIENTES" indica os Requisitos

de Segurança que devem estar satisfeitos no contexto quando a aplicação CLIENTES está sendo executada (valor de *InfoAplicacao* contém "CLIENTES"). Para iniciar a execução do aplicativo CLIENTES no tempo T2, o Prometheus atualizou o contexto com o nome aplicativo; executou as Regras de Segurança para obter o Requisito Representativo para o contexto corrente; e comparou o valor de cada *InfoContexto* no Requisito Representativo com o contexto. Verificou-se que os requisitos de segurança *InfoSaidaTcp* e *InfoTunelSsh* não estavam atendidos. Assim, o Prometheus selecionou e executou os adaptadores adequados para que esses requisitos de segurança fossem atendidos. A seleção dos adaptadores foi feita pelo mecanismo de seleção de adaptadores com base no repositório *Descrição dos Adaptadores* do Prometheus. O trecho desse repositório relacionado a essa seleção pode ser visto na Figura 4.3.

```
<adaptador
        classe="prometheus.adaptador.AdaptaFirewall"
        duracao="14" tamanho="1287" energia="2000" versao="1.01">
  <infoContexto>InfoSaidaTcp</infoContexto>
  <infoContexto>InfoSaidaUdp</infoContexto>
  <infoContexto>InfoPortaTcpAberta</infoContexto>
</adaptador>

<adaptador
        classe="prometheus.adaptador.AdaptaTunelSsh"
        duracao="740" tamanho="1514" energia="78000" versao="1.03">
  <infoContexto>InfoTunelSsh</infoContexto>
</adaptador>

<adaptador
        classe="prometheus.adaptador.AdaptaCriptografia"
        duracao="1" tamanho="1112" energia="12000" versao="1.02">
  <infoContexto>InfoCriptografia</infoContexto>
</adaptador>
```

**Figura 4.3. Adaptadores para *InfoSaidaTcp*, *InfoTunelSsh* e *InfoCriptografia***

Nesse trabalho o valor de "energia", indicando o consumo estimado de energia para efetuar uma adaptação, não foi considerado. Os dados do repositório indicam que o adaptador "AdaptaFirewall" pode ser usado para efetuar modificações no valor da Informação de Contexto *InfoSaidaTcp*. O código desse adaptador comanda a execução de Controles de Segurança pelo Componente de Firewall para que modifique as regras do *firewall* local do dispositivo.

Da mesma forma, os dados do repositório indicam que o adaptador "AdaptaTunelSsh" pode ser usado para efetuar modificações no valor de *InfoTunelSsh*. O código desse adaptador comanda a execução de Controles de Segurança pelo Componente de Conexões,

estabelecendo uma sessão com o servidor SSH e abrindo ou fechando túneis sobre essa sessão de modo a atender aos valores requisitados.

A caminho da empresa Glostora, o dispositivo móvel passou de um ambiente corporativo para um ambiente público no tempo T3. A troca de ambiente foi efetuada de forma transparente para o aplicativo CLIENTES. Ao detectar a mudança de um ambiente corporativo para um ambiente público, o Prometheus registrou a mudança no contexto (rede corporativa x rede pública) e selecionou um adaptador (AdaptaCriptografia). O adaptador selecionado mudou o nível de criptografia usado pelo sistema de túneis SSH de *Fraca* para *Forte*, garantindo o nível necessário de proteção à integridade e a confidencialidade das informações usadas pelo aplicativo CLIENTES nessa rede pública. A Figura 4.4 mostra as regras envolvidas na mudança do nível de criptografia utilizado.

```
<regra nome="DEFAULT">
  <acao requisito="InfoCriptografia">Fraca</acao>
</regra>

<regra nome="Rede Publica">
  <condicao contexto="InfoRedeEmUso"
                  op="naoContem">wifi:capivara</condicao>
  <acao requisito="InfoCriptografia">Forte</acao>
</regra>
```

**Figura 4.4. Regras para a criptografia em redes públicas**

Ao se afastar da sede da empresa o dispositivo se desconectou da rede WI-FI da empresa e passou a usar a rede 3G provida pelo operador de telefonia. Ao detectar essa mudança o Componente de Redes do Prometheus mudou o valor da Informação de Contexto *InfoRedeEmUso* para "3g:publica". As Regras de Segurança são processadas na ordem em que aparecem no repositório. A regra "Inicializacao" atribui, incondicionalmente, o valor "*Fraca*" ao requisito *InfoCriptografia*. A regra "Rede Publica", processada em seguida, verifica se a informação de contexto *InfoRedeEmUso* não contém "wifi:capivara". Como a condição é verdadeira, o valor para o Requisito de Segurança *InfoCriptografia* é atribuído novamente, dessa vez com "*Forte*".

Como o valor de *InfoCriptografia* no contexto ("*Fraca*") é diferente do valor do Requisito de Segurança ("*Forte*"), é selecionado um adaptador. A seleção do adaptador usa a informação do repositório *Descrição dos Adaptadores*. O trecho desse repositório mostrado na Figura 4.4 mostra que o adaptador "AdaptaCriptografia" é adequado para efetuar ajustes em *InfoCriptografia*. Ao ser executado esse adaptador faz com que o Componente de Conexões renegocie os algoritmos de criptografia para todas as sessões estabelecidas. O

algoritmo será selecionado pelo adaptador dentre os disponíveis para atender o nível de proteção desejado.

Ao chegar à Glostora, no tempo T4 o *smart phone* se conecta à rede WI-FI local, provocando a mudança no valor da Informação de Contexto *InfoRedeEmUso* para "wifi:glostora". Essa mudança, entretanto, não causa nenhuma adaptação, já que os Requisitos de Segurança continuam todos atendidos.

Em seguida, João executa o aplicativo VENDAS em seu *smart phone* no tempo T5. Os requisitos de segurança do aplicativo VENDAS, conforme mencionado são: necessidade de efetuar conexões TCP a "srv.capivara.com" na porta 22 (*InfoSaidaTcp*); necessidade do usar um túnel SSH até "srv.capivara.com" ligando conexões locais à porta 3307 com o servidor de banco de dados em "db2.intra" na porta 3306 (*InfoTunelSsh*); e a proibição de executar simultaneamente com o aplicativo BATE-PAPO (*InfoAplicacao*). Esses requisitos são obtidos do repositório de *Regras de Segurança*. O trecho de do repositório relativo a essas regras pode ser visto na Figura 4.5.

```
<regra nome="VENDAS">
  <condicao contexto="InfoAplicacao" op="contem">VENDAS</condicao>
  <acao requisito="InfoSaidaTcp">srv.capivara.com:22</acao>
  <acao requisito="InfoTunelSsh">srv:3307:db2.intra:3306</acao>
</regra>

<conflito>
  <condicao contexto="InfoAplicacao" op="contem">VENDAS</condicao>
  <condicao contexto="InfoAplicacao" op="contem">BATE-PAPO</condicao>
</conflito>
```

**Figura 4.5. Requisitos de Segurança para aplicação VENDAS**

Dentre os Requisitos de Segurança há a proibição da execução simultânea com o aplicativo "BATE-PAPO". Como o aplicativo BATE-PAPO não está em execução, a condição é falsa, e, portanto, não há proibição de executar o aplicativo.

Para satisfazer aos requisitos da aplicação "VENDAS", apenas o adaptador "AdaptaTunelSsh", para atender ao requisito *InfoTunelSsh,* foi selecionado, uma vez que o requisito *InfoSaidaTcp* já estava atendido. O adaptador "AdaptaTunelSsh" verifica que já existe uma sessão com o servidor indicado (criada ao executar o aplicativo CLIENTES), e apenas cria mais um túnel sobre a sessão existente.

Uma vez atendidos os Requisitos de Segurança o Controle de Execução dispara a aplicação.

Após efetuar o registro da venda, João tenta acionar o aplicativo BATE-PAPO no tempo T6. O requisito de segurança do BATE-PAPO, conforme já mencionado, é a permissão

para enviar e receber pacotes UDP no endereço bp.capivara.com na porta 4321 (*InfoSaidaUdp*). Ao examinar os requisitos da aplicação BATE-PAPO, o Prometheus detectou um conflito, pois a regra "VENDAS" proíbe a execução simultânea do aplicativo BATE-PAPO. O Prometheus então nega a execução do BATE-PAPO, e João recebe uma mensagem de erro. Nenhuma adaptação é efetuada.

Em seguida, no tempo T7, João encerra o aplicativo VENDAS. O final de execução é percebido pelo Controle de Execução, que verifica não haver mais nenhuma instância do aplicativo VENDAS em execução. O valor da Informação de Contexto *InfoAplicacao* é modificado de forma a não conter mais "VENDAS". Essa modificação provoca uma adaptação, visto que o valor de *InfoTunelSsh* mudou e o túnel ligando a porta local 3307 à porta 3306 no endereço db2.intra não é mais necessário. É então selecionado o adaptador *AdaptaTunelSsh* que fecha o túnel não utilizado.

No tempo T8, após o aplicativo VENDAS ser encerrado, João executa novamente o BATE-PAPO. Desta vez não há o conflito, já que a aplicação VENDAS não está mais em execução. É selecionado o adaptador "AdaptaFirewall", que, ao ser executado, modifica as permissões do Firewall para atender aos Requisitos de Segurança da aplicação. O trecho do repositório *Regras de Segurança* a partir do qual foram obtidos os Requisitos de Segurança da aplicação BATE-PAPO é mostrado na Figura 4.6.

```
<regra nome="BATE-PAPO">
  <condicao contexto="InfoAplicacao"
                    op="contem">BATE-PAPO</condicao>
  <acao requisito="InfoSaidaUdp"> bp.capivara.com:4321</acao>
</regra>
```

**Figura 4.6. Requisitos de Segurança da aplicação BATE-PAPO**

A Tabela 4.4 ilustra a evolução do valor das Informações de Contexto no tempo. A cada evento a coluna "Δ Contexto" indica a Informação de Contexto que foi modificada em conseqüência do evento. A coluna CRNA mostra os Requisitos de Segurança cujos valores ficaram diferentes do valor encontrado nas Informações de Contexto correspondente. As colunas agrupadas sob "Contexto" mostram em cada linha o valor das Informações de Contexto após o processamento das adaptações executadas em função do evento para satisfazer aos Requisitos de Segurança.

**Tabela 4.4. Evolução do valor das Informações de Contexto**

| | | | | Contexto | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Evento** | **Δ Contexto** | **CRNA** | *InfoAplicacao* | *InfoSaidaTcp* | *InfoSaidaUdp* | *InfoTunelSsh* | *InfoCriptografia* | *InfoRedeEmUso* |
| T0 | Prometheus Inicializado | *InfoCriptografia* | | | | | | *Fraca* | |
| T1 | Conexão à rede | *InfoRedeEmUso* | | | | | | *Fraca* | wifi:capivara |
| T2 | inicia CLIENTES | *InfoAplicacao* | *InfoSaidaTcp* *InfoTunelSsh* | CLIENTES | **srv.capivara.com:22** | | **srv:3306:db1.intra:3306** | *Fraca* | wifi:capivara |
| T3 | na rua | *InfoRedeEmUso* | *InfoCriptografia* | CLIENTES | srv.capivara.com:22 | | srv:3306:db1.intra:3306 | ***Forte*** | 3g:publica |
| T4 | chega à glostora | *InfoRedeEmUso* | | CLIENTES | srv.capivara.com:22 | | srv:3306:db1.intra:3306 | *Forte* | wifi:glostora |
| T5 | inicia VENDAS | *InfoAplicacao* | *InfoTunelSsh* | CLIENTES;VENDAS | srv.capivara.com:22 | | **srv:3306:db1.intra:3306; srv:3307:db2.intra.3306** | *Forte* | wifi:glostora |
| T6 | tenta executar BATE-PAPO | *InfoAplicacao* | **** conflito **** | CLIENTES;VENDAS | srv.capivara.com:22 | | srv:3306:db1.intra:3306; srv:3307:db2.intra.3306 | *Forte* | wifi:glostora |
| T7 | encerra VENDAS | *InfoAplicacao* | *InfoTunelSsh* | CLIENTES | srv.capivara.com:22 | | **srv:3306:db1.intra:3306** | *Forte* | wifi:glostora |
| T8 | inicia BATE-PAPO | *InfoAplicacao* | *InfoSaidaUdp* | CLIENTES;BATE-PAPO | srv.capivara.com:22 | **bp.capivara.com:4321** | srv:3306:db1.intra:3306 | *Forte* | wifi:glostora |

## 4.4 Considerações Finais

Na apresentação do caso de uso é mostrado como Prometheus gerencia os requisitos de disponibilidade, integridade e confidencialidade das aplicações do Grupo Capivara em presença de um ambiente altamente dinâmico e heterogêneo que reflete os diferentes ambientes computacionais envolvidos que faz com que controles de segurança sejam diferentes conforme o contexto corrente. Conforme já mencionado, as propriedades dinâmicas e heterogêneas dos ambientes relacionadas ao contexto de segurança se referem aos requisitos da aplicação, da rede e dos dispositivos. Dessa forma, todo o alicerce de sustentação dos requisitos de segurança das aplicações deve ser adaptado segundo o ambiente de execução.

No próximo Capítulo é apresentado o cenário de implementação, às métricas usadas na avaliação do Prometheus como a análise dos resultados obtidos com através dos experimentos realizados.

# 5 Avaliação do Prometheus

O Prometheus enfoca questões relativas à automação das adaptações necessárias à sustentação dos requisitos de segurança das aplicações ubíquas. O objetivo dessa avaliação é verificar se a segurança é provida adequadamente pelo Prometheus, e se o uso de adaptações dinâmicas para prover o nível adequado de segurança às aplicações trouxe algum benefício em relação à solução clássica, de configuração estática.

Um objetivo específico é avaliar o desempenho do Prometheus através do tempo utilizado para selecionar adaptadores, do tempo gasto para executar uma única adaptação, e do tempo de reação de uma adaptação. O desempenho do Prometheus é um requisito essencial para seu uso num ambiente de computação ubíqua, caracterizada por dispositivos com recursos computacionais limitados, tais como memória, processamento e energia residual.

Outro objetivo específico é avaliar o esforço de codificação do Prometheus fazendo uso de métricas como: o número de componentes; o número de linhas de código fonte do Prometheus; e o número de linhas de código fonte de cada Componente.

## 5.1 Premissas

O Serviço de Segurança Adaptativa está baseado em algumas premissas descritas a seguir.

a. É assumido que os Controles de Segurança utilizados pelo Prometheus são totalmente eficazes para prover a segurança adequada diante de mudanças de contexto. A discussão sobre a eficácia dos Controles de Segurança está fora do escopo desse trabalho.

b. Não é possível, em tempo de projeto, antecipar todas as respostas que um serviço de segurança ciente de contexto pode dar frente à ocorrência de eventos ainda não previstos, uma vez que novas vulnerabilidades são descobertas todos os dias. Portanto, é muito importante manter atualizado o conjunto de Regras de Segurança utilizadas para identificar vulnerabilidades, e o conjunto de Controles de Segurança usados para mitigar essas vulnerabilidades. Ou seja, incorporar dinamicamente novas regras de segurança e os componentes de segurança para satisfaze-las.

c. Controles de Segurança consomem recursos do dispositivo tais como memória e processador. O Prometheus aciona o mínimo necessário de Controles de Segurança de modo a garantir que os Requisitos de Segurança das aplicações sejam satisfeitos e, ao mesmo tempo, que sejam consumidos o mínimo de recursos.

d. Não é necessário ter os Controles de Segurança em funcionamento o tempo todo, mas apenas quando existem vulnerabilidades. Ter Controles de Segurança em funcionamento em contextos onde as vulnerabilidades não estão presentes não contribui para melhorar o nível de segurança.

A premissa (d) acima implica que os Controles de Segurança devem ser acionados quando uma vulnerabilidade é detectada, e devem estar em funcionamento antes que uma ameaça possa se concretizar, ou seja, a velocidade da reação é um fator determinante para o sucesso do Prometheus. A premissa (d) implica também que os Controles de Segurança devem ser desativados quando não forem mais necessários.

## 5.2 Ambiente e Informações de Contexto analisadas

Os experimentos do presente trabalho foram realizados num notebook Compaq Presario V2607CL com Windows XP Home Edition conectado a uma rede sem fio com um único roteador do tipo 802.11g, o qual está conectado à internet, possibilitando assim que o dispositivo móvel possa se conectar a um servidor corporativo.

Os experimentos foram configurados de modo ao Prometheus atender a uma ou mais aplicações em uso em um dispositivo móvel. O dispositivo móvel, ao ser ligado, inicia a execução do Prometheus. Durante a inicialização do Prometheus, os Componentes de Segurança, depois de carregados, obtém o valor inicial para cada uma das Informações de Contexto pelas quais são responsáveis, e passam a monitorar as mudanças nesses valores. Qualquer mudança significativa é registrada no Gestor de Contexto. Nos experimentos realizados, as Informações de Contexto analisadas foram: Aplicações em Execução (*InfoAplicacao*); Rede em Uso (*InfoRedeEmUso*); Túnel SSH (*InfoTunelSsh*); Conexões permitidas pelo *firewall* (*InfoSaidaTcp*); Criptografia Usada pelos Túneis SSH (*InfoCriptogafia*); e Memória Disponível (*InfoMemoria*).

## 5.3 Descrição das Métricas

Com o intuito de avaliar o Prometheus, nesse trabalho foram utilizadas métricas temporais, de código e de consumo de recursos. Essas métricas são descritas detalhadamente nas próximas subseções.

### 5.3.1 Métricas Temporais

As métricas temporais estão relacionadas com a seqüência de eventos entre o instante em que ocorre uma mudança no contexto e o efetivo funcionamento dos Controles de

Segurança acionados em conseqüência dessa mudança. A Figura 5.1 ilustra detalhadamente essa seqüência de eventos.

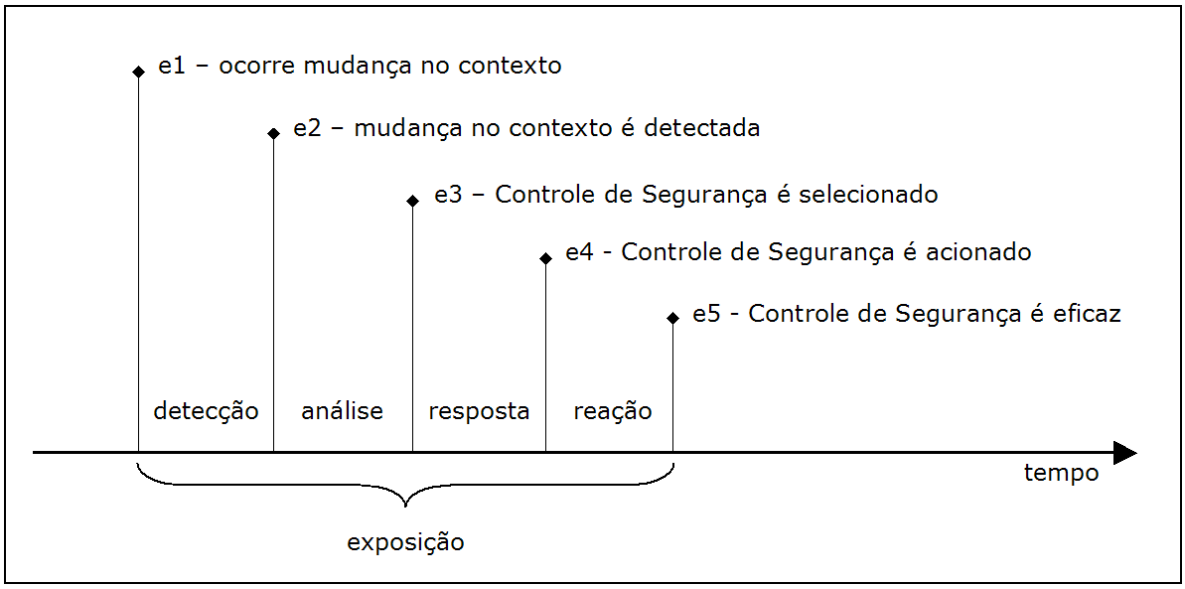


**Figura 5.1. Detecção, Análise e Resposta.**

O evento e1 é caracterizado por uma mudança no contexto de segurança dos aplicativos. O evento e2 ocorre quando, após detectar a mudança no contexto, os Componentes de Segurança registram a nova situação no Gestor de Contexto. ***Tempo de Detecção*** é definido como sendo o intervalo de tempo entre os eventos e1 e e2. Após a detecção é feita a análise do contexto. O processo de análise busca por vulnerabilidades no ambiente, e seleciona Controles de Segurança para mitigar as vulnerabilidades encontradas. O evento e3 ocorre quando é selecionada a reação do sistema à mudança de contexto registrada em e2. Observe que nem toda mudança de contexto constitui uma vulnerabilidade para as aplicações, portanto, pode não haver reação à mudança no contexto. ***Tempo de Análise*** é definido como sendo o intervalo de tempo entre os eventos e2 e e3. O evento e4 ocorre quando o Controle de Segurança é acionado em resposta à fragilidade detectada. ***Tempo de Resposta*** é definido como sendo o intervalo de tempo entre os eventos e3 e e4. Em algumas situações, a reação do sistema à mudança do sistema não é imediata, ou seja, a reação não é eficaz imediatamente após o acionamento do Controle de Segurança. O evento e5 ocorre no momento em que o Controle de Segurança passa a ser totalmente eficaz. ***Tempo de Reação*** é definido como sendo o intervalo de tempo entre os eventos e4 e e5.

O intervalo de tempo entre os eventos e1 e e5 é definido como sendo o ***Tempo de Exposição***. Duas estratégias são utilizadas para minimizar o risco das aplicações durante o Tempo de Exposição. A primeira é de minimizar o Tempo de Exposição. Um Tempo de Exposição pequeno reduz as chances de um ataque ser bem sucedido. A segunda estratégia é

de antecipação. Os Controles de Segurança devem ser acionados aos primeiros indícios de que uma situação potencialmente perigosa está se formando, e não quando a ameaça já é uma realidade.

Em suma, para avaliar o desempenho do Prometheus foram utilizadas 4 métricas temporais principais, a saber: Tempo de Detecção (TD); Tempo de Análise (TA); Tempo de Resposta (TRSP) e Tempo de Reação (TR).

Os tempos de detecção, análise, resposta e reação são examinados em mais detalhes nas seções a seguir.

- **Tempo de Detecção**

*O Tempo de Detecção* (TD) é o intervalo de tempo entre a ocorrência de uma mudança no contexto e o retorno do método que registra essa mudança no Gestor de Contexto. Esse tempo pode variar enormemente e depende da implementação do Controle de Segurança correspondente e da natureza da Informação de Contexto. De uma forma geral, quanto mais importante ou crítica é a vulnerabilidade potencial detectada pela mudança no valor de uma determinada Informação de Contexto, menor deve ser o tempo de detecção. Por exemplo, o tempo de detecção de uma mudança na Informação de Contexto *InfoVersao*, indicando a necessidade de atualizar Regras e Controles de Segurança, pode ser de algumas horas enquanto uma mudança na Informação de Contexto *InfoMemoriaDisponivel*, indicando a quantidade de memória disponível para os aplicativos, pode levar menos que 1 segundo. Já a detecção do início de execução de uma aplicação ocorre antes mesmo que a aplicação seja de fato iniciada (antecipação).

Quanto à estratégia para atualizar o valor das Informações de Contexto pelos Componentes de Segurança, ela pode ser ativa ou passiva. Na estratégia passiva, o Componente de Segurança é avisado quando o valor monitorado é modificado. Na estratégia ativa, o Componente de Segurança verifica periodicamente se o valor monitorado foi modificado. Ambas as estratégias têm vantagens e desvantagens.

Na estratégia ativa, o Componente de Segurança aguarda um tempo entre as verificações. O atraso entre o valor ser modificado e a detecção dessa mudança é de, em média, a metade desse tempo de espera. Assim, quanto menor o tempo de espera, menor será o Tempo de Detecção. A verificação, no entanto, tem um custo. Dessa forma, quanto mais freqüentes forem as verificações, menor será o atraso (menor Tempo de Detecção), porém, maior será o custo.

Na estratégia passiva, a verificação das informações de contexto só ocorre quando o valor monitorado muda de valor, o que parece mais interessante. Alguns valores, entretanto, são modificados com muita freqüência, mas as mudanças não são significativas como é o caso da memória disponível. Nesses casos a estratégia ativa consome uma grande quantidade de recursos inutilmente. Nem sempre é possível utilizar a estratégia passiva, já que ela depende de haver suporte para isso na entidade que produz a informação.

A decisão de qual estratégia de monitoração utilizar deve levar em conta os fatores apresentados em seguida.

**Tabela 5.1. Critérios para selecionar estratégia de monitoração**

| **Critério** | **Consideração** |
|---|---|
| Suporte para a estratégia passiva | A não existência de suporte para a monitoração passiva obriga o uso da monitoração ativa. |
| Urgência de atualizar a Informação de Contexto | O tempo de detecção tende a ser menor com a monitoração passiva. |
| Custo de verificar o valor da informação | Se o custo é alto, verificar com freqüência pode ser muito caro. |
| Freqüência com que a informação muda de valor | Monitorar passivamente informações que mudam com muita freqüência (voláteis) pode ser muito caro. |

Assim, o Tempo de Detecção depende da implementação do Componente de Segurança.

- **Tempo de Análise**

Ao registrar uma modificação no valor de uma Informação de Contexto, o Gestor de Contexto notifica o Gestor de Segurança Adaptativa (GSA) para que faça a análise do contexto. O procedimento GSA é composto por um conjunto de passos que são executados em um laço infinito. Esse laço é executado numa linha de processamento (*thread*) separada do Gestor de Contexto. Os passos executados nesse laço são:

a. aguarda notificação do Gestor de Contexto;
b. obtém cópia da TIC;
c. obtém Requisitos de Segurança;
d. obtém CRNA;
e. seleciona adaptadores;
f. executa adaptadores (Resposta);
g. volta para o passo (a).

O Tempo de Análise (TA) é definido como sendo o intervalo de tempo entre a notificação efetuada pelo Gestor de Contexto e a Seleção dos Adaptadores. O Tempo de Análise é composto pelo somatório dos seguintes tempos: Tempo para Copiar a TIC (TA.C); Tempo para Obter os Requisitos de Segurança (TA.RS); o Tempo para Obter o CRNA (TA.CRNA); o Tempo para Selecionar Adaptadores (TA.SA) e o Atraso da Análise (TA.A). O Atraso da Análise será detalhado em seguida.

O ***Tempo para Copiar a*** TIC (***TA.C***) é definido com sendo o intervalo de tempo entre a chamada do método do Gestor de Contexto que faz a cópia da TIC e o instante seguinte ao retorno desse método. O ***Tempo para Obter os Requisitos de Segurança*** (***TA.RS***) é definido como sendo o intervalo de tempo entre a chamada do método do Gestor de Regras que calcula os Requisitos de Segurança para um dado contexto e o instante seguinte ao retorno desse método. O ***Tempo para Obter o CRNA*** (***TA.CRNA***) é definido como sendo o intervalo de tempo entre a chamada do método do Gestor de Regras que obtém o CRNA e o instante seguinte ao retorno desse método. O ***Tempo para Selecionar Adaptadores*** (***TA.SA***) para os requisitos de segurança exigidos é definido como sendo o tempo decorrido entre o instante que o Gestor de Segurança Adaptativa (GSA) chama o método do Gestor de Política de Adaptação (GPA), que determina quais Controles de Segurança devem ser acionados, e o instante seguinte ao envio da lista de adaptadores pelo GPA. O ***Atraso da Análise (TA.A)*** é definido como sendo o tempo entre a notificação da mudança de contexto e o início de processamento do GSA para o tratamento dessa notificação. O Atraso da Análise é detalhado em seguida.

A linha de execução do GSA é notificada pelo Gestor de Contexto sempre que ocorre uma mudança no valor de uma Informação de Contexto. Entretanto, devido à possibilidade do GSA estar processando uma notificação anterior, o GSA pode só tomar conhecimento dessa mudança quando executar o passo (a), resultando num atraso para o inicio do processamento da análise da última notificação. Esse atraso é denominado de Atraso da Análise. Se no

momento da notificação a linha de execução estiver parada no ponto (a), então o processamento é prontamente iniciado e não há Atraso da Análise, ou seja, o Atraso da Análise é zero. Se, entretanto, ocorrer uma mudança no contexto quando a execução do GSA estiver, por exemplo, no ponto (e), então a análise da modificação no contexto só será iniciada após a conclusão das etapas (e), (f) e (g). O pior caso ocorre quando a notificação do GSA ocorre com a linha de execução no ponto (c). No pior caso, portanto, a duração do Atraso da Análise é igual à soma dos piores tempos de (c), (d), (e) e (f). A Figura 5.2 ilustra uma situação em que o GSA é notificado duas vezes num curto espaço de tempo, e a notificação (B) é enviada ao GSA antes que termine o processamento da notificação (A).

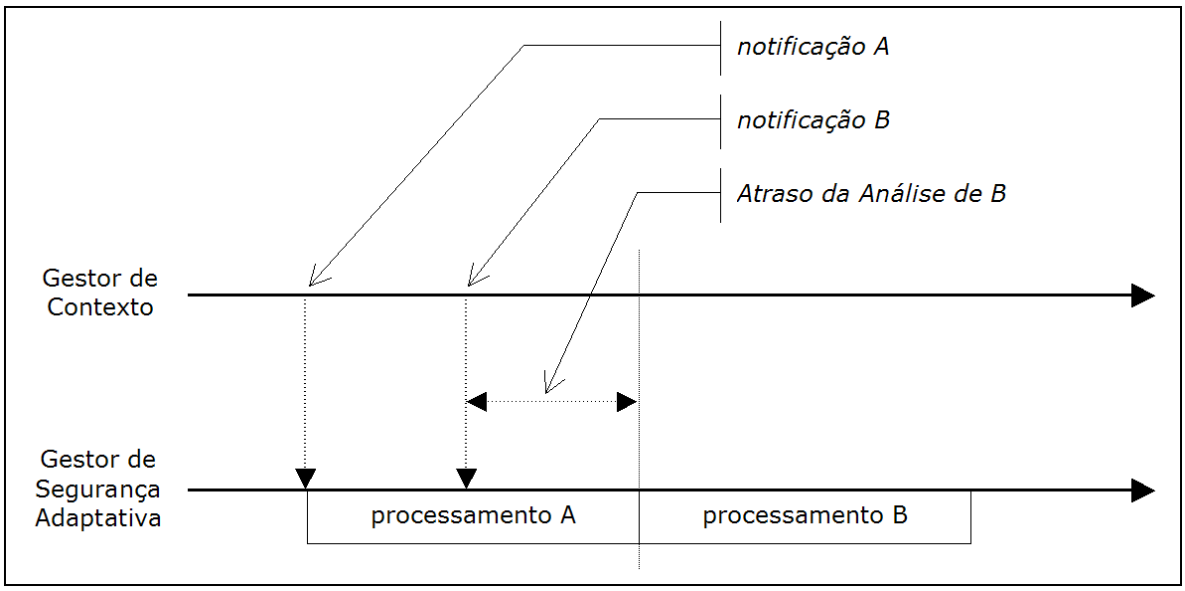


**Figura 5.2. Atraso da Análise**

Em suma, o Tempo de Análise (TA) é definido como sendo a soma dos seguintes intervalos de tempo: Atraso da Análise (TA.A); Tempo para obter a cópia da TIC (TA.C); Tempo para calcular os Requisitos de Segurança (TA.RS); Tempo para obter o CRNA (TA.CRNA); e Tempo para selecionar os adaptadores (TA.SA). O Tempo de Análise é influenciado por vários fatores, dentre eles, o número de Informações de Contexto na TIC; o número de regras no Repositório de Regras; e do número de Informações de Contexto no CRNA.

- **Tempo de Resposta**

O Tempo de Resposta é o intervalo de tempo durante o qual são acionados os Controles de Segurança selecionados pela análise. Vale lembrar que na fase de análise é selecionado um conjunto de Adaptadores que atuarão junto aos Componentes de Segurança para acionar os Controles de Segurança. O Tempo de Resposta (TRSP) é definido como sendo

o intervalo de tempo entre a chamada do primeiro adaptador selecionado até o retorno do último adaptador selecionado.

A duração do Tempo de Resposta é igual ao somatório dos tempos para executar cada um dos adaptadores selecionados. Portanto, o Tempo de Resposta varia conforme os adaptadores selecionados.

- **Tempo de Reação**

A maioria dos Controles de Segurança é eficaz logo após seu acionamento. Para alguns Controles de Segurança, entretanto, existe um intervalo de tempo entre o seu acionamento e o efetivo funcionamento. O efeito do fechamento de uma porta TCP no *firewall* local do dispositivo, por exemplo, é imediato. Já a mudança do algoritmo de criptografia dos túneis SSH, não é imediata, pois depende de negociações com o servidor. Ao receber a solicitação de troca, o Componente de Segurança retorna imediatamente, e, após retornar, inicia a negociação com o servidor. O Tempo de Reação (TR) é definido como sendo o instante após o acionamento do Controle de Segurança e seu efetivo funcionamento.

Embora não seja uma métrica de tempo, a métrica ***Número de Caracteres Expostos*** (***NCE***) está relacionada com o tempo de reação**.** Assim a métrica NCE mede o volume máximo de dados recebidos desde que foi solicitada a troca do algoritmo de criptografia até que o novo algoritmo tenha sido adotado. Exceto os adaptadores relacionados ao Gestor de Conexão, todos os outros adaptadores analisados nesse trabalho não envolvem nenhuma negociação com o servidor (troca de mensagens) para que a adaptação possa ser realizada. Portanto, os valores do TR e NCE*,* para esses outros adaptadores, são iguais a zero.

### 5.3.2 Métricas de Código

Para avaliar o esforço de codificação do Prometheus utilizaram-se as seguintes métricas de Código: Número de componentes, Número de classes, Número de instruções não incluindo comentário no código fonte (*NCSS – Non Commenting Source Statements*) do Prometheus bem como o NCSS dos principais componentes.

Embora o uso da métrica de linhas de código (SLOC – *Source Lines Of Code*) tenha sido muito criticado como critério de avaliação de projetos, ela é intuitiva. Não é difícil imaginar que o esforço para desenvolver um programa com 10.000 linhas seja maior do que aquele necessário para um programa de 500 linhas. Comparado com outras métricas, como Pontos de Função, a métrica SLOC é fácil de obter e fácil de entender, e serve para dar uma noção, ainda que imprecisa, do tamanho do projeto. A métrica SLOC, entretanto, é muito influenciada pelo estilo utilizado pelo programador na escrita do código fonte, já que inclui,

indiferentemente, linhas de código, linhas em branco e comentários. Como alternativa, a métrica NCSS (*Non Commenting Source Statements*) conta apenas o número de instruções da linguagem (*statements*), e dessa forma diminui ou elimina o impacto do estilo de formatação usado pelo programador sobre o valor medido.

### 5.3.3 Métricas de Consumo de Recursos

Para avaliar o impacto do uso do Prometheus sobre o desempenho dos aplicativos foram efetuadas medidas sobre o consumo de recursos, particularmente a memória utilizada pelo Serviço de Segurança (o próprio Prometheus) e o consumo de espaço de armazenamento para o código do Prometheus. O protótipo do Prometheus foi desenvolvido em Java. Para garantir a portabilidade, o Java cria uma máquina virtual (JVM) dentro da qual o aplicativo em Java é executado. O aplicativo Java se relaciona com a JVM assim como um aplicativo não Java se relaciona com o Sistema Operacional. O consumo de memória de um aplicativo em Java é gerenciado pela JVM. Quando um aplicativo Java aloca uma porção de memória, essa memória é obtida da JVM e não do sistema operacional. Somente se a JVM não dispuser de memória suficiente é que a quantidade de memória em uso pela JVM será aumentada. Da mesma forma, quando um aplicativo libera uma porção de memória ela não é retornada ao SO, mas para a JVM. Assim, medir a quantidade de memória utilizada por cada componente dentro da JVM significa muito pouco para avaliar o impacto desse componente em particular sobre o dispositivo. Diante disso, as medidas foram efetuadas sobre a quantidade de memória utilizada pela JVM do Prometheus e não por cada componente individualmente. A métrica ***Memória Utilizada (MU)*** indica quantidade de memória utilizada pelo Prometheus durante sua operação. A métrica MU é expressa em Mega Bytes (1 MB = $2^{20}$ bytes ou 1.048.576 bytes).

## 5.4 Descrição dos Experimentos

O protótipo do Prometheus foi implementado e submetido a dois experimentos que visam comprovar se metas específicas foram atingidas.

O primeiro experimento visa medir o atraso imposto ao início de execução de uma aplicação devido às adaptações efetuadas para garantir os Requisitos de Segurança. Nesse experimento as Informações de Contexto envolvidas foram *InfoAplicacao*, *InfoSaidaTcp*, e *InfoTunelSsh*. As condições iniciais do experimento foram:

- Não havia nenhuma instância da aplicação em execução.
- Não havia nenhum túnel SSH estabelecido.

- Todas as portas do *firewall* estavam fechadas.

Os Requisitos de Segurança da aplicação solicitam:

- Uma porta TCP de saída aberta.
- Um túnel SSH com um servidor remoto.

O segundo experimento verifica se há violação da integridade ou da confidencialidade das aplicações em execução durante o Tempo de Exposição em conseqüência da mudança da rede. Foram efetuadas medidas do Tempo de Exposição e da quantidade de informação trafegada entre o dispositivo e o servidor corporativo durante o Tempo de Exposição. Nesse segundo experimento as Informações de Contexto envolvidas foram *InfoRedeEmUso* e *InfoCriptografia*.

Para os dois experimentos será medido o Tempo de Exposição. Esse tempo, como já visto, é composto pela soma dos tempos de Detecção, de Análise, de Resposta e de Reação.

## 5.5 Análise dos Resultados

Essa seção apresenta os resultados da avaliação do Prometheus. Nesses experimentos, para cada medida, foram feitas pelo menos 30 amostragens e computados a média, o desvio padrão e o intervalo de confiança de 95%.

As medidas de tempo foram obtidas ao se calcular a diferença dos valores adquiridos ao invocar o método *System.nanoTime()* da biblioteca do Java antes e depois da execução do que se deseja medir. Cabe observar que embora a precisão da medida obtida por esse método seja de nanossegundos, a acurácia depende da implementação da Máquina Virtual do Java (JVM – *Java Virtual Machine*). Testes realizados no mesmo ambiente em que foram obtidas as medidas revelam que a menor diferença obtida entre leituras consecutivas de *System.nanoTime()* foi de 1676ns ($1{,}68\ 10^{-6}$s). Embora não seja suficiente para determinar a acurácia do relógio, o teste mostra que a acurácia é melhor que a obtida com *System.currentTimeMillis()*, e adequada para as medidas efetuadas.

As seções a seguir apresentam os resultados temporais das atividades Detecção, Análise, Resultado e Reação que estão relacionados com a seqüência de eventos entre o instante em que ocorre uma mudança no contexto e o efetivo funcionamento dos Controles de Segurança acionados em conseqüência dessa mudança.

### 5.5.1 Detecção

A seguir será apresentado o Tempo de Detecção para as Informações de Contexto *InfoAplicacao*; *InfoRedeEmUso*; *InfoMemoriaDisponivel*; e *InfoSaidaTcp*.

- ***InfoAplicacao***

O Tempo de Detecção da Informação de Contexto *InfoAplicacao* está relacionado ao primeiro experimento. ***InfoAplicacao*** é tratada pelo Componente de Execução e indica quais aplicações estão em execução. A estratégia utilizada é passiva, ou seja, o Componente de Execução é informado quando uma aplicação vai iniciar e quando termina. O tratamento de *InfoAplicacao* é antecipado, ou seja, antes de executar uma nova aplicação o Controle de Execução informa ao Gestor de Contexto o novo valor de *InfoAplicacao* através de um método *atualizaContextoSinc*. Esse método, diferente de *atualizaContexto*, só retorna depois que o tratamento do Gestor de Segurança Adaptativa para essa mudança foi efetuado, em outras palavras, só retorna depois que os Controles de Segurança relativos aos Requisitos de Segurança da aplicação que vai se iniciar foram acionados. Assim, podemos afirmarque o Tempo de Detecção para *InfoAplicacao* é zero.

- ***InfoRedeEmUso***

O valor da Informação de Contexto *InfoRedeEmUso* indica qual rede sem fio está em uso pelo dispositivo. InfoRedeEmUso é tratada pelo Componente de Redes. O Tempo de Detecção de *InfoRedeEmUso* está relacionado ao segundo experimento. A informação de qual rede está em uso é crítica para o Prometheus, pois pode, por exemplo, determinar qual o algoritmo de criptografia deve ser usado para garantir a integridade e a confidencialidade da informação. É portanto importante que o Gestor de Contexto seja atualizado o mais rápido possível. Considerando os critérios da Tabela 5.1, a estratégia utilizada pelo Controle de Redes é a passiva. Para obter a informação de qual rede está em uso o Controle de Redes usa, através de JNI (Java Native Interface) um componente em C++. Esse componente, por sua vez, obtém a informação por meio da API WMI (Windows Management Instrumentation).

No caso de *InfoRedeEmUso* o Tempo de Detecção é igual ao tempo para verificar a rede em uso mais o tempo para registrar o valor no Gestor de Contexto.

Na implementação utilizada nos testes essa informação é obtida do sistema operacional (Windows XP) através de uma API (WMI). Nos testes preliminares a obtenção da informação através dessa API se mostrou lenta (cerca de 20 ms). Foi então utilizada a abordagem passiva. Com essa abordagem é registrado no sistema operacional o interesse pela informação. Quando a informação muda de valor o Componente de Segurança é informado através de um mecanismo de *callback*. A informação é obtida então somente quando ocorre uma mudança no valor da informação.

Foi medido o tempo para obter a informação do sistema operacional através da API. O resultado da medição pode ser observado na Tabela 5.2.

**Tabela 5.2. Tempo para verificar a Rede em Uso**

| | |
|---|---|
| Mínimo | 15 ms |
| Médio | 20,80 ms |
| Máximo | 109 ms |
| número de amostras | 1.000 |
| Desvio Padrão | 7,97 ms |
| Intervalo de Confiança 95% | ±0,49 ms |

O Tempo de Detecção esperado para a *InfoRedeEmUso* é de 20,8ms ± 0,49ms para um intervalo de confiança de 95%.

- ***InfoMemoriaDisponivel***

A Informação de Contexto InfoMemoriaDisponivel informa quanto há de memória disponível no sistema, e é tratada pelo Componente de Memória. Por ser uma informação muito volátil, a estratégia utilizada é ativa. São feitas medições periódicas da quantidade de memória disponível. Se a diferença entre a medida atual e a anterior for maior que um determinado patamar, então o novo valor é registrado no Gestor de Contexto. Esse patamar é necessário para diminuir a volatilidade da informação, e assim, melhorar o desempenho.

O patamar utilizado pelo Componente de Memória é de 8 MB. O intervalo entre verificações é de 1.000 ms. O resultado das medições pode ser visto na Tabela 5.3.

**Tabela 5.3. Tempo para verificar a Memória Disponível**

| | |
|---|---|
| Mínimo | 31,0 µs |
| Médio | 36,7 µs |
| Máximo | 70,0 µs |
| número de amostras | 100 |
| Desvio Padrão | 5,4 µs |
| Intervalo de Confiança de 95% | ±2,1 µs |

O tempo para obter a informação é, portanto, muito baixo. O intervalo entre verificações é 4 ordens de grandeza maior que o tempo necessário para verificar.

- ***InfoSaidaTcp***

A Informação de Contexto ***InfoSaidaTcp*** informa quais portas TCP estão abertas para saída no *firewall* local, ou seja, através de quais portas é possível estabelecer uma conexão TCP com um endereço remoto. Essa Informação de Contexto é tratada pelo Componente de Firewall.

Espera-se que a freqüência com que portas são abertas ou fechadas no *firewall* do dispositivo seja muito baixa. A estratégia de monitoramento utilizada foi passiva.

Foi medido o tempo necessário para obter essa informação pela implementação do Componente de Firewall. O resultado das medições pode ser visto na Tabela 5.4

**Tabela 5.4. Tempo para verificar Portas de Saída TCP abertas no *firewall***

| | |
|---|---|
| Mínimo | 31,0 ms |
| Médio | 35,6 ms |
| Máximo | 78,0 ms |
| número de amostras | 1.000 |
| Desvio Padrão | 7,30 ms |
| Intervalo de Confiança 95% | ±0,45 ms |

O Tempo de Detecção esperado para *InfoSaidaTcp* é de 35,6ms ±0,45ms para um intervalo de confiança de 95%.

### 5.5.2 Análise

Como já visto, o Tempo de Análise (TA) é igual à soma do Atraso da Análise (TA.A); Tempo para obter a cópia da TIC (TA.C); Tempo para calcular os Requisitos de Segurança (TA.RS); Tempo para obter o CRNA (TA.CRNA); e Tempo para selecionar os adaptadores (TA.SA).

A seguir é apresentado o resultado das medições efetuadas para cada um desses tempos.

- **Tempo para Copiar a TIC (TA.C)**

Devido à natureza dinâmica da informação contida na Tabela de Informações de Contexto (TIC), a análise do contexto é feita sobre uma cópia dessa estrutura de dados. A TIC, conforme já mencionado, é a estrutura de dados usada pelo Gestor de Contexto para armazenar as Informações de Contexto. O tempo para efetuar a cópia depende do número de Informações de Contexto na TIC. Assim, foram efetuados experimentos variando o número

de Informações de Contexto entre zero e 60 em intervalos de 10. O gráfico na Figura 5.3 mostra o resultado das medições efetuadas.

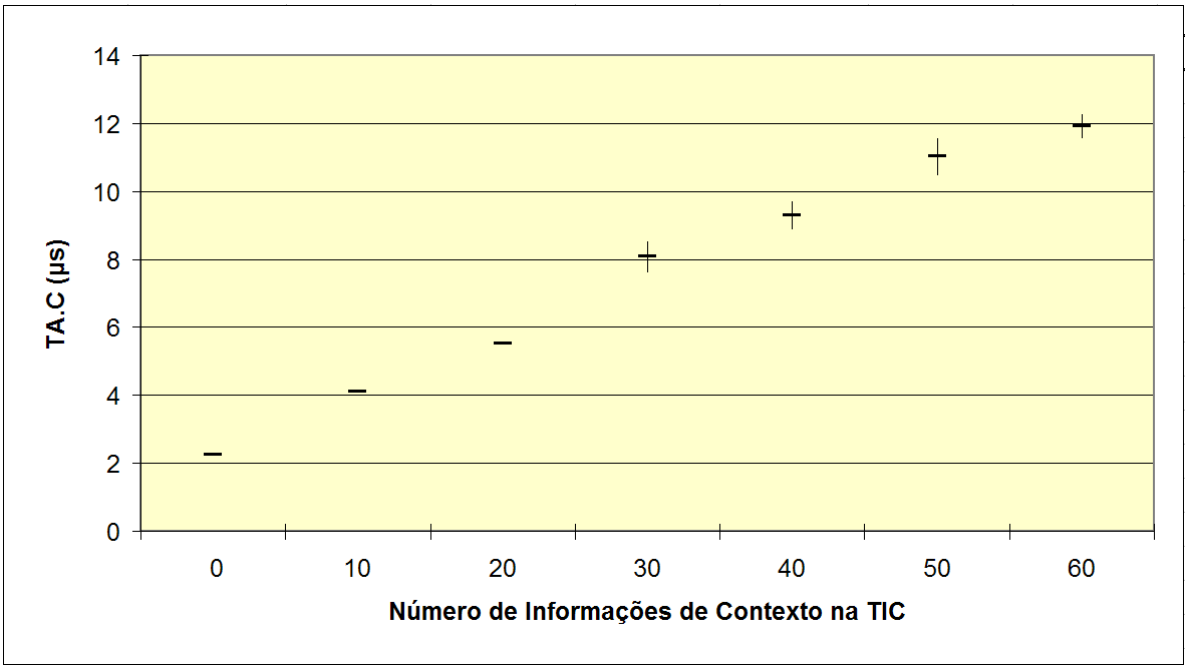


**Figura 5.3. Tempo para copiar a TIC (TA.C)**

No gráfico o eixo X mostra o número de Informações de Contexto na TIC e o eixo Y mostra o tempo em microssegundos (µs) para efetuar a cópia. Para cada medida foram efetuadas 1000 amostragens. Pode-se observar nessa figura que para uma tabela contendo 60 Informações de Contexto obteve-se o maior valor de tempo (11,9µs) para efetuar a cópia. Conforme esperado, o Tempo para Cópia da TIC cresce linearmente com o número de Informações de Contexto. Observa-se também que TA.C é um valor muito pequeno.

- **Tempo para calcular os Requisitos de Segurança (TA.RS)**

Os requisitos de segurança são obtidos executando as Regras de Segurança sobre o contexto. As regras são compostas por condições e ações. Quando as condições são satisfeitas as ações são executadas. Cada ação indica um valor para uma Informação de Contexto. Ao final do processo os valores combinados das Informações de Contexto serão o Requisito Representativo. A estrutura de dados utilizada para armazenar o contexto é acessada através de um HASH. O tempo para acessar o conteúdo da tabela, portanto, não varia com o seu tamanho. O tempo para calcular os requisitos, entretanto, varia conforme o número de regras e com o número de ações a executar.

O contexto sobre o qual o experimento foi executado foi montado de forma que as condições fossem verdadeiras em 50% dos casos. Para cada condição verdadeira foram

executadas duas ações. As medidas foram efetuadas variando-se o número de regras entre zero e 50 com intervalos de 10. Para cada ponto foram efetuadas 1000 amostragens. O resultado do experimento pode ser observado no gráfico da Figura 5.4.

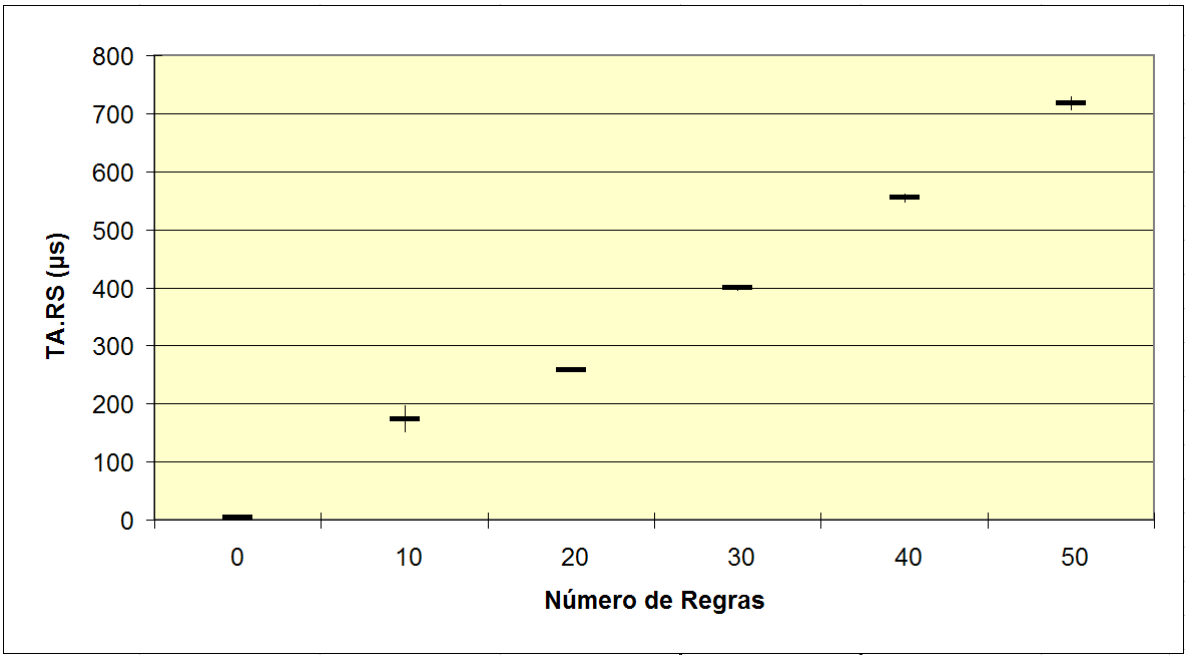


**Figura 5.4. Tempo para calcular os Requisitos de Segurança (TA.RS)**

O eixo X do gráfico representa o número de Regras de Segurança avaliadas, e o eixo Y o tempo para calcular os Requisitos de Segurança em microssegundos (µs). Como esperado, o TA.RS cresce linearmente com o número de Regras de Segurança. Observa-se também que o tempo para calcular os Requisitos de Segurança é muito pequeno (menor que um milissegundo), mesmo para um número de regras relativamente alto.

- **Tempo para obter o CRNA (TA.CRNA)**

O CRNA é obtido computando-se a diferença entre os Requisitos de Segurança e a Tabela de Informações de Contexto (TIC). O tempo para obter essa diferença (TA.CRNA) varia conforme o número de Requisitos de Segurança. As medidas foram efetuadas variando o número de Requisitos de Segurança de zero a 25 com incrementos de 5. Para cada ponto foram efetuadas 1000 amostras. O resultado dessas medidas pode ser visto no gráfico da Figura 5.5.

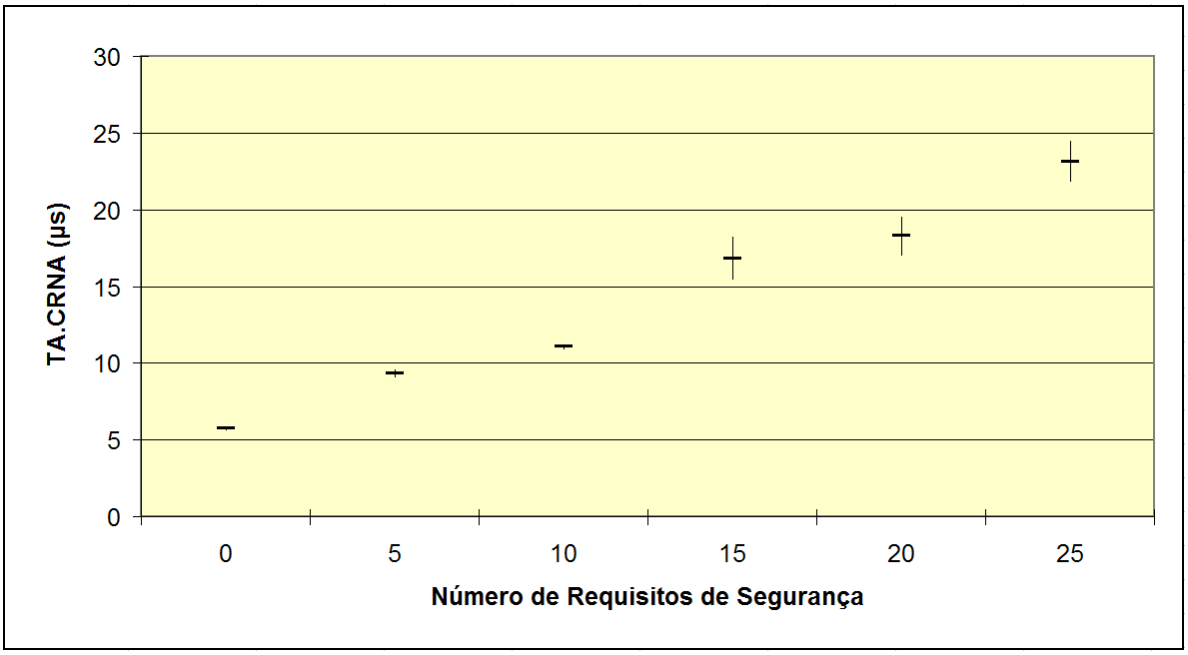


**Figura 5.5. Tempo para obter o CRNA (TA.CRNA)**

No gráfico o eixo X representa o número de Requisitos de Segurança no Requisito Representativo. O eixo Y representa o tempo para obter o CRNA (TA.CRNA) em microssegundos (µs). Para cada ponto a linha vertical representa o intervalo de confiança de 95% da medida. Como esperado, observa-se que o TA.CRNA cresce linearmente com o número de Requisitos de Segurança. Observa-se também que, mesmo para um número expressivo de Requisitos de Segurança (25), os tempos medidos são muito pequenos (21,8µs ± 1,3µs).

- **Tempo para Selecionar Adaptadores (TA.SA)**

Nessa etapa é avaliado o Tempo para Selecionar Adaptadores (TA.SA) para satisfazer aos requisitos não atendidos. Foram efetuadas medidas variando o número de Requisitos de Segurança de 0 a 30 em intervalos de 5. Para cada ponto foram efetuadas 2000 amostragens. O resultado das medições pode ser visto no gráfico da Figura 5.6. Em cada ponto no gráfico está representado o intervalo de confiança de 95% como uma linha vertical.

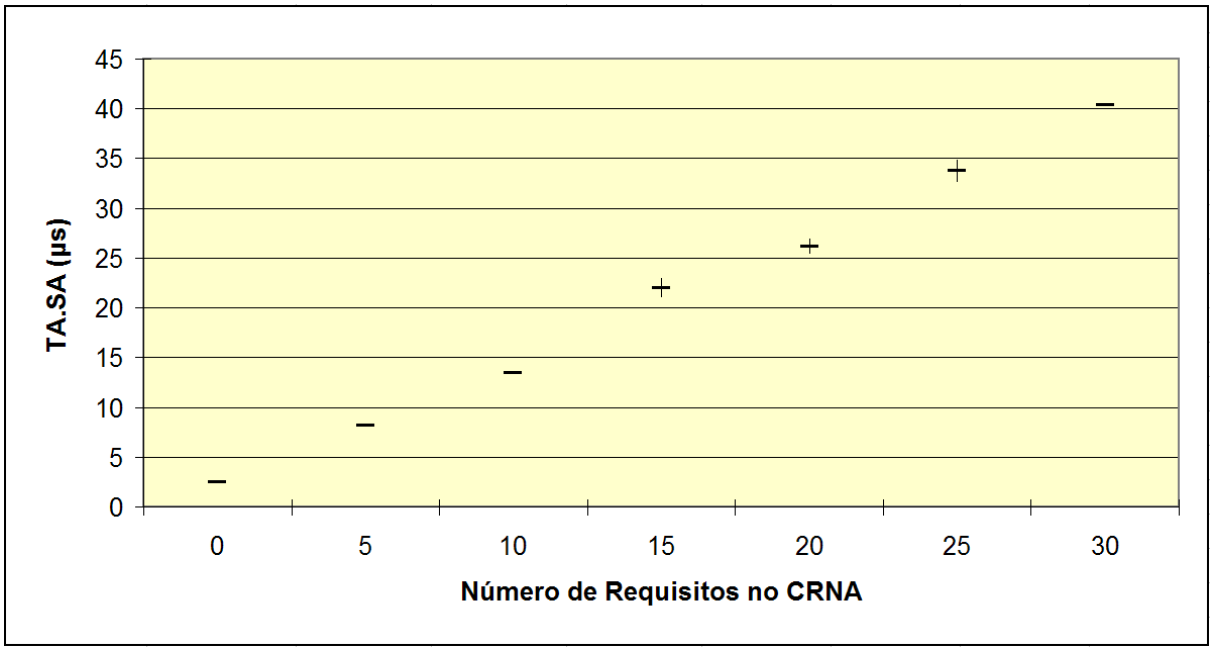


**Figura 5.6. Tempo para Selecionar Adaptadores (TA.SA)**

No gráfico o eixo X representa o número de Requisitos de Segurança no CRNA, e o eixo Y representa o tempo para selecionar os adaptadores (TA.SA). Como podemos observar o tempo cresce linearmente com o número de Requisitos de Segurança. Foram efetuados outros testes incluindo adaptadores que atendem a mais de um requisito, mas não foi observada nenhuma diferença significativa nos resultados.

- **Atraso da Análise (TA.A)**

Nos experimentos realizados em nenhum momento ocorreu atraso na análise. Não foi, portanto, possível efetuar uma medida desse tempo. Existe, entretanto, a possibilidade de calcular um atraso teórico, como sendo:

TA.A = TA.C + TA.RS + TA.CRNA + TA.SA + TRSP

Verificou-se que o valor esperado para a soma das parcelas relativas à análise é menor que 0,1ms. O valor do Tempo de Resposta (TRSP) será examinado a seguir.

### 5.5.3 Resposta

Essa etapa é responsável por executar as adaptações selecionadas e carregadas pelas etapas anteriores. O tempo para executar as adaptações varia conforme o Controle de Segurança que está sendo acionado. O Controle de Segurança acionado está relacionado ao valor desejado para uma Informação de Contexto. A seguir é examinado o TRSP para o estabelecimento de túneis SSH (*InfoTunelSsh*); para a configuração do *firewall*

(*InfoSaidaTcp*); para a execução de uma aplicação (*InfoAplicacao*); e para a mudança do algoritmo de criptografia usada pelos túneis SSH (*InfoCriptografia*).

- **TRSP para *InfoTunelSsh***

A métrica TRSP relacionada à adaptação que garante o estabelecimento de conexões TCP Segura através do mecanismo de túneis SSH é definida como sendo a soma de dois tempos: Tempo de Estabelecimento de Sessão SSH (***TSessão***) e Tempo de Estabelecimento do Túnel (***TTunel***). Define-se ***TSessão*** como sendo o tempo de conexão com o servidor SSH mais o tempo de autenticação. O ***TTunel*** é definido como sendo o tempo para o cliente SSH solicitar ao servidor SSH a criação de um túnel sobre uma sessão e o cliente SSH receber a confirmação do pedido.

Foram efetuadas medidas fixando o número de sessões em 1 e variando o número de túneis entre 1 e 5. Para cada medida foram efetuadas 300 amostras. O resultado das medições pode ser observado no gráfico da Figura 5.7.

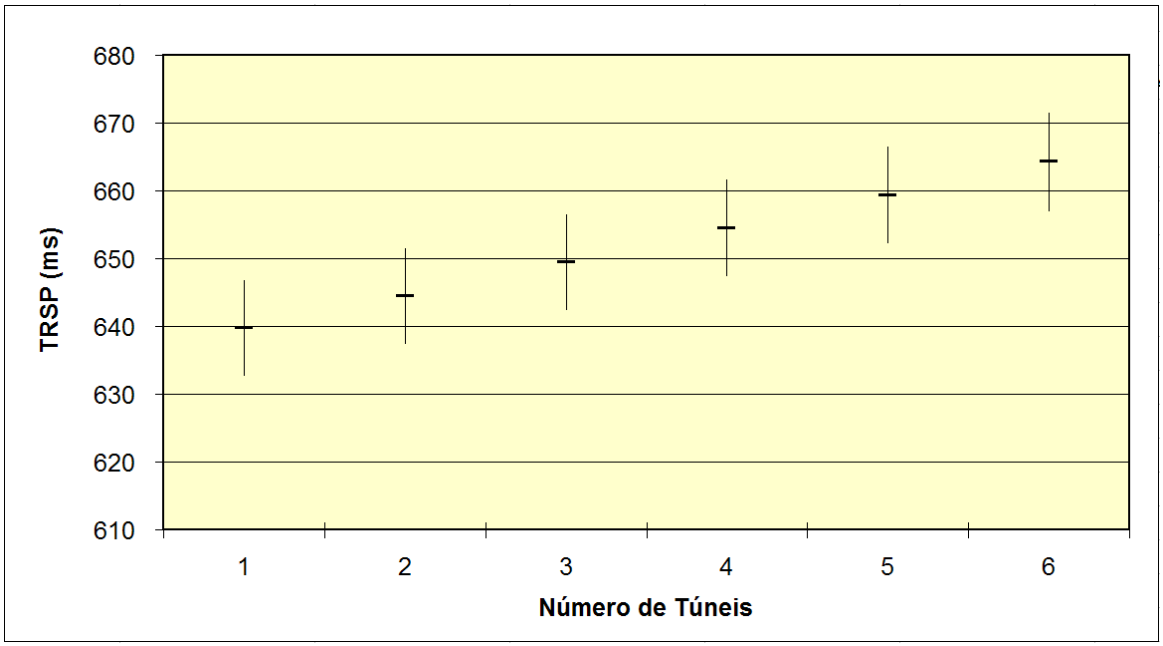


**Figura 5.7. TRSP para *InfoTunelSsh***

O gráfico mostra no eixo X o número de túneis estabelecidos, e no eixo Y o tempo em milissegundos (ms) para estabelecer esses túneis. A cada ponto é mostrado o intervalo de confiança de 95% da amostra. Como esperado o tempo para estabelecer o primeiro túnel é maior que o tempo para estabelecer os demais túneis. Isso se deve ao fato que para estabelecer o primeiro túnel é necessário antes estabelecer uma sessão SSH com o servidor. A partir do segundo túnel o tempo cresce linearmente com o número de túneis criados. O tempo para

criar um túnel sobre uma sessão (***TTunel***) é de 4,9 ms ± 0,2 ms, e o tempo para criar uma sessão SSH (***TSessão***) é de 634,8 ms ± 7,0 ms.

- **TRSP para *InfoSaidaTcp***

A Informação de Contexto ***InfoSaidaTcp***, conforme já exposto, informa quais portas TCP estão abertas para saída no *firewall* local, ou seja, através de quais portas é possível estabelecer uma conexão TCP com um endereço remoto. Foi medido o tempo gasto pelo adaptador para reprogramar o *firewall*. O resultado das medições efetuadas pode ser visto na Tabela 5.5.

**Tabela 5.5. Tempo para programar o *firewall***

| | |
|---|---|
| Mínimo | 31,0 ms |
| Médio | 34,3 ms |
| Máximo | 94,0 ms |
| número de amostras | 1.000 |
| Desvio Padrão | 6,69 ms |
| Intervalo de Confiança 95% | ±0,41 ms |

Assim, o tempo esperado para executar a adaptação que reprograma o firewall é de 34,3ms ±0,41ms para um intervalo de confiança de 95%.

- **TRSP para *InfoCriptografia***

Conforme já mencionado, a Informação de Contexto *InfoCriptografia* indica qual nível de criptografia deve ser usado para encriptar dados trafegados nos túneis SSH. O nível de criptografia é dado pelo algoritmo utilizado. Foi medido o tempo necessário para o adaptador acionar o uso do novo algoritmo junto ao Gestor de Conexões. O tempo para mudar o algoritmo está relacionado com o número de sessões SSH estabelecidas. Assim, foram efetuadas medidas variando o número de sessões de 1 a 5. Para cada um desses pontos foram efetuadas 240 amostras. Entre cada uma das amostras aguardou-se 1300ms para não contaminar a medida com o processamento relativo à mudança de algoritmo provocada pela amostra anterior. O tempo de cada amostra foi obtido medindo-se a diferença entre os valores da função *System.nanoTime()* obtidos antes e depois da chamada do adaptador. O resultado das medidas pode ser visto no gráfico da Figura 5.8.

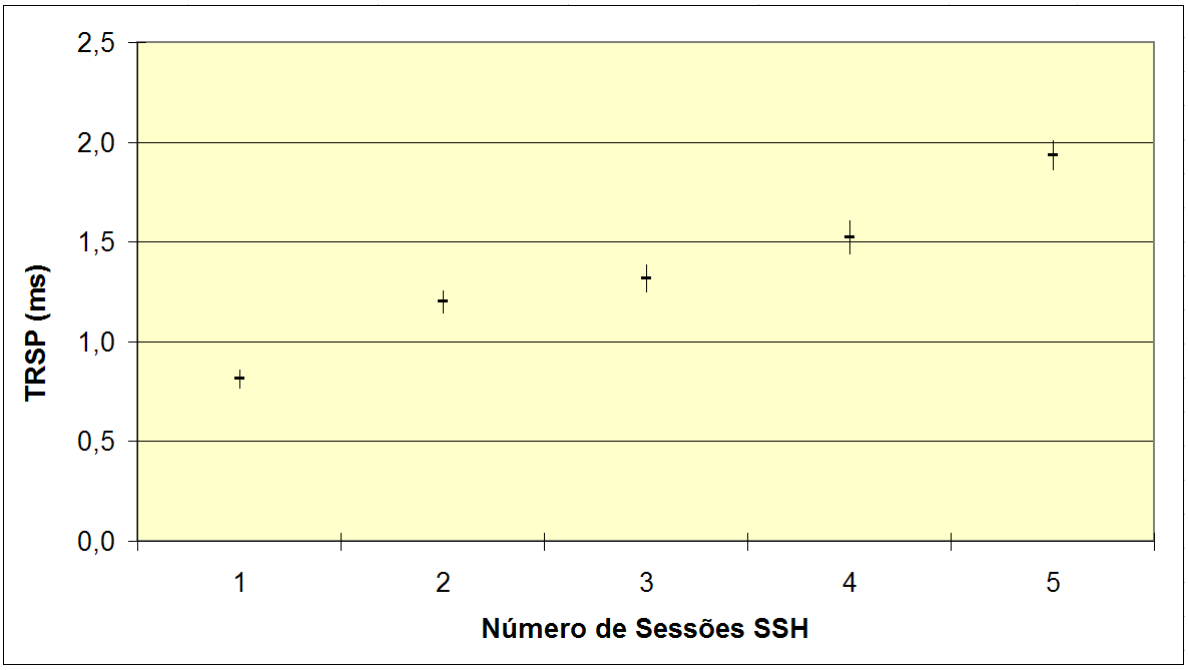


**Figura 5.8. TRSP para *InfoCriptografia***

No gráfico o eixo X representa o número de sessões SSH no momento da execução da adaptação. O eixo Y mostra o tempo em milissegundos (ms) gasto para efetuar a adaptação. Para cada ponto está ilustrado o intervalo de confiança de 95%. Como esperado, observamos que o TRSP cresce linearmente com o número de sessões SSH. Observamos também que o tempo é muito pequeno (de 1 a 2 ms, dependendo do número de sessões SSH).

### 5.5.4 Reação

O Tempo de Reação é de zero para todos os Controles de Segurança acionados pelo Prometheus, com a exceção de um. Isso significa que os Controles de Segurança são eficazes a partir do momento em que são acionados. A exceção fica por conta do Controle de Segurança responsável por mudar o valor de *InfoCriptografia*. Essa Informação de Contexto indica qual algoritmo de criptografia deve ser utilizado para garantir a confidencialidade e a integridade das informações trafegadas entre as aplicações e o servidor corporativo. Esse Controle de Segurança atua junto às instâncias de sessões SSH já estabelecidos para seja negociado com os respectivos servidores o uso do novo algoritmo. A biblioteca [Ganymed] utilizada na implementação dos túneis SSH faz a negociação do novo algoritmo de forma assíncrona, ou seja, a negociação continua após o retorno do método que solicita a mudança de algoritmo. Com isso, os dados que já transmitidos pelo dispositivo, mas, por algum motivo, ainda não recebidos pelo servidor (*buffers*, tempo de propagação na rede, roteadores intermediários, etc), bem como os dados já transmitidos pelo servidor, mas ainda não recebidos pelo dispositivo, continuarão trafegando pela rede encriptados com o algoritmo

antigo. A demora para a mudança do algoritmo de criptografia pode representar um risco se: (i) a mudança é de um algoritmo mais fraco para um mais forte (ii) há alguma sessão SSH estabelecida quando é solicitada a mudança do algoritmo de criptografia; (iii) há túneis SSH estabelecidos sobre essas sessões; (iv) há conexões TCP sobre esses túneis; (v) a quantidade de informação trafegada com o algoritmo mais fraco é suficiente para que um observador mal intencionado consiga quebrar a chave; (vi) a porção de dados trafegados com o algoritmo mais fraco continha informações de fato confidenciais.

Para medir o Tempo de Reação foi feito um teste em que são transmitidos caracteres para um aplicativo especial no servidor. Esse aplicativo no servidor apenas ecoa os caracteres recebidos. O teste começa enviando um fluxo contínuo de caracteres "A". O eco dos caracteres enviados é recebido numa linha de processamento separada (*thread*). Após o envio de 100.000 caracteres é solicitada a troca do algoritmo de criptografia; mudado o caractere enviado para "B"; e disparado um temporizador. Foi observado que o primeiro caractere "B" enviado já foi criptografado com o novo algoritmo. O teste mediu o tempo para receber o primeiro caractere "B" (Tempo de Reação) e o número de caracteres "A" recebidos após a solicitação de troca de algoritmo (quantidade de dados expostos). A Tabela 5.6 contém os resultados do experimento.

**Tabela 5.6. Medidas para TR**

| Medida | Tempo de Reação | Número de Caracteres Expostos |
|---|---|---|
| Mínimo | 313 ms | 0 |
| Médio | 601 ms | 17.103 |
| Máximo | 969 ms | 30.000 |
| Número de amostras | 82 | 82 |
| Desvio Padrão | 159 ms | 9311 |
| Intervalo de confiança 95% | ±34,75 ms | ±2016 |

Nota-se uma flutuação forte nos tempos e no volume de dados expostos. Isso foi atribuído ao fato de que os testes foram efetuados através da Internet por meio de uma conexão doméstica de banda larga (2 Mb/s) do lado do cliente, e uma conexão de 3 Mb/s não dedicada exclusivamente ao experimento do lado do servidor.

Constata-se que o Tempo de Reação é relativamente curto (601 ± 34,75ms), e a quantidade máxima de dados exposta, relativamente pequena.

Duas abordagens podem ser utilizadas para diminuir o risco relacionado ao Número de Caracteres Expostos. A primeira é de nunca trafegar os dados confidenciais em claro, utilizando como nível mínimo de criptografia um algoritmo que elimine o risco relacionado ao Número de Caracteres Expostos. A segunda abordagem é de disparar a mudança da criptografia não quando mudar a rede WI-FI, mas quando a potencia do sinal recebido ficar abaixo de um certo limiar para, dessa forma, se antecipar a uma iminente troca de rede.

### 5.5.5 Resultados para o Primeiro Experimento

O primeiro experimento consistiu de medir o atraso imposto pelo Prometheus devido à execução de uma aplicação devido a adaptações para preparar o ambiente de segurança. O atraso é a soma dos tempos de Detecção; Análise; e Resposta. Os tempos obtidos estão resumidos na Tabela 5.7.

**Tabela 5.7. Tempos para o Primeiro Experimento**

| **Tempo de Detecção** | **0ms** |
|---|---|
| **Tempo de Análise** | **79,2 µs** |
| TA.A | 0 µs |
| TA.C | 2,8 µs |
| TA.RS | 64 µs |
| TA.CRNA | 7,4 µs |
| TA.SA | 4,9 µs |
| **Tempo de Resposta** | **674 ms** |
| reprogramar *firewall* | 34 ms |
| criar túnel SSH | 640 ms |
| **Tempo de Reação** | **0 ms** |
| **Tempo Total** | **674 ms** |

Como podemos constatar pelos números na tabela, o tempo de análise é 4 ordens de magnitude menor que o Tempo de Resposta.

O maior responsável pelo atraso foi a criação do túnel SSH, com quase 95% do tempo. Se, entretanto, a sessão SSH já estivesse estabelecida (por exemplo para outra aplicação) o tempo para criar um túnel seria de 4,9 ms.

Concluímos que o atraso imposto pelas adaptações antes da execução da aplicação é aceitável. Outra conclusão é que o mecanismo de seleção de adaptadores, descrito na seção

3.5.7, embora complexo, causa muito pouco impacto no tempo total, e por isso mesmo, não deve ser o primeiro alvo de uma eventual otimização.

### 5.5.6 Resultados relacionados ao Código e ao Consumo de Recursos

Com o intuito de demonstrar que o Serviço de Segurança Adaptativa, Prometheus, pode ser usado em ambiente ubíquo sem impactar as aplicações, constatou-se o pequeno tamanho do código do Prometheus em Java e em C++ (respectivamente 4.750 e 832 linhas sem comentários – NCSS) e que o Prometheus requerer uma sobrecarga baixa em termos de armazenamento em disco (996 KB) e *footprint* de memória (28 MB de memória máxima usada pela JVM do Prometheus).

A maior parte do código do Prometheus foi desenvolvida em Java. Entretanto, alguns componentes utilizam particularidades do sistema operacional hospedeiro, e por esse motivo foram desenvolvidos em C++. A Tabela 5.8 mostra o número de linhas de código sem os comentários (*NCSS – Non Commenting Source Statements*) dos principais componentes e do código do fonte do projeto, incluindo as rotinas de teste. Ainda nessa tabela é ilustrado o número de componentes usado em Java (21 componentes) e em C++ (2 componentes); o número de classes em Java (143 classes) e o número de métodos (566 métodos).

**Tabela 5.8. Métricas de Código do Prometheus e de Consumo de Recursos**

| Métricas | Valor | Unidade |
|---|---|---|
| NCSS do Gestor de Contexto | 47 | NCSS |
| NCSS do Gestor de Políticas de Adaptação | 218 | NCSS |
| NCSS do Gestor de Regras de Segurança | 102 | NCSS |
| NCSS do Gestor de Segurança Adaptativa | 33 | NCSS |
| NCSS do Gestor de Componentes | 29 | NCSS |
| NCSS do Controle de Execução | 79 | NCSS |
| NCSS do Controle de Versão | 53 | NCSS |
| NCSS do Controle de Conexões | 263 | NCSS |
| NCSS do Controle de Firewall | 107 | NCSS |
| NCSS do Controle de Memoria | 56 | NCSS |
| NCSS do Controle de Redes | 44 | NCSS |
| NCSS de todo o código fonte em Java | 4.750 | NCSS |
| Número de classes Java | 143 | Classes |
| Número de métodos | 566 | Métodos |
| Número de Componentes em C++ | 4 | Componentes |
| NCSS do código em C++ | 573 | NCSS |
| Tamanho do código executável em disco | 212 | KB |
| Tamanho da Bibliotecas utilizadas pelo Prometheus | 784 | KB |
| Memória máxima usada pela JVM do Prometheus | 16,3 | MB |

Os números para Java foram obtidos com a ferramenta [JavaNCSS]. Os números para C++ foram obtidos usando um critério semelhante ao usado pelo NCSS e coletados com a ajuda de um editor de textos.

A quantidade de memória utilizada foi observada com auxilio da ferramenta *Windows Task Manager*, parte integrante do sistema operacional hospedeiro do Prometheus.

## 5.6 Considerações Finais

Nesse capítulo foram descritos os experimentos realizados, as métricas, e expostos os resultados obtidos e as conclusões.

O próximo capítulo apresenta as conclusões e os trabalhos futuros.

# 6 Conclusão e Trabalhos Futuros

Este trabalho apresentou um Serviço de Segurança Adaptativa cujo objetivo principal foi atender dinamicamente aos Requisitos de Segurança das aplicações conforme o contexto de execução. Com o Prometheus é possível garantir os requisitos de disponibilidade, integridade e confidencialidade das aplicações diante de um ambiente altamente dinâmico e heterogêneo, que faz com que controles de segurança sejam diferentes conforme o contexto corrente.

O uso da Segurança Adaptativa como um serviço traz dois benefícios imediatos:

- O primeiro benefício é de liberar os projetistas e programadores da tarefa complexa de lidar com a ciência de contexto para adaptar a segurança de suas aplicações.
- O segundo benefício é que, uma vez adotada uma infra-estrutura de adaptação, os procedimentos de teste e validação efetuados para essa infra-estrutura valem para todas as aplicações que se beneficiam dela. A alternativa seria de validar os procedimentos adaptativos de cada aplicação.

A arquitetura proposta é flexível, e permite que sejam acrescentados ou substituídos componentes do sistema em tempo de execução. A Política de Segurança foi formalizada num conjunto de Regras de Segurança, e sintetizada num arquivo em XML. As Regras de Segurança também podem ser atualizadas dinamicamente.

Uma contribuição foi a criação de operações para manipular as Informações de Contexto. As Informações de Contexto são implementadas como especializações de uma classe base. Essa classe base define operações que são implementadas nas especializações. Este encapsulamento permite abstrair os detalhes de cada implementação e as particularidades de cada Informação de Contexto, tornando assim possível tratar Informações de Contexto com características diferentes de modo uniforme.

Os experimentos relatados no capítulo 5 mostraram que o Prometheus pode ser usado para garantir os Requisitos de Segurança das aplicações, e que o porte da implementação de referência é pequeno o suficiente ser executado na maioria dos *smart phones* hoje disponíveis.

O tratamento de conflitos feito pelo Prometheus nos Requisitos de Segurança é muito primário. Investigar formas melhores de expressar os conflitos de requisitos e como lidar com esses conflitos está entre as prioridades de estudos futuros. Uma das possibilidades aventadas é a de se atribuir prioridades às aplicações. Assim, aplicações menos prioritárias seriam selecionadas para que cedam os recursos alocados em favor de aplicações mais prioritárias.

No estágio atual do Prometheus, decisões mais complexas da Política de Adaptação foram delegadas ao código dos adaptadores. Um trabalho futuro seria fornecer a Política de Adaptação numa linguagem declarativa, à semelhança das Regras de Segurança. Essa linguagem deveria ter expressividade suficiente para atender a questões como "a quantidade energia residual está crítica: devo desativar uma aplicação que usa muita energia ou desligar algum equipamento?".

A velocidade dos processadores parece estar chegando ao limite físico. Os progressos obtidos recentemente na velocidade dos novos processadores são muito menos impressionantes do que já foram no passado. A capacidade de processamento dos equipamentos, entretanto, continua crescendo. Este crescimento, porém, está sendo obtido com processadores de vários núcleos, capazes de executar mais de um programa simultaneamente. É, portanto, essencial que as novas aplicações sejam capazes de executar algoritmos que se beneficiem da capacidade de processamento paralelo. Na análise do Prometheus efetuada no Capítulo 5 ficou claro que o gargalo no processo de adaptação está na execução das adaptações. No estado atual da arquitetura as adaptações devem ser executadas seqüencialmente. Um estudo futuro é o de montar uma árvore de dependência entre as adaptações, e fazer o Prometheus lidar com essa árvore, identificando as adaptações que podem ser executadas em paralelo.

Ao final da execução de um aplicativo as adaptações efetuadas em favor dessa aplicação não são imediatamente desfeitas, mas somente após um tempo. Isso é feito para tentar atenuar o custo de adaptar e "desadaptar" quando aplicações são executadas repetidamente. Manter as adaptações já efetuadas indefinidamente na expectativa de reutilizar os recursos já alocados é uma possibilidade. Recursos alocados, entretanto, têm um custo. Um estudo futuro será o de pesar (i) o custo de manter os recursos alocados; (ii) o custo para desalocar os recursos; (iii) o impacto dos recursos alocados sobre a disponibilidade das outras aplicações. Com base nesses pesos, tomar a decisão de, se e quando, os recursos devem ser desalocados.

A execução de uma adaptação tem como conseqüência a modificação de uma ou mais Informações de Contexto. Um novo valor para uma Informação de Contexto leva o Gestor de Contexto a notificar o Gestor de Segurança Adaptativa para que analise o contexto e, possivelmente, execute outra adaptação. Esse processo pode levar a um laço infinito, no qual os recursos do sistema são consumidos inutilmente até a parada total do equipamento. Um estudo futuro é o de investigar mecanismos para identificar essa situação e contornar o problema. Uma possibilidade seria de validar as Regras de Segurança, na tentativa de

identificar um laço infinito. Outra possibilidade seria o de instalar um mecanismo que identifique o laço e o interrompa, sem no entanto comprometer o funcionamento do serviço ou a segurança.

O Prometheus utiliza uma solução proprietária para o gerenciamento de componentes. Uma possível evolução do Prometheus seria a utilização de uma solução aberta. Soluções abertas são amplamente discutidas e tendem a ser muito melhores que soluções proprietárias. Uma possível alternativa é o modelo [OSGI]. O OSGI é um modelo bastante maduro; tem tido aceitação cada vez maior; provê facilidades para lidar com a atualização dinâmica e versionamento de componentes; é razoavelmente simples; e há implementações pequenas o suficiente não onerar demasiadamente uma aplicação do porte do Prometheus.

## Referências Bibliográficas

BULCÃO NETO, R. F.; TEIXEIRA, C. A. C.; PIMENTEL, M. G. C. A Semantic web-based infrastructure for supporting context-aware applications. In: YANG, L.T. et.al. (Ed.) **Embedded and ubiquitous computing – EUC 2005** International Conference EUC 2005, Nagasaki, Japan, Dec. 6-9 2005 Proceedings. Berlin: Springer, 2005. p. 900-909. (Lecture Notes on Computer Science, 3824).

CANAL, C.; Murillo, J. M.; Poizat, P. Software adaptation. **L'Objet**, Cachan, v. 12, n. 1, p. 9–31, 2006. Special Issue on Coordination and Adaptation Techniques for Software Entities.

CARTON, A. et al. Aspect-oriented model-driven development for mobile context-aware computing In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING OF PERVASIVE COMPUTING APPLICATIONS, SYSTEMS AND ENVIRONMENTS, 1., 2007, Minneapolis. **Proceedings …** Minneapolis: IEEE, 2007.

CHIGAN, C.; LEIYUAN, L.; YINGHUA, Y. Resource-aware self-adaptive security provisioning in mobile ad hoc networks. In: WIRELESS COMMUNICATIONS AND NETWORKING CONFERENCE, 2005 New Orleans. **Proceedings …** Hoes Lane: IEEE; p. 2118-2124, 2005. v. 4.

DEY, A. K.; ABOWD, G. D. The context toolkit: aiding the development of context-aware applications. In: WORKSHOP ON SOFTWARE ENGINEERING FOR WEARABLE AND PERVASIVE COMPUTING, 2000, Limerick. **Proceedings …** , Seatle: University of Chicago, 2000.

DOOMUN, M R.; SOYJAUDAH, K. M. S. IEEE 802.11i security for energy-security optimization, In: ADVANCED INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 3., 2007, Morne. **Proceedings …** Hoes Lane: IEEE, 2007.

ELKHODARY A.; WHITTLE, J. Survey of approaches to adaptive application security. In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR ADAPTIVE AND SELF-MANAGING SYSTEMS, 2007, Minneapolis. **Proceedings …** Hoes Lane: IEEE, 2007.

FISMA. **Federal Information System Management Act**. Disponível em: <csrc.nist.gov/drivers/documents/FISMA-final.pdf>. Acesso em: ago. 2009.

GLBA. **The Gramm-Leach Bliley Act**. Disponível em <banking.senate.gov/conf/confrpt.htm>. Acesso em: ago. 2009.

HIPAA. **The Health Insurance Portability and Accountability Act of 1996**. Disponível em: <www.hhs.gov/ocr/privacy>. Acesso em: ago. 2009.

HINTON, H. et al. SAM: security adaptation manager. ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 15. 1999, Phoenix. **Proceedings …** Los Alamitos: IEEE, 1999. p. 361 – 370.

HOH, S . TAN, J. S.; HARTLEY, M. Context-aware systems - a primer for user-centred services. **BT Technology Journal**, London, v. 24, n. 2, p. 186-194, Apr. 2006.

ISO 27000. **International Organization for Standardization.** Disponível em: http://www.iso.org. Acesso em: jun. 2009.

IZQUIERDO, A.: SIERRA, J. M.: TORRES, J. Providing security for digital ecosystems with adaptative encryption, In: DIGITAL ECOSYSTEMS AND TECHNOLOGIES CONFERENCE, 2007, Cairns. **Proceedings …** Hoes Lane: IEEE, 2007. p. 329 – 333.

JAVANCSS - A source measurement suite for java L. Clemens. 2009 Disponível em: http://www.kclee.de/clemens/java/javancss/. Acesso em: 25 jun. 2009.

MAIA, R. F. **Um framework para adaptação dinâmica de sistemas baseados em componentes distribuídos**. 2007. Dissertação (Mestrado em Informática) - Departamento de Informática, Programa de Pós-Graduacão em Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

MARCUS, L. Introduction to logical foundations of an adaptive security infrastructure. In: WORKSHOP ON LOGICAL FOUNDATIONS OF AN ADAPTIVE SECURITY INFRASTRUCTURE, 2004, Turku. **Proceedings …** .Turku: [s.n.], 2004. p. 251-266.

MOURA, A. L. et al. Dynamic support for distributed auto-adaptive applications In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 22, 2002, Vienna. **Proceedings …** Los Alamitos: IEEE, 2002. p. 451–458.

OSGI Aliance. **The Dynamic module system for java**. Disponível em: <www.osgi.org> Acesso em: 25 jun. 2009.

PCI. **The Payment Card Industry Security Standards Council**. Disponível em: <www.pcisecuritystandards.org> Acesso em: ago. 2009.

PEREIRA FILHO, J.G. et al. Infraware: um middleware de suporte a aplicações móveis sensíveis ao contexto. In.: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 24., 2006, Curitiba. **Anais ...** , Curitiba: SBC, 2006.

PIRMEZ, M., et al. Prometheus: Um Serviço de Segurança Adaptativa. 2008. In: VII SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO. Gramado, RS, Brasil. **Anais ...** , Porto Alegre: SBC, 2008.

PLATTNER, C. **Ganymed SSH-2 for java build 210**. Disponível em: http://nixbit.com/cat/programming/libraries/ganymed-ssh-2-for-java-build/. Acesso em: 25 jun. 2009.

PREUVENEERS, D. et al. Context-aware adaptation for component-based pervasive computing systems In: INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING, 4., 2006. Dublin. **Proceedings …** Dublin: University College Dublin, 2006.

RFC2196 - **Site security handbook.** September 1997. Disponível em: http://www.faqs.org/rfcs/rfc2196.html. Acesso em: jun. 2009.

RFC2828 - **Internet security glossary.** May 2000. Disponível em: http://www.faqs.org/rfcs/rfc2828.html. Acesso em: jun. 2009.

RFC4251 - **The secure shell (SSH) protocol architecture**. January 2006. Disponível em: http://www.faqs.org/rfcs/rfc4251.html. Acesso em : jun. 2009.

ROBERTSON, P.; SHROBE, H. ; LADDAGA, R., (Ed). **Self-adaptive software**. Berlin: Springer-Verlag, 2001. (Lecture Notes in Computer Science, 1936). ISBN: 3-540-41655-2.

SACRAMENTO, V. et al. MoCA a middleware for developing collaborative applications for mobile users. In: INTERNATIONAL MIDDLEWARE CONFERENCE, 5., 2004. Toronto. **Proceedings …** .Toronto: ACM/IFIP/USENIX, 2004. p. 262-326.

SANTOS, I. L.A. et al. **Usando aspectos e composição dinâmica para prover adaptação ciente ao contexto em sistemas ubíquos.** 2008. Dissertação (Mestrado em Sistemas e Computação) - Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, 2008

SBC.Grandes desafios da pesquisa em computação 2006 – 2016., **Computação Brasil**, Porto Alegre, Ano .7, n. 23, set./out./nov. 2006.

SILVA, F. J. S. **Adaptação dinâmica de distemas distribuídos.** 2003. Tese (Doutorado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2003.

SPRINGER, T. et al. Middleware support for context-awareness in 4G. In: INTERNATIONAL WORKSHOP ON WIRELESS MOBILE MULTIMEDIA ARCHIVE, 2006, Buffalo. **Proceedings …** Los Vaqueros Circle: IEEE, 2006 p. 203 – 211,

STALLINGS, W. **Network security essentials** 2nd ed. Upper Ssddle River: Prentice Hall, 2003.

TAYLOR, C; ALVES-FOSS, J. Attack recognition for system survivability: a low-level approach In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 36., 2003, Buffalo. **Proceedings …** Los Alamitos: IEEE, 2003

X.800.CCITT / ITU. Security architecture for open systems interconnection for CCITT applications. Geneva 1991

WEISE, J. **Designing an adaptive security architecture** Sun Microsystems, November 2008. Disponível em: http://wikis.sun.com/display/BluePrints/Designing+an+Adaptive+Security+Architecture. Acesso em: jun. 2009.

WEISER, M. The computer for the twenty-first century. **Scientific American**, New York, v. 265, Sept. 1991.

XIAO, L. An adaptive security model using agent-oriented MDA. **Information and Software Technology**, Volume 51, Issue 5, SPECIAL ISSUE: Model-Driven Development for Secure Information Systems, 2009, p. 933-955.

ZHANG, J.; CHENG, B. H. C. Model-based development of dynamically adaptive software. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 28., 2006. Shanghai. **Proceedings …** New York: ACM, 2006.

ZHANG, J et al. Adding safeness to dynamic adaptation techniques. In: WORKSHOP ON ARCHITECTING DEPENDABLE SYSTEMS, 2004, Edinburgh. In: **Proceedings …** New York: ACM/IEEE, 2004.

_____. Enabling safe dynamic component-based software adaptation. In: LEMOS, R.; GACEK, C.; ROMANOVSKI, A. (Ed.). **Architecting Dependable Systems III**, Berlin: Springer, 2005. (Lecture Notes in Computer Science, 2677).