UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GIOVANE CÉSAR MOREIRA MOURA

Uma Proposta para Medição de Complexidade e Estimação de Custos de Segurança em Procedimentos de Tecnologia da Informação

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Luciano Paschoal Gaspary
Orientador

Porto Alegre, julho de 2008

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Moura, Giovane César Moreira

Uma Proposta para Medição de Complexidade e Estimação de Custos de Segurança em Procedimentos de Tecnologia da Informação / Giovane César Moreira Moura. – Porto Alegre: PPGC da UFRGS, 2008.

74 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2008. Orientador: Luciano Paschoal Gaspary.

1. Gerência de redes e serviços, medição de complexidade, segurança de redes e computadores, modelo de custos, análise de regressão. I. Gaspary, Luciano Paschoal. II. Título.

Ao meu padrinho, Carlos Augusto Monteiro, por todo apoio, incentivo e motivação desde os primeiros passos no mundo da computação.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a meus pais, Juvenal e Neusa, por sempre terem apoiado e incentivado todas as minhas escolhas. Muito obrigado por acreditarem em mim.

Um agradecimento especial a minha namorada, Carla, pelo amor, compreensão, paciência e apoio durante o mestrado.

A meus irmãos, Thales e Vinícius, pela força e amizade sempre presentes, apesar da distância física.

Ao professor Luciano, por ter sido um orientador de fato, presente, amigo e incentivador. Muito obrigado por esta oportunidade que rendeu resultados tão positivos.

Gostaria de agradecer também ainda aos professores do Instituto de Informática que contribuíram para este trabalho, em especial ao Professor Lisandro, Professor Jürgen e Professor Weber.

A Alexander Keller e Yixin Diao, do IBM Thomas J. Watson Research Center, pelos valiosos comentários e sugestões durante o curso deste trabalho.

A todos os amigos na UFRGS: Raniery, Ewerton, Clarissa, Jéferson, José Rafael, Cristiano, Bruno, Jerônimo, Machado, Weverton, Paulo e Alex. Obrigado a todos pela ajuda, pelos bons momentos de diversão e todo apoio na hora dos “deadlines”, onde todo mundo sempre se ajudava em prol do grupo.

E a todos meus amigos e familiares de Goiás que lá ficaram, mas sempre estiveram comigo durante o mestrado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 7

LISTA DE FIGURAS 8

LISTA DE TABELAS 9

RESUMO 10

ABSTRACT 11

1 INTRODUÇÃO 12

2 TRABALHOS RELACIONADOS 15
2.1 Análise de prejuízos causados por indisponibilidade 15
2.2 Modelo de custos para administração de sistemas 16
2.3 Arcabouço conceitual para obtenção de um *benchmark* de complexidade de configuração 17
2.4 Complexidade de procedimentos de configuração 18
2.5 Modelo para previsão de custos de procedimentos de configuração . . . 20
2.6 Complexidade de processos de tecnologia da informação 20
2.7 Considerações parciais 22

3 MODELO DE SEGURANÇA DE TECNOLOGIA DE INFORMAÇÃO E MÉTRICAS DE COMPLEXIDADE 25
3.1 Modelo de Segurança de TI 25
3.1.1 Modelo de infra-estrutura 26
3.1.2 Modelo de atividades 27
3.2 Métricas de complexidade 27
3.2.1 Métricas em nível de procedimentos 28
3.2.2 Métricas em nível de ações 31

4 FERRAMENTA *SECURITY COMPLEXITY ANALYZER* 32
4.1 Arquitetura 32
4.2 Componentes da SCA 32
4.3 Implementação da SCA 34
4.3.1 Estrutura dos arquivos de entrada e saída 34

5 AVALIAÇÃO EXPERIMENTAL 37
5.1 Complexidade agregada por mecanismos de segurança em procedimentos gerais 37
5.1.1 Complexidade em nível de procedimentos 38
5.1.2 Complexidade em nível de ações 40
5.2 Medida de complexidade de procedimentos relacionados a mecanismos de segurança isolados 41
5.3 Comparação de valores de complexidade de procedimentos relacionados a diferentes ferramentas que suportam os mesmos mecanismos de segurança 43
5.4 Considerações parciais 44

6 PREVISÃO DE CUSTOS COM USO DE MÉTRICAS DE COMPLEXIDADE DE PROCEDIMENTOS DE TI 46
6.1 Metodologia para criação de modelo de previsão de custos 47
6.1.1 Etapa 1: Avaliação de complexidade e captura de tempos 47
6.1.2 Etapa 2: Construção do modelo quantitativo 47
6.1.3 Etapa 3: Extrapolação do modelo para realizar previsões 50
6.2 Estudo de caso: previsão de custos associados à segurança 50
6.2.1 Análise de complexidade e captura de tempos 51
6.2.2 Construção do modelo quantitativo 51
6.2.3 Extrapolação do modelo 54
6.2.4 Previsão de custos de segurança em nível de procedimentos 55
6.3 Considerações parciais 55

7 CONCLUSÃO E TRABALHOS FUTUROS 57

REFERÊNCIAS 59

APÊNDICE A ANÁLISE DE REGRESSÃO LINEAR 62
A.1 Introdução à análise de regressão 62
A.1.1 Método dos Mínimos Quadrados Ordinários 63
A.1.2 Avaliação da qualidade de modelos de regressão linear 64
A.2 Ferramenta utilizada: R 64

APÊNDICE B ARTIGO PUBLICADO 66

LISTA DE ABREVIATURAS E SIGLAS

BI	Business Item
CA	Certificate Authority
CI	Configuration Item
CSV	Comma-separated values
CMDB	Configuration Management Database
DMZ	Demilitarized Zone
GUI	Graphical User Interface
IPSec	IP Security
IT	Information Technology
ITIL	Information Technology Infrastructure Library
NIST	National Institute of Standards and Technology
OGC	Office of Government Commerce
RMSE	Root Mean Square Error
ROI	Return-On-Investiment
SCA	Security Complexity Analyzer
SLA	Service Level Agreement
SSL	Secure Sockets Layer
TCO	Total Cost of Ownership
TI	Tecnologia da Informação
VPN	Virtual Private Network
XMI	XML Metadata Interchange
XML	Extended Markup Language

LISTA DE FIGURAS

Figura 2.1: Tempo necessário para solucionar as requisições 17
Figura 2.2: Efeito que cem requisições trazem para o sistema quando o mesmo está quase saturado 17
Figura 2.3: Procedimento parcial de configuração do *SPECjAppServer* 18
Figura 2.4: Histograma de complexidade de ações 19
Figura 2.5: Exemplo parcial de um processo de mudança 21
Figura 2.6: Histograma de complexidade de processo 23

Figura 3.1: Modelo de infra-estrutura de TI 26
Figura 3.2: Procedimento completo para instalar Joomla com mecanismos de segurança 28
Figura 3.3: Árvore de hierarquia de contêineres. 29

Figura 4.1: Componentes internos do SCA 33
Figura 4.2: Representação XML do procedimento 35
Figura 4.3: Arquivo de resultado de complexidade gerado pela SCA 36

Figura 5.1: Procedimento para instalar o Joomla no cenário A 38
Figura 5.2: Procedimento para instalar o Joomla no cenário B 39
Figura 5.3: Complexidade em nível de procedimentos para os cenários A, B e C . 40
Figura 5.4: Complexidade de ações para o procedimento do cenário C 41
Figura 5.5: Procedimento referente ao cenário D 41
Figura 5.6: Procedimento referente ao cenário E 42
Figura 5.7: Procedimento referente ao cenário F 42
Figura 5.8: Procedimento referente ao cenário G 42
Figura 5.9: Procedimento referente ao uso do OpenVPN 44
Figura 5.10: Procedimento referente ao uso do Openswan 44

Figura 6.1: Análise de complexidade do procedimento base 52
Figura 6.2: Modelo criado com o cenário base 52
Figura 6.3: Correlação das métricas de complexidade com o tempo 53
Figura 6.4: Validação do modelo quantitativo com a previsão dos custos associados à segurança 54

LISTA DE TABELAS

Tabela 2.1: Métricas de complexidade de processos 22

Tabela 3.1: Mecanismos de segurança utilizados no cenário exemplo 26
Tabela 3.2: Valores de *SourceScore* para avaliação dos parâmetros 30

Tabela 5.1: Medidas de complexidade para os cenários A, B e C 40
Tabela 5.2: Medidas de complexidade para os procedimentos dos cenários D, E, F e G 43
Tabela 5.3: Medidas de complexidade para a instalação do OpenVPN e Openswan 45

Tabela 6.1: Evolução na qualidade do modelo a medida que mais métricas são adicionadas ao mesmo 53
Tabela 6.2: Tempo previsto e estimado para os cenários base e futuro 55

RESUMO

Segurança de TI tornou-se nos últimos anos uma grande preocupação para empresas em geral. Entretanto, não é possível atingir níveis satisfatórios de segurança sem que estes venham acompanhados tanto de grandes investimentos para adquirir ferramentas que satisfaçam os requisitos de segurança quanto de procedimentos, em geral, complexos para instalar e manter a infra-estrutura protegida. A comunidade científica propôs, no passado recente, modelos e técnicas para medir a complexidade de procedimentos de configuração de TI, cientes de que eles são responsáveis por uma parcela significativa do custo operacional, freqüentemente dominando o *total cost of ownership*. No entanto, apesar do papel central de segurança neste contexto, ela não foi objeto de investigação até então. Para abordar este problema, neste trabalho aplica-se um modelo de complexidade proposto na literatura para mensurar o impacto de segurança na complexidade de procedimentos de TI. A proposta deste trabalho foi materializada através da implementação de um protótipo para análise de complexidade chamado *Security Complexity Analyzer* (SCA). Como prova de conceito e viabilidade de nossa proposta, o SCA foi utilizado para avaliar a complexidade de cenários reais de segurança. Além disso, foi conduzido um estudo para investigar a relação entre as métricas propostas no modelo de complexidade e o tempo gasto pelo administrador durante a execução dos procedimentos de segurança, através de um modelo quantitativo baseado em regressão linear, com o objetivo de prever custos associados à segurança.

Palavras-chave: Gerência de redes e serviços, medição de complexidade, segurança de redes e computadores, modelo de custos, análise de regressão.

An approach to measure the complexity and estimate the cost associated to Information Technology Security Procedures

ABSTRACT

IT security has become over the recent years a major concern for organizations. However, it doest not come without large investments on both the acquisition of tools to satisfy particular security requirements and complex procedures to deploy and maintain a protected infrastructure. The scientific community has proposed in the recent past models and techniques to estimate the complexity of configuration procedures, aware that they represent a significant operational cost, often dominating total cost of ownership. However, despite the central role played by security within this context, it has not been subject to any investigation to date. To address this issue, we apply a model of configuration complexity proposed in the literature in order to be able to estimate security impact on the complexity of IT procedures. Our proposal has been materialized through a prototypical implementation of a complexity scorer system called Security Complexity Analyzer (SCA). To prove concept and technical feasibility of our proposal, we have used SCA to evaluate real-life security scenarios. In addition, we have conducted a study in order to investigate the relation between the metrics proposed in the model and the time spent by the administrator while executing security procedures, with a quantitative model built using multiple regression analysis, in order to predict the costs associated to security.

Keywords: Network and service management, complexity analysis, computer and network security, cost models, regression analysis.

1 INTRODUÇÃO

Segurança tem se tornado nos anos recentes uma preocupação central para as empresas em geral, que cada vez mais confiam e dependem de complexas infra-estruturas de Tecnologia da Informação (TI) para realizar suas atividades comerciais (CANNON; WHEELDON, 2007). Neste contexto, tanto a segurança física quanto a lógica são cuidadosamente empregadas nas operações diárias das empresas para obter diferentes níveis de proteção da informação. Confidencialidade, autenticidade, integridade, não-repúdio, controle de acesso e disponibilidade (STALLINGS, 2006) são exemplos de serviços de segurança lógicos que diretores e gerentes de TI podem demandar para que se preservem os dados sensíveis mantidos pelos *Configuration Itens* (CI) presentes na infra-estrutura de TI.

As demandas (ou requisitos), expressas por meio de *Service Level Agreements* (SLA) e/ou políticas de segurança, são materializadas através de um conjunto de mecanismos tais como criptografia de sistemas de arquivos e tráfego de rede, filtragem de pacotes e redundância de hardware e software. Esses mecanismos, por sua vez, podem ser implantados com ferramentas produzidas por diferentes fornecedores. A instalação, a configuração e a manutenção de tais ferramentas podem fazer parte de procedimentos mais gerais (por exemplo, junto à configuração de uma aplicação Web) ou de procedimentos focados especificamente nas próprias ferramentas de segurança.

Intuitivamente, à medida que se aumenta a quantidade de mecanismos de segurança a serem implantados, mais complexos, longos e onerosos os respectivos procedimentos tendem a se tornar. Considere, como exemplo, a instalação de um sistema operacional: o aumento no número de mecanismos de segurança agregados à instalação padrão acarreta procedimentos com um maior número de passos de execução e com mais parâmetros a serem fornecidos pelo administrador. Além disso, existem situações em que um mecanismo de segurança pode ser implantado utilizando diferentes ferramentas (por exemplo, a autenticação de usuários pode ser feita utilizando o software OpenLDAP ou o Active Directory da Microsoft), que podem diferir no que diz respeito à complexidade de instalação/manutenção.

Diante do exposto, a determinação da medida de complexidade de procedimentos para se estimar o custo de segurança em TI é fundamental por uma série de razões. Primeiramente, pode ser utilizada pelos diretores e gerentes para que os SLAs e as políticas de segurança sejam revisados levando em consideração os custos previstos associados com a implantação real dos mesmos. Além disso, provê uma diretriz para a equipe de TI acerca da escolha das ferramentas mais convenientes (por exemplo, as de menores valores de complexidade). Por fim, é possível identificar as ações relacionadas à segurança presentes nos procedimentos que apresentam os maiores valores de complexidade, para que as mesmas sejam classificadas como candidatas à automação (BROWN; KELLER;

HELLERSTEIN, 2005).

Diversos trabalhos têm sido realizados com o objetivo de medir a complexidade associada a procedimentos de configuração mais gerais. No primeiro deles, Brown e Hellerstein (BROWN; HELLERSTEIN, 2004) propõem um arcabouço para se obter um *benchmark* baseado em procedimentos que possibilite estimar a complexidade de configuração de sistemas. Procedimento, neste contexto, é definido como o conjunto de interações entre o operador e o sistema computacional para se atingir um estado específico de configuração. Como exemplo de um procedimento de configuração, têm-se as tarefas necessárias para configurar o servidor Web httpd da Apache Foundation, que abrange desde a criação de usuários (e definição das respectivas permissões) até a determinação dos diretórios e disponibilização do serviço. Um *benchmark* baseado em procedimentos deve permitir responder a perguntas como: "Qual a dificuldade envolvida no procedimento de configuração do httpd, especificamente na tarefa de configurar as permissões?".

Dando seqüência ao trabalho recém mencionado, Brown *et. al* (BROWN; KELLER; HELLERSTEIN, 2005) propõem um modelo para determinar a complexidade de configuração de software e realizam um estudo de caso com os softwares DB2 e Websphere, ambos da IBM. Para tal, cada etapa da configuração é quantificada segundo um conjunto de métricas determinadas, divididas em três grupos: memória (referente à memória do próprio administrador), parâmetros e execução. Além disso, esta avaliação permite determinar a medida de complexidade associada ao procedimento como um todo.

O trabalho mais recente no que se refere à complexidade de procedimentos foi o de Diao *et. al* (DIAO, Y. et al., 2007), onde os autores, seguindo ainda o arcabouço inicial sobre complexidade de procedimentos (BROWN; HELLERSTEIN, 2004), propõem um modelo quantitativo que estabelece uma relação entre as métricas de complexidade definidas anteriormente (BROWN; KELLER; HELLERSTEIN, 2005) e métricas de nível de negócios, tais como tempo de execução e custo de mão-de-obra.

Os estudos realizados até o momento se concentram na área de configuração em geral. Entretanto, apesar do papel central que segurança ocupa, ela ainda não foi sujeita a uma investigação sistemática e minuciosa. Diante desse contexto, é fundamental que se possa determinar, de forma mais precisa, o real impacto que segurança ocasiona nos procedimentos de TI. Neste trabalho são propostas formas de se mensurar a complexidade ocasionada por segurança em diferentes dimensões. Além disso, é apresentada também uma abordagem para se obter um modelo quantitativo que permita correlacionar os valores obtidos para as métricas de complexidade com o tempo de execução de procedimentos de segurança. Este modelo, por sua vez, permite realizar previsões acerca do tempo de execução esperado para procedimentos de segurança, sem a necessidade prévia de execução dos mesmos.

Esta dissertação está organizada da seguinte forma: no Capítulo 2 são apresentados os trabalhos que fundamentam a área de análise de complexidade de procedimentos de configuração de TI. Já no Capítulo 3 é introduzido o modelo de segurança de Tecnologia de Informação e as métricas utilizadas para mensurar os níveis de complexidade. No Capítulo 4 é apresentada a ferramenta *Security Complexity Analyzer* (SCA), desenvolvida para automatizar a processo de medição da complexidade dos procedimentos de segurança, incluindo sua estrutura e os resultados produzidos por ela. Já no Capítulo 5 é apresentado um estudo de caso em que são tratados diversos cenários com diferentes requisitos de segurança e, então, é mensurado o valor de complexidade de seus respectivos procedimentos. No Capítulo 6 introduz-se o modelo quantitativo desenvolvido para estimar o tempo necessário para se executar as tarefas de segurança a partir da avaliação

de complexidade com as métricas apresentadas no Capítulo 4. Por fim, no Capítulo 7 são apresentadas as conclusões e as perspectivas de trabalhos futuros.

2 TRABALHOS RELACIONADOS

Nos últimos anos vários esforços têm sido feitos para se obter uma metodologia que (i) permita mensurar a complexidade tanto de procedimentos quanto de processos de TI e (ii) relacione esta medida de complexidade com o tempo gasto na execução dos procedimentos. Apesar do reconhecido potencial que segurança pode ter sobre os indicadores de complexidade e de tempo, este problema não foi investigado em trabalhos anteriores. Em paralelo, entretanto, diferentes perspectivas de segurança em TI têm sido analisadas. Refletindo estes esforços, este capítulo introduz os trabalhos mais proeminentes relacionados ao tema desta dissertação.

2.1 Análise de prejuízos causados por indisponibilidade

O primeiro trabalho relacionado ao tema desta dissertação é o apresentado por Patterson (PATTERSON, 2002). Neste trabalho, é proposta uma equação que permite estimar os prejuízos causados quando um sistema gerencial em uma empresa torna-se indisponível (Equação 2.1). Ao contrário de outros trabalhos, que consideravam apenas a perda de faturamento como prejuízo causado pela indisponibilidade, a equação proposta por Patterson também leva em conta o tempo que os funcionários deixam de realizar seu trabalho devido à indisponibilidade.

$$C = (CustEmp/h) * (taxaEmpAfet) + FatMedio/h * (taxaFatAfet) \qquad (2.1)$$

Nesta fórmula, o termo C representa o custo médio de 1 hora de indisponibilidade do sistema. O termo $CustEmp/h$ representa o custo de médio de um empregado por hora, que é obtido através da divisão do total gasto com salários e benefícios de todos os funcionários por semana pelo número de horas de trabalho por semana. Este termo é então multiplicado pela porcentagem de empregados afetados pela indisponibilidade ($taxaEmpAfet$) . Já o termo $FatMedio/h$ corresponde ao faturamento mensal de uma empresa dividido pelo número de horas por semana que a empresa se encontra aberta. Este termo, por sua vez, é multiplicado pelo percentual do faturamento que é devido aos serviços afetados pela indisponibilidade ($taxaFatAfetado$).

Com o uso da equação, o autor conseguiu estimar os prejuízos que a loja eletrônica Amazon.com e a fabricante de computadores e software Sun Microsystems teriam, por hora, caso seus sistemas se tornassem indisponíveis: US$ 550.000,00 e US$ 825.000,00, respectivamente. Patterson reconhece que o modelo é simplista e afirma que um modelo mais complexo, que englobe diversos outros cenários e seja muito mais difícil de calcular, não necessariamente trará uma maior precisão no cálculo do custo de indisponibilidade. Como exemplo, o autor cita que, dependendo da empresa, uma hora de indisponibilidade

pode não causar queda no faturamento, desde que os clientes tentem comprar posteriormente. Dependendo da estrutura da empresa, apenas certas partes podem ser afetadas; por exemplo, a parte de vendas pode estar disponível, enquanto outras não.

2.2 Modelo de custos para administração de sistemas

Dando prosseguimento na criação de modelos de custos, o trabalho apresentado por Couch *et al.* (COUCH; WU; SUSANTO, 2005) provê um modelo, baseado no trabalho de Patterson (PATTERSON, 2002), que prevê o custo de administração de sistemas tendo por base decisões gerenciais específicas (como, por exemplo, o custo resultante da decisão de se ter 1 ou dois administradores do sistema). Este custo pode ser visto como a soma de dois termos: o *custo de operação* e *custo de espera por mudanças*. O custo de operação engloba gastos relacionados a salários, benefícios, contratos e equipamentos em uma empresa. Esse valor é relativamente previsível e se mantém constante por períodos consideráveis. Já o custo de espera, proposto pelos autores, é uma generalização do *custo de downtime* proposto por Patterson (Equação 2.1).

Os autores então utilizam dados obtidos a partir de um sistema de *trouble-ticket* real para gerar um modelo de sistema e, através do modelo de custos, prevêr o tempo gasto com administração do mesmo, isto é, o tempo que os administradores deste sistema gastam para atender cada *ticket* que é aberto. Este sistema é modelado como um ambiente em que há c administradores de sistemas trabalhando 24 horas por dia, sete dias por semana.

Este modelo de sistema, por sua vez, é então simulado por uma ferramenta desenvolvida pelos autores. O simulador gera requisições para os administradores (variando a quantidade de administradores disponíveis para resolver as requisições, de 1 a 4), e, como saída, tem-se o tempo que eles gastam para processar as requisições. Como resultado foi determinado o tempo que n administradores necessitam para atender as requisições, ilustrado na Figura 2.1 (extraída de (COUCH; WU; SUSANTO, 2005)). Nesta figura, pode-se notar que um administrador sozinho tem um desempenho muito pior do que 2 ou mais, quando submetido a mesma taxa de requisições. Desta forma, os clientes internos são obrigado a esperar um maior tempo para solucionar suas demandas, o que ocasiona um aumento no *custo de espera por mudanças* do modelo de custos desenvolvido pelos autores.

No mesmo trabalho, foi ainda realizada outra simulação, em que o sistema recebeu uma grande quantidade de requisições (100) quando o mesmo se encontrava quase em estado de saturação. Os resultados deste teste podem ser observados na Figura 2.2 (extraída de (COUCH; WU; SUSANTO, 2005)). Nesta figura, o eixo y denota tempo incremental (Δt) entre uma requisição e outra. Pode-se notar na figura que, quando há apenas um administrador, o tempo de resposta atinge valores altos, gerando atrasos e prejuízos para empresa através do aumento no tempo de espera por mudanças. A principal contribuição deste trabalho foi constatar que, em se tratando da instalação de software em massa, muitas vezes é melhor realizar a instalação de forma planejada e manual do que de forma automatizada quando os administradores se encontram quase saturados. Segundo os autores, é melhor controlar a taxa de requisições atendendo um pequeno número de usuários de cada vez do que perder o controle e gerar uma grande quantidade de requisições não gerenciáveis, que pode ocorrer quando uma atualização automatizada em massa de um software gera problemas.

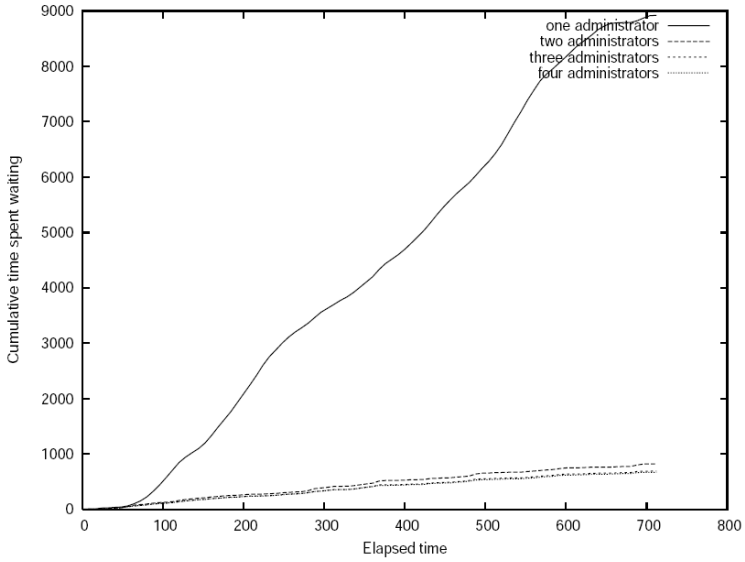

Figura 2.1: Tempo necessário para solucionar as requisições

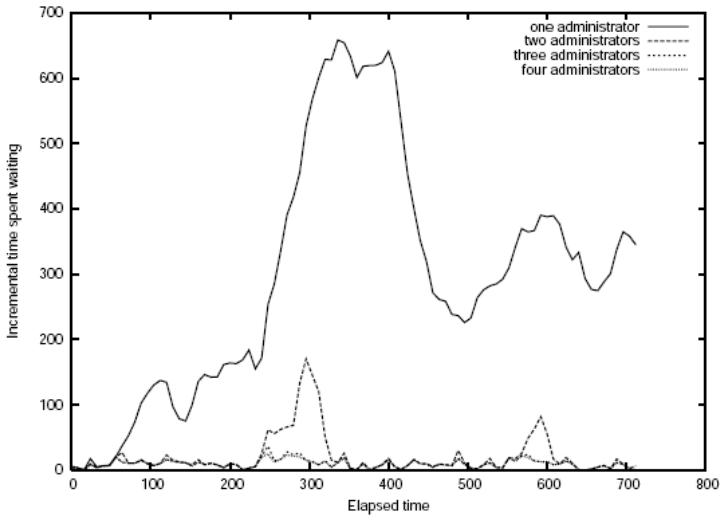

Figura 2.2: Efeito que cem requisições trazem para o sistema quando o mesmo está quase saturado

2.3 Arcabouço conceitual para obtenção de um *benchmark* de complexidade de configuração

O trabalho apresentado por Brown e Hellerstein (BROWN; HELLERSTEIN, 2004) alerta para a crescente complexidade vivenciada pelos administradores durante a configuração de soluções de softwares atuais. A magnitude deste problema é melhor ilustrada na Figura 2.3 (adaptada de (BROWN; KELLER; HELLERSTEIN, 2005)), que ilustra parte do procedimento de configuração do *SPECjAppServer* utilizando o servidor de aplicação Websphere. Para este cenário em particular, são necessárias mais de 50 ações (representadas em caixas na Figura 2.3), sendo este um dos cenários mais simples de serem configurados.

Esta complexidade associada à configuração de sistemas computacionais é tida como o maior impedimento à adoção de novos produtos na área de TI, além de contribuir para

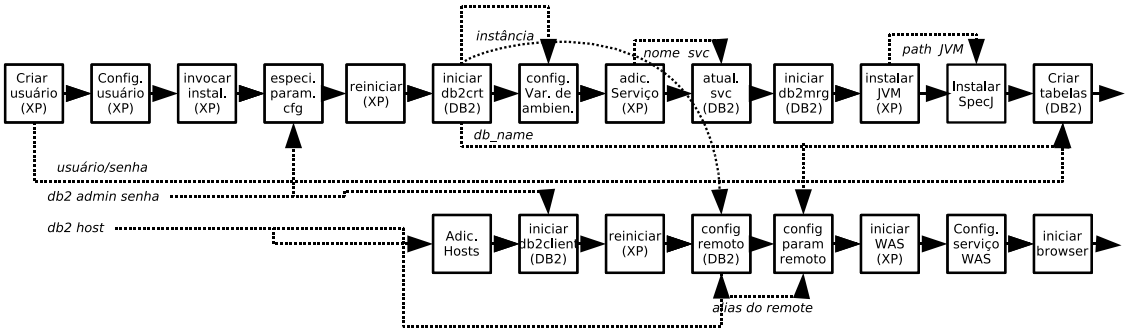

Figura 2.3: Procedimento parcial de configuração do *SPEC jAppServer*

o aumento dos custos dos serviços de TI (BROWN; KELLER; HELLERSTEIN, 2005). Neste contexto, os autores propuseram um arcabouço conceitual para se obter um *benchmark* que possibilite determinar a complexidade de procedimentos de configuração (BROWN; HELLERSTEIN, 2004). Diferentemente dos *benchmarks* orientados à carga de trabalho (*workload*) – como os produzidos pela SPEC (SPEC, 2008) – um *benchmark* de complexidade de configuração deve ser orientado a procedimento, onde as métricas produzidas devem ser capazes de abstrair detalhes relacionados ao procedimento em si e não ao tempo de resposta do sistema às ações.

Neste arcabouço, os autores sugerem que o processo de quantificação das métricas seja feito em quatro etapas, definidas a seguir.

1. **Captura do procedimento**: o procedimento de configuração é executado e as etapas são documentadas em um arquivo de resposta (*response file*).

2. **Decomposição do procedimento e análise**: as etapas de interação descritas no arquivo de respostas são mapeadas em ações (como as “caixas” na Figura 2.3). No fim desta etapa, gera-se o procedimento, que é o fluxo seqüencial das ações.

3. **Quantificação do procedimento**: o procedimento, então, é quantificado de acordo com um conjunto de métricas previamente definidas, e a relação entre estas métricas e o tempo gasto para a execução de cada ação é determinada estatisticamente através de um modelo quantitativo.

4. **Validação do procedimento**: o *benchmark* verifica se o procedimento em questão foi capaz de atingir os objetivos especificados inicialmente, através de requisitos de qualidade previamente estabelecidos. Por exemplo, pode ser incluído como requisito de qualidade de um servidor Web uma faixa de valores para vazão e latência.

Este arcabouço foi, então, explorado em trabalhos posteriores, abordado nas próximas seções.

2.4 Complexidade de procedimentos de configuração

Dando continuidade ao trabalho descrito na Seção 2.3, Brown *et. al* (BROWN; KELLER; HELLERSTEIN, 2005) propõem um modelo que permite capturar e quantificar a complexidade de um procedimento de configuração de TI (uma representação de um procedimento parcial de configuração pode ser observado na Figura 2.3). Além do modelo,

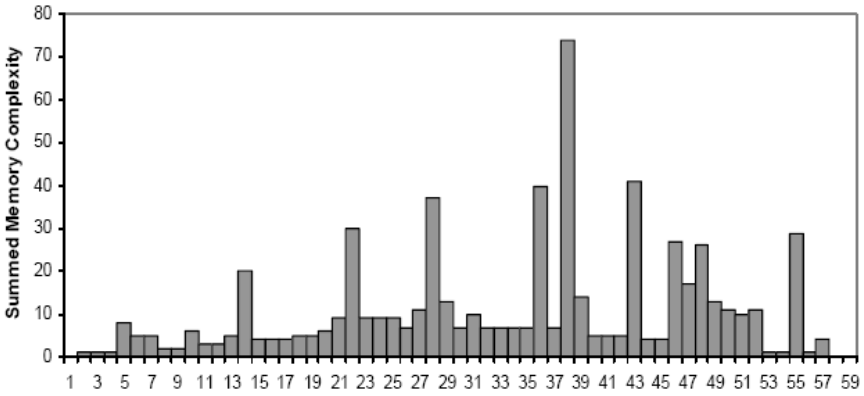

Figura 2.4: Histograma de complexidade de ações

os autores propõem três categorias de métricas para avaliar a complexidade relacionada aos procedimentos de configuração:

- complexidade de execução: se refere à complexidade relacionada à execução das ações de configuração que fazem parte do procedimento, caracterizada tipicamente pela quantidade de ações e o número de trocas de contexto entre as ações;

- complexidade de parâmetros: trata da complexidade envolvida no fornecimento de informações durante a execução do procedimento de configuração;

- complexidade de memória: contabiliza o número de parâmetros que devem ser lembrados e a quantidade de tempo que eles devem permanecer na memória do administrador.

Este modelo é aplicado na avaliação de complexidade do procedimento de configuração de uma ferramenta baseada em J2EE. Os resultados desta avaliação, referentes às métricas de memória, são exibidos na forma de um histograma na Figura 2.4 (extraída de (BROWN; KELLER; HELLERSTEIN, 2005)). Nesta figura, o eixo *x* representa o número de cada ação do procedimento enquanto o eixo *y* representa o grau de complexidade de cada ação.

A partir desse histograma, é possível notar quais ações mais contribuem para a complexidade total do procedimento (representadas pelas barras mais elevadas). Essas ações são potenciais candidatas a serem automatizadas ou serem redefinidas a fim de diminuir sua complexidade. Os autores ainda destacam como a automatização de algumas ações em um sistema de gerenciamento de mudanças pode reduzir significativamente a complexidade total.

Por fim, os autores afirmam que esses resultados servem como um ponto inicial para futuras pesquisas. O objetivo final, segundo os mesmos, é produzir um arcabouço para realizar um *benchmark* de configuração de complexidade de sistemas heterogêneos e autoconfiguráveis. Como trabalho futuro, é sugerido desenvolver modelos de complexidade de configuração mais efetivos e produzir tecnologias que permitam reduzir a complexidade apontada por tais modelos. Em um trabalho posterior, Keller *et. al* (KELLER; BROWN; HELLERSTEIN, 2007) descrevem uma ferramenta desenvolvida para automatizar a análise de complexidade dos procedimentos de configuração.

2.5 Modelo para previsão de custos de procedimentos de configuração

Dando prosseguimento aos trabalhos apresentados nas seções 2.3 e 2.4 , Diao *et. al* (DIAO, Y. et al., 2007) propuseram um modelo quantitativo onde as métricas apresentadas no trabalho recém mencionado (BROWN; KELLER; HELLERSTEIN, 2005) são utilizadas como entrada para a criação de um modelo quantitativo. Este modelo tem por objetivo relacionar métricas de desempenho no nível de negócios de um procedimento, como tempo e custos de execução, com as métricas de complexidade.

Os autores afirmam que tal abordagem traz vários benefícios como o uso do modelo para ajudar a se determinar o retorno do investimento (*return-on-investiment* – ROI) tanto antes quanto depois da execução do procedimento, bem como utilizar o modelo para realizar previsões sobre o custo de execução do procedimento. Além disso, um modelo calibrado pode revelar quais os fatores que têm um maior impacto na complexidade do procedimento.

Neste trabalho, os autores propõem uma seqüência composta de quatro etapas para se obter o modelo quantitativo, descritas a seguir.

1. **Coleta das métricas de complexidade:** o procedimento é executado por um administrador que define as ações e documenta os parâmetros utilizados ao longo da execução. A partir desses dados é então feita a análise de complexidade com as métricas de complexidade propostas por Brown *et. al* (BROWN; KELLER; HELLERSTEIN, 2005).

2. **Medição das métricas de desempenho de nível de negócios:** durante a execução do procedimento, o tempo gasto na execução de cada ação é, então, documentado.

3. **Construção do modelo através da calibração:** as métricas de complexidade e as métricas de nível de negócios são correlacionadas na forma de um modelo de regressão linear múltipla onde a variável *tempo* é explicada pelos valores das métricas de complexidade de cada ação.

4. **Extrapolação do modelo para se realizar previsões de métricas de nível de negócios:** o modelo criado é utilizado para realizar previsões de tempo de execução (e custos) de versões modificadas do procedimento original (como, por exemplo, de uma variação onde determinadas ações são automatizadas).

Os autores então realizam um estudo de caso onde são avaliados diversos procedimentos de configuração ligeiramente diferentes entre si. Além disso, a experiência de cada administrador em executar o procedimento (visto que isto é um fator que altera significativamente o tempo gasto durante a execução de cada ação) é também avaliada. Como resultado, o modelo gerado pelos autores foi capaz de previr o tempo de execução associado a ações desses procedimentos ligeiramente diferentes. Essa previsão foi então confrontada com o tempo real gasto pelos administradores na execução do experimento, e foi possível observar que o modelo foi capaz de explicar uma parte significante da variabilidade do tempo observado.

2.6 Complexidade de processos de tecnologia da informação

Dando prosseguimento ao trabalho mencionado na Seção 2.4, Diao e Keller propõem uma extensão ao modelo inicial (proposto por Brown *et. al* (BROWN; KELLER; HEL-

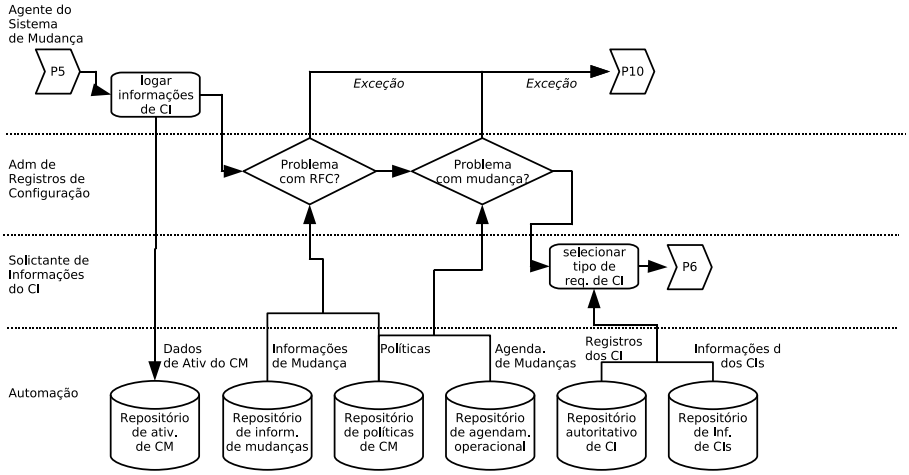

Figura 2.5: Exemplo parcial de um processo de mudança

LERSTEIN, 2005)) para estimar a complexidade de processos de TI (DIAO; KELLER, 2006). Diferentemente de procedimentos (mais relacionados ao nível operacional – incluindo instalação, configuração, etc.), os processos de TI são relacionados ao nível de operações de negócios. Um exemplo parcial de um processo pode ser visto na Figura 2.5 (extraída de (DIAO; KELLER, 2006)). Pode-se notar a diferença deste processo com o procedimento ilustrado na Figura 2.3:o primeiro se caracteriza por estrutura de decisões e ações em um nível mais elevado, não relacionado à configuração de um dispositivo, enquanto o último é composto por caixas que representam ações que o administrador deve executar para atingir uma configuração.

Processos, com freqüência, são modelados segundo recomendações de melhores práticas, tais como o ITIL (CANNON; WHEELDON, 2007). A partir destas recomendações, as empresas têm buscado automatizar suas operações tendo por base processos como uma forma de conter ou mesmo reduzir os custos com o gerenciamento de sistemas. Com esta abordagem, os custos dos processos só podem ser determinados depois que os mesmos são implantados. Neste contexto, Diao e Keller (DIAO; KELLER, 2006) propõem um modelo que ajuda a estimar a complexidade e o custo de capital humano para executar o gerenciamento de serviços de TI, sem que haja a necessidade de execução prévia dos processos associados aos mesmos. A diferença deste trabalho com o descrito na Seção 2.4 reside no fato de que o modelo anterior fornece métricas para capturar a interação entre o administrador de sistemas e o sistema em si em tempo de execução, enquanto o novo modelo foi estendido para prover uma estimativa de complexidade de gerenciamento de TI que consegue abordar:

- interação entre 2 ou mais papéis em um processo (isto é, responsáveis);
- troca de dados em diversos formatos (isto é, *Business Itens* – BI) entre ações; e
- tomada de decisão entre diversos papéis.

Para atingir este objetivo, são definidas três categorias de métricas: complexidade de execução, complexidade de coordenação e complexidade de item de negócio. A Tabela 2.1 sumariza estas métricas.

A concretização da proposta começa com a modelagem de um processo utilizando o software IBM Websphere Business Modeler. O modelo de processo é exportado em

Tabela 2.1: Métricas de complexidade de processos

Categoria	*Complexidade*	*Definição*
Complexidade de execução	Execução base	Indica a complexidade de cada ação de acordo com o tipo de execução. Ações automatizadas tem valor menor que as manuais.
	Decisão	Determina a complexidade associada a cada tomada de decisão em um processo. Ações sem decisões possuem valor zero. Caso haja diversas opções de execução numa ação, ela terá um valor maior.
Complexidade de coordenação	Coordenação de *link*	Indica a complexidade associada à coordenação efetuada por vários papéis.
	Ação compartilhada	Estima a complexidade de ações associadas a vários papéis.
Complexidade de item de negócio	Base de BI	É o produto entre a quantidade *business items* e papéis envolvidos em uma ação.
	Fonte de BI	Relaciona a complexidade de cada BI com a sua origem.

formato *XML Metadata Interchange* (XMI) para o *Integrated Complexity Analyzer*, que é responsável por mensurar a complexidade dos processos. Os resultados desta etapa podem ser sumarizados em um histograma, como o ilustrado na Figura 2.6 (extraída de (DIAO; KELLER, 2006)). Nesta figura, os resultados da avaliação de complexidade estão agrupados pela categoria de complexidade (como a primeira coluna da Tabela 2.1) para cada ação em particular. Pode-se observar na mesma figura que as ações de maior complexidade são a 10 e a 11 – e são, então, classificadas como ações candidatas a sofrerem mudanças, com o objetivo de diminuir a medida de complexidade total do processo.

Como benefícios desta abordagem, pode-se mencionar que esta análise de complexidade revela quais são as atividades dos processos de TI que possuem maior custo. A partir disso, essas atividades podem ser automatizadas ou redefinidas. Além disso, o modelo permite comparar a complexidade antes das mudanças e depois, destacando desta forma a redução da complexidade. Por fim, este modelo pode ser aplicado a processos já implantados e obter os mesmos benefícios.

2.7 Considerações parciais

Nos últimos anos vários esforços científicos foram conduzidos para que se pudesse chegar ao atual estágio de análise de complexidade de procedimentos e processos de TI. Apesar do impacto potencial reconhecido que segurança acarreta nos indicadores de complexidade, esta linha não foi investigada anteriormente a este trabalho. Em paralelo, entretanto, diversas perspectivas de complexidade e modelos de custos foram propostas. Refletindo estes desenvolvimentos, este capítulo apresentou as investigações mais proe-

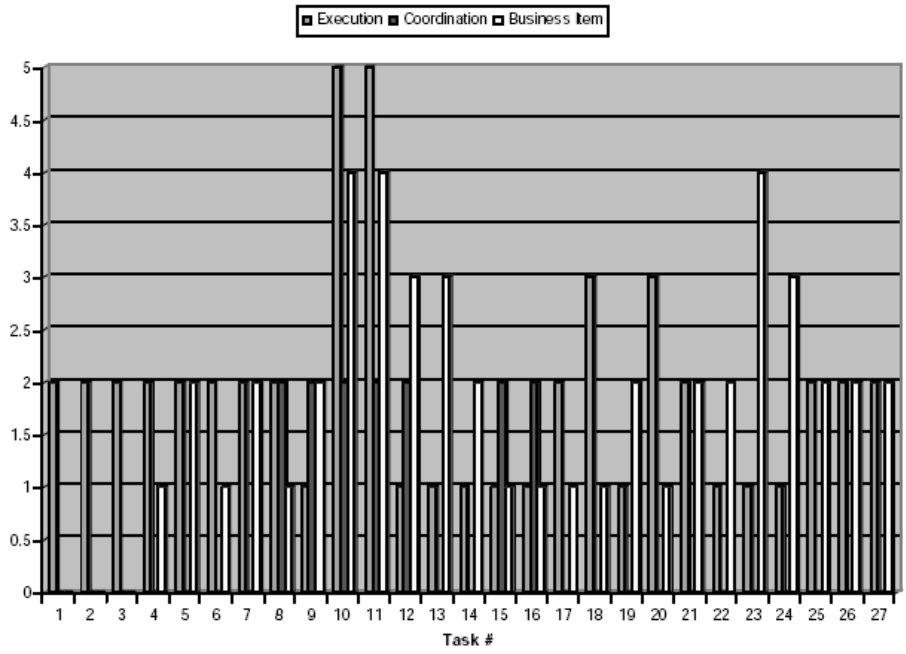

Figura 2.6: Histograma de complexidade de processo

minentes em se tratando de modelos de custos de administração de sistemas e complexidade de processos/procedimentos mais gerais de TI.

O trabalho de Patterson, apresentado na Seção 2.1, aborda os custos ocasionados pela indisponibilidade dos serviços em uma infra-estrutura de TI, enquanto o de Couch *et. al*, apresentado na Seção 2.2, trata de um modelo de operação e atendimento em um sistema de *trouble ticket*. Brown e Hellerstein (BROWN; HELLERSTEIN, 2004) propuseram um arcabouço conceitual para se obter um *benchmark* de complexidade de configuração, que foi investigado por estudos subseqüentes. No primeiro deles – desenvolvido por Brown *et al.* (BROWN; KELLER; HELLERSTEIN, 2005) – foi proposto um conjunto de métricas, inspiradas na área de engenharia de software (ZUSE, 1991), para abstrair a complexidade de procedimentos de configuração. Os autores demonstraram que este conjunto de métricas foi capaz de capturar a diminuição da complexidade associada à procedimentos de configuração depois que algumas das ações dos mesmos foram automatizadas. Um passo adiante foi dado por Diao *et al.* (DIAO, Y. et al., 2007), onde as métricas propostas no trabalho de Brown *et al.* (BROWN; KELLER; HELLERSTEIN, 2005) foram utilizadas como entrada para a construção de um modelo quantitativo para derivar métricas de desempenho em nível de negócios, como custos e tempo de trabalho. Por fim, Keller *et al.* (KELLER; BROWN; HELLERSTEIN, 2007) descreveram a ferramenta utilizada para ajudar na captura e análise de complexidade dos dados coletados.

Tratando do problema de complexidade a partir de uma perspectiva de negócios, Diao e Keller (DIAO; KELLER, 2006) estenderam as métricas propostas em (BROWN; KELLER; HELLERSTEIN, 2005) para ser possível quantificar a complexidade dos processos de gerenciamento de serviços de TI (CANNON; WHEELDON, 2007). Diferentemente dos procedimentos, que são diretamente relacionados ao nível operacional (por exemplo, instalação, configuração e manutenção de um CI), processos são relacionados em nível de negócios. Neste contexto, as novas métricas capturam aspectos como tomada de decisões entre múltiplos cargos (ou *roles*) e a interação entre dois ou mais cargos.

Em relação à segurança de TI, não há trabalhos conhecidos que relacionem as me-

didas de complexidade para estimar o quanto os procedimentos podem ser afetados pela manipulação de diferentes mecanismos de segurança. Ao contrário disto, por exemplo, os *benchmarks* CIS (CIS, 2008) são destinados a determinar o grau de proteção de um sistema já em operação em relação a uma extensa lista de vírus, *worms* e vulnerabilidades. Em relação a aspectos econômicos, Cavusoglu *et al.* (CAVUSOGLU; MISHRA; RAGHUNATHAN, 2004) propuseram um modelo a fim de estimar a quantidade ótima a ser investida em segurança, tendo como entrada, entre outras variáveis, os prejuízos esperados no caso de haver falhas de segurança, bem como parâmetros de qualidade e custo de instalação e configuração dos mecanismos de segurança. O raciocínio por trás dos "custos" de instalação e configuração dos mecanismos de segurança – problema que poderia comparado ao tema desta dissertação – não é abordado pelo autor.

Nos próximos capítulos é descrita uma proposta para mensurar a complexidade de segurança nos procedimentos relacionados a TI. Além disso, é proposto um modelo quantitativo que permite relacionar as métricas de complexidade com métricas de negócio, como tempo de execução.

3 MODELO DE SEGURANÇA DE TECNOLOGIA DE INFORMAÇÃO E MÉTRICAS DE COMPLEXIDADE

Neste capítulo são apresentados o modelo de Segurança de Tecnologia da Informação e as métricas de complexidade de procedimentos. O modelo fornece o substrato necessário para que as métricas possam ser efetivamente utilizadas na abstração da complexidade dos procedimentos relacionados à segurança de TI. Para tal, ambos devem ser aplicados em conjunto, conforme descrito neste capítulo.

3.1 Modelo de Segurança de TI

Para mensurar objetivamente complexidade de segurança em procedimentos de TI é fundamental ter acesso a uma visão precisa da infra-estrutura e, sobretudo, dos procedimentos sob análise. Nesta seção é apresentado um modelo de segurança de TI que serve justamente a esse propósito.

Mais especificamente, o modelo de segurança de TI permite expressar uma abstração da infra-estrutura de TI da corporação – incluindo os mecanismos de segurança – a ser implantada ou modificada, bem como os procedimentos necessários para atingir tais metas. Baseado na proposta de Brown *et al.* (BROWN; KELLER; HELLERSTEIN, 2005), este modelo é composto de duas partes complementares: o *modelo de infra-estrutura* e o *modelo de atividade*. O primeiro representa os componentes de hardware e software da infra-estrutura (CIs) e o grau em que eles são relacionados aos serviços de segurança. Já o segundo abstrai a seqüência de atividades que devem ser executadas a fim de atingir um objetivo em particular.

Para ilustrar este modelo, apresenta-se um cenário exemplo que consiste na instalação e configuração da aplicação Web Joomla (JOOMLA, 2008) – que é um sistema de gerenciamento de portais desenvolvido em PHP – e todo o software necessário para que ele seja executado (por exemplo, o sistema gerenciador de banco de dados MySQL e o servidor Web Apache httpd). O cenário é composto por dois computadores (um hospeda o servidor de banco de dados e o outro, o servidor Web) e um *firewall*, que filtra os pacotes destinados aos computadores. Além disso, o ambiente é incrementado com um conjunto de mecanismos de segurança, sumarizados na Tabela 3.1. Por exemplo, Dm-crypt (DM-CRYPT, 2008) é empregado para criptografar o conteúdo do sistema de arquivos do computador 1 e o OpenSSL (OPENSSL, 2008) é utilizado para criptografar as conexões com o banco servidor Web. As duas partes são apresentadas a seguir.

Tabela 3.1: Mecanismos de segurança utilizados no cenário exemplo

Computador 1	
Mecanismo de Segurança	*Ferramenta*
Criptografia de sistema de arquivos	dm-crypt e OpenSSL
Criptografia da conexão com o banco de dados	MySQL e OpenSSL

Computador 2	
Mecanismo de Segurança	*Ferramenta*
Criptografia de sistema de arquivos	dm-crypt e OpenSSL
Criptografia da conexão com o servidor Web	httpd e OpenSSL

Firewall	
Mecanismo de Segurança	*Ferramenta*
Filtro de Pacotes	netfilter/iptables

3.1.1 Modelo de infra-estrutura

A infra-estrutura de TI pode ser modelada como um conjunto de contêineres, que representam recursos do sistema ou contêineres hospedeiros. A Figura 3.1 ilustra o modelo de infra-estrutura de TI para o cenário exemplo recém mencionado. O contêiner `Computador 1` é uma representação do computador onde o servidor de banco de dados é executado, enquanto o contêiner `Computador 2` representa o servidor que hospeda a aplicação Web. Por sua vez, estes contêineres hospedam outros contêineres, como `Slackware_app` e `Slackware_bd` (sistema operacional de ambos os computadores). Pode-se notar que são utilizadas linhas pontilhadas e sólidas na figura, cuja finalidade é diferenciar os contêineres hospedeiros daqueles que representam recursos do sistema (como, por exemplo, `MySQL tables`).

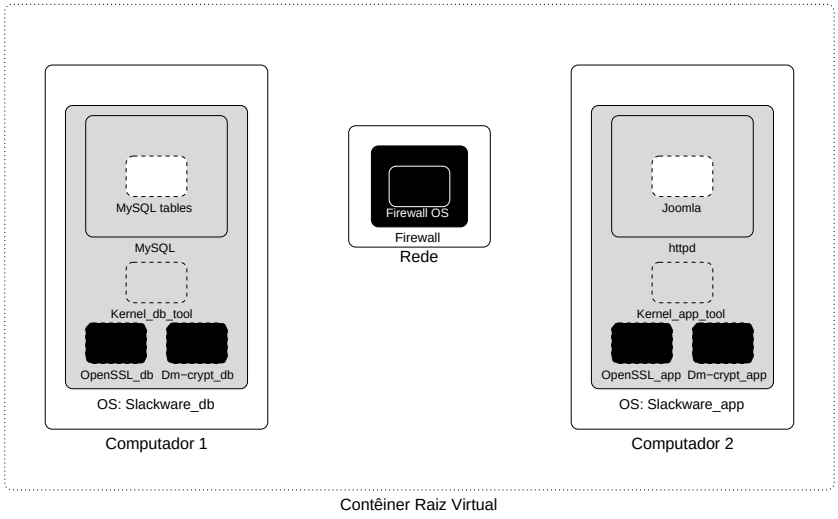

Figura 3.1: Modelo de infra-estrutura de TI

Na mesma figura, a cor dos contêineres reflete como eles se relacionam com segurança. Na verdade, esta proposta é uma extensão do modelo de infra-estrutura de TI proposto em (BROWN; KELLER; HELLERSTEIN, 2005). Contêineres em preto são

aqueles cuja funcionalidade principal é diretamente relacionada com os mecanismos de segurança (por exemplo, o contêiner `Dm-crypt_db`, responsável por criptografar o sistema de arquivos do computador hospedeiro do banco de dados). Em contraste, contêineres representados em branco não são afetados quando mecanismos de segurança são empregados (por exemplo, `MySQL tables`). Por fim, contêineres em cinza caracterizam CIs que, apesar de não serem relacionados diretamente com segurança, podem necessitar de alterações para que os mecanismos de segurança demandados sejam suportados. Por exemplo, o servidor de banco de dados MySQL requer uma configuração especial para suportar o uso de criptografia em suas conexões (com o uso de OpenSSL).

3.1.2 Modelo de atividades

O modelo de atividade provê uma abstração para a interação entre o administrador e o sistema a ser manipulado, e é baseado em três pilares: *objetivos*, *procedimentos* e *ações* (ou *tarefas*). Objetivo pode ser definido como o estado final a ser atingido pela infra-estrutura de TI (como, por exemplo, a ativação do protocolo SSL para comunicações com o servidor Web). Por sua vez, procedimento pode ser definido como uma seqüência de passos a serem seguidos para se atingir um objetivo específico. Por fim, uma ação denota um passo individual em um procedimento (como, por exemplo, a criação de uma chave privada).

A Figura 3.2 ilustra o procedimento para instalar a aplicação Web Joomla com os mecanismos de segurança especificados na Tabela 3.1 . Ele é composto de 77 ações (representadas por retângulos). Cada ação é identificada por um título e o contêiner em que ela é executada, sendo que este último é indicado pelo uso de parênteses. Os parâmetros consumidos por uma ação são destacados por setas pontilhadas que se ligam às respectivas ações. Por exemplo, a ação de número 1 possui o título “Select kernel”, seu contêiner é o `Slackware_db` e ela consome o parâmetro `kernel_db`.

As ações são representadas em preto, branco e cinza, seguindo a mesma semântica utilizada no modelo de infra-estrutura de TI. Contêineres pretos e brancos são expandidos em ações pretas e brancas, respectivamente. Contêineres cinza são expandidos em ações cinzas e brancas, em que as primeiras representam ações que precisaram ser adicionadas ao procedimento para dar suporte aos mecanismos de segurança. Por exemplo, o sub-procedimento referente à instalação do contêiner `Slackware_db` (representado em cinza no modelo de infra-estrutura de TI, na Figura 3.1) é composto de ações relacionadas aos mecanismos de segurança (tais como as ações 16 a 22 e de 25 a 27) e ações ordinárias (como, por exemplo, as de 1 a 15). Em um primeiro momento, poderia-se questionar o porquê da ação de número 73 na Figura 3.2 não ser representada em preto, uma vez que é nela que são informadas as credenciais para autenticação (`db_name` e `db_passwd`). A razão para isso é que esta ação não é relacionada à implantação de quaisquer mecanismos de segurança especificados na Tabela 3.1. Além disso, ela não é uma ação opcional, uma vez que necessita ser obrigatoriamente executada para que se possa instalar o Joomla – com ou sem os mecanismos de segurança.

3.2 Métricas de complexidade

O modelo apresentado na sub-seção anterior deve ser aplicado com um conjunto de métricas capaz de capturar a complexidade associada aos procedimentos abstraídos, relacionados à TI. As métricas utilizadas neste trabalho foram propostas por *Brown et al.* (BROWN; KELLER; HELLERSTEIN, 2005), que aplicaram as mesmas para capturar a

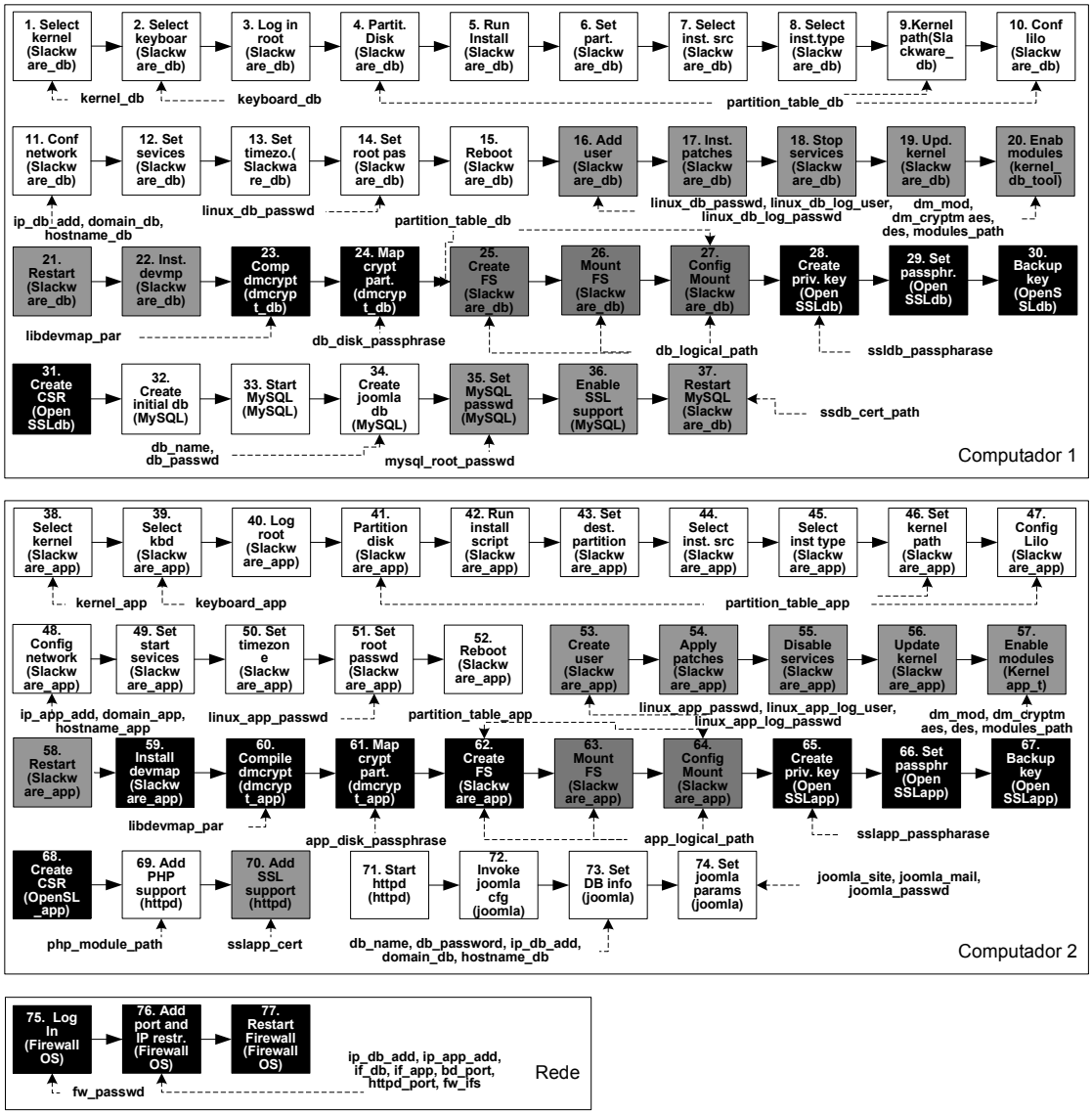

Figura 3.2: Procedimento completo para instalar Joomla com mecanismos de segurança

complexidade de procedimentos gerais de configuração. Em uma primeira tentativa em se determinar a complexidade associada aos mecanismos de segurança, não foram propostas novas métricas. Ao invés disso, foram aplicadas aquelas utilizadas com sucesso no contexto dos procedimentos de configuração mais gerais.

As métricas de complexidade empregadas neste trabalho são divididas em três grupos: *execução*, *parâmetro* e *memória*. Essas métricas podem ser empregadas para realizar o cálculo de complexidade tanto no nível de *procedimentos* quanto no nível de *ações*. Mais detalhes sobre o uso das métricas de complexidade em cada nível são apresentados nas próximas sub-seções.

3.2.1 Métricas em nível de procedimentos

Esta subseção apresenta as métricas de complexidades utilizadas para capturar a complexidade dos procedimentos. A avaliação de complexidade neste nível permite obter um panorama geral da complexidade associada aos respectivos procedimentos.

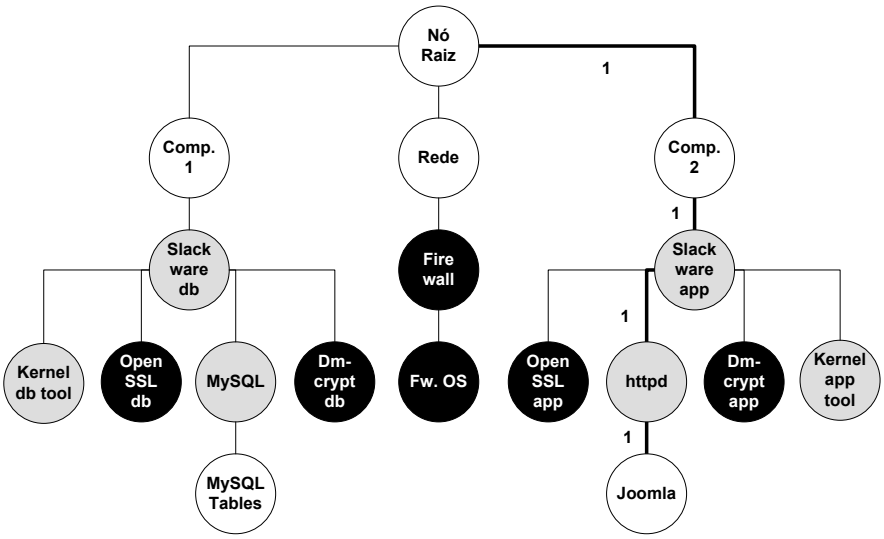

Figura 3.3: Árvore de hierarquia de contêineres.

3.2.1.1 Complexidade de Execução

As métricas pertencentes ao grupo de execução têm por função capturar a complexidade relacionada ao ato de executar a seqüência de ações definidas nos procedimentos. Elas são descritas a seguir.

- *NumActions*: contabiliza o número de ações no procedimento. Para o procedimento da Figura 3.2, este valor é 77.
- *ContextSwitchSum*: efetua o somatório dos valores associados a todas as trocas de contextos *(ContextSwitch*) presentes em um procedimento. Uma troca de contexto ocorre quando um contêiner de uma ação difere do contêiner da ação anterior, assim como entre as ações 74 e 75 da Figura 3.2. O valor desta troca de contexto é calculado através da distância entre o contêiner da ação de origem e o contêiner raiz que o interliga com a ação de destino na *árvore hierárquica de coitêineres*. Essa árvore é derivada diretamente a partir do modelo de Infra-estrutura de TI ilustrado na Figura 3.1. A Figura 3.3 representa a árvore hierárquica referente ao cenário usado como exemplo. Pode-se observar, na mesma figura, que o valor da troca de contexto entre as ações 74 e 75 é 4.

3.2.1.2 Complexidade de parâmetros

As métricas de complexidade definidas no grupo dos parâmetros são utilizadas para mensurar a complexidade associada ao fornecimento de parâmetros durante a execução do procedimento (seja pelo operador humano ou pelo sistema). As métricas são definidas a seguir.

- *ParamCount*: contabiliza o número de parâmetros distintos utilizados durante a execução do procedimento. O procedimento da Figura 3.2 apresenta o valor de 46 para esta métrica.
- *ParamUseCount*: contabiliza o número de parâmetros, incluindo as repetições, usados na execução do procedimento. No caso do procedimento usado como exemplo,

Tabela 3.2: Valores de *SourceScore* para avaliação dos parâmetros

Valor	*Propriedade do parâmetro*
0	Sugerido - preenchido automaticamente
1	Seleção (por exemplo, um *combo-box*)
2	Não sugerido - no mesmo contêiner - documentado
3	Não sugerido - no mesmo contêiner - não documentado
4	Não sugerido - em outro contêiner - documentado
5	Não sugerido - em outro contêiner - não documentado
6	Não sugerido - fora do ambiente

esta métrica acumula o valor 72, ou seja, há 26 repetições de parâmetros no procedimento.

- *ParamCrossContext*: contabiliza o somatório das ocorrências de trocas de contextos associadas a ações, pertencentes a diferentes contêineres, que consomem o mesmo parâmetro. Por exemplo, na Figura 3.2, o parâmetro `db_password` é utilizado tanto na ação 34 quanto na ação 73.

- *ParamAdaptCount*: sumariza o número de parâmetros utilizados de forma sintática diferente ao longo do procedimento (como, por exemplo, o caminho absoluto e relativo do *script* de inicialização do httpd).

- *ParamSourceScore*: efetua o somatório das medidas de *SourceScore* associadas a cada parâmetro distinto utilizado no procedimento. *SourceScore* reflete a dificuldade de se determinar os valores dos parâmetros, variando de 0 a 6. Neste trabalho é proposta uma forma de quantificar esta métrica, de acordo com as regras apresentadas na Tabela 3.2. Por exemplo, o parâmetro da ação `php_module_path` possui um valor de 6, já que é obtido fora do contexto em questão. Para o procedimento da Figura 3.2, o valor calculado para a métrica *ParamSourceScore* é 110.

3.2.1.3 Complexidade de memória

O grupo de métricas de memória é utilizado para avaliar o número de parâmetros que devem ser lembrados durante a execução do procedimento, bem como o intervalo que eles necessitam permanecer armazenados em memória. Para realizar esta quantificação, a memória do administrador é modelada como uma pilha *Last-In-First-Out* (LIFO) com busca não-associativa (apesar de se utilizar neste trabalho o termo pilha, é importante destacar que ela se difere da tradicional estrutura de dados pilha, uma vez que os elementos podem ser removidos em qualquer posição que eles estejam, e não apenas do topo. Entretanto, o processo de adição de elementos é idêntico ao de uma pilha tradicional). Seis métricas associadas à complexidade de memória são empregadas no contexto deste trabalho:

- *MemSizeAverage* e *MemSizeMax*: representam os valores médio e máximo relativos ao tamanho da pilha durante a execução do procedimento. No caso do procedimento exemplo, os valores para estas métricas são, respectivamente, 7,38 e 14.

- *MemDepthAverage* e *MemDepthMax*: se referem aos valores médio e máximo relativos à profundidade dos elementos acessados na pilha durante a execução do

procedimento. Os valores destas métricas no caso do procedimento exemplo são de 1,2 e 40, respectivamente.

- *MemLatAverage* e *MemLatMax*: são relacionadas à latência média e máxima dos parâmetros buscados na pilha. Latência indica o intervalo, medido em número de ações, que um parâmetro precisa ser retido na memória do administrador antes de ser utilizado novamente numa ação subseqüente. Para o procedimento em questão, os resultados são 7,38 e 264, nesta ordem.

3.2.2 Métricas em nível de ações

Para determinar a complexidade no nível das ações é necessário que o cálculo de alguma das métricas previamente apresentados seja levemente modificado. Em se tratando das métricas que avaliam a complexidade de execução, *NumActions* sempre recebe 1 como valor, enquanto *ContextSwitchSum* recebe o valor de *ContextSwitch* da ação em relação à sua predecessora.

As métricas relacionadas à complexidade de parâmetros também são ligeiramente diferenciadas. *ParamCount* não é aplicado no contexto de ações, enquanto *ParamUseCount* enumera o número de parâmetros que a ação em questão consume. *ParamAdaptCount* contabiliza o número de parâmetros que uma ação reutiliza de forma sintática diferente. Já *ParamSourceScore* contabiliza a soma dos valores de *SourceScore* relacionados aos parâmetros consumidos pela ação que aparece no procedimento pela primeira vez. Por fim, as métricas do grupo de memória (*MemSize*, *MemDepth* e *MemLat*) já são calculadas em relação a ações e, desta forma, não requerem nenhuma mudança. Os valores médios e máximos destas métricas não são aplicados a ações.

4 FERRAMENTA *SECURITY COMPLEXITY ANALYZER*

Durante este trabalho foi desenvolvida uma ferramenta para auxiliar no processo de captura e quantificação de complexidade dos procedimentos de TI relacionados à segurança, denominada *Security Complexity Analyzer* (SCA). Neste capítulo apresenta-se a arquitetura da ferramenta e como pode ela ser utilizada para automatizar o processo de análise de complexidade.

4.1 Arquitetura

A arquitetura da SCA é baseada no modelo de arquitetura de três camadas, em que funcionalidades similares são agrupadas nas camadas de apresentação, negócios/aplicação e de dados. O principal benefício desta abordagem reside na independência das camadas: uma alteração em uma camada não requer que sejam feitas alterações nas demais. Estas camadas são definidas a seguir.

- *Camada de apresentação*: parte responsável pela interação com o usuário, em que os dados relativos aos procedimentos, tais como ações, contêineres e parâmetros são capturados. Além disso, os resultados gerados pela SCA são exibidos nesta camada.

- *Camada de negócios/aplicação*: parte dedicada ao processamento dos dados inseridos pelo usuário e pela análise de complexidade dos procedimentos.

- *Camada de dados*: parte responsável por armazenar e consultar os dados provenientes das análises efetuadas pela camada de aplicação.

Durante o andamento deste trabalho, foram realizadas a modelagem e a implementação da camada de negócios da SCA, a parte primordial para a análise dos dados. Em relação às camadas de apresentação e de dados, apenas protótipos foram implementados. Entretanto, devido às características da arquitetura, o desenvolvimento dessas camadas pode ser realizado desacopladamente da camada de negócios – o que foi deixado como trabalho futuro. Nas próximas seções são descritos os componentes da SCA.

4.2 Componentes da SCA

A Figura 4.1 ilustra os componentes internos previstos para a SCA, bem como a relação entre eles e o administrador. Os componentes *GUI/XML editor* e *Viewer* compõem a camada de apresentação, enquanto *Parser* e *Complexity Scorer* perfazem a camada de

negócios. Por fim, o componente *Security Complexity DB* representa a camada de dados, conforme definido na seção anterior.

Enquanto executa os procedimentos de segurança, o administrador é capaz de interagir paralelamente com a ferramenta, através dos componentes *GUI* ou *XML editor*. Durante esta interação, são cadastradas todas as ações, contêineres e parâmetros que precisam ser inseridos ou são produzidos pelo sistema (e os classificando com base no sistema de pesos proposto na Tabela 3.2) – como pode ser visto na fase 1 da figura. Além disso, a ferramenta pode ser utilizada em modo "*live*", em que o sistema pode capturar o tempo real gasto na execução de cada ação – dados que são essenciais para a criação dos modelos quantitativos, tópico que é abordado no Capítulo 6.

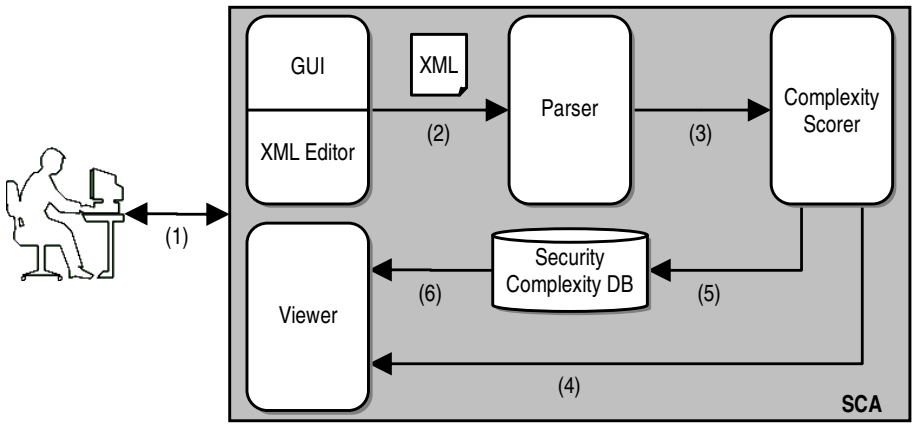

Figura 4.1: Componentes internos do SCA

Os dados de complexidade que o administrador insere no sistema através da *Graphical User Interface* (GUI) ou do editor *Extended Markup Language* (XML) são gravados pela SCA num esquema XML (fase 2 na Figura 4.1). A próxima fase consiste no *parsing* do arquivo XML produzido e então na criação de estrutura de dados conveniente para a análise de complexidade. O cálculo da complexidade para cada uma das métricas métricas (fase 3) é feito pelo componente *Complexity Scorer*, conforme descrito na Seção 3.2.

Em seguida, os dados tomam dois caminhos: eles podem ser exibidos diretamente ao usuário (fase 4) ou serem armazenados em banco de dados (fase 5) e, então, serem entregues à camada de apresentação e exibidos para o usuário. Como os resultados das avaliações de complexidade são persistidos pela SCA em banco de dados, é possível para o administrador comparar diferentes cenários de segurança. Através de tal funcionalidade (fase 6), ele pode, por exemplo, estimar o impacto do uso de diferentes combinações de mecanismos de segurança nas operações diárias ou decidir qual ferramenta usar para implementar um mecanismo de segurança específico, baseado nas medidas de complexidade de instalação, configuração e manutenção de cada um. Além disso, a SCA pode ser utilizada para destacar as ações mais complexas de um procedimentos fornecendo uma diretriz sobre quais ações seria razoável automatizar para reduzir a complexidade total.

Na próxima seção são descritos os detalhes pertinentes à implementação da ferramenta, bem como as tecnologias e ferramentas utilizadas.

4.3 Implementação da SCA

A ferramenta SCA foi desenvolvida em ambiente Linux, utilizando a linguagem de programação Java versão 5.0. A IDE utilizada durante o desenvolvimento foi o Eclipse (ECLIPSE, 2008), e a biblioteca externa JDOM (JDOM, 2008) foi empregada no projeto do componente *Parser*.

Apesar da ferramenta atualmente possuir um estágio funcional e operacional, a implementação de alguns elementos não essenciais para análise de complexidade da Figura 4.1 foram deixados para trabalho futuro. Os componentes atualmente implementados em todas suas funcionalidades previstas são o *Parser* e *Complexity Scorer*. Desta forma, a ferramenta espera como entrada um arquivo XML com a descrição do procedimento a ser analisado e gera outro arquivo XML com o resultado da avaliação de complexidade.

A seguir são descritos os detalhes do arquivo descritor de procedimentos e do arquivo de resultados de complexidade gerado pelo SCA, bem como a forma de cálculo do tempo gasto em cada ação.

4.3.1 Estrutura dos arquivos de entrada e saída

O arquivo descritor de procedimentos a ser fornecido à SCA deve ser formatado em um esquema XML específico. Neste esquema, os dados de cada ação individual são representados, além da descrição tanto dos contêineres quanto dos parâmetros consumidos no procedimento. A Figura 4.2 ilustra um exemplo parcial do arquivo XML utilizado para expressar o procedimento apresentado na Figura 3.2. Ele é organizado em três partes: *Tasks* (linhas 5 a 28, se referem às ações do modelo de atividades), *paramList* (linhas de 30 a 41, se referem à lista de parâmetros utilizados no procedimento) e *containerList* (linhas 43 a 52), que se refere à lista de *containers* do modelo de infra-estrutura de TI.

Para ilustrar o conteúdo de cada *tag* XML, tome-se como exemplo a ação (ou tarefa) número 23, que pode se vista na linha 5 da Figura 4.2. O título desta ação (`taskTitle`) é "Compile dm-crypt"; ela é executada no contêiner (`taskContainer`) `dm-crypt_db`, seu número de seqüência (`taskSeqNumber`) é 23 e utiliza apenas o parâmetro (*tag* `paramUsed`) `libdevmap_par_bd`. Sua relação com segurança (`securityType`) é 2, ou seja, classificada como uma ação diretamente relacionada com segurança, destacada em preto no modelo de atividades, como pode ser visto na Figura 3.2. Além disso, esta ação tem três valores de tempos, 5,98, 3,2 e 9,18 segundos, que se referem, respectivamente, aos tempos de interação entre o administrador e o sistema durante a execução desta ação em particular (`interactionTime`), o tempo de processamento gasto pelo sistema (`processingTime`) e o tempo total, que se refere a soma dos tempos de interação e execução (`totalTime`).

Cada parâmetro, por sua vez, tem atribuído um valor para a métrica *SourceScore*, descrito na Tabela 3.2 (`paramSourceScore`). No caso do parâmetro `libdevmap_par_bd` este valor é 4, como pode ser observado na linha 33 da mesma figura. Por fim, cada contêiner possui uma descrição (`containerTitle`, como pode ser observado na linha 49) e uma uma posição (`containerTreePos`). No exemplo, o contêiner descrito como `dm_crypt_db` possui o valor 0.1.1.3 para esta posição (linha 50), que se refere a posição do mesmo na árvore hierárquica de contêineres a ser montada para o cálculo das métricas *ContextSwitch* e *ParamCrossContext*, conforme definido na Seção 3.2 e ilustrado na Figura 3.3.

Depois de receber um arquivo de entrada, a SCA realiza o cálculo das métricas de complexidade, tanto em nível de ações quanto em nível de procedimentos. Estes resul-

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Procedure>
3    <desc>Joomla install with all security mechanisms</desc>
4  ...
5    <Task>
6     <taskTitle>Compile dm-crypt_</taskTitle>
7     <taskContainer>dm-crypt_db</taskContainer>
8     <taskSeqNumber>23</taskSeqNumber>
9     <paramUseList>
10      <paramUsed>libdevmap_par_bd</paramUsed>
11    </paramUseList>
12    <securityType>2</securityType>
13    <interactionTime>5.98</interactionTime>
14    <processingTime>3.2</processingTime>
15    <totalTime>9.18</totalTime>
16   </Task>
17   <Task>
18     <taskTitle>map crypt partition</taskTitle>
19     <taskContainer>dm-crypt_db</taskContainer>
20    <taskSeqNumber>24</taskSeqNumber>
21     <paramUseList>
22      <paramUsed>db_disk_passphrase</paramUsed>
23     </paramUseList>
24     <securityType>2</securityType>
25     <interactionTime>62.1</interactionTime>
26     <processingTime>17.7</processingTime>
27     <totalTime>79.8</totalTime>
28   </Task>
29   ...
30   <paramList>
31     <parameter>
32       <paramTitle>libdevmap_par_bd</paramTitle>
33       <paramSourceScore>4</paramSourceScore>
34       <paramTask>23</paramTask>
35     </parameter>
36     <parameter>
37       <paramTitle>dm-crypt_db</paramTitle>
38       <paramSourceScore>4</paramSourceScore>
39       <paramTask>24</paramTask>
40     </parameter>
41   </paramList>
42 ...
43   <containerList>
44     <container>
45       <containerTitle>Slackware_db</containerTitle>
46       <containerTreePos>0.1.1</containerTreePos>
47     </container>
48     <container>
49       <containerTitle>dm-crypt_db</containerTitle>
50       <containerTreePos>0.1.1.3</containerTreePos>
51     </container>
52   </containerList>
53 ...
54 </procedure>
```

Figura 4.2: Representação XML do procedimento

tados, por sua vez, são então armazenados no arquivo de resultados de complexidade em formato XML.

A Figura 4.3 ilustra parte do arquivo XML gerado pela SCA, onde os resultados provenientes da análise de complexidade do procedimento referente ao cenário exemplo (ilustrado na Figura 3.2) são armazenados. Na mesma figura, pode-se notar que o arquivo é organizado em duas partes: `ProcedureLevelResults` (linhas 4 a 18) e `TaskLevelResults` (linhas 19 a 51). A primeira parte se refere aos resultados da avaliação de complexidade em nível de procedimento, enquanto a última é relacionada ao resultado no nível das ações. Por exemplo, para o procedimento em questão, a quantidade de parâmetros consumidos é 72 (linha 8) e o número máximo de itens a ser armazenado em memória (capturado pela métrica *MemSizeMax*) é 14 (linha 13). Já a primeira ação do procedimento (de valor 1 na linha 21, referente à *tag* `taskSeqNumber`) utiliza um parâmetro apenas (linha 24), enquanto o parâmetro consumido na ação de número de seqüência igual a 2 (linha 36) tem o valor de *paramSourceScore* igual a 1.

```
1 <?xml version= ?1.0 ? encoding= ?UTF-8 ??>
2 <Procedure>
3   <desc>Joomla install with all security mechanisms</desc>
4   <ProcedureLevelResults>
5     <numActions> 77 </numActions>
6       <contextSwitch> 21 </contextSwitch>
7       <paramCount> 46 </paramCount>
8       <paramUseCount> 72 </paramUseCount>
9       <paramAdaptCount> 0 </paramAdaptCount>
10      <paramCrossContext> 45 </paramCrossContext>
11      <paramSourceScore>  110 </paramSourceScore>
12      <memSizeAvg> 7.38 </memSizeAvg>
13      <memSizeMax> 14 </memSizeMax>
14      <memLatAvg> 7.38 </memLatAvg>
15      <memLatMax> 264 </memDepthMax>
16      <memDepthAvg> 1.21 </memDepthAvg>
17      <memDepthMax> 40 </memDepthMax>
18   </ProcedureResults>
19   <TasksLevelResults>
20     <Task>
21       <taskSeqNumber> 1 </taskSeqNumber>
22       <taskTitle>Select kernel </taskTitle>
23       <TaskResults>
24         <numActions> 1 </numActions>
25         <contextSwitch> 0 </contextSwitch>
26         <paramUseCount> 1 </paramUseCount>
27         <paramAdaptCount> 0 </paramAdaptCount>
28         <paramCrossContext> 0 </paramCrossContext>
29         <paramSourceScore> 1 </paramSourceScore>
30         <memSizeAvg> 0 </memSizeAvg>
31         <memLatAvg> 0 </memLatAvg>
32         <memDepthAvg> 0 </memDepthAvg>
33       </TaskResults>
34     </Task>
35     <Task>
36       <taskSeqNumber> 2 </taskSeqNumber>
37       <taskTitle> Select Keyboard </taskTitle>
38       <TaskResults>
39         <numActions> 1 </numActions>
40         <contextSwitch> 0 </contextSwitch>
41         <paramUseCount> 1 </paramUseCount>
42         <paramAdaptCount> 0 </paramAdaptCount>
43         <paramCrossContext> 0 </paramCrossContext>
44         <paramSourceScore> 1 </paramSourceScore>
45         <memSizeAvg> 0 </memSizeAvg>
46         <memLatAvg> 0 </memLatAvg>
47         <memDepthAvg> 0 </memDepthAvg>
48       </TaskResults>
49     </Task>
50      ...
51   </TasksLevelResults>
52 </Procedure>
```

Figura 4.3: Arquivo de resultado de complexidade gerado pela SCA

5 AVALIAÇÃO EXPERIMENTAL

Neste capítulo é apresentada uma proposta para mensurar a complexidade associada à segurança dos procedimentos de TI, baseada nas métricas descritas na Seção 3.2. Foi realizada uma série de experimentos, explicados ao longo do capítulo, para que fosse possível avaliar a complexidade de segurança em três dimensões distintas:

- aumento de complexidade imposto pelo uso de mecanismos de segurança em procedimentos relacionados a TI;
- medida de complexidade dos procedimentos que manipulam mecanismos de segurança isoladamente;
- comparação de medidas de complexidade associada a procedimentos demandados por diferentes ferramentas que satisfazem um mesmo mecanismo de segurança.

Em seguida é apresentada cada uma dessas dimensões e discutido como as métricas de complexidade devem ser utilizadas em cada caso. São apresentados e discutidos, também, os resultados obtidos com a avaliação de complexidade de cenários reais vivenciados pelas empresas em geral.

5.1 Complexidade agregada por mecanismos de segurança em procedimentos gerais

Segurança (e suas atividades) podem fazer parte de procedimentos mais gerais, como, por exemplo, na instalação e configuração de uma aplicação Web segura. Para determinar a medida de complexidade que os mecanismos de segurança podem agregar a procedimentos mais gerais, primeiramente deve-se modelar um procedimento base, que não contém quaisquer mecanismos de segurança e, então, proceder com a avaliação de complexidade. Em seguida, um novo procedimento deve ser modelado – consistindo no procedimento base enriquecido com ações e parâmetros demandados pelos mecanismos de segurança especificados – e, então, deve-se realizar a sua avaliação de complexidade novamente. A diferença entre os valores observados para o novo procedimento e o procedimento base ($\Delta_{complexidade}$) reflete a complexidade adicional imposta pelos mecanismos de segurança no contexto específico em questão.

Para avaliar esta abordagem, foram analisados três cenários típicos, comumente encontrados em empresas. Eles compartilham um objetivo em comum – a instalação e configuração da aplicação Web Joomla e todo o software do qual ela depende. Entretanto, os cenários diferem em relação aos mecanismos de segurança implantados. O primeiro,

cenário A, é o cenário base e, portanto, não inclui qualquer mecanismo de segurança dentre os especificados na Tabela 3.1. Os cenários B e C podem ser vistos como versões, com o nível de segurança incrementado, do cenário A. O primeiro satisfaz parte dos mecanismos de segurança especificados na Tabela 3.1 (somente aqueles aplicados ao computador 1), enquanto o segundo implanta todos os mecanismos listados na mesma tabela.

Para prosseguir com a avaliação de complexidade, foram definidos procedimentos e árvores de hierarquias específicos para cada cenário, que então foram utilizados como entrada para a SCA. Como exemplo, a Figura 5.1 ilustra o procedimento referente ao cenário A, com um total de 38 ações, enquanto a Figura 5.2 apresenta o cenário B, com 55 ações. Por fim, o procedimento do cenário C pode ser visto na Figura 3.2, com 77 ações. Como se pode notar, a quantidade de ações de segurança, denotadas por caixas pretas e cinzas, aumenta do cenário A para o C. A seguir são apresentados e discutidos os resultados da avaliação de complexidade tanto para o nível de ações quanto para o de procedimentos.

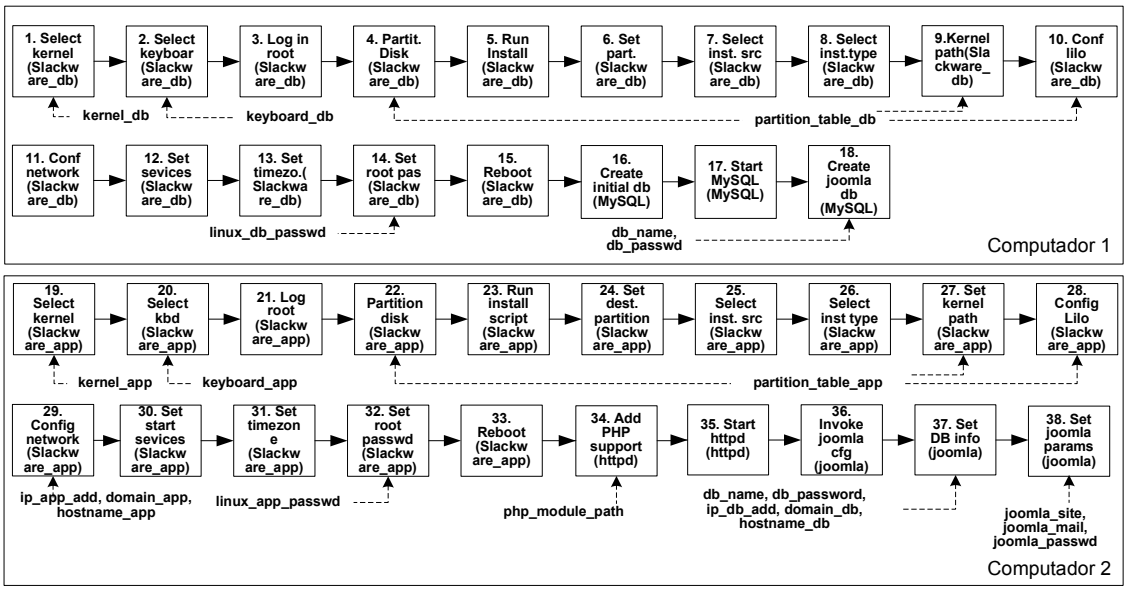

Figura 5.1: Procedimento para instalar o Joomla no cenário A

5.1.1 Complexidade em nível de procedimentos

Depois de definir os procedimentos, o administrador os repassou em formato XML para a SCA, que, por sua vez, efetuou o cálculo de complexidade utilizando as métricas apresentadas na Seção 3.2. O resultado da avaliação dos cenários A, B e C para o contexto dos procedimentos pode ser observado na Tabela 5.1. Nesta tabela, os valores em parênteses indicam a variação percentual da complexidade que os cenários B e C apresentam em relação ao cenário A.

Examinando a tabela, pode-se notar que a implantação dos mecanismos de segurança impactou significativamente as medidas de complexidade dos cenários B e C em relação aos valores observados para o cenário base A (para quase todas as métricas). Por exemplo, com exceção da métrica *ParamAdaptCount*, os valores medidos para o cenário C foram, no mínimo, o dobro quando comparados ao cenário A. Em relação ao cenário B, as diferenças foram, no geral, também altas. Pode-se observar, entretanto, que o valor da métrica *MemDepthAvg* foi menor do que o valor calculado para o cenário A. Isto se

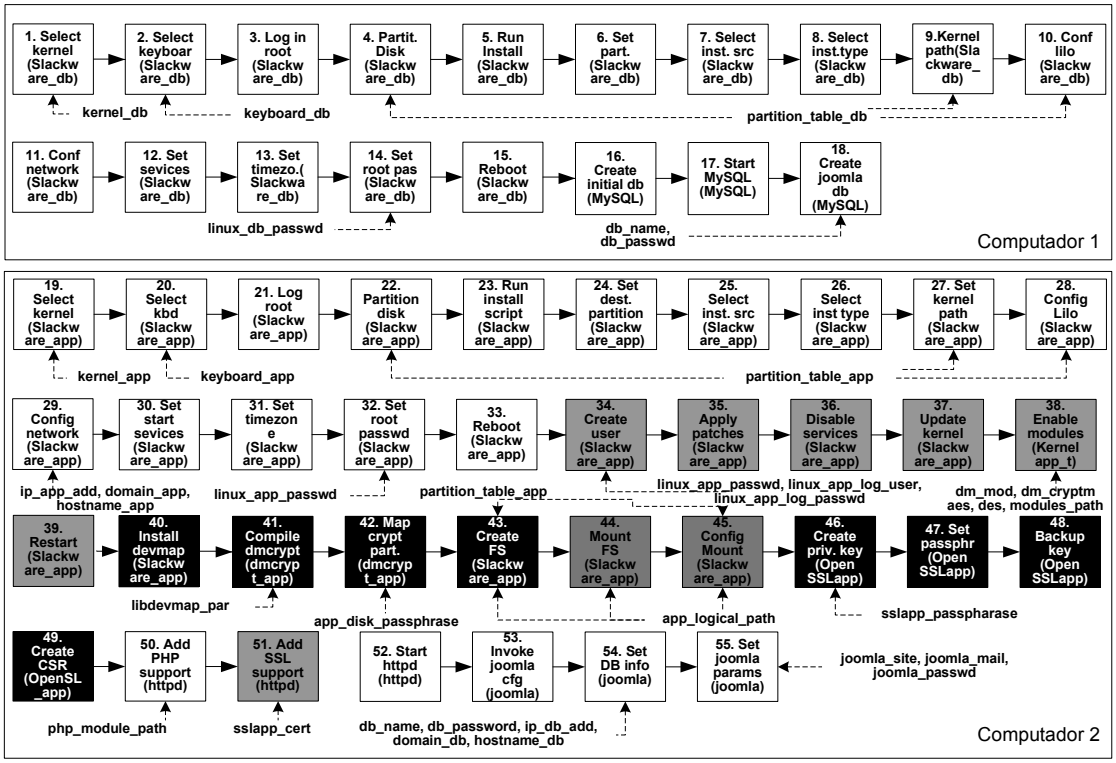

Figura 5.2: Procedimento para instalar o Joomla no cenário B

deve ao fato de que a soma dos valores associados a *MemDepth* no cenário B não ter sido suficientemente alta, quando comparada ao cenário A, para acompanhar o valor maior do denominador (*NumActions*) utilizado para calcular as médias (53 para o cenário B e 38 para o cenário A).

Estes resultados também destacam o grande volume de informações que um administrador deve manter em memória enquanto executa os procedimentos, capturado pela métrica *MemSizeMax*. No pior caso, ele deve armazenar 14 itens, o que excede em no mínimo 5 itens o valor comum aceito da capacidade da memória de curto prazo dos humanos, conforme descrito por Miller (MILLER, 1956). Efetuando-se uma análise mais detalhada dos itens em memória, 6 deles são diretamente associados aos mecanismos de segurança. Além do aumento do número de itens a serem lembrados, o intervalo de tempo que eles devem ser retidos em memória também aumentou, refletido pela métrica *MemLatAvg*.

A Figura 5.3 ilustra a medida de complexidade total para os três cenários avaliados, computadas no nível de procedimentos. Os valores observados para cada grupo de métricas (execução, parâmetro e memória) foram somados. Apesar do gráfico não espelhar a influência de cada métrica na complexidade resultante do procedimento – o que é demonstrado no Capítulo 6 quando se realiza a seleção de métricas para o modelo quantitativo – ele fornece ao leitor uma primeira aproximação de como os cenários A, B e C se diferenciam no que se refere à complexidade final.

Os mecanismos de segurança incluídos no cenário B – mais precisamente a criptografia do sistema de arquivos e na conexão com o servidor Web – são responsáveis por um aumento de 50%, 47% e 58% nos subgrupos de métricas de execução, parâmetro e memória. Quando se compara o cenário C com o cenário A, esses valores são ainda maiores: 123%, 133% e 131% para os mesmos subgrupos de métricas.

Tabela 5.1: Medidas de complexidade para os cenários A, B e C

Métrica	*Cenário A*	*Cenário B*	*Cenário C*
NumActions	38	55 (44,7%)	77 (102,6%)
ContextSwitchSum	6	11 (83,3%)	21 (250,0%)
ParamCount	20	32 (60,0%)	46 (130,0%)
ParamUseCount	31	45 (48,3%)	72 (132,2%)
ParamAdaptCount	0	0 (0%)	0 (0%)
ParamCrossContext	21	21 (0%)	45 (114,2%)
ParamSourceScore	45	74 (64,4%)	110 (144,4%)
MemSizeAvg	3.42	4.23 (23,6%)	7,38 (115,7%)
MemSizeMax	6	7 (16,6%)	14 (133,3%)
MemLatAvg	3.42	4,23 (23,6%)	7,38 (115,7%)
MemLatMax	116	201 (73,2%)	264 (127,5%)
MemDepthAvg	0.52	0.41 (-21,1%)	1,21 (132,6%)
MemDepthMax	15	15 (0%)	40 (166,6%)

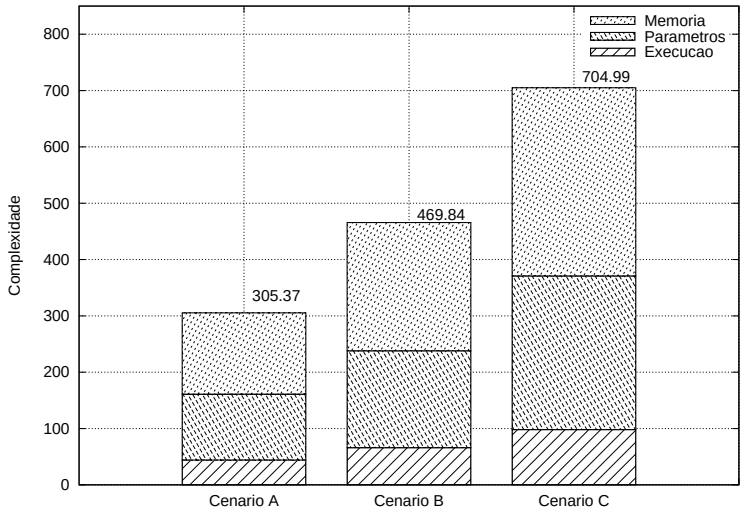

Figura 5.3: Complexidade em nível de procedimentos para os cenários A, B e C

5.1.2 Complexidade em nível de ações

Uma visão mais detalhada dos pontos de maior complexidade de um procedimento pode ser obtida com um gráfico que ilustre a complexidade no nível das ações, como o que pode ser observado na Figura 5.4. Ele ilustra as medidas de complexidade para um subconjunto de ações do cenário C. As três barras representam os valores medidos para complexidade de execução, parâmetro e memória, respectivamente, enquanto suas cores denotam a relação com segurança (como indicado no modelo de segurança de TI). Por exemplo, a ação 62 – `create filesystem` – é um passo final que precisa ser executado para a cifragem do sistema de arquivos da máquina e, portanto, diretamente relacionado à segurança. O alto valor observado para a complexidade de memória se deve ao fato da necessidade do administrador em manter na memória o parâmetro “partition_table_app” consumido 15 ações anteriormente.

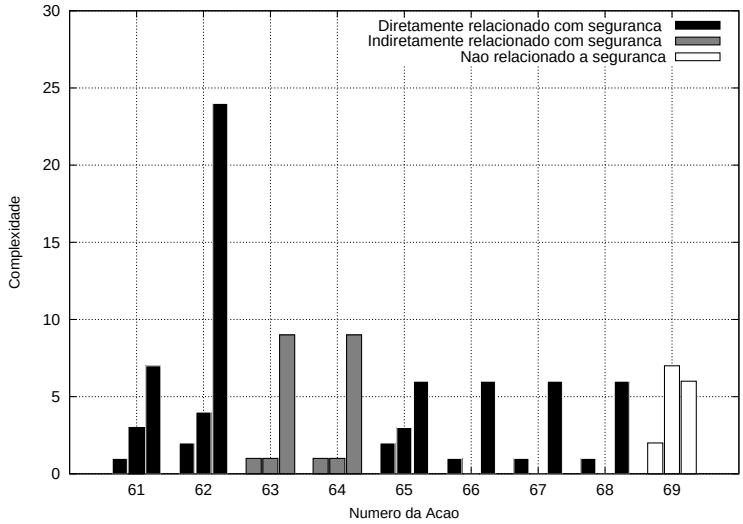

Figura 5.4: Complexidade de ações para o procedimento do cenário C

5.2 Medida de complexidade de procedimentos relacionados a mecanismos de segurança isolados

Ao contrário do problema formulado na subseção anterior, há situações em que mecanismos de segurança precisam ser instalados/desinstalados e/ou manutenidos em uma infra-estrutura de TI já existente e em operação. Por exemplo, considere o caso em que um gerente de TI determina o uso de criptografia no sistema de arquivos de um servidor (onde dados confidenciais são armazenados). Para que se possa mensurar a complexidade de procedimentos como esse, deve-se, primeiramente, especificar os mesmos isoladamente – isto é, ignorando as ações não relacionadas aos próprios mecanismos de segurança – e, então, efetuar a avaliação de complexidade.

Para esta dimensão em particular foi realizada a avaliação de complexidade de quatro cenários que implementam diferentes mecanismos de segurança em um servidor em produção, executando o sistema operacional Slackware Linux com os serviços básicos de sistema e rede (incluindo o servidor Web httpd da Apache). No primeiro cenário, denominado D, foi instalado e configurado o uso de OpenSSL (OPENSSL, 2008) para permitir uma comunicação segura dos clientes com o servidor Web. O procedimento referente a este cenário, com um total de 7 ações, pode ser observado na Figura 5.5. Observe que neste caso todo o software é hospedado no computador 1.

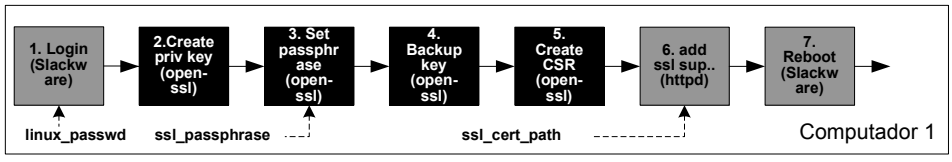

Figura 5.5: Procedimento referente ao cenário D

O segundo cenário, E, consiste no uso de criptografia no sistema de arquivos do Computador 1, utilizando dm-crypt (DM-CRYPT, 2008). O procedimento associado a este cenário, composto por 10 ações, pode ser visualizado na Figura 5.6. Por sua vez, o terceiro cenário, F, é caracterizado pelo uso de um filtro de pacotes netfilter/iptables (NETFILTER/IPTABLES, 2008), em que regras são adicionadas ao firewall existente. O pro-

cedimento associado a este cenário pode ser observado na Figura 5.7. Apesar de possuir apenas 3 ações, o procedimento utiliza diversos parâmetros durante a configuração do firewall.

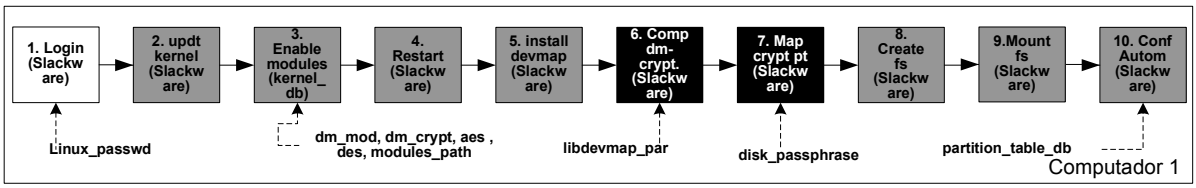

Figura 5.6: Procedimento referente ao cenário E

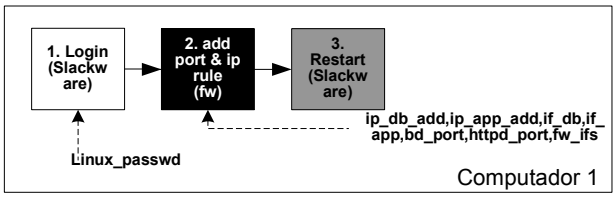

Figura 5.7: Procedimento referente ao cenário F

Por fim, no cenário G os três mecanismos são instalados simultaneamente. O respectivo procedimento resultante, composto de 17 ações, pode ser visualizado na Figura 5.8.

A Tabela 5.2 sumariza os resultados obtidos. Comparando os cenários D, E e F, pode-se notar que o cenário E – referente à criptografia do sistema de arquivos – apresenta as maiores medidas de complexidade, refletindo o tempo e o esforço que os administradores percebem ao executá-lo. Para instalar o dm-crypt, foi necessário, dentre outras ações, compilar o mesmo a partir do seu código fonte e ativar módulos específicos no kernel do Linux, o que resultou em um procedimento contendo mais ações, parâmetros e trocas de contextos (quando comparados com aqueles executados para os cenários D e F). Além disso, o mesmo procedimento demandou mais parâmetros (10) que foram, em geral, os mais difíceis de se obter (refletido pelo valor da métrica *ParamSourceScore*).

A partir da tabela, pode-se notar também que a medida de complexidade para o cenário G é, em geral, menor do que aquela representada pela soma dos valores computados para os cenários D, E e F. Por exemplo, *NumActions* resultou em 17 para G, enquanto que para D+E+F o valor foi 20. Num primeiro momento, pode causar supresa o fato de o resultado derivado da combinação dos três mecanismos de segurança não representar a soma dos resultados individuais . Isto é devido à existência de ações compartilhadas (e parâmetros)

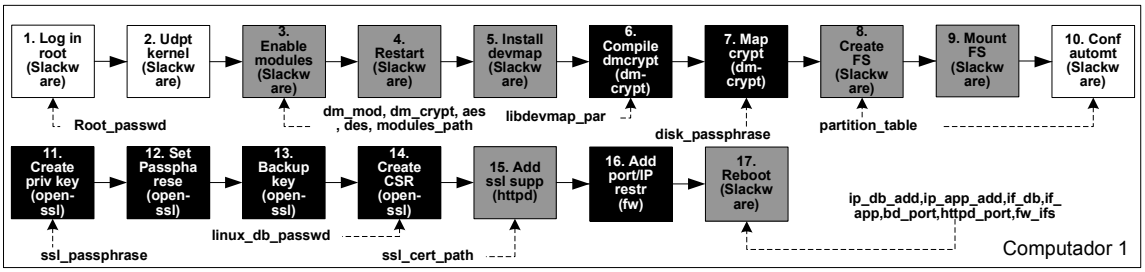

Figura 5.8: Procedimento referente ao cenário G

Tabela 5.2: Medidas de complexidade para os procedimentos dos cenários D, E, F e G

Métrica/Cenário	*D*	*E*	*F*	*G*
NumActions	7	10	3	17
ContextSwitchSum	2	4	1	7
ParamCount	3	10	8	19
ParamUseCount	3	13	8	22
ParamAdaptCount	0	0	0	0
ParamCrossContext	0	0	0	0
ParamSourceScore	9	31	16	52
MemSizeAvg	0	0,4	0	0,23
MemSizeMax	0	2	0	2
MemLatAvg	0	0,4	0	0,23
MemLatMax	0	3	0	3
MemDepthAvg	0	0,5	0	0,9
MemDepthMax	0	3	0	3

quando os três mecanismos são executados em um único procedimento, como a ação de se autenticar em um sistema operacional utilizando nome de usuário e senha como parâmetros.

5.3 Comparação de valores de complexidade de procedimentos relacionados a diferentes ferramentas que suportam os mesmos mecanismos de segurança

Quando se avalia as possibilidades de satisfazer os requisitos de segurança definidos em um SLA ou em política de segurança, pode haver situações em que mais de uma ferramenta de segurança pode ser utilizada para atingir o mesmo objetivo. Neste contexto, a comparação das medidas de complexidade associadas aos procedimentos de instalação, remoção, configuração e manutenção demandados pelas diferentes ferramentas pode ajudar durante a escolha de uma ferramenta em detrimento de outra.

Para ilustrar com um exemplo concreto, foi realizada uma avaliação de complexidade dos procedimentos associados à instalação e configuração de diferentes ferramentas que implementam *Virtual Private Networks* (VPNs). As ferramentas utilizadas nos experimentos foram OpenVPN (OPENVPN, 2008) e Openswan (OPENSWAN, 2008). Para prover canais de comunicação seguros, a primeira emprega *Secure Sockets Layer* (SSL), enquanto a segunda utiliza *IP Security* (IPSec). Com o intuito de realizar uma comparação justa, as ferramentas foram avaliadas sob as mesmas condições. OpenVPN e Openswan foram instaladas em duas máquinas equivalentes, permitindo aos usuários remotos realizar conexões utilizando a Internet (modo *road warrior* de VPN). Ambas as ferramentas foram configuradas para utilizar certificados X.509 para autenticação de sessões, processo que inclui, também, a criação de uma *Certificate Authority* (CA) e a emissão dos certificados.

A Figura 5.9 ilustra o procedimento associado à configuração do cenário da ferramenta OpenVPN. Observe que são necessárias 13 ações que demandam uma quantidade significativa de parâmetros. Por sua vez, o procedimento de configuração da Openswan pode ser visto na Figura 5.10. De antemão é notável que ele apresenta uma maior quantidade de ações que o procedimento associado ao OpenVPN.

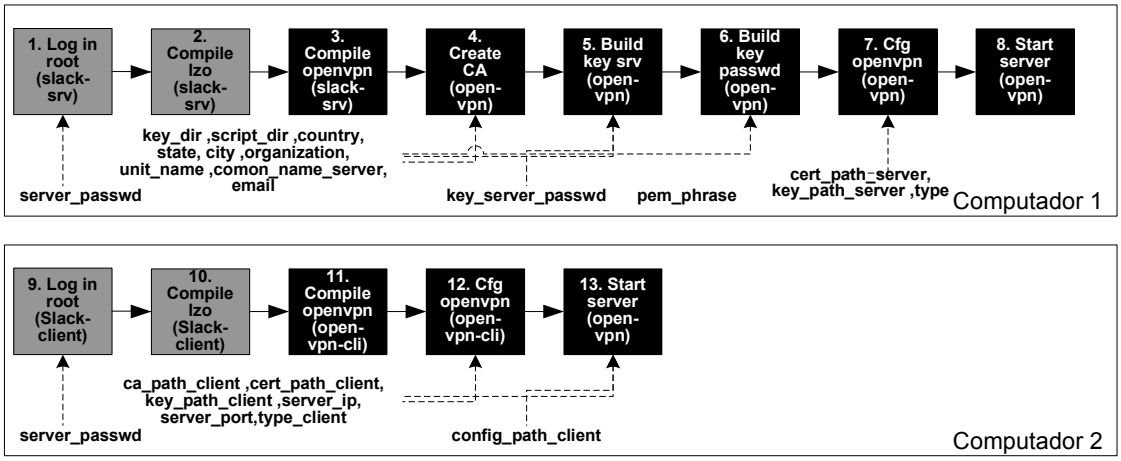

Figura 5.9: Procedimento referente ao uso do OpenVPN

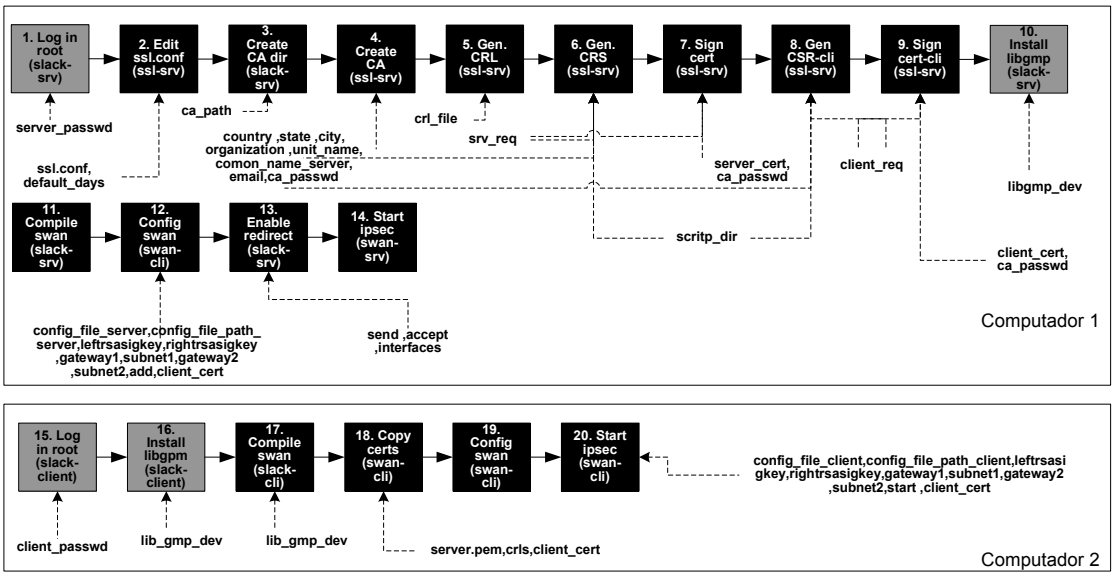

Figura 5.10: Procedimento referente ao uso do Openswan

A Tabela 5.3 sumariza os resultados obtidos neste experimento. Comparando as ferramentas, pode-se observar que a instalação e configuração da Openswan é significativamente mais complexa de se realizar do que a da OpenVPN (demandando 53% mais ações, 180% mais trocas de contexto e 58% mais parâmetros). Isto se deve ao fato da última prover *scripts* que automatizam a criação da CA e a emissão dos certificados, quanto que para a primeira é necessário realizar as operações manualmente. Além disso, a Openswan requer a instalação do gmp (GNU *Multiple Precision Arithmetic Library*) e a configuração explícita da interface de rede para aceitar redirecionamento de pacotes.

5.4 Considerações parciais

Os resultados obtidos com a avaliação de complexidade dos procedimentos relacionados a segurança de TI podem ser correlacionados com o tempo gasto pelo administrador durante a execução dos procedimentos. Este tópico é investigado no próximo capítulo, em que o impacto ocasionado pela complexidade dos mecanismos de segurança no tempo de execução é mensurado, e, a partir destes dados, é criado um modelo quantitativo para

Tabela 5.3: Medidas de complexidade para a instalação do OpenVPN e Openswan

Métrica	*OpenVPN*	*Openswan*
NumActions	13	20
ContextSwitchSum	5	14
ParamCount	24	38
ParamUseCount	40	69
ParamAdaptCount	0	0
ParamCrossContext	0	5
ParamSourceScore	66	94
MemSizeAvg	1,23	4,85
MemSizeMax	8	10
MemLatAvg	1,23	4,85
MemLatMax	8	43
MemDepthAvg	5,53	6,7
MemDepthMax	36	45

realizar previsões acerca do tempo de execução de novos procedimentos, sem que haja a necessidade de executar os mesmos.

6 PREVISÃO DE CUSTOS COM USO DE MÉTRICAS DE COMPLEXIDADE DE PROCEDIMENTOS DE TI

Uma das grandes promessas da área de TI é que ela é capaz de melhorar a eficiência das operações de negócios das empresas, gerando economia em termos financeiros, processando rapidamente as informações e possibilitando a aplicação de novas funcionalidades (DIAO, Y. et al., 2007). Em se tratando de segurança, essas promessas se traduzem em obter a melhor relação custo/benefício entre o nível de proteção da infra-estrutura de TI e o uso de recursos, tais como hardware/software, pessoas e recursos financeiros. Entretanto, quando os administradores propõem o uso de determinado mecanismo de segurança (e, por fim, uma ferramenta que o implante), não é possível saber de antemão os reais custos associados à implantação do mesmo. Isto traz a tona questões cruciais por parte dos diretores de TI: como é possível quantificar, medir e, por fim, prever os custos associados à introdução de um determinado mecanismo de segurança?

Nos capítulos anteriores foi apresentada uma abordagem para mensurar a complexidade que a implantação dos mecanismos de segurança adicionam tanto em procedimentos mais gerais de TI quanto em procedimentos específicos de segurança. Entretanto, os resultados obtidos a partir desta análise não permitem determinar diretamente as economias ou gastos gerados com a adoção de um mecanismo de segurança específico. Neste capítulo é apresentada uma proposta para a criação de um modelo quantitativo baseado nas métricas apresentada no Capítulo 3, que permite estimar os custos associados à instalação/manutenção e configuração de ferramentas de segurança específicas, tendo por base o trabalho de Diao *et al.* (DIAO, Y. et al., 2007).

Esta abordagem apresenta vários benefícios: o modelo quantitativo pode ser utilizado para determinar o *return-on-investment* (ROI), antes ou depois da implantação das ferramentas de segurança. Um exemplo seria o cenário da Seção 5.3, em que diferentes ferramentas suportam o mesmo mecanismo de segurança: neste caso, o modelo poderia ser utilizado para comparar o impacto do uso da ferramenta no orçamento da organização. Além disso, modelos calibrados para cenários específicos podem revelar quais são os fatores que apresentam a maior contribuição na complexidade final do procedimento – que pode ser utilizado como uma diretriz para a melhoria dos produtos por parte dos fornecedores. Por fim, as previsões obtidas com o modelo podem ser utilizadas pelos administradores e gerentes de TI para revisarem os SLAs e as políticas de segurança à luz dos custos associados à sua real implantação.

Este capítulo se divide da seguinte forma: na próxima seção é apresentada uma abordagem para a criação de um modelo quantitativo. Em seguida, é apresentado um estudo de caso onde a abordagem é aplicada e avaliada tanto no nível das ações quanto dos procedimentos. Por fim, na Seção 6.3, as considerações parciais são apresentadas. Além disso,

no Apêndice A é revisada a técnica de análise de regressão linear múltipla e introduzida a ferramenta R (IHAKA; GENTLEMAN, 1996; R, 2008), utilizadas no decorrer deste trabalho para realizar as análises estatísticas.

6.1 Metodologia para criação de modelo de previsão de custos

Nesta seção é descrita a abordagem para a criação de um modelo quantitativo, tendo por base as métricas de complexidade apresentadas no Capítulo 3 e a técnica de regressão linear múltipla, analisada em mais detalhes no Apêndice A. Esta abordagem, baseada no trabalho de Diao *et al.* (DIAO, Y. et al., 2007), pode ser resumida em três etapas: avaliação de complexidade e medição do tempo em um cenário base, construção do modelo de regressão e avaliação de qualidade do modelo e, por fim, extrapolação do modelo para previsão de custos. Mais detalhes destas etapas são apresentados a seguir.

6.1.1 Etapa 1: Avaliação de complexidade e captura de tempos

O primeiro passo para a criação do modelo quantitativo é a definição dos cenários de segurança a serem avaliados. Neste estudo é proposto o uso de dois tipos de cenários: *base* e *futuro*. Os dados do cenário base são utilizados para se criar o modelo quantitativo, enquanto que os dados do cenário futuro, por sua vez, são utilizados para avaliar a qualidade das previsões geradas pelo modelo.

A escolha do cenário base depende do contexto a ser avaliado. Por exemplo, caso uma organização queira saber o custo de agregar determinados mecanismos de segurança a um procedimento já existente, o cenário base, neste caso, pode ser o procedimento sem os mecanismos de segurança, enquanto que o procedimento com segurança, por sua vez, seria o cenário futuro. Por outro lado, se a organização deseja determinar a economia de custos com manutenção e instalação de mecanismos de segurança quando estes são removidos de procedimentos com segurança, o cenário base, neste caso, seria o cenário com os mecanismos de segurança enquanto que o cenário futuro, a ter seu custo estimado, seria o cenário sem os mecanismos de segurança.

Após a definição e a escolha dos cenários, o cenário base, então, deve ser executado pelos administradores. Durante essa execução, os administradores modelam o procedimento referente a este cenário – anotando ações, parâmetros e contêineres. Além disso, é medido o tempo gasto pelo administrador durante a execução de cada ação, descontando-se o tempo gasto com a documentação do procedimento (por exemplo, definição de ações e parâmetros). Por fim, a análise de complexidade do procedimento base deve ser feita, conforme apresentado na Seção 3.2. Estes valores (tempo mensurado e valores das métricas) são utilizados como entrada para a criação do modelo na próxima etapa.

6.1.2 Etapa 2: Construção do modelo quantitativo

A relação entre os dados obtidos na etapa 1 – valores para as métricas de complexidade do cenário base e as medidas de tempo – é investigada nesta etapa, com o uso da técnica de regressão linear múltipla. Esta técnica permite gerar uma equação (ou modelo) linear que correlaciona o tempo gasto na execução de cada ação com as métricas de complexidade. A equação de regressão apresenta a seguinte forma:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{6.1}$$

Nesta equação, as variáveis explanatórias são representadas pelos termos x_i, ao passo

que a variável dependente é representada pelo termo y. A relação entre elas é caracterizada pelos parâmetros β_i, que indicam quantitativamente como a variável dependente pode ser interpretada linearmente através das variáveis explanatórias (FRANK E.; HARRELL, 2006).

Neste trabalho, o uso principal deste modelo é prever o tempo associado ao uso de mecanismos de segurança através das métricas de complexidade de TI. Para isso, a variável dependente (a ser estimada) é o tempo de execução das ações (y, na Equação 6.1). As variáveis explanatórias, neste estudo, são as métricas apresentadas na Seção 3.2 (x_i, na mesma Equação). Para construir o modelo quantitativo, deve-se utilizar o método dos mínimos quadrados ordinários (descrito em mais detalhes no Apêndice A). Este método produz como resultado os parâmetros β_i da Equação 6.1 – que são os valores que acompanham cada uma das métricas de complexidade.

A medida que mais métricas são utilizadas, mais medições de tempo devem ser executadas. Como a quantidade de medições é normalmente limitada, se faz necessário identificar um subconjunto das métricas de complexidade de TI que possa ser utilizado como variáveis explanatórias. Além disso, o uso de uma menor quantidade de métricas evita o *overfitting* dos dados de entrada, preservando a capacidade de adaptação do modelo para novos cenários (DIAO, Y. et al., 2007). Apesar do uso de mais variáveis explanatórias permitir gerar um modelo com menor erro, isso não implica necessariamente em um aumento da qualidade do modelo. As variáveis extras tendem a modelar o "ruído" ao invés da relação entre as métricas, o que prejudica a extrapolação do modelo para predição de custos associados aos procedimentos de segurança.

É desejável, inclusive, que o modelo final possua o menor número de variáveis explanatórias possível ao mesmo tempo que consiga realizar previsões satisfatoriamente, uma vez que uma menor quantidade de variáveis explanatórias (as métricas de complexidade, de fato) facilita o uso do modelo e diminui os custos de manutenção do mesmo (MONTGOMERY; RUNGER, 2006). O compromisso entre esses objetivos conflitantes é conhecido como busca pela "melhor" equação de regressão. Entretanto, em vários problemas, nenhum modelo de regressão é o "melhor" no que diz respeito a todos os critérios de avaliação. Neste trabalho, a seleção de métricas é feita em uma abordagem incremental, baseada na proposta de Diao *et. al* (DIAO, Y. et al., 2003). É utilizado um algoritmo para criação do modelo quantitativo, onde são utilizadas as seguintes variáveis:

- `candidateMetrics`, que é responsável por armazenar as métricas de complexidade candidatas a participarem do modelo quantitativo;
- `responseVariable`, que armazena a variável dependente, ou seja, o tempo de execução das ações;
- `explanatoryVariables`, variável que recebe o conjunto de métricas a serem adicionadas ao modelo;
- `bestMetric`, que é utilizada para armazenar temporariamente as métricas de complexidade utilizadas no processo de criação do modelo.

A seguir são apresentados os detalhes do algoritmo de seleção de métricas utilizado neste trabalho.

1. **Inicialização**

- É atribuído à variável `candidateMetrics` o conjunto de todas as métricas de complexidade.
- A variável `responseVariable` recebe a variável dependente, isto é, o tempo de execução das ações.
- Não é atribuída nenhuma variável a `explanatoryVariables`.

2. **Definição da métrica que melhor explica `responseVariable`**

   - É feito o cálculo da correlação entre cada métrica de `candidateMetrics` e `responseVariable`.
   - É atribuída a `bestMetric` métrica com o maior valor de correlação em módulo.
   - `bestMetric` é adicionada a `explanatoryVariables`.

3. **Incremento do modelo**

   - É feita uma regressão linear para `responseVariable` utilizando `explanatoryVariables`.
   - `bestMetric` é removida de `candidateMetrics`.

4. **Teste para finalização**

   - Deve-se avaliar a qualidade do modelo gerado, utilizando os critérios definidos na subseção 6.1.2.1. Se a qualidade do modelo começa a piorar ou se parâmetros β_i negativos começarem a sugir, a seleção de métricas está encerrada. Caso contrário, deve-se voltar ao passo 2 até que `candidateMetrics` fique vazio.

6.1.2.1 Avaliação da qualidade do modelo

Para se determinar qual o subconjunto de métricas devem fazer parte do modelo quantitativo, deve-se observar como a qualidade do modelo varia a medida que mais métricas são adicionadas ao modelo, como definido no processo incremental de criação do modelo recém mencionado. Para auxiliar nesta escolha, deve-se utilizar duas métricas empregadas no contexto de regressão linear múltipla: R^2 e RMSE. R^2 é definido a proporção da varibilidade em um conjunto de dados que pode ser explicado por um modelo quantitativo. Esta métrica pode ser calculada da seguinte forma:

$$R^2 = 1 - \frac{var(y - \hat{y})}{var(y)} \tag{6.2}$$

onde $var()$ denota a variância, y representa o tempo observado e $\hat{y}$ se refere ao tempo estimado. O valor de R^2 denota a porcentagem da variabilidade de y que pode ser explicada pelo modelo. Um valor de R^2 igual a 1 indica que o modelo foi capaz de capturar toda a variabilidade e um valor de R^2 igual a 0 indica que modelo não foi capaz de capturar a variabilidade.

A outra métrica a ser utilizada em conjunto com R^2 é o erro médio quadrático (*root mean square error* – RMSE), que é uma medida utilizada para comparar as diferenças entre os valores previstos por um modelo e os valores reais observados. RMSE pode ser calculado da seguinte forma:

$$RMSE = \sqrt{\frac{1}{k}\sum_{k=1}^{K}(y(k) - \hat{y}(k))^2} \tag{6.3}$$

Valores de RMSE mais próximos de zero são mais desejáveis, pois indicam uma melhor precisão do modelo.

Para escolher o conjunto de métricas a ser utilizadas no modelo quantitativo, deve-se observar como os valores de R^2 e RMSE se comportam à medida que as variáveis explanatórias são adicionadas ao modelo. O modelo ideal é aquele que apresenta um maior valor de R^2 e um menor RMSE ao mesmo tempo que não possui parâmetros β_i negativos, que tendem a modelar o ruído, prejudicando a extrapolação do modelo para realização de previsões (DIAO, Y. et al., 2007).

Uma vez definido o modelo (e verificado se os valores de R^2 e RMSE são aceitáveis), o próximo passo é extrapolar o mesmo para realizar previsões, como abordado na próxima etapa.

6.1.3 Etapa 3: Extrapolação do modelo para realizar previsões

Nesta etapa o modelo criado é, então, extrapolado para estimar o tempo de execução do cenário futuro, definido na Etapa 1. Para realizar a previsão, o resultado da análise de complexidade do cenário futuro é utilizado como entrada para o modelo quantitativo gerado e é obtido, como saída, uma estimativa do tempo de execução das ações.

Opcionalmente, o administrador pode, ainda, desejar determinar a qualidade da previsão gerada pelo modelo para o cenário futuro. Neste caso, o procedimento do cenário futuro deve ser executado por completo e os tempos gastos em cada ação devem ser anotados. Estes valores reais devem, então, ser confrontados com os tempos previstos pelo modelo, utilizando o mesmo método apresentado na Subseção 6.1.2.1.

A próxima seção apresenta um estudo de caso onde a metodologia descrita é aplicada em um cenário real de segurança.

6.2 Estudo de caso: previsão de custos associados à segurança

Nesta seção aplica-se a abordagem recém mencionada para produzir o modelo quantitativo em um estudo de caso com procedimentos de segurança reais, vivenciados na organizações. Foram definidos os dois cenários especificados pela abordagem (base e futuro), que consistem em versões simplificadas dos cenários A e C apresentados na Seção 5.1 (em que a aplicação Web Joomla era instalada variando apenas os requisitos de segurança). A diferença é que nos cenários base e futuro o sistema operacional das máquinas já estava instalado e não foi necessário configurar o firewall da rede. Desta forma, o cenário base consiste na configuração do banco de dados MySQL em uma máquina e a configuração do servidor Web httpd e da aplicação Joomla na outra máquina. Por sua vez, o cenário futuro pode ser visto como uma versão modificada do cenário base, onde o ambiente é incrementado com os mecanismos de segurança, tais como o uso de criptografia tanto nas conexões com o banco de dados (com o uso de OpenSSL) quanto nas conexões com o servidor Web. Além disso, é utilizado também sistema de arquivos criptografados em ambas as máquinas, com o auxílio da ferramenta dm-crypt. O objetivo, neste estudo de caso, é realizar uma previsão do tempo adicional necessário para agregar mecanismos de segurança em um procedimento base.

A infra-estrutura avaliada é composta de dois computadores com sistema operacional e os serviços básicos de rede devidamente configurados e em produção (foi utilizada a instalação completa padrão do Slackware Linux 10.2). A seguir é demonstrado como as quatro etapas definidas na Seção 6.1 são aplicadas para realizar previsões.

6.2.1 Análise de complexidade e captura de tempos

Depois de definidos os cenários a serem utilizados, eles foram modelados em procedimentos e executados por três administradores distintos, experientes em configuração de soluções como as dos cenários base e futuro. A medida que cada ação era executada, os administradores interagiam com um subcomponente da SCA e efetuavam três tomadas de tempo para cada ação. Com isso, foi possível categorizar o tempo total gasto durante a execução de uma ação em *tempo ocupado* (tempo que o administrador efetivamente permanecia ocupado) e *tempo de processamento* (tempo em que o administrador aguardava o fim da execução da ação pela máquina).

Estes procedimentos, por sua vez, foram, então, manualmente codificados em formato XML (de forma análoga ao procedimento da Figura 4.2) e submetidos para análise de complexidade pela SCA. Os resultados provenientes desta análise foram armazenados para serem utilizados na etapa de criação do modelo quantitativo.

Os dois procedimentos instanciados foram executados pelos administradores, totalizando seis medidas de tempo distintas. É importante observar que apenas os dados referentes ao cenário base (ou seja, 3 medições) foram utilizados para a criação do modelo, enquanto que os dados obtidos com o cenário futuro foram utilizados para avaliar a qualidade das previsões gerada pelo modelo. Estes dados permitiram correlacionar as métricas de complexidade e as medições de tempo para realizar a construção do modelo quantitativo (com os dados do cenário base), confrontar as previsões obtidas com os dados observados para o cenário futuro.

6.2.2 Construção do modelo quantitativo

Para criar o modelo quantitativo foram necessários tanto os resultados provenientes da análise de complexidade quanto os valores das medições de tempo obtidas durante a execução das ações, referentes ao cenário base. Para esta análise, o tempo em que o administrador permanecia ocioso (tempo de processamento) foi descartado e não considerado. A Figura 6.1 ilustra a complexidade, por ação, do procedimento referente ao cenário base. Nesta figura, as métricas de complexidade são agrupadas levando em consideração sua categoria (execução, parâmetro e memória), como visto na Seção 3.2.

O tempo gasto na execução das ações pode ser observado na linha contínua na Figura 6.2. Nesta figura, o eixo *x* indica o número da ação e o eixo *y* indica o tempo de interação gasto pelo administrador para executar a ação. Este valor se refere à média obtida dos tempos de execução dos três administradores.

Seguindo o processo de criação do modelo descrito na Seção 6.1.2, foi determinada a correlação entre cada métrica de complexidade e o tempo de execução das ações. A Figura 6.3 ilustra o valor de correlação (em módulo) entre as métricas e o tempo. A métrica *ParamSourceScore* possui uma correlação de 0,62 e é então adicionada ao modelo. É importante observar que o valor da correlação pode variar de -1 a 1. Os coeficientes de correlação relativamente altos indicam uma relação linear mais forte entre as métricas de complexidade e o tempo necessário para a execução, o que sugere que a criação do modelo quantitativo é viável. Incrementalmente, as métricas foram adicionadas ao modelo e a qualidade do mesmo foi observada em cada iteração.

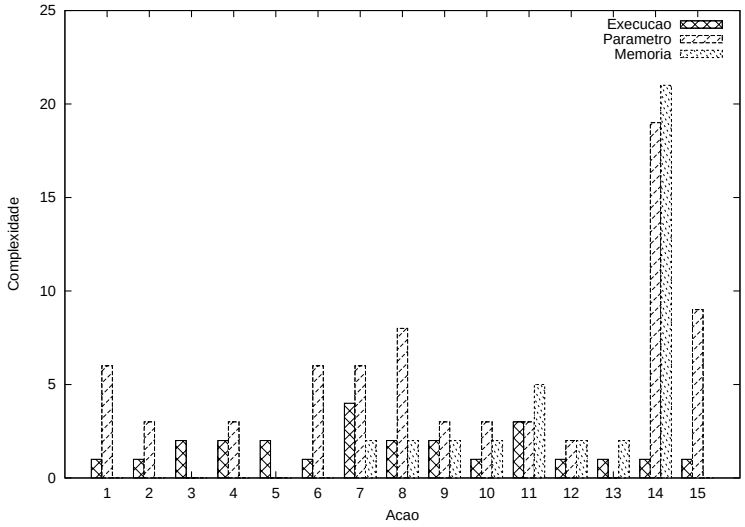

Figura 6.1: Análise de complexidade do procedimento base

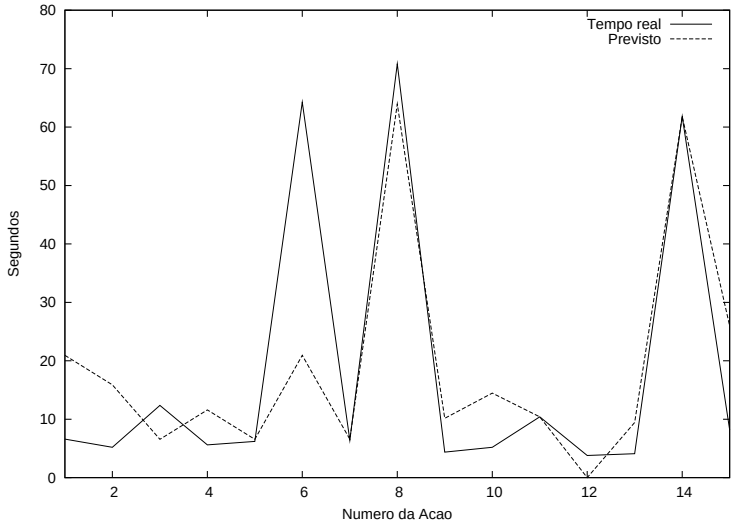

Figura 6.2: Modelo criado com o cenário base

A Tabela 6.1 sumariza a avaliação obtida para a qualidade do modelo, a medida que cada métrica era adicionada ao modelo a cada iteração. Na tabela, a primeira coluna indica o número da iteração, ao passo que a segunda coluna indica a métrica que foi adicionada ao modelo, em relação ao modelo anterior. Por sua vez, a terceira coluna indica o valor de R^2 para o modelo, a quarta coluna contém o valor de RMSE e, por fim, a última coluna indica a existência ou não de termos negativos. Por exemplo, na etapa 3 o modelo é composto de três métricas (*ParamSourceScore*, *MemLat* e *ParamUseCount*), apresenta um valor de R^2 igual a 0,68, um valor de RMSE igual a 15,02 segundos e possui parâmetros β_i negativos.

Analisando a tabela, pode-se notar que o valor de R^2 aumenta, como esperado, a medida que mais métricas são adicionadas ao modelo, ao passo que o valor de RMSE diminui. Entretanto, pode-se observar que o valor de R^2 é influenciado principalmente pelas primeiras métricas. Além disso, as métricas são ordenadas de acordo com sua contribuição para o modelo final, que difere da ordem de correlação da Figura 6.3. Apesar de a métrica *ParamUseCount* possuir um coeficiente de correlação com o tempo maior do

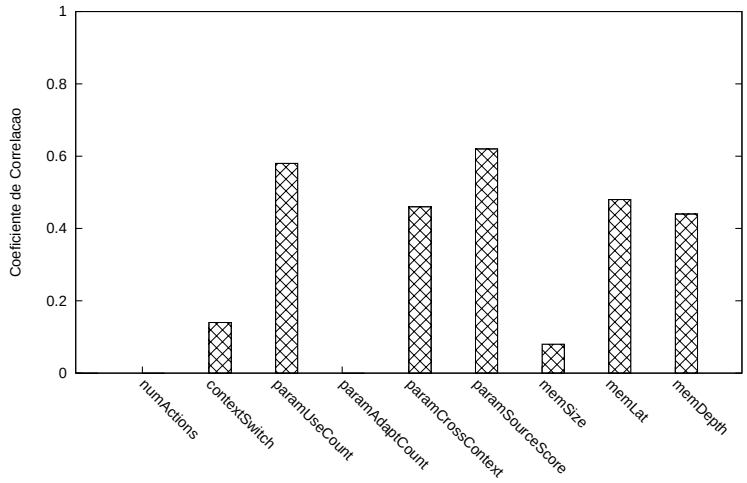

Figura 6.3: Correlação das métricas de complexidade com o tempo

Tabela 6.1: Evolução na qualidade do modelo a medida que mais métricas são adicionadas ao mesmo

Etapa de Iteração	*Métrica Adicionada*	R^2	*RMSE*	*Parâmetros β_i negativos?*
1	ParamSourceScore	0,61	18,70	Não
2	MemLat	0,65	17,70	Não
3	ParamUseCount	0,68	16,83	Sim
4	ParamCrossContext	0,75	15,02	Sim
5	MemDepth	0,75	15,02	Sim
6	ContextSwitch	0,75	14,95	Sim
7	MemSize	0,75	14,86	Sim
8	ParamAdaptCount	0,75	14,86	Sim
9	NumActions	0,78	14,06	Sim

que *MemLat*, ela apresenta uma correlação muito forte com *ParamUseCount*, o que acaba por restringir o uso da métrica na busca de um modelo de melhor qualidade. A partir da terceira etapa de iteração, termos com parâmetros negativos começam a surgir, indicando *overfitting*. Desta forma, a criação do modelo quantitativo deve parar na etapa 2 – que apresenta valores de R^2 e RMSE satisfatórios e não possui parâmetros β_i negativos. O modelo, então, apresenta a seguinte forma (observe que os parâmetros β_i são obtidos a partir do método dos mínimos quadrados ordinários):

$$y = 6,017 \times ParamSourceScore + 1,641 \times MemLat \tag{6.4}$$

Uma vez escolhido o modelo, decidiu-se plotar os resultados das previsões de tempo de execução geradas pelo mesmo para o cenário base. Estes resultados podem ser observados na linha pontilhada da Figura 6.2. Pode-se notar um *fit* considerável entre o tempo previsto e o observado. O valor de R^2 de 0,65 indica que este modelo é capaz de explicar 65% da variabilidade do tempo de execução do cenário base. Este modelo, por fim, é utilizado na próxima subseção, onde ele é extrapolado com o objetivo de prever o custo associado à implantação dos mecanismos de segurança do cenário futuro.

6.2.3 Extrapolação do modelo

Para validar o modelo quantitativo construído na subseção anterior, foi feita a extrapolação do mesmo para prever o tempo associado à execução do procedimento do cenário futuro. A Figura 6.4 ilustra o tempo de trabalho real (linha sólida) e o previsto pelo modelo (em linha pontilhada). Na figura, pode-se notar que a curva pontilhada, em geral, acompanha a mesma tendência descrita pela linha sólida (que descreve o medido de execução), o que caracteriza um *fit* considerável do modelo previsto em relação ao tempo real. Para avaliar analiticamente a qualidade do modelo, utilizou-se a função `postResample` presente no pacote CARET do R, conforme apresentado na Seção A.2. O modelo obtido apresentou um valor igual a 0,60 para R^2, que significa que o modelo foi capaz de explicar 60% da variabilidade do tempo. Já o valor de RMSE obtido foi de 17,31, ou seja, a diferença média entre os valores medidos e previstos é de 17,31 segundos em um total de 810,44 segundos.

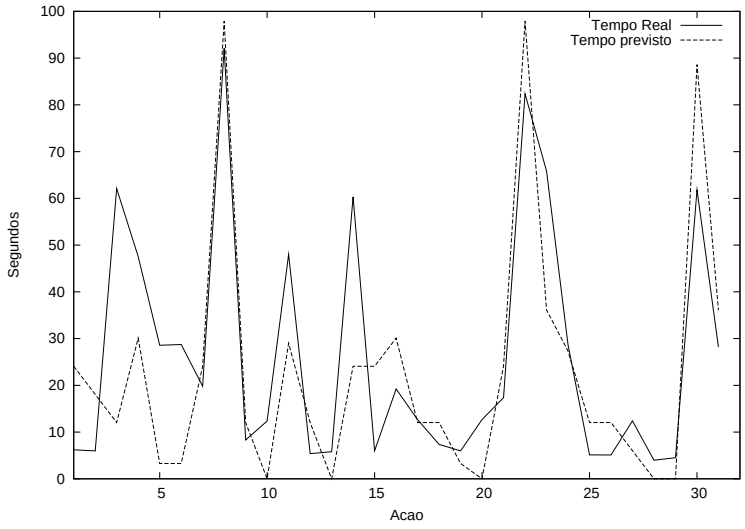

Figura 6.4: Validação do modelo quantitativo com a previsão dos custos associados à segurança

Pode-se notar na Figura 6.4 que o modelo foi capaz de prever os picos de tempo na ação 8 (diretamente relacionada com segurança), que se refere à requisição de assinatura do certificado SSL. Esta ação requer que sejam fornecidos diversos parâmetros (tais como nome do arquivo a ser criado, informações sobre a localização da organização, nome, email, administrador, país, estado e senha) manualmente, via linha de comando (*shell*). A mesma justificativa também vale para o pico da ação 22, onde é criada a requisição de assinatura do certificado SSL no servidor de aplicação. O valor de *ParamSourceScore* para estas ações contabiliza 16, sendo que são utilizado 9 parâmetros. Baseado nestes valores, o modelo é capaz de prever com certo grau de precisão o tempo gasto durante a execução desta ação.

Analisando as ações para as quais o modelo obteve um pior desempenho, pode-se notar que é na ação 3 que reside a maior diferença entre o valor observado e previsto. Esta ação se refere ao mapeamento da partição do disco a ser criptografada. Neste caso, o modelo previu um valor para esta atividade de 12,03 segundos (visto que são necessários um parâmetro apenas, de valor 2 de *ParamSourceScore*). Entretanto, no exercício em particular, a média observada de tempo gasto pelos participantes foi de 62,1 segundos.

Diferentemente das outras ações, nesta dois participantes tiveram de consultar documentação na Internet para executar manualmente os comandos para o mapeamento da partição a ser criptografada. Este problema – caracterizado como a diferença de experiência de cada administrador – tem impacto considerável na criação dos modelos quantitativos, e foi abordado no trabalho de Diao *et al.* (DIAO, Y. et al., 2007). No nosso estudo, foram escolhidos administradores com experiência na utilização dos softwares em questão. Entretanto, em uma entrevista após a análise dos dados, dois deles argumentaram que, apesar de saberem de forma geral dos comandos do dm-crypt, eles não se lembraram dos pormenores necessários para efetuar o mapeamento da partição criptografada. Estes resultados, apesar de influenciarem negativamente a previsão, refletem a realidade observada no cotidiano das organizações, onde administradores diferentes executam as ações em tempos diferentes. A solução para estes problemas é uma análise que permita categorizar os administradores segundo seu nível de *expertise* e calibrar o modelo de acordo com essas variações, o que constitui um tema para investigação em trabalhos futuros.

6.2.4 Previsão de custos de segurança em nível de procedimentos

Na subseção anterior foi apresentado como o modelo quantitativo pode ser utilizado para estimar o tempo de execução das ações individuais que compõem um procedimento. Entretanto, uma visão mais geral do tempo de execução pode ser obtida realizando a previsão no nível de procedimentos.

A Tabela 6.2 apresenta os resultados obtidos para a análise em nível de procedimentos dos cenários base e futuro. Os valores de tempo apresentados nas colunas foram obtidos através do somatório do tempo de execução das ações que compõem os procedimentos. Analisando a tabela, é possível observar que o tempo de execução do cenário base foi de 275,29 segundos, ao passo que do cenário futuro foi 810,44 segundos. Desta forma, os mecanismos de segurança ocasionaram um aumento de 194,43% no tempo de execução. O modelo obtido no cenário base gerou uma previsão de 711,65 segundos, ou seja, um aumento de 158,50% em relação ao tempo medido do cenário base. Em uma outra perspectiva, pode-se observar que o tempo previsto pelo modelo corresponde a 87,81% do tempo real observado para a execução procedimento (quociente obtido através da divisão do tempo previsto pelo estimado), o que fornece uma boa precisão na estimativa.

Tabela 6.2: Tempo previsto e estimado para os cenários base e futuro

Tempo	*Cenário Base*	*Cenário Futuro*
Medido	275,29	810,44 (194,43%)
Estimado	-	711,65 (158,50%)

6.3 Considerações parciais

Neste capítulo foi descrita uma abordagem para relacionar as métricas de complexidade com o tempo gasto na execução das ações pertinentes aos procedimentos. Para isso, foi desenvolvido um modelo quantitativo utilizando os valores associados a estas métricas e os tempos observados durante a execução do procedimento do cenário base (sem requisitos de segurança). O modelo foi determinado utilizando-se o método dos mínimos quadrados ordinários e, então, extrapolado para realizar a previsão do tempo

de execução das ações do cenário futuro (é importante observar que o custo de execução do procedimento pode ser obtido multiplicando-se o tempo de execução pelo valor da hora de trabalho de um administrador). Além disso, foi feita uma análise no nível de procedimentos. Os resultados obtidos sugerem a viabilidade da abordagem, uma vez que:

- levando em consideração as métricas de complexidade coletadas no cenário base, o modelo foi capaz de explicar 65% da variabilidade do tempo;
- quando o modelo foi extrapolado para estimar o tempo gasto no cenário com segurança, ele foi capaz de explicar 60% da variabilidade do tempo em questão;
- observou-se, assim como no trabalho de Diao *et at.* (DIAO, Y. et al., 2007), que apenas poucas métricas são suficientes para produzir um modelo sem afetar consideravelmente sua qualidade;
- foi possível identificar o custo adicionado pelos mecanismos de segurança no cenário futuro em relação ao cenário base, em nível de procedimentos: eles ocasionaram um aumento de 194,43% no tempo de execução e o modelo gerado foi capaz de prever um aumento de 158,80%.

7 CONCLUSÃO E TRABALHOS FUTUROS

Apesar do reconhecido papel ocupado por segurança na complexidade dos procedimentos de TI, esta tem sido caracterizada ao longo dos anos de maneira absolutamente intuitiva, sem o suporte de uma forma mais objetiva e mensurável (a exemplo do que se atingiu na área de Engenharia de Software para mensurar o "custo"associado ao desenvolvimento de sistemas). Em uma primeira iteração para tratar este problema, foi aplicado neste trabalho um modelo de complexidade de configuração proposto por Brown *et al.* (BROWN; KELLER; HELLERSTEIN, 2005) para compreender a extensão da influência de segurança nos procedimentos de TI. A avaliação de complexidade dessa proposta foi auxiliada pela SCA – uma ferramenta que automatiza a análise de complexidade de procedimentos de segurança – que permitiu realizar diversos experimentos. Além disso, a relação entre as métricas de complexidade e o tempo de execução dos procedimentos foi também investigada e determinada por meio de um modelo quantitativo. Este modelo permitiu realizar previsões acerca do tempo de execução e, indiretamente, dos custos relacionados à instalação/configuração e ou manutenção dos mecanismos de segurança.

Neste trabalho, foi proposta uma abordagem para se determinar a complexidade induzida pelos mecanismos de segurança em três dimensões distintas:

- complexidade adicionada por mecanismos de segurança em procedimentos de TI mais gerais;
- medidas de complexidade associadas à execução de procedimentos relacionados somente à segurança;
- comparação de complexidade entre ferramentas que implementam os mesmos mecanismos de segurança.

Pode-se mencionar importantes conclusões aprendidas no contexto desta dissertação. Primeiro, é difícil isolar e determinar a complexidade dos mecanismos de segurança quando estes permeiam procedimentos de TI mais gerais. A razão disto é (*i*) que os valores calculados para as métricas (especialmente as do grupo de parâmetro e de memória) são muito sensíveis à ordem em que as ações são executadas e (*ii*) devido ao fato de existirem ações de segurança em contêineres que não possuem relação com segurança (como as ações cinzas presentes na Figura 3.2). Para superar tais empecilhos, foi utilizada uma abordagem sistemática e objetiva, descrita na Seção 5.1.

Outra conclusão foi que as medidas de complexidade calculadas para os mecanismos de segurança são fortemente dependentes da infra-estrutura disponível e, devido a isto, não podem ser facilmente generalizadas para outros cenários. Por exemplo, a complexidade associada à instalação do OpenSSL pode ser diferente dependendo do sistema operacional utilizado. Isto reflete a experiência relatada pelos administradores.

Alem disso, em se tratando de realizar previsões quanto ao tempo de execução dos procedimentos de segurança, duas métricas de complexidade (dentre as nove) foram suficientes para criar um modelo quantitativo para os cenários avaliados. Isto sugere que talvez seja possível realizar simplificações no modelo de segurança de TI, o que levaria a uma maior adoção do mesmo. Por fim, foi possível constatar o impacto no tempo de execução dos procedimentos ocasionado pela diferença do nível de experiência dos participantes. Entretanto, apesar destas diferenças, o modelo foi capaz de gerar previsões com uma precisão aceitável.

Os resultados obtidos com a análise de complexidade foram condensados na forma de um artigo, entitulado *Applying a Model of Configuration Complexity to Measure Security Impact on IT Procedures*[1]. Este artigo foi publicado em uma das principais conferências de Gerência de Redes de Computadores, de padrão Qualis A Internacional, o *IFIP/IEEE Network Operations and Management Symposium* (NOMS 2008) (MOURA; GASPARY, 2008), em Salvador, Brasil (uma cópia deste artigo pode ser vista no Apêndice B).

Em relações a trabalhos futuros, pode-se citar algumas sugestões para posteriores investigações a serem realizadas pelo grupo de pesquisa. Em se tratando do modelo quantitativo, primeiramente deve ser feito um estudo mais amplo, avaliando diversos cenários envolvendo diferentes requisitos de segurança, bem como procedimentos com mais ações e parâmetros. Além disso, um estudo sobre os administradores deve ser conduzido de forma que seja possível classificar os mesmos de acordo com seu nível de experiência e determinar como esta experiência afeta os resultados de tempo. Mais que isso, talvez seja necessário propor novas métricas para capturar detalhes ainda não observados durante a execução de procedimento, através de entrevistas com administradores e observações durante a execução de procedimentos. Por fim, este modelo deve ser estendido para ser possível determinar o custo de procedimentos em ambientes distribuídos, onde mais de um administrador opera para alcançar os objetivos de segurança.

Como citado no Capítulo 5, há ainda algumas melhorias a serem implementadas na ferramenta SCA em futuros trabalhos, principalmente com relação à usabilidade. Por exemplo, a implementação do componente *GUI* tornará mais amigável a interação entre o usuário e a SCA, possibilitando, dentre outras facilidades, o cadastramento de procedimentos diretamente na SCA (ao invés de editar um arquivo XML manualmente). Além disso, a *GUI* poderia, automaticamente, realizar as coletas dos tempos associados à execução de cada ação dos procedimentos, necessários para a criação do modelo quantitativo.

Uma outra integração vantajosa seria entre a SCA e a ferramenta R (R, 2008), utilizada neste trabalho para criar o modelo quantitativo de previsão de custos descrito no Capítulo 6. Neste estudo, a criação do modelo e a extrapolação do mesmo ocorreu de forma externa à SCA, utilizando diretamente a R. Uma integração entre a R e o módulo *Viewer* permitiria ao administrador realizar toda a análise de complexidade e estimativa de custos dentro de um único ambiente integrado.

[1] Além deste artigo, durante o primeiro ano de mestrado na UFRGS, tivemos outro artigo aprovado em outra conferência Qualis A internacional de Gerência de Redes. O artigo, intitulado *On the Performance of Web Services Management Standards for Network Management - An Evaluation of MUWS and WS-Management* (MOURA et al., 2007), foi apresentado durante o *IFIP/IEEE International Symposium on Integrated Network Management (IM)* em Munique, Alemanha. Ele trata da análise de desempenho de *web services* padronizados quando aplicados a gerência de redes.

REFERÊNCIAS

BECKER, R. A.; CHAMBERS, J. M.; WILKS, A. R. **The new S Language**: a Programming Environment for Data Analysis and Graphics. [S.l.: s.n.], 1998.

BRETSCHER, O. **Linear Algebra with Applications**. 3rd ed. New York: Prentice Hall, 2004.

BROWN, A. B.; HELLERSTEIN, J. L. An approach to Benchmarking Configuration Complexity. In: ACM SIGOPS EUROPEAN WORKSHOP, 2004, Leuven, Belgium. **Proceedings...** New York: ACM Press, 2004. p.18.

BROWN, A. B.; KELLER, A.; HELLERSTEIN, J. L. A Model of Configuration Complexity and its Application to a Change Management System. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2005, Nice, France. **Proceedings...** Piscataway:IEEE, 2005. p.631–644.

CANNON, D.; WHEELDON, D. **Service Operation Itil, Version 3 (Itil)**. UK: Stationery Office, 2007.

CARET: Classification and Regression Training. Disponível em: <http://cran.r-project.org/src/contrib/Descriptions/caret.html>. Acesso em: 15 fev. 2008.

CAVUSOGLU, H.; MISHRA, B.; RAGHUNATHAN, S. A Model For Evaluating IT Security Investments. **Communications of ACM**, New York, NY, USA, v.47, n.7, p.87–92, July 2004.

CIS: Center for Internet Security. Disponível em: <http://www.cisecurity.org/>. Acesso em: 15 fev. 2008.

COUCH, A. L.; WU, N.; SUSANTO, H. Toward a Cost Model for System Administration. In: LARGE INSTALLATION SYSTEM ADMINISTRATION , LISA, 2005, Monterey,CA,USA. **Proceedings...** [S.l.]: USENIX Association, 2005.

CRAN: The Comprehensive R Archive Network. Disponível em: <http://cran.r-project.org/>. Acesso em: 15 fev. 2008.

DIAO, Y. et al. Generic On-line Discovery of Quantitative Models for Service Level Management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 8., 2003, Colorado, USA. **Integrated Network Management VIII: Managing it all**. Boston: Kluwer Academic, 2003. p.157–170.

DIAO, Y.; KELLER, A. Quantifying the Complexity of IT Service Management Processes. In: IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, 2006, Dublin, Ireland. **Proceedings...** [S.l.]: IEEE, 2006.

DIAO, Y. et al. Predicting Labor Cost throught IT Management Complexity Metrics. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2007, Munich, Germany. **Proceedings...** [S.l.]: IEEE, 2007.

DM-CRYPT. Disponível em: <http://www.saout.de/misc/dm-crypt/>. Acesso em: 15 fev. 2008.

ECLIPSE. Disponível em: <http://www.eclipse.org/>. Acesso em: 20 fev. 2008.

FRANK E.; HARRELL, J. **Regression Modeling Strategies**. Secaucus, NJ, USA: Springer-Verlag, 2006.

GPLV2: The GNU General Public License, Version 2. Disponível em: <http://www.fsf.org/licensing/licenses/info/GPLv2.html>. Acesso em: 20 fev. 2008.

IHAKA, R.; GENTLEMAN, R. R: a Language for Data Analysis and Graphics. **Journal of Computational and Graphical Statistics**, [S.l.], v.5, n.3, p.299–314, 1996.

JDOM. Disponível em: <http://www.jdom.org>. Acesso em: 20 fev. 2008.

JOOMLA. Disponível em: <http://www.joomla.org/>. Acesso em: 15 fev. 2008.

KELLER, A.; BROWN, A. B.; HELLERSTEIN, J. L. A Configuration Complexity Model and Its Application to a Change Management System. **IEEE Transactions on Network and Service Management**, [S.l.], v.4, n.1, p.13–27, 2007.

LJUNG, L. **System Identification**: Theory for the User. 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

MILLER, G. A. The Magical Number Seven, Plus or Minus Two: some limits on our capacity for processing information. **The Psychological Review**, [S.l.], v.63, p.81–97, 1956.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied Statistics and Probability for Engineers**. 4th ed. [S.l.]: John Wiley & Sons, 2006.

MOURA, G. C. M.; GASPARY, L. P. Applying a Model of Configuration Complexity to Measure Security Impact on IT Procedures. In: IFIP/IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 11., 2008, Salvador, Brazil. **Proceedings...** [S.l.:s.n.], 2008. p.97–104.

MOURA, G. C. M.; SILVESTRIN, G.; SANCHEZ, R. N.; GASPARY, L. P.; GRANVILLE, L. Z. On the Performance of Web Services Management Standards - An Evaluation of MUWS and WS-Management for Network Management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 10., 2007, Munich, Germany. **Proceedings...** [S.l.]: IEEE, 2007. p.459–468.

NETFILTER/IPTABLES. Disponível em: <http://www.netfilter.org/>. Acesso em: 15 fev. 2008.

OPENSSL. Disponível em: <http://www.openssl.org>. Acesso em: 15 fev. 2008.

OPENSWAN. Disponível em: <http://www.openswan.org/>. Acesso em: 15 fev. 2008.

OPENVPN. Disponível em: <http://www.openvpn.org/>. Acesso em: 15 fev. 2008.

PATTERSON, D. A Simple Way to Estimate the Cost of Downtime. In: LARGE INSTALLATION SYSTEM ADMINISTRATION, LISA, 2002, Philadelphia,PA,USA. **Proceedings...** [S.l.]: USENIX, 2002. p.185–188.

R. Disponível em <http://www.r-project.org/>. Acesso em: 20 fev. 2008.

SPEC: Standard Performance Evaluation Corporation. Disponível em: <http://www.spec.org/>. Acesso em: 5 fev. 2008.

STALLINGS, W. **Network Security Essentials**: Applications and Standards. 3rd ed. [S.l.]: Prentice Hall, 2006.

ZHANG, J.; MORRIS, J. Process Modelling and Fault Diagnosis Using Fuzzy Neural Networks. **Fuzzy Sets Syst.**, Amsterdam, The Netherlands, v.79, n.1, p.127–140, 1996.

ZUSE, H. **Software Complexity**: Measures and Methods. Hawthorne, NJ, USA: Walter de Gruyter & Co., 1991.

APÊNDICE A ANÁLISE DE REGRESSÃO LINEAR

A.1 Introdução à análise de regressão

Diversos métodos de modelagem de sistemas foram estudados e aplicados na resolução de problemas de áreas como identificação de sistemas e aproximação de funções (LJUNG, 1999; ZHANG; MORRIS, 1996). Entretanto, a maioria destes modelos são voltados à investigação de relações de entrada e saída de variáveis estritamente quantitativas (tais como tempo de resposta, atraso, etc.). As métricas apresentadas na Seção 3.2, apesar de possuírem uma escala e intervalos definidos (como variáveis quantitativas), possuem uma forte relação qualitativa e são representadas com descrições categóricas, como observado por Diao *et al.* (DIAO, Y. et al., 2007), o que sugere o uso de outras técnicas de modelagem de sistemas.

A técnica utilizada neste trabalho foi a análise de regressão múltipla (FRANK E.; HARRELL, 2006), devido ao fato de modelos lineares apresentarem a tendência de serem mais robustos do que modelos não lineares (como modelos utilizando redes neurais), especialmente com o uso de múltiplas variáveis (DIAO, Y. et al., 2007). Por definição, análise de regressão (categoria que engloba regressão linear simples e múltipla) é uma técnica de investigação da relação entre a variável dependente (ou variável de resposta) e variáveis independentes (ou variáveis explanatórias).

Em regressão linear, a relação entre as variáveis explanatórias e a variável dependente é representada pela equação de regressão, que possui a seguinte forma:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{A.1}$$

Nesta equação, as variáveis explanatórias são representadas pelos termos x_i, ao passo que a variável dependente é representada pelo termo y. A relação entre elas é caracterizada pelos parâmetros β_i, que indicam quantitativamente como a variável dependente pode ser interpretada linearmente através das variáveis explanatórias (FRANK E.; HARRELL, 2006).

Neste trabalho, o uso principal deste modelo é prever o custo associado ao uso de mecanismos de segurança através das métricas de complexidade de TI. Para isso, a variável dependente (a ser estimada) é o tempo de execução do procedimento. As variáveis explanatórias, neste estudo, são as métricas apresentadas na Seção 3.2.

Nas próximas subseções é descrito como pode ser realizado o cálculo dos coeficientes β_i da equação de regressão. Além disso, é mostrado como a qualidade de um modelo de regressão linear pode ser avaliada..

A.1.1 Método dos Mínimos Quadrados Ordinários

Para se estimar os valores dos parâmetros β_i no modelo quantitativo da Equação A.1, utiliza-se o método dos *mínimos quadrados ordinários*, descrito primeiramente por Gauss em 1794, quando este tinha apenas 18 anos (BRETSCHER, 2004; LJUNG, 1999). O método consiste em uma técnica de otimização matemática que procura encontrar o melhor ajuste para um conjunto de dados, com o objetivo de minimizar a soma dos quadrados das diferenças entre os dados reais e os valores previstos pelo modelo (tais diferenças são chamadas resíduos).

Suponha que existam $n > k$ observações de um experimento, e que x_{ij} seja a *i*ésima observação da variável x_j. As observações são:

$$(x_{i1}, x_{i2}, ..., x_{ik}, y_i), \quad i = 1, 2, ..., n \quad \text{e} \quad n > k$$

Cada observação $(x_{i1}, x_{i2}, ..., x_{ik}, y_k)$ satisfaz a seguinte equação:

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_k x_{ik} + \varepsilon_i \\ &= \beta_0 + \sum_{i=1}^{n} \beta_j x_{ij} + \varepsilon_i \qquad i = 1, 2, ..., n \end{aligned} \tag{A.2}$$

A equação dos quadrados mínimos ordinários, então, é:

$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ij})^2 \tag{A.3}$$

O método busca minimizar L com respeito a $\beta_0, \beta_1, ..., \beta_k$. A estimativa dos quadrados mínimos de $\beta_0, \beta_1, \beta_k$ deve satisfazer a equação:

$$\frac{\partial L}{\partial \beta_0}|_{\hat{\beta}_0, \hat{\beta}_1, ... \hat{\beta}_k} = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ij}) = 0 \tag{A.4}$$

e a equação:

$$\frac{\partial L}{\partial \beta_j}|_{\hat{\beta}_0, \hat{\beta}_1, ... \hat{\beta}_k} = -2 \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{k} \hat{\beta}_j x_{ij}) x_{ij} = 0 \qquad j = 1, 2, ..., k \tag{A.5}$$

Simplificando as Equações A.4 e A.5, as equações normais dos quadrados mínimos ficam da seguinte forma:

$$\begin{aligned} \sum_{i=1}^{n} (y_i) &= n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_{i1} + \hat{\beta}_2 \sum_{i=1}^{n} x_{i2} + ... + \hat{\beta}_k \sum_{i=1}^{n} x_{ik} \\ \sum_{i=1}^{n} x_{i1} y_i &= \hat{\beta}_0 \sum_{i=1}^{n} x_{i1} + \hat{\beta}_1 \sum_{i=1}^{n} x_{i1}^2 + \hat{\beta}_2 \sum_{i=1}^{n} x_{i1} x_{i2} + ... + \hat{\beta}_k \sum_{i=1}^{n} x_{i1} x_{ik} \\ \sum_{i=1}^{n} x_{ik} y_i &= \hat{\beta}_0 \sum_{i=1}^{n} x_{ik} + \hat{\beta}_1 \sum_{i=1}^{n} x_{ik} x_{i1} + \hat{\beta}_2 \sum_{i=1}^{n} x_{ik} x_{i2} + ... + \hat{\beta}_k \sum_{i=1}^{n} x_{ik}^2 \end{aligned} \tag{A.6}$$

Pode-se notar que há $p = k+1$ equações normais, uma para cada um dos coeficientes de regressão desconhecidos. A solução destas equações normais é o valor de quadrados mínimos dos coeficientes de regressão β_0, β_1, ..., β_k. Estas equações podem ser resolvidas por quaisquer métodos de resolução de equações de sistemas lineares. Assumindo que os dados obedecem a uma distribuição normal, este método fornece um valor estimado imparcial dos parâmetros β_i do modelo. Atualmente diversos softwares estatísticos possuem este método implementado. No nosso estudo, foi utilizado o R, que é descrito na Seção A.2.

A.1.2 Avaliação da qualidade de modelos de regressão linear

O coeficiente de determinação R^2 é utilizado para julgar a adequação de um modelo de regressão. Ele é definido pela seguinte equação:

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T} \tag{A.7}$$

onde:

- $SS_T = \sum_{i=1}^{n}(y_i - \overline{y})^2$ é a soma quadrática corrigida total
- $SS_E = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ é a soma quadrática dos erros
- $SS_R = \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2$ é a soma quadrática dos resíduos

Desta forma, a Equação A.7 pode ser reescrita da seguinte forma:

$$R^2 = 1 - \frac{var(y - \hat{y})}{var(y)} \tag{A.8}$$

onde $var()$ denota a variância. O valor de R^2 quantifica a quantidade da variabilidade da variável dependente que foi capturada pelo modelo. Um valor de R^2 igual a 1 indica que o modelo foi capaz de capturar toda a variabilidade e um valor de R^2 igual a 0 indica que modelo não foi capaz de capturar a variabilidade.

Outra métrica a ser utilizada em conjunto com R^2 é o erro médio quadrático, definido como:

$$RMSE = \sqrt{\frac{1}{k}\sum_{k=1}^{K}(y(k) - \hat{y}(k))^2} \tag{A.9}$$

Um modelo que apresente um valor alto para R^2 e um valor baixo de RMSE indica um mapeamento consistente entre as métricas de complexidade e tempo de execução das ações e procedimentos (DIAO, Y. et al., 2007). Por outro lado, um modelo de baixa qualidade pode indicar problemas na avaliação das métricas de complexidade ou uma previsão muito diferente do real devido a um estudo incompleto do comportamento do usuário que, no caso, é o administrador.

A.2 Ferramenta utilizada: R

A ferramenta escolhida para auxiliar a criação dos modelos quantitativos foi o R (IHAKA; GENTLEMAN, 1996; R, 2008). O R é software livre (licenciado sob os termos da *GNU General Public License Version 2* (GPLv2)) (GPLV2, 2008) e pode ser executado

em diversos sistemas operacionais, como Linux, Unix, FreeBSD, MacOS e Windows. O R pode ser visto não só como um software em si, mas também como uma implementação da linguagem de programação estatística S (BECKER; CHAMBERS; WILKS, 2008), que se tornou o padrão *de facto* entre os estatísticos no desenvolvimento de aplicações para análise de dados e como um ambiente para o desenvolvimento de técnicas estatísticas. O R possui bibliotecas que suportam diversas abordagens estatísticas: modelagem linear e não-linear, testes estatísticos, análise de séries temporais, entre outras. Além disso, o R é bem conhecido pelo suporte à geração de gráficos de alta qualidade.

Na instalação padrão do R são incluídos um conjunto de pacotes denominados '*recommended packages*", que contém um conjunto de bibliotecas e funções para análise estatística. Além desses pacotes padrão, há ainda, atualmente, mais de 1200 outros pacotes (denominados "*contributed packages*"), que são desenolvidos pela comunidade, geralmente voltados para casos estatísticos específicos, muitas vezes representando o estado da arte em diversas áreas da estatística computacional. Esses pacotes são disponibilizados através do *Comprehensive R Archive Network* (CRAN) (CRAN, 2008).

Durante este estudo foi utilizado o *recommended package* `stats` para a criação dos modelos quantitativos, uma vez que o mesmo possui implementado o método dos mínimos quadrados ordinários, como definido na Seção A.1.1. Já para avaliar a qualidade dos modelos gerados, isto é, para calcular o valor de R^2 e RMSE, como definido na Seção A.1.2, foi utilizado o *contributed package Classification and Regression Training* (`caret`) (CARET, 2008).

Outra vantagem em se utilizar o R é que ele também possui uma lingaguem de programação, o que torna possível realizar a integração com a ferramenta SCA. Isto perimitiria ao administrador executar a análise de complexidade e, em seguida, realizar a calibração dos modelos quantitativos para predizer os custos associados à instalação de mecanismos de segurança em um mesmo ambiente – a camada de apresentação do SCA – como pode ser visto na Figura 4.1. Para isso, bastaria desenvolver programas em R que recebessem os resultados das análise de complexidade e gerassem os modelos quantitativos. Além disso, o R também pode ser configurado para gerar os gráficos de complexidade (como histogramas) de procedimentos para que possam ser exibidos ao usuário no módulo *Viewer*.

APÊNDICE B ARTIGO PUBLICADO

Neste apêndice é apresentado o artigo "*Applying a Model of Configuration Complexity to Measure Security Impact on IT Procedures*" desenvolvido durante o mestrado. O artigo é resultado do segundo ano de mestrado e apresenta uma análise de complexidade para estimar o impacto que os requisitos segurança acabam por agregar nos procedimentos de TI. Ele apresenta também o medida de complexidade associada a manipulação de ferramentas de segurança isoladas de procedimentos mais gerais bem como uma análise comparativa de duas ferramentas que implementam os mesmos mecanismos de segurança numa perspectiva de complexidade.

- Títutlo: *Applying a Model of Configuration Complexity to Measure Security Impact on IT Procedures*
- Nome: IFIP/IEEE Network Operations and Management Symposium (NOMS)
- URL: http://www.noms2008.org
- Data: De 7 a 11 de abril de 2008
- Local: Salvador, Bahia, Brasil.
- Páginas: 97-104
- ISBN: à definir
- Localizador IEEE Xplore: a definir

Applying a Model of Configuration Complexity to Measure Security Impact on IT Procedures

Giovane César Moreira Moura, Luciano Paschoal Gaspary
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brazil
Email: {gcmmoura, paschoal}@inf.ufrgs.br

***Abstract*— IT security has become over the recent years a major concern for organizations. However, it doesn't come without large investments on both the acquisition of tools to satisfy particular security requirements and complex procedures to deploy and maintain a protected infrastructure. The scientific community has proposed in the recent past models and techniques to estimate the complexity of configuration procedures, aware that they represent a significant operational cost, often dominating total cost of ownership. However, despite the central role played by security within this context, it has not been subject to any investigation so far. To address this issue, we apply a model of configuration complexity proposed in the literature in order to be able to estimate security impact on the complexity of IT procedures. Our proposal has been materialized through a prototypical implementation of a complexity scorer system called Security Complexity Analyzer (SCA). To prove concept and technical feasibility of our proposal, we have used the SCA to evaluate real-life security scenarios.**

I. Introduction

Security has become over the recent years a major concern for organizations, which increasingly rely on complex IT infrastructures to run most of their current business transactions [1]. In this context, both *physical* and *logical* security are carefully introduced into daily operations in order to achieve different levels of information protection. Confidentiality, authenticity, integrity, non-repudiation, access control, and availability [2] are examples of logical security services that IT directors and managers may demand in order to preserve sensitive data maintained by Configuration Items (CI) within the IT infrastructure.

The demands (or requirements), expressed by means of Service Level Agreements (SLA) and/or security policies, are materialized through a bunch of mechanisms such as file system/network traffic encryption, packet filtering, and hardware/software redundancy. These mechanisms, in turn, can be implemented with tools released by different vendors. Installation, configuration, and maintenance of such tools are performed either interleaved in more general procedures (e.g., configuration of a web-based application) or in procedures targeted to deploy the security tools themselves.

Intuitively, the more the security mechanisms to be handled, the more complex, long, and costly the corresponding procedures tend to become. For example, as the number of mechanisms required for the installation of an operating system is increased, it is expected that the procedure presents a larger number of execution steps and demands additional parameters to be supplied and remembered. Besides, there are situations where a mechanism can be deployed using different tools (for example, user authentication through OpenLDAP or Microsoft's Active Directory), which may also differ in regards to deployment complexity.

Determining a *complexity measure* to estimate the cost of IT security is, in this context, fundamental for several reasons, out of which we highlight three. First, it can be used by directors/managers to revise SLAs and policies taking into account the predicted costs associated to their real implementation. Second, it provides the IT staff with guidance for choosing the most suitable tools (e.g., the less cumbersome ones) to implement specific mechanisms. Third, it allows complexity spots present within procedures related to security mechanisms to be identified and classified as candidates for automation [3].

The scientific community has proposed in the recent past models and techniques to estimate the complexity of configuration procedures and establish its relationship to business-level performance metrics like labor cost and time [4]. However, despite the central role played by security within this context, it has not been subject to any investigation so far.

In this paper we apply the methodology introduced by Keller *et al.* [3], with some extensions, in order to be able to both assess the potential impact that security tasks may cause to the overall complexity of IT-related procedures and determine/compare complexity measures associated to the manipulation of tools that implement specific security mechanisms. Our proposal has been materialized through a prototypical implementation of a complexity scorer system called Security Complexity Analyzer (SCA). To prove concept and technical feasibility of our proposal, we have used the SCA to evaluate real-life security scenarios.

The reminder of this paper is organized as follows. Section II discusses related work. Section III introduces the IT security model that abstracts the interaction between the system administrator and the hardware and software to be deployed. Section IV details the set of metrics used to measure both procedure and task complexity. Section V emphasizes implementation aspects of SCA. Section VI presents results obtained from experiments performed using SCA. Section VII concludes the paper with remarks and perspectives for future work.

II. Related Work

In recent years, some research efforts have been conducted towards proposing a measure to estimate complexity of both IT-related procedures and processes. Despite the recognized potential impact of security on complexity indicators, this issue has not been addressed in previous investigations. In parallel, however, different perspectives of IT security have indeed been analyzed. Reflecting these research developments, this section firstly revisits the most prominent investigations on general IT procedure/process complexity assessment and then discusses important contributions focused on security.

Brown and Hellerstein [5] have proposed a conceptual framework for configuration complexity benchmarking, which has been further developed in subsequent studies. In the first of them – carried out by Brown *et al.* [3] – it has been proposed a set of metrics, inspired on software engineering ones, to abstract the complexity of configuration procedures. The authors have demonstrated that this set of metrics was able to capture the decrease of complexity associated to a configuration procedure after some of its tasks were automated. A step further was taken by Diao *et al.* [4], where the metrics proposed in the aforementioned work were used as input for a quantitative model to derive business-level performance metrics like labor cost and time. Finally, Keller *et al.* have described in [6] the tooling developed to support the capture and analysis of complexity data.

Looking at the problem from a business perspective, Diao and Keller [7] extended the metrics proposed in [3] to be able to quantify the complexity of IT service management processes [8]. In contrast to procedures, which are directly related to the system level (e.g., installation, configuration, and maintenance of CIs), processes are associated to the business level. In this context, the new metrics capture aspects such as decision making among multiple roles and interaction between two or more roles.

In regards to IT security, to our knowledge there is no existing work on complexity measures to estimate how procedures may be affected by the manipulation of different security mechanisms. Instead, for example, CIS Security Benchmarks [9] are intended to assess the degree in which an already deployed system is protected against an extensive list of threats. Concerning economic aspects of information security, Cavusoglu *et al.* [10] have introduced an economic model to estimate the optimal amount to invest in information security, which takes as input, among other variables, the expected damages due to security breaches, as well as quality parameters and deployment costs of candidate security mechanisms. The rationale behind the estimation of these deployment "costs" – problem that could be compared to this work – is not described in the paper.

In the following sections we describe our proposal to evaluate the complexity of security within IT-related procedures.

III. IT Security Model

The IT security model allows one to express an abstraction of the enterprise IT infrastructure – including the required security mechanisms – to be deployed or modified, as well as the procedures to accomplish that. Based on the proposal by Brown *et al.* [3], this model is composed of two complementary parts: *infrastructure* and *activity*. The former represents the hardware/software components and the degree in which they are related to security services. The latter abstracts the sequence of activities that must be executed in order to achieve a particular goal.

To illustrate the model, we introduce an example scenario that consists of the deployment of the web application Joomla [11] – which is a modular open source content management system developed in PHP – and the software upon which it depends (e.g., MySQL and Apache's httpd). The scenario is composed of two machines (one hosts the database server and the other hosts the web server) and a network firewall that controls the access to them (since they are located within a demilitarized zone). In addition, the environment is enriched with a set of security mechanisms as summarized in Table I. For example, Dm-crypt [12] and OpenSSL [13] are employed to encrypt the file system of Machine 1. Henceforth we provide more details about the IT security model.

TABLE I
Security mechanisms employed in the example scenario

Machine 1

Security Mechanism	*Tools*
File system encryption	dm-crypt and OpenSSL
Database connection encryption	MySQL and OpenSSL

Machine 2

Security Mechanism	*Tools*
File system encryption	dm-crypt and OpenSSL
Web server connection encryption	httpd and OpenSSL

Firewall

Security Mechanism	*Tools*
Packet filter	netfilter/iptables

A. IT infrastructure model

The IT infrastructure can be modeled as a set of *containers*, which represent either system resources or hosting containers. Figure 1 depicts the IT infrastructure model for the example scenario previously introduced. Container $Machine1$ is a representation of the computer responsible for hosting the database server, while $Machine2$ denotes the computer that hosts the web application. In turn, these containers host other ones, such as $Slackware\_app$ and $Slackware\_bd$ (operating system of both machines). Note that solid and dashed lines are used, respectively, to distinguish hosting containers from those representing system resources (e.g., $MySQL\ tables$).

In the same figure, the color of containers denotes how they are related with security. Actually, this represents an extension to the original IT infrastructure model proposed in [3]. The black containers are those whose main functionality is directly related to security mechanisms (e.g., $Dm\text{-}crypt\_db$, responsible for file system encryption in $Machine\_2$). In contrast, containers represented in white are not affected when

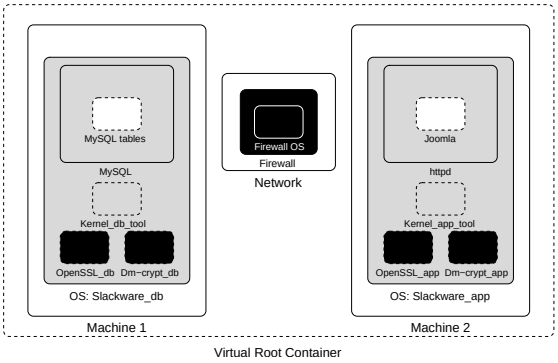

Fig. 1. IT infrastructure model

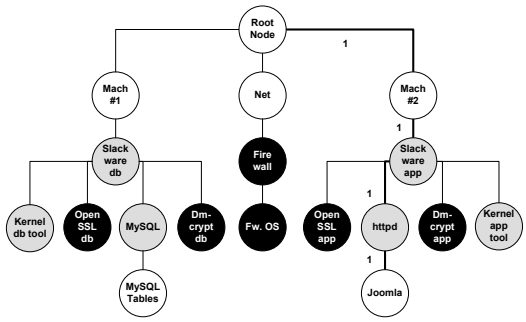

Fig. 3. Container hierarchy tree.

security mechanisms are deployed (e.g., $MySQL\ tables$). At last, gray containers characterize CIs, which although not related to security, may need to be changed in order to support security mechanisms demanded elsewhere. For example, MySQL database server requires special setup to support connection encryption (through OpenSSL).

B. Activity model

The Activity model provides an abstraction for the interaction between the administrator and the system to be deployed, and is based on three pillars: *goals*, *procedures*, and *actions* (or *tasks*). Goal can be defined as the state to be reached by an IT infrastructure (e.g., activation of SSL protocol for web server connections). Procedure is a series of steps that must be followed, in a regular definite order, to achieve a specific goal. Finally, action denotes an individual step of a procedure (e.g., creation of a private key).

Figure 2 illustrates the procedure to deploy Joomla enhanced with the security mechanisms specified in Table I (this procedure is subject of complexity measurements later on in Section VI). It is composed of 77 actions (represented by boxes). Each action is characterized by a title and the container it is associated to (in parentheses). The parameters consumed by an action are indicated by dotted arrows directed to the corresponding box.

In addition, actions are represented in black, white, and gray, following the same semantics applied to the IT infrastructure model. Black and white containers are expanded into black and white actions, respectively. Gray containers expand into both gray and white actions, where the former represent those aggregated to support the operation of security mechanisms. For example, the installation of $Slackware\_db$ (represented as a gray container in the infrastructure model) is composed of actions related *to security mechanisms* (16–22 and 25–27) and ordinary ones (e.g., 1–15). At a first glance, one could question, for example, why action 73 in Figure 2 is not represented in black, since it consists of informing database authentication credentials (db_name and db_passwd). The reasons for this is that this action is not related to the implementation of any security mechanism specified in Table I. Moreover, it is not an optional action, since one must mandatorily execute it in order to deploy Joomla – with or without any security mechanism.

IV. Complexity Metrics

The model introduced in the previous section must be employed in conjunction with a set of metrics that are able to capture the complexity associated to the abstracted IT-related procedures. In this section, we revisit these metrics, which are organized in three groups: *execution*, *parameter*, and *memory complexity*, borrowed from Brown *et al.* [3]. As a first attempt to assess the complexity associated to security mechanisms, we decided not to come up with new metrics, but to further explore those successfully employed in the context of general configuration procedures.

The complexity quantification process can be executed in both *procedure* and *task* levels. The former provides a general measure of procedure complexity, while the latter provides a more detailed view, focused on each task. Next we review the metrics and how they are applied in each level.

A. Procedure-level complexity

1) Execution complexity: This group of metrics evaluates the complexity related to the procedure execution. They are defined as follows.

- *NumActions*: number of actions of a procedure.
- *ContextSwitchSum*: sum of values associated to each *ContextSwitch* present within a procedure. A context switch occurs when the container of an action differs from the container of the previous one, such as actions 74 and 75 in Figure 2. The value associated to a context switch is calculated as the distance between both containers, considering they are organized in a hierarchy tree. For example, in Figure 3 one can observe that the value associated to the context switch between actions 74 and 75 is 4.

2) Parameter complexity: This subset of metrics is used to measure the complexity associated to the task of providing a procedure with parameters (by either the human operator or the system itself). The metrics are the following.

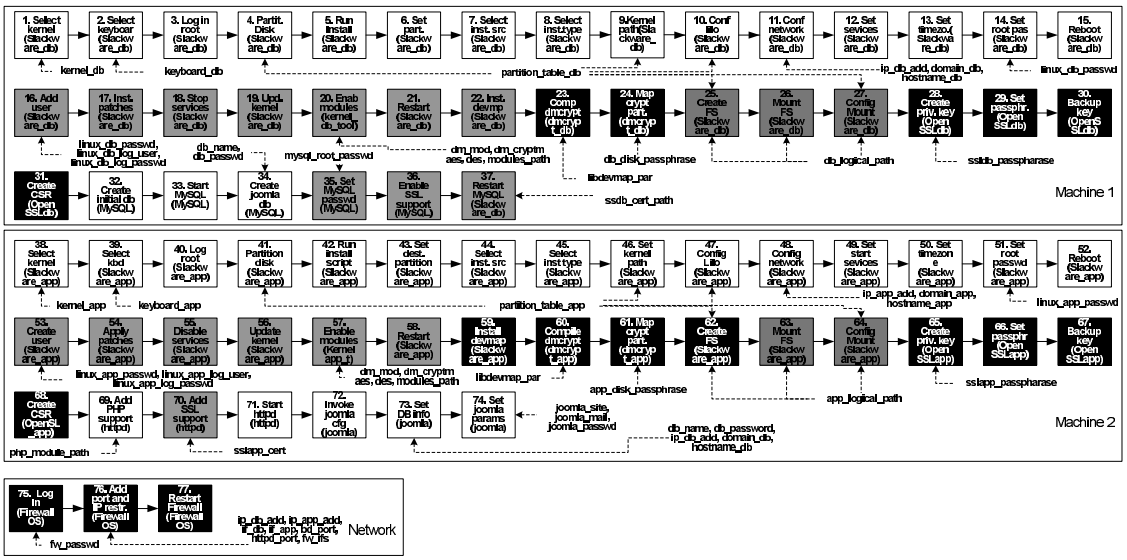

Fig. 2. Procedure to deploy Joomla enhanced with security mechanisms

- *ParamCount*: number of dissimilar parameters used in a procedure.
- *ParaUseCount*: number of parameters, including repetitions, used in a procedure.
- *ParamCrossContext*: sum of context switch values associated to actions, belonging to distinct containers, which consume a same parameter.
- *ParaAdaptCount*: number of parameters used in different syntactic forms along a procedure (e.g., relative and absolute paths to the httpd start script).
- *ParaSourceScore*: sum of the *SourceScore* measure associated to each distinct parameter used in a procedure. *SourceScore* indicates the difficulty to determine parameter values, ranging between 0 and 6. We propose this grading to be done according to the rules presented in Table II.

TABLE II
SOURCESCORE VALUES FOR PARAMETER EVALUATION

Value	*Parameter properties*
0	Suggested - automatically fulfilled
1	Selection (e.g., combo-box)
2	Not suggested - in the same container - documented
3	Not suggested - in the same container - not documented
4	Not suggested - in other container - documented
5	Not suggested - in other container- not documented
6	Not suggested - outside the environment

3) Memory complexity: This group of metrics is used to evaluate the number of parameters that must be remembered during the execution of a procedure, as well as the interval they must be retained in memory. To compute these measures, the administrator's memory is modeled as a Last-In-First-Out (LIFO) stack with non-associative lookup. Six metrics are associated to memory complexity, as summarized below.

- *MemSizeAverage* and *MemSizeMax*: average and maximum size of stack during the execution of a procedure.
- *MemDepthAverage* and *MemDepthMax*: average and maximum depth in the stack of the accessed parameters during the execution of a procedure.
- *MemLatAverage* and *MemLatMax*: average and maximum latency of parameters pushed onto the stack. Latency indicates the interval a parameter must be retained in the administrator's memory before being used again in a subsequent action.

B. Task-level complexity

Determining complexity at task level requires the measures of some of the metrics previously introduced to be computed in a slightly different way, as follows.

1) Execution complexity: NumActions is always assigned the value 1, while *ContextSwitchSum* is assigned the value of *ContextSwitch* of the action being measured in relation to its predecessor.

2) Parameter complexity: ParamCount is not applicable to actions, while *ParamUseCount* enumerates the number of parameters that the action being measured consumes. *ParamAdaptCount* is assigned the number of parameters the action *reuses* in a different syntactic form. Finally, *ParamSourceScore* receives the sum of *SourceScore* values associated to those parameters consumed by the action, which appear in the procedure for the first time.

3) Memory complexity: MemSize, *MemDepth*, and *MemLat* are already computed on a per-task basis and, therefore, do not

require changes. *Average* and *Max* values are not applicable to actions.

V. Security Complexity Analyzer

To prove concept and technical feasibility of our proposal we have developed the SCA (Security Complexity Analyzer) tool. It was implemented under Linux using the Java programming language. Figure 4 illustrates the SCA architecture including its components and the interactions among them.

Before start using the tool, the human administrator must capture data, needed to instantiate both the IT infrastructure and Activity models that represent the procedure to be evaluated. Next, he/she interacts with an editor (flow 1 of Figure 4), either a Graphical User Interface (GUI) or XML editor, in order to specify the mentioned models. The result of this process is the generation of a XML file (described below), which is processed by a parser (2) and transformed into an in-memory data structure. This is then consumed by the complexity scorer module (3) to compute the complexity measures defined in Section IV. The results are displayed by a viewer module (4) and stored, together with the input XML files, in a repository (5).

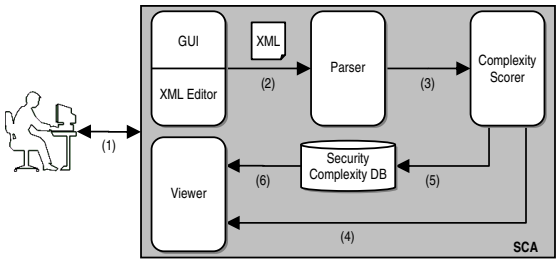

Fig. 4. SCA internal components

Figure 5 shows as an example parts of a XML file used to specify a procedure. It is organized in three parts: tasks (lines 5–18), containers (lines 20–23), and parameters (lines 25–29). Each task has as attributes a sequence number, a name, and an indication of its relation to security ($securityrelation$). In addition, it is necessary to specify to which container it belongs and the parameters it consumes. In turn, the specification of containers requires them to be named and their position within the hierarchy tree to be explicitly informed. Finally, all parameters of the procedure must be enumerated and their respective *SourceScore* values determined (according to Table II).

Since the results of evaluations are stored by SCA, it is possible for the administrator to compare different assessments. By means of such functionality (flow 6 of Figure 4), as described in further detail in the next section, he/she can, for example, estimate the impact of using different combinations of security mechanisms on the complexity of daily IT procedures or decide upon a tool to implement a certain security mechanism based on its reduced complexity

```
1  <?xml version="1.0" encoding="UTF-8"?>
2   <procedure>
3     <desc>"Scenario C procedure"</desc>
4   ...
5   <task seqnumber="22" title="Install dev-mapper"
6      securityrelation="1">
7      <container title="Slackware_db"/>
8   </task>
9   <task seqnumber="23" title="Compile dm-crypt"
10      securityrelation="2">
11     <container title="dm-crypt_db"/>
12     <parameter title="libdevmap_par"/>
13  </task>
14  <task seqnumber="24" title="Map crypt part."
15     securityrelation="2">
16     <container title="dm-crypt_db"/>
17     <parameter title="db_disk_passphrase"/>
18  </task>
19   ...
20  <container title="Slackware_db" treepos="0.1.1">
21  </container>
22  <container title="dm-crypt_db_db" treepos="0.1.1.3">
23  </container>
24   ...
25  <parameter title="libdevmap_par" paramsourcescore="4">
26  </parameter>
27  <parameter title="db_disk_passphrase"
28  paramsourcescore="2">
29  </parameter>
30 ...
31  </procedure>
```

Fig. 5. XML representation of procedure

to install, configure, and maintain (in comparison with its counterparts). Moreover, SCA can be used to spot the most complex tasks within a procedure, providing the IT staff with "clues" of which are worth trying to automate.

VI. Experimental Evaluation

In this section we present how to measure the complexity associated to security in IT procedures, based on the metrics described in Section IV. We have conducted several experiments, explained throughout this section, in order to evaluate security complexity in three dimensions: (i) complexity overhead that security mechanisms impose in general IT-related procedures; (ii) complexity measure of procedures that manipulate isolated security mechanisms; and (iii) comparison of complexity scores associated to procedures demanded by different tools to support a same security mechanism. Next, we elaborate each of these dimensions, introducing how to use the metrics and discussing the results obtained in the evaluation of real-life scenarios.

A. Complexity aggregated by security mechanisms in general IT-related procedures

Security may take part of more general procedures, as in the installation and setup of a secure web application, for example. In order to measure the complexity that security mechanisms may add to more general procedures, one must firstly model a base procedure, which does not involve any security mechanism, and proceed with the complexity evaluation. Then, a new procedure must be modeled – consisting of the base procedure interleaved with the actions/parameters required by the desired security mechanisms – and the complexity evaluation

performed again. The difference between the scores of the new security-enhanced procedure and the base procedure indicates the complexity overhead imposed by the security mechanisms considering the specific context.

We have conducted a set of experiments using three real life scenarios, based on the example described in Section III. They share a common goal – deploying Joomla and the software upon which it depends – but differ in relation to the security mechanisms implemented. The first one, A, is the base scenario, thus it does not include any of the security mechanisms specified in Table I. Scenarios B and C can be regarded as security-enhanced versions of scenario A. The former supports part of the security mechanisms specified in Table I (only those applied to Machine 2), while the latter implements all mechanisms listed in the same table.

To carry out the evaluation, we have defined specific procedures and container hierarchy trees for each scenario, which were used as input for our tool, SCA. As an example, Figure 2 illustrates the procedure for scenario C and Figure 3 depicts its hierarchy tree. The results (per metric) obtained for scenarios A, B, and C are summarized in Table III. The values in parentheses indicate the percentage variation of complexity that scenarios B and C present in relation to scenario A.

TABLE III
COMPLEXITY SCORES FOR SCENARIOS A, B, AND C

Metric	*Scenario A*	*Scenario B*	*Scenario C*
NumActions	38	55 (44.7%)	77 (102.6%)
ContextSwitchSum	6	11 (83.3%)	21 (250.0%)
ParamCount	20	32 (60.0%)	46 (130.0%)
ParamUseCount	31	45 (48.3%)	72 (132.2%)
ParamAdaptCount	0	0 (0%)	0 (0%)
ParamCrossContext	21	21 (0%)	45 (114.2%)
ParamSourceScore	45	74 (64.4%)	110 (144.4%)
MemSizeAvg	3.42	4.23 (23.6%)	7.38 (115.7%)
MemSizeMax	6	7 (16.6%)	14 (133.3%)
MemLatAvg	3.42	4.23 (23.6%)	7.38 (115.7%)
MemLatMax	116	201 (73.2%)	264 (127.5%)
MemDepthAvg	0.52	0.41 (-21.1%)	1.21 (132.6%)
MemDepthMax	15	15 (0%)	40 (166.6%)

Examining the table, one can note that the deployment of security mechanisms has impacted significantly the complexity measures of both scenarios B and C in relation to those observed for A (for almost all metrics). For example, except for the metric *ParamAdaptCount*, the values measured for scenario C were at least double those of A. In regards to scenario B, the differences were, overall, also large. Note, however, that the value of *MemDepthAvg* was lower than that calculated for scenario A. This is explained by the fact that the sum of values associated to *MemDepth* in scenario B was not higher enough, when compared to A, to follow the much higher value of the denominator (*NumActions*) used to compute the average (53 for scenario B against 38 for A).

The results also highlight the large amount of information that an administrator must remember while executing these procedures (*MemSizeMax*): in the worst case, 14 items. This value exceeds at least by five the common accepted capacity of humans' short-term memory [14]. Looking closer at these items, we have observed that 6 are directly associated to the deployment of security mechanisms. Following the increasing number of items to be remembered, the time interval they must be retained in the administrator's memory (*MemLatAvg*) also expands as the scenario gets more sophisticated.

Figure 6 depicts the overall complexity measures for the three scenarios evaluated, computed on a per group of metrics basis. The values observed for the metrics of each group (execution, parameter, and memory) have been summed. Although the plot may not precisely capture the influence of each metric on the resulting procedure complexity – topic of a future investigation – it provides the reader with a first approximation of how scenarios A, B, and C differentiate themselves. The security mechanisms deployed in B – namely file system and web connection encryption – are responsible for an increase of 50%, 47%, and 58% in execution, parameter, and memory complexity, respectively, in relation to scenario A. These overheads reach 123%, 133%, and 131% when comparing scenarios C and A.

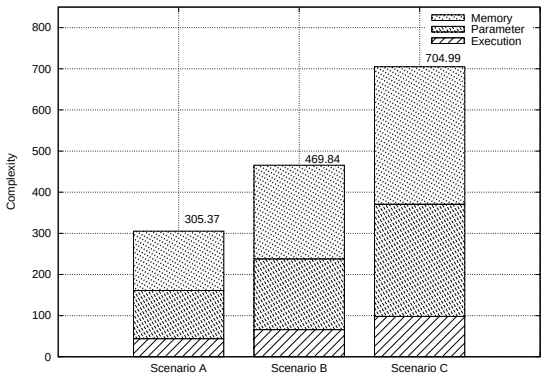

Fig. 6. Overall procedure complexity

A more detailed view of complexity hotspots within a procedure can be obtained through a per-task plot, as the one illustrated in Figure 7. It reveals the values measured for a subset of tasks of scenario C. The three bars per task represent execution, parameter, and memory complexity, respectively, while their colors denote the relation of the task in regards to security (as indicated in both the IT infrastructure and activity models). Task 62, $Create\ filesystem$, is the final step that need to be executed to cipher the file system of Machine 2, i.e., it is directly related to security. The high value measured for memory complexity is due to the need of remembering and reusing a parameter ($partition\_table\_app$) consumed 15 steps before.

B. Complexity measure of procedures that manipulate isolated security mechanisms

As opposed to the problem formulated in the previous subsection, there are situations where security mechanisms need to

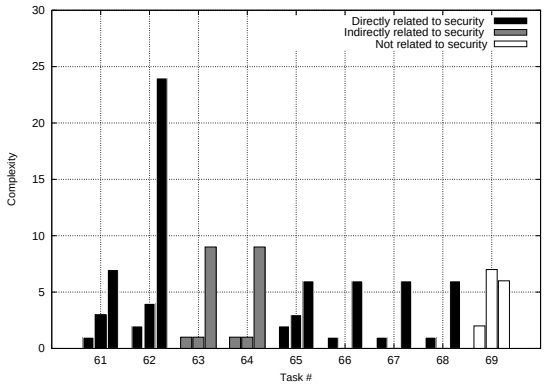

Fig. 7. Task complexity

TABLE IV
COMPLEXITY SCORES FOR PROCEDURES OF SCENARIOS D, E, F, AND G

Metric/Scenario	*D*	*E*	*F*	*G*
NumActions	7	10	3	17
ContextSwitchSum	2	4	1	7
ParamCount	3	10	8	19
ParamUseCount	3	13	8	22
ParamAdaptCount	0	0	0	0
ParamCrossContext	0	0	0	0
ParamSourceScore	9	31	16	52
MemSizeAvg	0	0.4	0	0.23
MemSizeMax	0	2	0	2
MemLatAvg	0	0.4	0	0.23
MemLatMax	0	3	0	3
MemDepthAvg	0	0.5	0	0.29
MemDepthMax	0	3	0	3

be (un)installed and/or maintained on top of an infrastructure already in place. For example, consider the case of an IT manager who demands the deployment of encryption support for the file system of a server (where sensitive information of the organization is stored). In order to measure the complexity of such procedures, one must specify them isolatedly – i.e., ignoring tasks not focused on the target security mechanisms – and compute the associated scores.

We have conducted the complexity evaluation of four scenarios, consisting of the deployment of different security mechanisms on top of a server running Slackware Linux and basic system/network services (including the Apache httpd web server). In the first scenario, D, we have installed and configured OpenSSL [13] to enable secure communications with the web server. The second scenario, E, comprised the encryption of the server's file system using dm-crypt [12], while the third, F, encompassed the setup of the netfilter/iptables packet filter [15]. Finally, in scenario G the three security mechanisms have been deployed at once.

Table IV summarizes the results obtained. Comparing scenarios D, E, and F, one can notice that E – file system encryption – presented the higher complexity scores, reflecting the time and effort expended to set it up. To install dm-crypt we had, among other tasks, to compile it from source and enable modules in the Linux kernel, which resulted in a procedure containing more tasks, parameters, and context switches (when compared to those executed for scenarios D and F). Not only this particular procedure demanded more parameters (10), but they were in general difficult to obtain (as reflected by the value of the metric *ParamSourceScore*).

From the table, it is also seen that the complexity measure for scenario G is, in general, less than that represented by the sum of the values computed for D, E, and F. For example, *NumActions* resulted 17 for G and 20 for D+E+F. In a first glance, one may find surprising that results derived from combination of three security mechanisms do not represent the sum of their individual results. This is due to the possibility of sharing actions and parameters when the three mechanisms are deployed together in a single procedure.

C. Comparison of complexity scores associated to procedures demanded by different tools to support a same security mechanism

When evaluating possibilities of fulfilling requirements defined in SLA/security policies, one may have to deal with situations where more than one security tool can be used, achieving equivalent or similar results. In this context, the comparison of complexity scores associated to procedures (e.g., install, uninstall, configure, and maintain) demanded by different tools can help the decision for one or another.

To illustrate with a concrete example, we have measured the complexity scores associated to the procedures required to install and configure two different tools that implement Virtual Private Networks (VPNs). The tools used in our experiments were OpenVPN [16] and Openswan [17]. To provide secure communication channels, the former employs Secure Sockets Layer (SSL), while the latter uses IP Security (IPSec). In order to make a fair comparison, both tools were evaluated under the same conditions. OpenVPN and Openswan have been installed on two equivalent machines, allowing remote users to connect to the VPN gateways through the Internet ("road warrior" VPN mode). We have configured them to employ X.509 certificates for session authentication, process that included the deployment of a Certificate Authority (CA) and the issuance of user certificates.

Table V summarizes the results obtained for this experiment. Comparing OpenVPN and Openswan, one can notice that the deployment of Openswan is significantly more cumbersome to perform than OpenVPN (demanding 53% more tasks, 180% more context switches, and 58% more parameters). This is due to the fact that the latter provides scripts that automate the creation of both the CA and certificates, while for the deployment of the former it is necessary to proceed manually. Moreover, Openswan requires the installation of gmp (GNU Multiple Precision Arithmetic Library) and the explicit configuration of the gateway's network interface card to accept packet redirection.

TABLE V
COMPLEXITY SCORES FOR INSTALLING OPENVPN AND OPENSWAN

Metric	*OpenVPN*	*Openswan*
NumActions	13	20
ContextSwitchSum	5	14
ParamCount	24	38
ParamUseCount	40	69
ParamAdaptCount	0	0
ParamCrossContext	0	5
ParamSourceScore	66	94
MemSizeAvg	1.23	4.85
MemSizeMax	8	10
MemLatAvg	1.23	4.85
MemLatMax	8	43
MemDepthAvg	5.53	6.7
MemDepthMax	36	45

VII. CONCLUSIONS AND FUTURE WORK

Despite the central role played by security on the complexity of IT procedures, it has been very hard, if at all possible, to measure it in a fair and accurate way. The reason for such difficulty comes from the lack of a quantitative assessment methodology. In our first iteration to address this problem, we have applied a model of configuration complexity proposed by Keller *et al.* [3] in order to be able to understand to which extent security influences complexity of IT procedures. Our proposal is supported by SCA, a prototypical implementation of a security complexity scorer system, which allowed us to conduct several experiments.

The results obtained, although preliminary, are encouraging. We have been able to exercise three dimensions of analysis: overhead induced by security mechanisms on general IT procedures, complexity measures of isolated security-related procedures, and comparison between security mechanisms based on scores associated to the procedures to manipulate them.

Important lessons have been learned from our experiments. Fist, it is difficult to isolate and compute the overall complexity of security mechanisms when they are interleaved within general IT procedures. This is due to the fact that the values measured for the metrics (especially the parameter and memory-related ones) are very sensitive to the order in which tasks are executed. In addition, security tasks embed themselves within containers that have no relation to security (e.g., gray tasks depicted in Figure 2). To overcome such issues, we had to resort on the delta-based approach described in Section VI-A, whose feasibility would have been compromised had we not developed SCA to assist the process. Second, complexity scores computed for security mechanisms are highly dependent on the available infrastructure in place and therefore are not easily generalizable to other scenarios. Third, although the results obtained seem to reflect the complexity perceived for the execution of the several procedures reported in this paper, it is still unclear what (combination of) metrics better capture the subtleties associated to IT security.

In the future, we intend to calibrate the model, i.e., tune the weights of the metrics so that the resulting measures reflect what is perceived by human operators when dealing with IT security. To accomplish that, we will correlate data collected from a multitude of real-life scenarios with values obtained by the computation of the metrics. A first proposal for such calibration problem has been proposed by Diao *et al.* [4], but the authors recognize that further work is needed to improve its accuracy. It is also our intent to validate the proposal with IT security officers and operators so as to ensure the model is complete.

VIII. ACKNOWLEDGMENTS

We would like to thank Alexander Keller and Yixin Diao for their valuable comments and suggestions during the course of this work.

REFERENCES

[1] Jacques A. Cazemier, Paul L. Overbeek, and Louk M. Peters. *Security Management (IT Infrastructure Library Series)*. Stationery Office, UK, March 2000.
[2] William Stallings. *Network Security Essentials: Applications and Standards (3rd Edition)*. Prentice Hall, July 2006.
[3] Aaron B. Brown, Alexander Keller, and Joseph L. Hellerstein. A Model of Configuration Complexity and its Application to a Change Management System. In IEEE, editor, *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, IFIP/IEEE International Symposium on Integrated Network Management, pages 631–644, Nice, France, 2005.
[4] Yixin Diao, Alexander Keller, Sujay Parekh, and Vladislav V. Marinov. Predicting Labor Cost throught IT Management Complexity Metrics. In IEEE, editor, *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 2007. IEEE.
[5] Aaron B. Brown and Joseph L. Hellerstein. An approach to Benchmarking Configuration Complexity. In *Proceedings of the 11th ACM SIGOPS European Workshop*, page 18, Leuven, Belgium, Sept 2004. ACM Press.
[6] A. Keller, A. B. Brown, and J. L. Hellerstein. A Configuration Complexity Model and Its Application to a Change Management System. *Network and Service Management, IEEE Transactions on*, 4(1):13–27, 2007.
[7] Yixin Diao and Alexander Keller. Quantifying the Complexity of IT Service Management Processes. In IEEE, editor, *Proc. of IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Dublin, Ireland, 2006. IEEE.
[8] Office of Government Commerce. Service Support (IT Infrastructure Library Series), June 2000.
[9] Center for Internet Security. http://www.cisecurity.org/.
[10] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. A model for evaluating it security investments. *Commun. ACM*, 47(7):87–92, July 2004.
[11] Joomla. http://www.joomla.org/, 2007.
[12] Dm-crypt. http://www.saout.de/misc/dm-crypt/, 2007.
[13] OpenSSL. http://www.openssl.org, 2007.
[14] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.
[15] Netfilter/iptables. http://www.netfilter.org/, 2007.
[16] OpenVPN. http://openvpn.net/, 2007.
[17] Openswan. http://www.openswan.org/, 2007.