

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

SOFTWARE LIVRE PARA VERIFICAÇÃO DE ADEQUAÇÃO
DE SERVIDORES GNU/LINUX À NORMA DE SEGURANÇA
NBR ISO/IEC 27002

FERNANDO LUCIO CUGIK

BLUMENAU
2007

2007/2-16

FERNANDO LUCIO CUGIK

**SOFTWARE LIVRE PARA VERIFICAÇÃO DE ADEQUAÇÃO
DE SERVIDORES GNU/LINUX À NORMA DE SEGURANÇA
NBR ISO/IEC 27002**

Trabalho de Conclusão de Curso submetido à
Universidade Regional de Blumenau para a
obtenção dos créditos na disciplina Trabalho
de Conclusão de Curso II do curso de Ciências
da Computação — Bacharelado.

Prof. Fabio Rafael Segundo, Mestre - Orientador

**BLUMENAU
2007**

2007/2-16

**SOFTWARE LIVRE PARA VERIFICAÇÃO DE ADEQUAÇÃO
DE SERVIDORES GNU/LINUX À NORMA DE SEGURANÇA
NBR ISO/IEC 27002**

Por

FERNANDO LUCIO CUGIK

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente: _____
Prof. Fabio Rafael Segundo, Mestre – Orientador, FURB

Membro: _____
Prof. Paulo Fernando da Silva, Mestre – FURB

Membro: _____
Prof. Everaldo Artur Grahl, Mestre – FURB

Blumenau, 6 de dezembro de 2007

Este trabalho é dedicado à minha família e à comunidade de software livre.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que sempre me apoiou quando eu mais precisei. Ao meu pai Estanislau e a minha mãe Luzia, pelo caráter, esforço, trabalho e dedicação sempre pensando na gente.

Aos meus antepassados, que mesmo sem os ter conhecido, sua história e esforço são contadas e repassadas pelas gerações, servindo de inspiração, superação e mantendo os valores da família.

Aos todos os meus amigos, pelos empurrões e cobranças. Ao Fischer que ajudou com seu conhecimento em PHP.

À Fran, pela pessoa maravilhosa que é, pela sua paciência e ajuda na escrita deste trabalho.

Aos professores que contribuíram com seus conhecimentos técnicos e experiência de vida para a minha formação, em especial ao professor Paulo Fernando pelo seu empenho e conhecimento em segurança.

Ao meu amigo e orientador, Fabio, pelos anos de trabalho junto compartilhando a sua experiência e conhecimento, pela sua determinação e empenho em software livre e por ter acreditado e contribuído para a realização deste trabalho.

Mantenha-se simples, bom, puro, sério, livre de afetação, amigo da justiça, temente aos deuses, gentil, apaixonado, vigoroso em todas as suas atitudes. Lute para viver como a filosofia gostaria que vivesse. Reverencie os deuses e ajude os homens. A vida é curta. (Marco Aurélio)

RESUMO

Este trabalho apresenta o estudo da norma de segurança NBR ISO/IEC 27002, segurança em redes de computadores e das melhores práticas de segurança em servidores baseados no sistema operacional GNU/Linux. Mostra também o projeto, modelagem e implementação de um software livre com interface web para auditoria de segurança nestes servidores de acordo com tal norma. Pretende-se com este trabalho elucidar questões sobre a aderência de sistemas informatizados, mas especificamente, os servidores de rede, à norma de segurança. O software permitirá que tais verificações dos controles da norma possam ser adicionadas ou removidas, personalizando a auditoria para cada caso e também permitindo uma integração posterior com outras ferramentas. Para tanto, utilizou-se a linguagem PHP, o framework CakePHP e comandos do Bash, para realizar a implementação dos *scripts* e requisitos de segurança.

Palavras-chave: Segurança da informação. Norma NBR ISO/IEC 27002. Software livre. GNU/Linux.

ABSTRACT

This work presents the study of the security standard ISO/IEC 27002, network security and the best practices of server security, based in the GNU/Linux operational system. It also presents the project, the model and the implementation of an open source software with a web interface to security audit in this servers according to that standard. It intends to clear questions about the compliance of information systems – more specifically network servers – to the security standard. The software will allow those verifications of standard control to be added or removed, customizing the audit to each case and also permitting a later integration with other tools. To do so, it uses the PHP language, the CakePHP framework and the Bash commands, to make the implementation of the security requirements.

Key-words: Information security. Standard ISO/IEC 27002. Open source. GNU/Linux.

LISTA DE ILUSTRAÇÕES

Figura 1 – Digrama de casos de uso do auditor.....	27
Figura 2 – Digrama de classes.....	28
Figura 3 – Digrama de seqüência efetua <i>login</i>	29
Figura 4 – Digrama de seqüência cadastra servidor.....	30
Figura 5 – Digrama de seqüência realiza auditoria	31
Figura 6 – Digrama de seqüência visualiza resultado	32
Figura 7 – Digrama de entidade e relacionamento	33
Figura 8 – Comparativo entre <i>frameworks</i> PHP.....	35
Quadro 1 – Ferramenta Bake	36
Quadro 2 – <i>Script</i> de exemplo	38
Quadro 3 – Falso negativo nos <i>scripts</i>	39
Quadro 4 – <i>Script</i> 10.10.3.php verificação dos arquivos de <i>log</i>	40
Quadro 5 – <i>Script</i> 10.10.6.php sincronização dos relógios	41
Quadro 6 – <i>Script</i> 11.4.6.php controle de conexões de rede	42
Quadro 7 – Calcula adequação	43
Quadro 8 – Diretrizes para implementação do controle 15.3.1	45
Quadro 9 – Diretrizes para implementação do controle 15.3.2.....	45
Figura 9 – Página de <i>login</i> com <i>captcha</i>	46
Quadro 10 – Função de <i>login</i> na <i>Controller</i> <i>User</i>	47
Quadro 11 – Validação dos dados de entrada no <i>Model</i> <i>User</i>	47
Quadro 12 – Validação dos dados de entrada na <i>View</i> <i>User</i>	48
Quadro 13 – Diretrizes para implementação do controle 12.2.2.....	49
Quadro 14 – Diretrizes para implementação do controle 12.2.4.....	49
Quadro 15 – Trilha de auditoria	51
Figura 10 – Página inicial do software	52
Figura 11 – Página de <i>login</i>	53
Figura 12 – Página autenticado	54
Figura 13 – Página cadastro de servidores	55
Figura 14 – Página de auditorias	56
Figura 15 – Página nova auditoria.....	57
Figura 16 – Página visualizando o resultado da auditoria.....	58

Figura 17 – Página inicial do <i>site</i> do projeto	59
Quadro 16 – Comparativo da execução em distribuições GNU/Linux	60
Quadro 17 – Instalação do software	69
Quadro 18 – Detalhamento do caso de uso Efetua Login	70
Quadro 19 – Detalhamento do caso de uso Cadastra Servidor.....	71
Quadro 20 – Detalhamento do caso de uso Visualiza resultado da auditoria.....	71
Quadro 21 – Detalhamento do caso de uso Realiza Auditoria.....	72
Quadro 22 – Manual para a personalização ou criação de novos <i>scripts</i>	74

LISTA DE SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

AES – *Advanced Encryption Standard*

BASH – *Bourne-Again Shell*

BS – *British Standard*

CAPTCHA – *Completely Automated Public Turing test to tell Computers and Humans Apart*

COBRA – *Consultive, Objective and Bi-functional Risk Analysis*

GNU – *GNU's Not Unix*

ISO – *International Organization for Standardization*

IEC – *International Electrotechnical Commission*

MVC – *Model, View and Controller*

NIDS – *Network Intrusion Detect System*

OO – *Orientação a Objetos*

PHP – *Hypertext Preprocessor*

SGSI – *Sistema de Gestão de Segurança da Informação*

SSH – *Secure Shell*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO.....	13
1.1 JUSTIFICATIVA	14
1.2 OBJETIVOS DO TRABALHO	15
1.3 ESTRUTURA DO TRABALHO	16
2 FUNDAMENTAÇÃO TEÓRICA	17
2.1 SEGURANÇA DA INFORMAÇÃO	17
2.2 NORMA NBR ISO/IEC 27002	19
2.3 MECANISMOS DE SEGURANÇA.....	21
2.4 DOCUMENTOS DE SEGURANÇA PARA O GNU/LINUX.....	21
2.5 SOFTWARE LIVRE.....	22
2.6 AUDITORIA DA TECNOLOGIA DA INFORMAÇÃO.....	23
2.7 TRABALHOS CORRELATOS	23
3 DESENVOLVIMENTO DO SOFTWARE LIVRE	25
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA	25
3.2 ESPECIFICAÇÃO	26
3.2.1 Diagrama de casos de uso	26
3.2.2 Diagrama de classes	27
3.2.3 Diagramas de seqüência.....	29
3.2.4 Diagrama de entidade e relacionamento	32
3.3 IMPLEMENTAÇÃO	33
3.3.1 Técnicas e ferramentas utilizadas.....	34
3.3.2 Implementação do software	36
3.3.3 Implementação dos <i>scripts</i>	38
3.3.3.1 <i>Script</i> 10.10.3.php verificação dos arquivos de <i>log</i>	40
3.3.3.2 <i>Script</i> 10.10.6.php sincronização dos relógios	40
3.3.3.3 <i>Script</i> 11.4.6.php controle de conexões de rede	42
3.3.4 Cálculo da adequação.....	43
3.3.5 Considerações quanto à auditoria de sistema de informação.....	44
3.3.6 Segurança implementada na aplicação.....	45
3.3.6.1 Captcha	46
3.3.6.2 Autenticação	46

3.3.6.3 Validação dos dados de entrada.....	47
3.3.6.4 Controle do processamento interno	48
3.3.6.5 Validação de dados de saída	49
3.3.6.6 Criptografia.....	49
3.3.6.7 Controle de acesso ao código-fonte de programa.....	50
3.3.6.8 Gestão de vulnerabilidades técnicas	50
3.3.6.9 Trilha de verificação	50
3.3.7 Operacionalidade da implementação	51
3.4 DESENVOLVIMENTO DO <i>WEBSITE</i> PARA O SOFTWARE LIVRE.....	58
3.5 RESULTADOS E DISCUSSÃO	59
4 CONCLUSÕES.....	63
4.1 EXTENSÕES	64
REFERÊNCIAS BIBLIOGRÁFICAS	66
APÊNDICE A – Manual de instalação do software Check 27002	68
APÊNDICE B – Detalhamento dos casos de uso do auditor	70
APÊNDICE C – Manual para a personalização ou criação de novos <i>scripts</i>.....	73

1 INTRODUÇÃO

Atualmente, na sociedade da informação, ao mesmo tempo que as informações são consideradas o principal patrimônio de uma organização, estão também sob constante risco, de forma como nunca estiveram antes. Com isso, a segurança de informações tornou-se um ponto crucial para a sobrevivência das instituições (DIAS, 2000, p. 40).

A evolução tecnológica permitiu o aumento da capacidade, barateamento e larga utilização dos computadores e redes, mas também trouxe maior preocupação com a segurança das informações. Esta se torna mais complexa quando se tem uma grande quantidade de informações, que podem estar disponíveis de diversas formas e acessadas simultaneamente por pessoas com privilégios diferentes. Além disso, são descobertas um número cada vez maior de vulnerabilidades e ameaças explorando estes sistemas informatizados.

Dias (2000, p. 40) diz que nunca foi tão fácil atacar os sistemas informatizados, já que os sistemas de informações institucionais conectados em redes externas aumentam significativamente os riscos de segurança, além de muitas ferramentas de ataque utilizadas antigamente apenas por agências de inteligência hoje estarem disponíveis a qualquer pessoa.

Observando os números do Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (2007), em 2006 o total de incidentes reportados foi 191% maior do que o total de 2005 e ainda assim 37% superior a soma dos anos de 2004 e 2005. Um aumento significativo que leva a questionar se as ações de segurança seriam tomadas na mesma proporção. Para o ano de 2007, a estatística parcial até o mês de setembro mostra uma estabilização e redução de 6 % em relação ao mesmo período de 2006.

Percebe-se que as preocupações com segurança aumentam e a possibilidade de uma invasão, ataque, ou a quebra de confidencialidade impõe responsabilidades, exigindo das organizações ações apropriadas. Para orientar estas ações, surgiu a necessidade de criar padrões ou normas com o objetivo de definir e organizar os aspectos de segurança da informação.

Um dos primeiros esforços para criar recomendações de segurança, foi o Orange Book, conhecido oficialmente como *Trusted Computer Evaluation Criteria – DoD 5200.28-STD*, publicado pelo departamento de defesa dos Estados Unidos. O lançamento de seu primeiro rascunho foi em 1978 e pode ser considerado o marco inicial na busca de controles que permitam um ambiente computacional ser classificado de acordo com níveis de segurança.

Atualmente, as normas de segurança *British Standard (BS) 7799* e a *International*

Organization for Standardization (ISO) / International Engineering Consortium (IEC) 17799, renomeada para 27002, são as mais citadas quando se fala de segurança da informação. Em primeiro de julho de 2007, a ISO e a IEC publicaram uma correção técnica para a norma ISO/IEC 17799:2005, que altera o seu nome para ISO/IEC 27002 e a coloca na série 27000, criada para agrupar as normas de segurança da informação. Embora as pesquisas foram baseadas na norma anterior, este trabalho já se refere a nova norma, ou ao novo nome corrigido.

Os administradores do sistema operacional GNU/Linux muitas vezes implementam a segurança do servidor de uma forma independente, básica ou otimizada, mas nem sempre padronizada com a política de segurança da organização, ou com as melhores práticas recomendadas. Sendo assim, como a norma visa atender de uma forma mais abrangente a todos os sistemas e processos das organizações, verificou-se a possibilidade de aplicar a especificidade dos controles e requisitos da norma ao sistema operacional GNU/Linux, ajudando seus administradores no processo de auditoria e verificação de segurança nesses sistemas.

Uma ferramenta que seja construída de acordo com as recomendações da norma, com o objetivo de verificar o nível de aderência de servidores, serviços ou sistemas à norma de segurança NBR ISO/IEC 27002¹ seria de grande importância para as organizações que pretendem melhorar sua segurança em servidores GNU/Linux.

1.1 JUSTIFICATIVA

A norma de segurança NBR ISO/IEC 27002 agrupa as melhores práticas de segurança da informação aceitas mundialmente, visando atender de uma forma mais abrangente a todos os sistemas e processos da organização. A ferramenta a ser desenvolvida, verifica a implementação das recomendações dos controles e requisitos da norma ao sistema operacional GNU/Linux.

Visando atender uma necessidade dos administradores de sistema GNU/Linux, de analisar a segurança dos servidores de acordo com as melhores práticas e recomendações da norma, a ferramenta deverá analisar a segurança do servidor e indicar o grau de adequação à

¹ Assume-se que a norma NBR ISO/IEC 27002 poderá ser referenciada como “a norma”.

norma através de um percentual.

Por ser uma área relativamente nova, a normatização de segurança não dispõe de muitas ferramentas de suporte a esta atividade, não existindo atualmente nenhum projeto de software livre ativo com este objetivo. Este trabalho tem sua relevância criando uma alternativa como software livre de uma ferramenta de suporte para a verificação de aderência à norma NBR ISO/IEC 27002 em servidores GNU/Linux.

1.2 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo desenvolver uma aplicação com interface web para auditoria de segurança em servidores GNU/Linux, verificando a sua adequação com a norma de segurança NBR ISO/IEC 27002 através da implementação e verificação dos controles da norma no servidor GNU/Linux.

Os objetivos específicos do trabalho são:

- a) trazer mais informações sobre a aplicação das normas da família ISO/IEC 27002²;
- b) levantar informações sobre as melhores práticas de segurança em servidores GNU/Linux;
- c) desenvolver uma ferramenta baseando sua implementação nos controles da norma NBR ISO/IEC 27002 e seguindo as recomendações das melhores práticas de segurança em servidores GNU/Linux;
- d) identificar quais os controles da norma serão utilizados na verificação de aderência dos servidores GNU/Linux;
- e) implementar os controles identificados no item anterior para os servidores GNU/Linux;
- f) emitir um relatório com o resultado do teste de adequação do servidor aos controles da norma;
- g) comprovar o funcionamento da ferramenta em diferentes servidores GNU/Linux, com diferentes níveis de segurança e comparar os resultados obtidos;
- h) permitir que todos se beneficiem com o resultado desse estudo, publicando a

² Família ISO/IEC 27002 compreende as normas internacionais ISO/IEC 17799, ISO/IEC 17799:2005 e ISO/IEC 27002, além das normas nacionais NBR ISO/IEC 17799, NBR ISO/IEC 17799:2005 e NBR ISO/IEC 27002.

ferramenta como um projeto de software livre.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos. O segundo capítulo aborda os assuntos estudados para o desenvolvimento deste trabalho, como conceitos de segurança da informação, a norma NBR ISO/IEC 27002, mecanismos de segurança, alguns documentos de segurança para o GNU/Linux como o Linux Security HOWTO e o Red Hat Enterprise Linux 4: *Security Guide*, software livre, auditoria da tecnologia da informação e alguns trabalhos correlatos.

O terceiro capítulo descreve o desenvolvimento do trabalho, iniciando pelos requisitos que a ferramenta deve atender sua especificação, implementação, os resultados obtidos e discussão.

No quarto capítulo é apresentada a conclusão e as sugestões de extensão para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos de segurança da informação, a norma NBR ISO/IEC 27002, mecanismos de segurança, alguns documentos de segurança para o GNU/Linux como o Linux Security HOWTO e o Red Hat Enterprise Linux 4: *Security Guide*, software livre, auditoria da tecnologia da informação e na última seção serão apresentados alguns trabalhos correlatos.

2.1 SEGURANÇA DA INFORMAÇÃO

Primeiramente de adentrar na segurança da informação, deve-se reforçar a importância da informação para as organizações, de forma a demonstrar o motivo pelo qual ela deve ser protegida.

“A informação representa a inteligência competitiva dos negócios e é reconhecida como ativo crítico para a continuidade operacional da empresa” (SÊMOLA, 2003, p. 45).

Segundo Associação Brasileira de Normas Técnicas (2005, p. ix), a informação é um ativo que, como qualquer outro ativo importante, é essencial para os negócios de uma organização e conseqüentemente necessita ser adequadamente protegida. Isto é especialmente importante no ambiente dos negócios, cada vez mais interconectado. Como um resultado deste incrível aumento da interconectividade, a informação está agora exposta a um número cada vez maior de ameaças e vulnerabilidades.

Vale ressaltar que a informação necessita ser protegida adequadamente, independente da sua forma que pode ser impressa, escrita ou armazenada eletronicamente, ou ainda o meio que é compartilhada ou armazenada. (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2005, p. ix).

“Segurança da informação é a proteção da informação de vários tipos de ameaças para garantir a continuidade do negócio, minimizar o risco ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio.” (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2005, p. ix).

Segundo Peixoto (2006, p. 37) define o termo Segurança da Informação como uma área do conhecimento que salvaguarda os chamados ativos da informação, contra acessos

indevidos, modificações não autorizadas ou até mesmo sua não disponibilidade. Define também que ativo é tudo que manipula direta ou indiretamente a informação, inclusive ela própria, no contexto de segurança da informação, um ativo pode ser um computador, impressora, fichário ou até mesmo um usuário. Neste contexto, não deve ser confundido com ativo patrimonial.

Segundo Campos (2006, p. 5-6), para garantir a segurança da informação, necessita-se atender a três princípios básicos:

- a) **confidencialidade:** é respeitada quando apenas as pessoas autorizadas têm acesso à informação. Casos típicos de quebra de confidencialidade ocorrem quando pessoas invadem sistemas de computador das empresas ou de pessoas físicas e obtêm seus dados;
- b) **integridade** é respeitada quando a informação acessada está completa, sem alterações e, portanto, confiável. Quando uma informação é indevidamente alterada, como pela falsificação de um documento, da alteração de um registro em um banco de dados, configura um acidente de segurança da informação por quebra de integridade;
- c) **disponibilidade:** é alcançada quando a informação está acessível, por pessoas autorizadas, sempre que necessário. Como exemplo, a perda de documentos, sistemas indisponíveis, ou ainda ataques de negação de serviço são incidentes de segurança da informação por quebra de disponibilidade.

Conforme Associação Brasileira de Normas Técnicas (2005, p. ix), a segurança da informação é obtida a partir da implementação de um conjunto de controles adequados. Estes controles não são somente os informatizados, incluem políticas, processos, procedimentos, estruturas organizacionais e funções de software e hardware. Este não é apenas um projeto, com início e fim, os controles precisam ser estabelecidos, implementados, monitorados, analisados criticamente e melhorados, para garantir que os objetivos do negócio e de segurança da organização sejam atendidos. Convém que isto seja feito em conjunto com outros processos de gestão do negócio.

Para organizar a aplicação de controles e mecanismos de segurança foram desenvolvidas normas e padrões que ajudam na gestão da informação quanto à segurança.

2.2 NORMA NBR ISO/IEC 27002

A atual norma NBR ISO/IEC 27002 passou por diversas etapas, seus estudos têm origem em 1987 na Inglaterra, mais especificamente no Departamento de Indústria e Comércio onde um grupo foi criado com objetivos de elaborar um conjunto de critérios que pudessem validar a segurança da informação e proporcionar também certificação. Outro objetivo era criar um código de boas práticas que apoiasse a implementação de ações em segurança da informação, o que foi alcançado em 1989. Este trabalho foi aprimorado e mais tarde após várias revisões o código de prática foi publicado como um padrão britânico para gestão de segurança da informação com o título de PD 0003, e em 1995 essa norma foi publicada como BS7799:1995 (CAMPOS, 2006, p. 97).

Depois de uma nova revisão que gerou a BS7799:1999, foi proposto à ISO que a primeira parte desta norma, a BS7799-1, que estabelece o código de práticas, fosse homologada. O que aconteceu em 2000, sendo publicada como norma ISO/IEC 17799:2000, também em 2000 foi publicada a sua equivalente nacional, a norma NBR ISO/IEC 17799:2001, sendo ela a tradução literal da norma internacional (CAMPOS, 2006, p. 98).

A ISO continuou evoluindo o código de práticas e em 2005 foi disponibilizada a segunda versão da norma ISO/IEC 17799:2005 e a sua versão nacional NBR ISO/IEC 17799:2005. Nessa versão os controles são estruturados de forma a separar a definição do controle, as diretrizes de implementação e as suas informações adicionais (CAMPOS, 2006, p. 98).

Em primeiro de julho de 2007 a ISO e a IEC publicaram uma correção técnica para a norma ISO/IEC 17799:2005, a qual substitui em todo o documento as ocorrências de 17799 para 27002, alterando o seu nome para ISO/IEC 27002 e colocando-a na série de normas ISO/IEC 27000, um conjunto de normas criado para agrupar as normas de segurança.

Segundo Associação Brasileira de Normas Técnicas (2005, p. 1) a norma estabelece diretrizes e princípios gerais para iniciar, implementar, manter e melhorar a gestão de segurança da informação em uma organização. Os objetivos definidos nesta Norma provêm diretrizes gerais sobre as metas geralmente aceitas para a gestão da segurança da informação. É estruturada contendo 11 seções de controles de segurança, estas subdivididas em 39 categorias principais de segurança e uma seção introdutória. Cada seção contém um número das principais categorias de segurança. As seções (seguidas pelo número de categorias) são:

- a) política de segurança da informação (1);

- b) organizando a segurança da informação (2);
- c) gestão de ativos (2);
- d) segurança em recursos humanos (3);
- e) segurança física e do ambiente (2);
- f) gestão das operações e comunicações (10);
- g) controle de acesso (7);
- h) aquisição, desenvolvimento e manutenção de sistemas de informação (6);
- i) gestão de incidentes de segurança da informação (2);
- j) gestão da continuidade do negócio (1);
- k) conformidade (3).

Como subdivisões das seções ressaltam-se as principais categorias de segurança da informação, onde cada uma dessas categorias contém: um objetivo de controle, definindo o que deve ser alcançado, e um ou mais controles que podem ser aplicados visando alcançar o objetivo do controle.

Para melhor entendimento, vale definir que controle é a forma de gerenciar o risco, incluindo políticas, procedimentos, diretrizes, práticas ou estruturas organizacionais, que podem ser de natureza administrativa, técnica, de gestão ou legal. Controle é também usado como um sinônimo para proteção ou contramedida (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2005, p. 1).

Os controles de cada categoria são estruturados da seguinte forma:

- a) nome do controle;
- b) diretrizes para implantação, que contém as informações mais detalhadas para apoiar a implementação do controle e atender ao objetivo do controle;
- c) informações adicionais, como por exemplo considerações legais ou referências a outras normas.

A norma resume as melhores práticas aceitas internacionalmente quanto à segurança em todo o ambiente organizacional, incluindo não somente os servidores e equipamentos de informática, mas todo o tipo de informação e ativo que tem importância para a empresa. Para melhor aplicar as recomendações da norma é importante conhecer os mecanismos que possibilitam a segurança ser aplicada aos sistemas.

2.3 MECANISMOS DE SEGURANÇA

Mecanismo de segurança é o meio utilizado para atender um serviço de segurança, isto é, o mecanismo existe para prover e suportar os serviços de segurança. Como exemplo, cita-se o mecanismo de criptografia que pode ser utilizado para garantir o serviço de confidencialidade (DIAS, 2000, p. 74).

Soares, Lemos e Colcher (1995, p. 452-465) relacionam alguns mecanismos de segurança, entre quais pode-se destacar:

- a) criptografia: processo de modificar o texto original, gerando um texto ininteligível (criptografado) na origem, sendo que após isso envia-se o texto criptografado ao destino para então fazer o processo inverso, descriptografá-lo para retornar ao texto original;
- b) assinatura digital: envolve dois processos, a assinatura do dado pelo signatário e a verificação de assinatura por quem deseje confirmar a autenticidade do documento;
- c) autenticação: significa a identificação de uma entidade como um usuário ou computador, através de uma senha;
- d) controle de acesso: mecanismo utilizado para confirmar que o acesso somente é permitido para usuários autorizados;
- e) integridade de dados: verifica por exemplo se um dado armazenado continua íntegro, ou não sofreu nenhuma alteração, através de técnicas de verificação de alterações.

2.4 DOCUMENTOS DE SEGURANÇA PARA O GNU/LINUX

O documento Linux Security Howto segundo Fenzi e Wreski (2007), provê uma visão geral dos assuntos de segurança direcionados ao administrador de sistemas GNU/Linux. Retrata a filosofia da segurança e exemplos de como proteger o sistema GNU/Linux de intrusos. Abrange aspectos desde a segurança física, local, de arquivos, a importância de senhas e criptografia, alcançando tópicos de segurança do *kernel* e da rede. Apresenta os procedimentos a executar antes de colocar o servidor em produção, visando caso a máquina

for invadida, corrigir e disponibilizar o serviço rapidamente. Relata também os procedimentos a serem executados antes, durante e depois de uma invasão do sistema.

Com este conteúdo o Linux Security Howto pretende orientar, de uma forma abrangente, aos administradores de todas as distribuições GNU/Linux quanto à segurança. É também importante verificar as recomendações específicas a cada distribuição GNU/Linux, onde o guia de segurança de Red Hat (2007) vem contribuir com este trabalho.

Segundo Red Hat (2007), o guia de segurança Red Hat Enterprise Linux 4 tem como objetivo ajudar os seus usuários a aprender os processos e práticas de configurar a segurança em estações de trabalho e servidores, contra intrusão local e remota, contra exploração de vulnerabilidades e atividades maliciosas. Detalha o planejamento e as ferramentas usadas para criar um ambiente seguro, em um *data center*, no local de trabalho, ou em casa. O guia aborda tópicos de segurança incluindo: *firewalls*, criptografia, aplicação de segurança em serviços críticos, *Virtual Private Networks* (VPNs) e detecção de intrusão aplicados ao servidor GNU/Linux com a distribuição Red Hat Enterprise.

2.5 SOFTWARE LIVRE

Segundo GNU (2007), é importante definir que "Software Livre" é uma questão de liberdade, não de preço. Para entender o conceito, deve-se pensar em "liberdade de expressão", não em "cerveja grátis". "Software livre" se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade para os usuários do software:

- a) liberdade de executar o programa, para qualquer finalidade (liberdade número 0);
- b) liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades (liberdade número 1). Acesso ao código-fonte é um pré-requisito para esta liberdade;
- c) liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade número 2);
- d) liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade número 3). Acesso ao código-fonte é um pré-requisito para esta liberdade.

Seguindo estas quatro regras o software pode ser considerado como Software Livre.

2.6 AUDITORIA DA TECNOLOGIA DA INFORMAÇÃO

“A auditoria da tecnologia da informação é um tipo de auditoria operacional, isto é, que analisa a gestão de recursos enfocando os aspectos de eficiência, eficácia, economia e efetividade.” (DIAS, 2000, p. 12).

Dias (2000, p. 12) descreve que dependendo da área de verificação escolhida pelo auditor, a auditoria da tecnologia da informação, pode abranger o ambiente de informática como um todo, incluindo desde a segurança física, lógica e o plano de contingência. Pode ainda visar a organização do departamento de informática analisando aspectos administrativos como políticas, padrões, procedimentos e responsabilidades. Ou então pode focar em controles sobre banco de dados, redes de comunicação e de microcomputadores.

Solution TI (2007) diz que auditor é a pessoa, ou departamento, que foi designado pela administração da empresa para avaliar, examinar e descobrir os pontos falhos e a real eficácia dos departamentos por ela vistoriados. Este deve ser uma pessoa com grande conhecimento da área de processamento de dados e todas as suas fases, deve ter objetividade, discrição, raciocínio lógico e principalmente independência.

Segundo Solution TI (2007), a auditoria em um departamento, principalmente na área de processamento de dados, é de vital importância para a empresa, já que através desta a administração pode efetuar o planejamento, evitar fraudes e garantir o bom desempenho dos setores auditados.

Pode-se dizer que a segurança e a auditoria são interdependentes, pois enquanto a segurança tem a função de garantir a integridade dos dados, a auditoria vem garantir que estes dados estão realmente íntegros, propiciando o processamento correto e alcançando o resultado esperado (SOLUTION TI, 2007).

2.7 TRABALHOS CORRELATOS

Em pesquisas na biblioteca da FURB e através de ferramentas de busca na internet, foram encontrados poucos trabalhos desenvolvidos com o objetivo de verificar o grau de aderência de uma organização à norma 27002 ou ao seu nome antigo 17799, dentre eles cita-se: Rosemann (2002) e Gonçalves (2005). Existem também algumas ferramentas comerciais

com este objetivo, dentre elas o software *Consultive, Objective and Bi-functional Risk Analysis* (COBRA) da empresa C & A Systems Security (2007), e a ferramenta Módulo Risk Manager (MÓDULO, 2007).

Rosemann (2002) em seu trabalho desenvolve uma ferramenta para auxiliar na avaliação da adequação de uma empresa à norma NBR ISO/IEC 17799, na forma de um *check-list*, incluindo tópicos a serem respondidos, que ao final geram uma nota do grau de adequação da empresa com a norma. É possível incluir, alterar e ou excluir tópicos do *check-list*, bem como quantificar o grau de importância do tópico para a organização. A nota é calculada através da média ponderada das respostas fornecidas ao *check-list*.

Gonçalves (2005) propõe um modelo para verificação, homologação e certificação de aderência à norma nacional de segurança de informação NBR-ISO/IEC-17799. Diferente da maioria das propostas nesse sentido, como a ferramenta COBRA, que se utiliza de *check-lists* ou questionários eletrônicos para verificar o nível de adequação com a norma, a ferramenta visa permitir a análise automática do ambiente a ser verificado.

C & A Systems Security (2007) implementa um software de análise de risco de segurança, avaliação e adequação às normas ISO 17799 e BS7799, denominado COBRA. Este software permite uma análise de segurança pela própria empresa, sem a necessidade de contratar consultores externos. A análise é feita através de questionários eletrônicos, que com base nos dados informados realiza a verificação dos controles que devem ser aplicados, e se foram aplicados corretamente.

A ferramenta Módulo Risk Manager (MÓDULO, 2007) se propõe a implementar uma análise de riscos e o apoio à verificação de requisitos para certificações como *Control Objectives for Information and related Technology* (COBIT), ISO 27001, Sarbanes-Oxley, entre outros. Segundo o prospecto da ferramenta, esta é bastante completa para a gestão de riscos, porém sua avaliação não foi possível pois não é disponibilizada uma versão de demonstração.

3 DESENVOLVIMENTO DO SOFTWARE LIVRE

Este capítulo contém os requisitos do software desenvolvido, a sua especificação, desenvolvimento da ferramenta e dos *scripts*, e por fim a discussão dos resultados obtidos com o desenvolvimento.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA

Os requisitos foram elaborados observando as recomendações da própria norma NBR ISO/IEC 27002, os níveis de segurança desejados para esta aplicação e de forma a atender aos objetivos propostos neste trabalho, para isso, a ferramenta deverá:

- a) ser desenvolvida de forma a considerar a categoria 15.3 da norma NBR ISO/IEC 27002 - considerações quanto à auditoria de sistemas de informação. Atendendo as recomendações como acesso somente leitura de software e dados, todo acesso seja monitorado de forma a produzir uma trilha de referência, e a correta proteção das ferramentas de auditoria (Requisito Funcional - RF).
- b) ser desenvolvida com base na categoria 12.2 da norma NBR ISO/IEC 27002 - processamento correto nas aplicações, aplicando controles ao projeto que incluam a validação dos dados de entrada, do processamento interno e dos dados de saída (Requisito Não Funcional - RNF);
- c) gerar um relatório com o nível de aderência à norma do servidor GNU/Linux auditado, discriminando os controles verificados e o grau de adequação (RF);
- d) permitir o acesso ao sistema somente por usuários autorizados e autenticados (RF);
- e) ser implementada utilizando a linguagem de programação PHP e *Bourne-Again Shell* (BASH) (RNF);
- f) ser compatível com o sistema operacional GNU/Linux, distribuições Debian, Fedora, Red Hat e Ubuntu (RNF);
- g) possibilitar ao usuário selecionar quais controles serão aplicados a fim de verificar sua adequação (RF);
- h) implementar os controles da norma NBR ISO/IEC 27002 para o servidor GNU/Linux, seguindo as recomendações da própria norma e as melhores práticas

de segurança em servidores GNU/Linux (RNF).

3.2 ESPECIFICAÇÃO

A especificação do software foi feita utilizando os diagramas da *Unified Model Language* (UML): diagramas de casos de uso, diagrama de classes e diagrama de seqüência. A ferramenta utilizada para realizar esta especificação foi Enterprise Architect da empresa Sparx Systems. É também apresentado o diagrama de entidade e relacionamento utilizado, que foi construído com a ferramenta *open source* DBDesigner. Em seqüência são apresentados estes diagramas.

3.2.1 Diagrama de casos de uso

No software desenvolvido foram identificados os casos de usos do auditor que é exibido na Figura 1 e o detalhamento desses diagramas pode ser verificado no Apêndice B.

Para o ator auditor, são identificados os seguintes casos de uso:

- a) efetua *login*: o auditor acessa a página de *login* e entra com os dados para acessar o sistema;
- b) cadastra o servidor: o auditor depois de autenticado no sistema cadastra o servidor em que vai ser feita a auditoria, que pode ser local ou remoto;
- c) realiza a auditoria: o auditor seleciona o servidor alvo da auditoria, os controles que vão ser verificados e executa a auditoria;
- d) visualiza o resultado da auditoria: o auditor pode visualizar o resultado de uma auditoria previamente realizada.

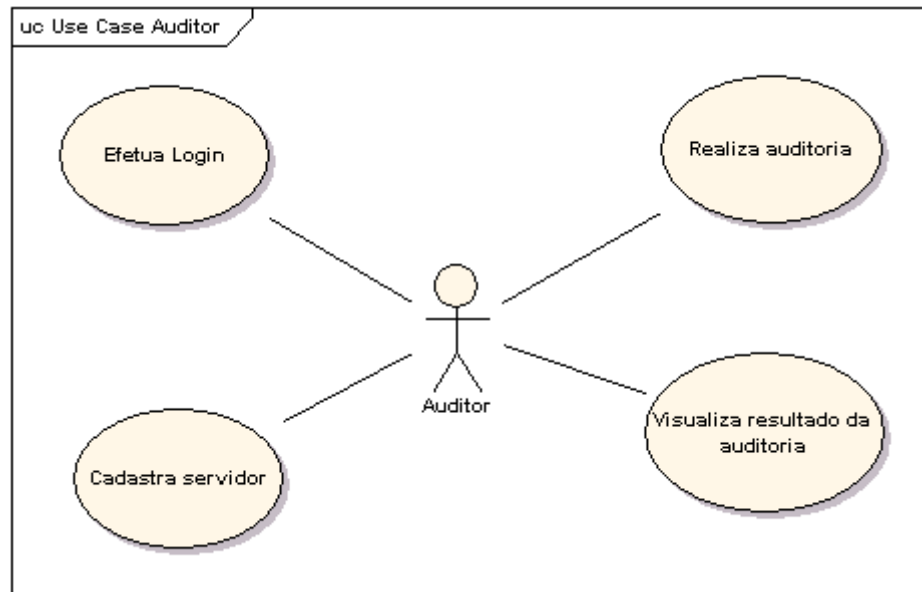


Figura 1 – Diagrama de casos de uso do auditor

3.2.2 Diagrama de classes

No desenvolvimento deste trabalho foi utilizado o *framework* CakePHP, abordado na seção 3.3.1. Para sua utilização na criação das classes específicas desta aplicação, o *framework* CakePHP disponibiliza uma classe padrão que deve ser usada como base na criação das outras, herdando assim as propriedades e funcionalidades disponibilizadas pelo *framework*. O nome desta classe padrão é `AppController`, e todas as classes específicas desta aplicação estendem as suas funcionalidades, como demonstrado na Figura 2. Nesse diagrama não são mostrados os atributos das classes para facilitar a visualização.

A classe `User` é responsável por manter as informações sobre os usuários e gerenciar o acesso através do `login()` e `logout()`, adicionalmente o método `captchaImage()` gera uma imagem com o captcha, o método `checkPassAes()` verifica a senha com a criptografia AES, e o `checkPerm()` verifica se o usuário está autenticado e tem permissão de acesso aos métodos.

Na classe `Auditoria`, responsável por manter as auditorias, destacam-se os métodos `addAuditoria()` e `executaAuditoria()` para adicionar e executar a auditoria. A `Auditoria` pode ter uma ou mais verificações.

A classe `Verificação` é responsável pela verificação dos controles nos servidores. Destacando-se os métodos `executaVerificação()` que verifica a segurança do controle da norma no servidor, `calculaAdequação()` para gerar o resultado dessa verificação e o

método `pegaVerificação()` que traz as verificações de uma auditoria.

Na classe `Controle` encontram-se os controles da norma NBR ISO/IEC 27002 que são verificados no software. E na classe `Script` esses controles são implementados pelos *scripts*. Destaque para o método `verificaIntegridade()` que obtém e compara o *hash* do arquivo de *script* antes de executá-lo.

A classe `So` mantém os sistemas operacionais suportados pela ferramenta, o método `verifica()` pesquisa se o sistema operacional do servidor é suportado pela ferramenta. A classe `Servidor` mantém os servidores para auditoria, o método `addLocal()` reconhece e cadastra automaticamente o servidor local e o `addRemoto()` reconhece e cadastra servidores remotos. No reconhecimento do servidor, são relacionadas as classes `Portas` e `InterfaceRedes`. Elas são responsáveis por manter as portas e interfaces de rede reconhecidas pelo servidor, através dos métodos `reconhecePortas()`, `pegaPortas()`, `salvaPortas()`, `reconheceRede()`, `pegaRede()` e `salvaRede()`.

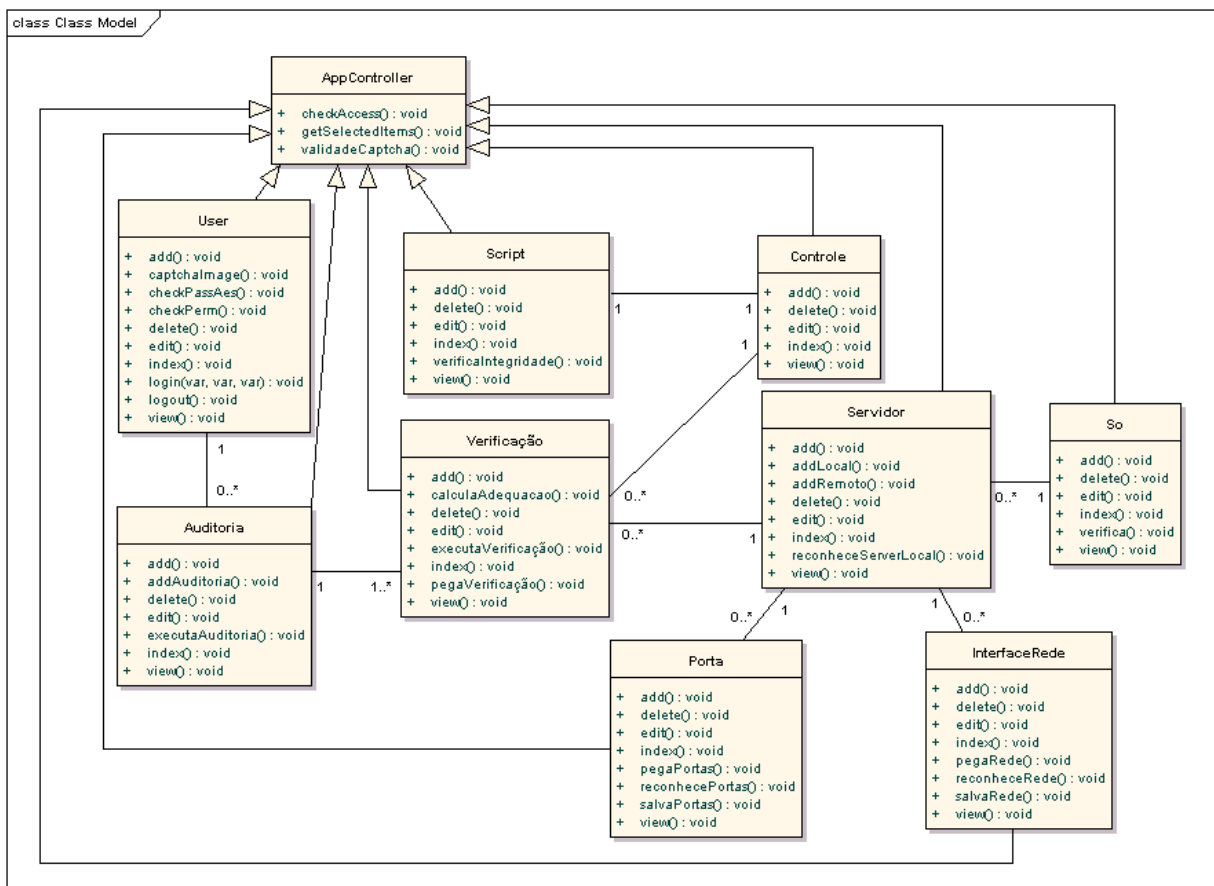


Figura 2 – Diagrama de classes

3.2.3 Diagramas de seqüência

Os diagramas de seqüência dos casos de uso do auditor relacionados na Figura 1 são demonstrados a seguir. Na Figura 3 pode-se visualizar o diagrama de seqüência para o caso de uso *Efetua login*.

Neste diagrama o auditor acessa qualquer página do sistema sem estar autenticado, o sistema então redireciona para a página de *login* chamando a função `captchaImage()` que exibirá uma imagem com o conteúdo do controle de segurança *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA) e retornará esta página para o usuário.

Após o auditor entrar com o usuário, senha e CAPTCHA, clicar no botão *login* o sistema chama a função de *login* que valida usuário, senha e também verifica o CAPTCHA digitado. Se as informações estiverem ok o acesso é autorizado.

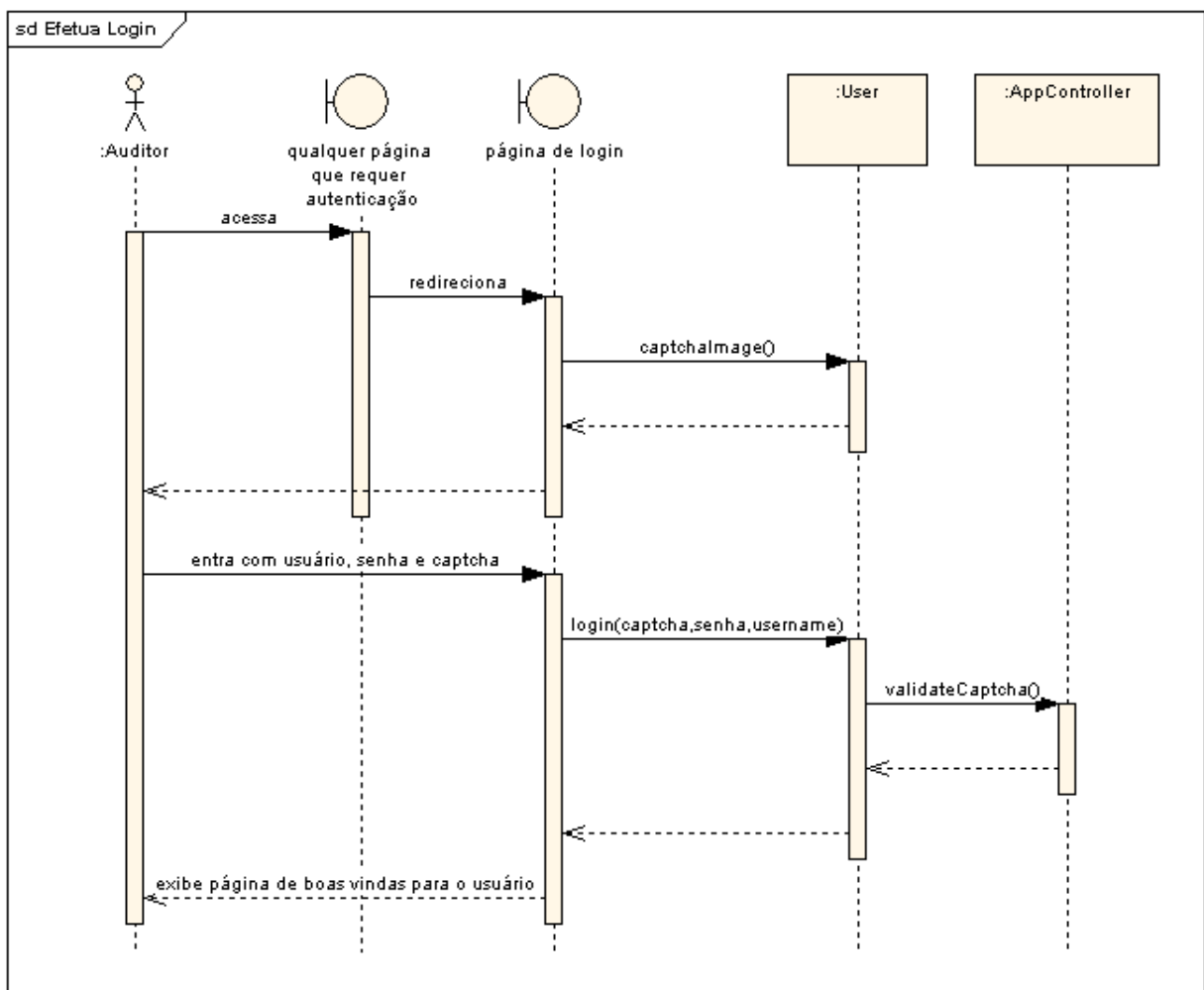


Figura 3 – Diagrama de seqüência efetua *login*

Na Figura 4 pode-se visualizar o processo em que o auditor cadastra o servidor.

Quando o auditor acessa o *link* Servidores, o sistema verifica se ele tem permissão para isso na função `checkPerm()`. Após o auditor clicar na opção para cadastramento local, o sistema verifica o sistema operacional através da função `verifica()`, chama a função `reconheceRede()` e depois `reconhecePortas()`, para fazer o reconhecimento das interfaces de rede da máquina e as portas abertas do servidor.

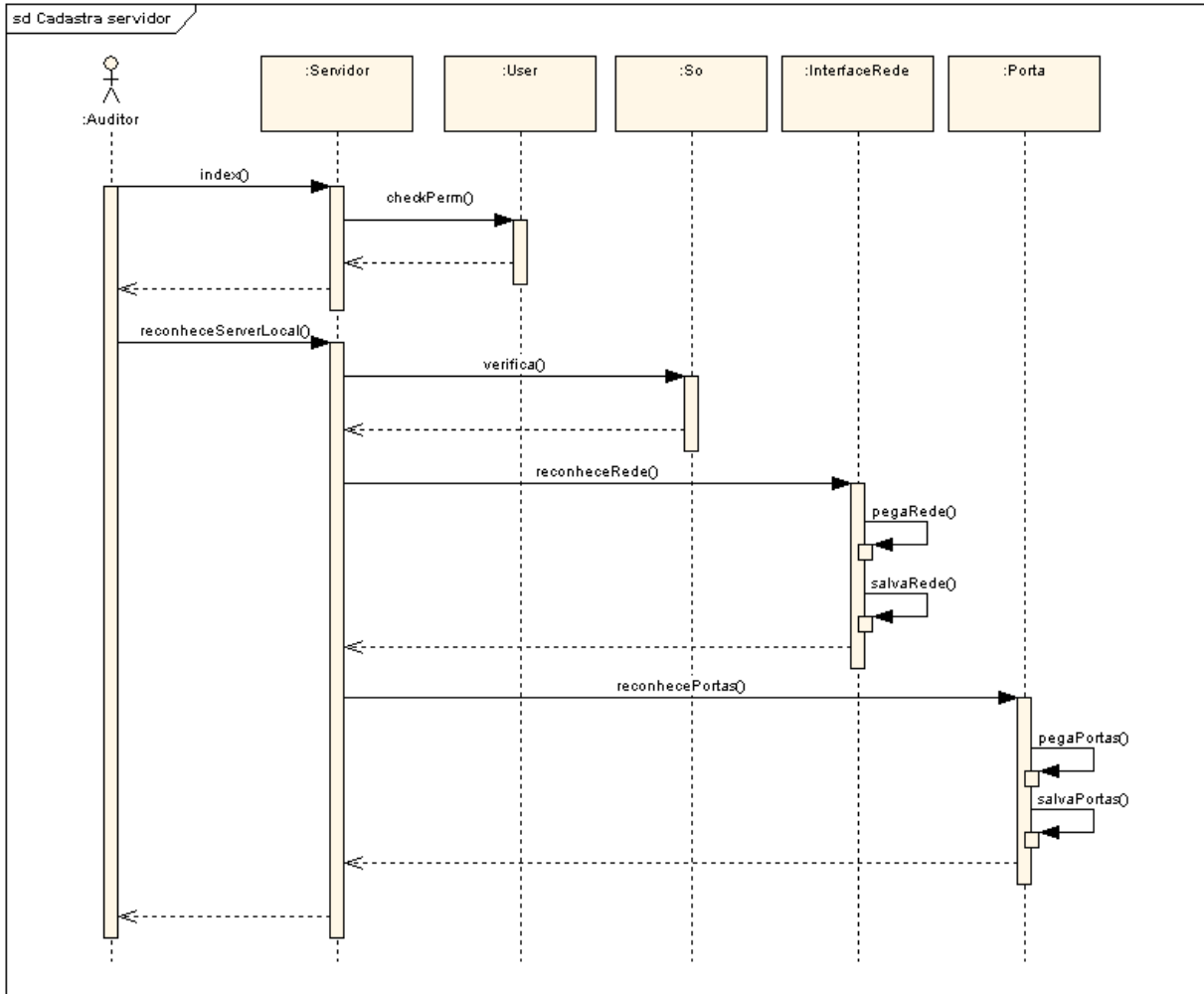


Figura 4 – Digrama de seqüência cadastra servidor

A Figura 5 demonstra a interação entre as classes para realizar o caso de uso de realizar a auditoria. O auditor, acessa o *link* `novaAuditoria()` de auditoria, o sistema pega a lista de servidores cadastrados na classe `Servidor`, através do método `generateList()`, que é provido pelo *framework* `CakePHP`, em seguida pega a relação dos controles cadastrados no sistema através do método `generateList()` na classe `Controle`, e exibe a página de auditoria possibilitando o auditor selecionar o servidor alvo da auditoria e os controles a serem verificados.

Tendo feito isso, o auditor clica em `Executar Auditoria`, que chama a função `executaAuditoria()`, para cada controle selecionado para verificação o sistema executa a

função `add()` da classe `Verificação`, que chama o método `executaVerificação()`. Em seguida chama `verificaIntegridade()` do `script` para antes de executar a verificação ter certeza de que o `script` está íntegro. Se ele estiver ok o método `executaScript()` é chamado e o `calculaResultado()` para calcular o nível de adequação obtido com a execução do `script`.

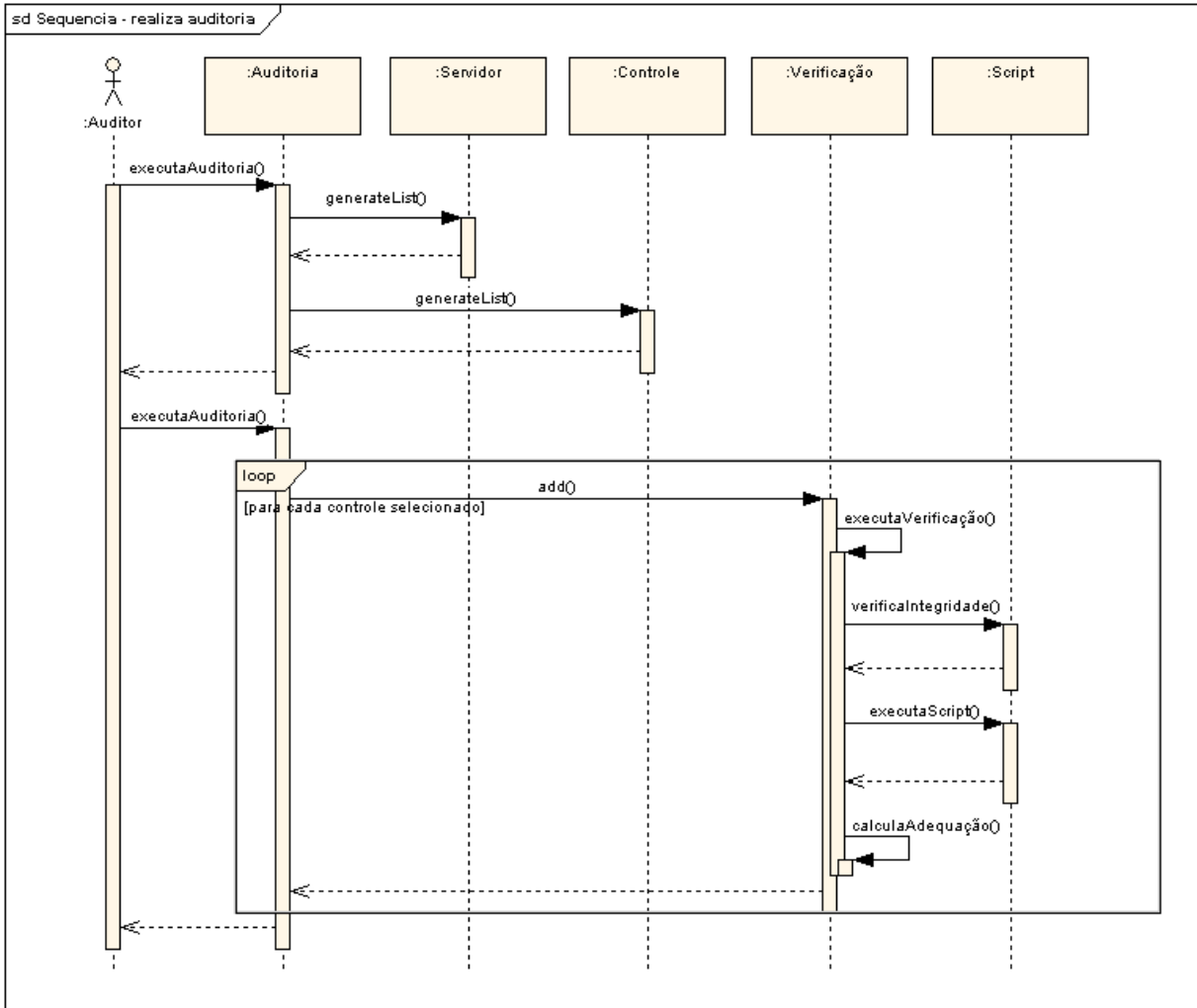


Figura 5 – Digrama de seqüência realiza auditoria

Na Figura 6 é demonstrado o diagrama de classes do caso de uso visualiza resultado, onde o auditor acessa a página principal da classe `Análise`, chamando a função `index()`, o sistema verifica se ele tem permissão de acesso através da função `checkPerm()` da classe `User`. Caso positivo, o sistema lista as auditorias realizadas, o usuário escolhe a auditoria a visualizar e clica em `view`, o sistema chama o método `visualizaAuditorias()` e para cada verificação pertencente a esta auditoria ele executa o `pegaVerificação()`. Ao final a página contendo os detalhes da auditoria e as suas verificações é exibida.

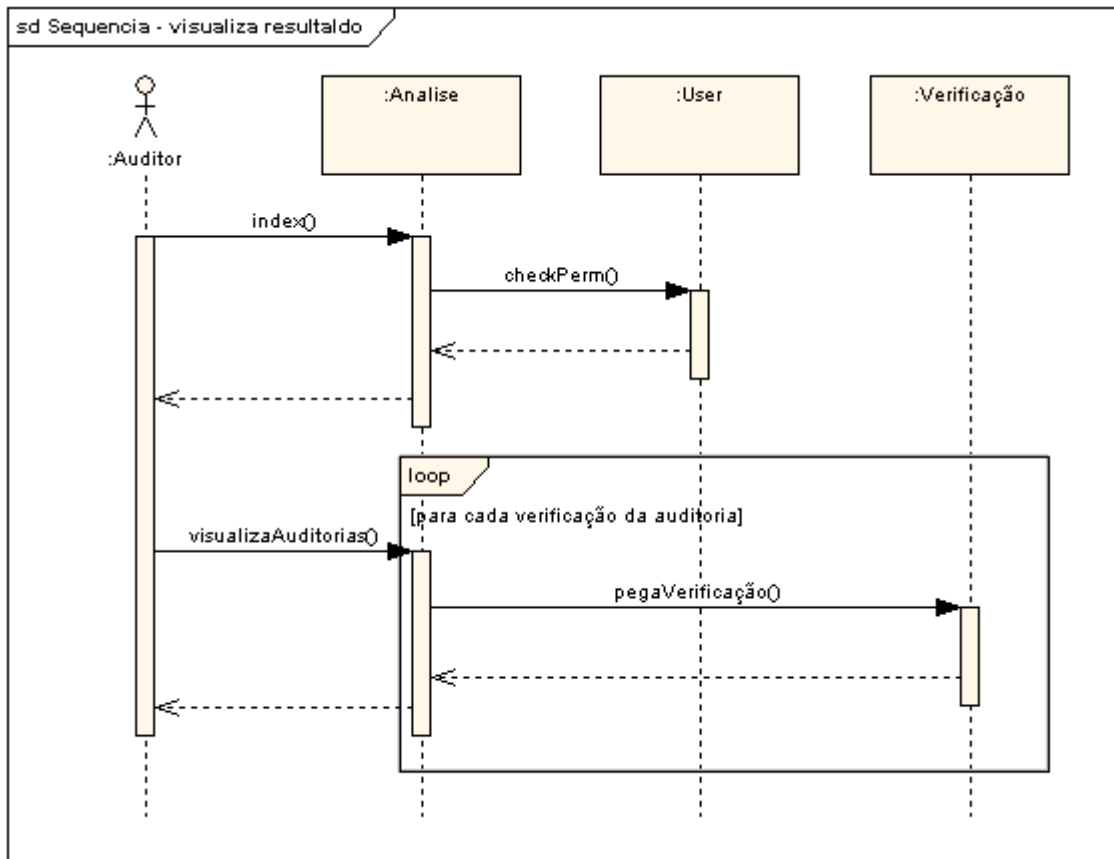


Figura 6 – Digrama de seqüência visualiza resultado

3.2.4 Diagrama de entidade e relacionamento

Este tópico contém o modelo físico utilizado no projeto, construído com a ferramenta DBDesigner, que pode ser visto na Figura 7. Para este modelo do banco foi consultado IBM (2007) e Gonçalves (2005). Adaptações foram feitas simplificando o modelo proposto por Gonçalves (2005), que se mostrou bastante completo, mas neste momento com complexidade desnecessária para esta aplicação, e ajustando o modelo básico da tabela `users` proposto por IBM (2007).

Como padrão, a chave primária das tabelas é sempre o campo `id`, do tipo `INT` e tamanho de 11. O nome utilizado nas tabelas é sempre no plural. E as chaves estrangeiras são definidas como `tabeladeorigemnosingular_id`. Esta padronização é importante para facilitar o desenvolvimento com o *framework* CakePHP, abordado na seção 3.3.1.

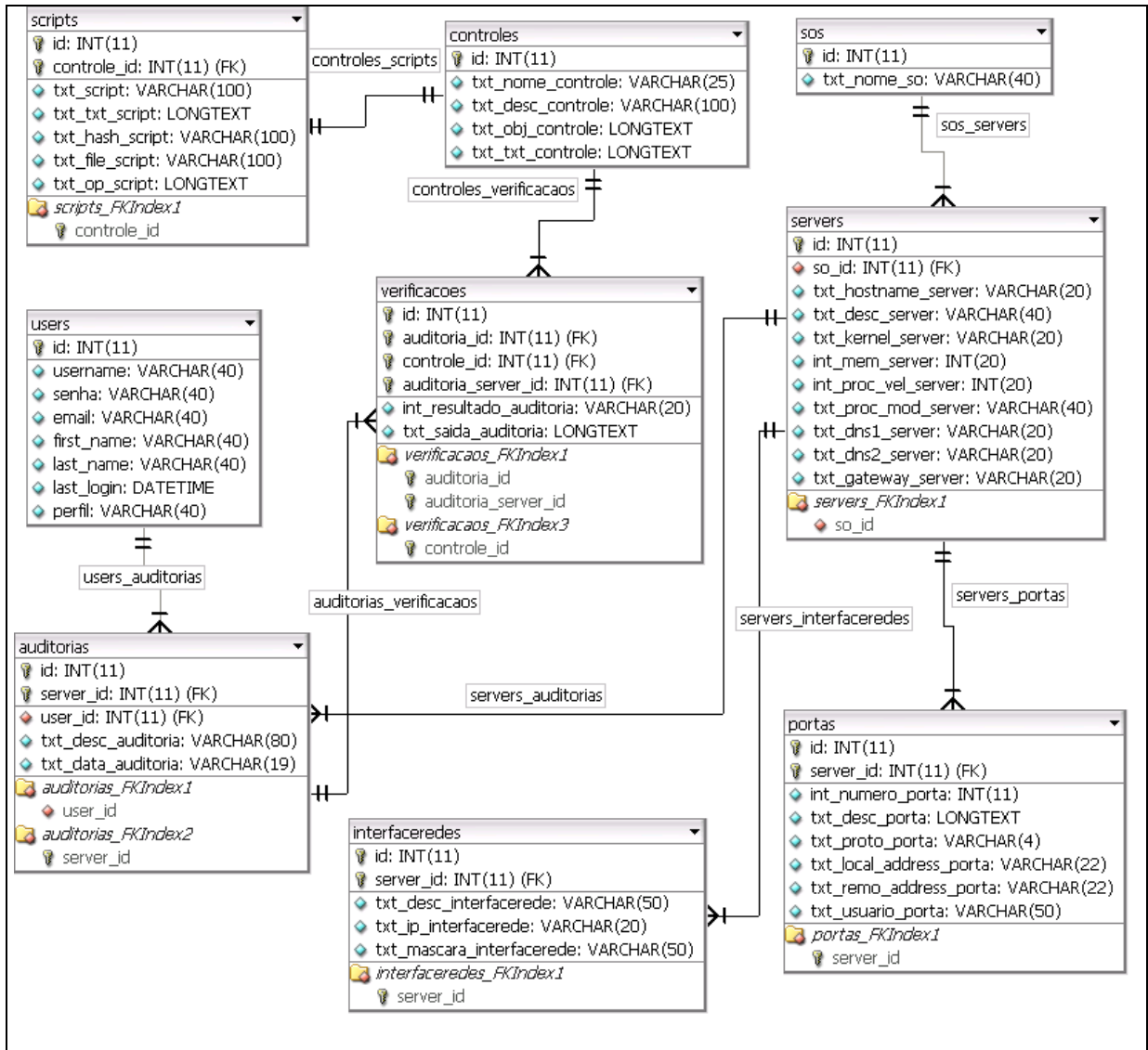


Figura 7 – Digrama de entidade e relacionamento

3.3 IMPLEMENTAÇÃO

Como a norma NBR ISO/IEC 27002 pretende abranger os aspectos de segurança de toda a organização, ela inclui controles e considerações não somente aos sistemas informatizados, como por exemplo Segurança em Recursos Humanos, e Segurança

Física e do Ambiente. Com isso, através da ferramenta, não é possível verificar se estes controles são implementados, pois esta verifica os controles computacionais.

Estes controles computacionais podem ainda ser implementados com a ajuda de ferramentas alternativas. Ficando sob a responsabilidade do proprietário do ativo, assumir os riscos quando esta ferramenta não reconhece um controle implementado, mas o mesmo considera que ele estaria satisfeito por outros meios.

Os controles da norma que foram implementados nos *scripts* são:

- a) 10.10.3 Proteção das informações dos registros (*log*);
- b) 10.10.6 Sincronização dos relógios;
- c) 11.2.2 Gerenciamento de privilégios;
- d) 11.3.1 Uso de Senhas;
- e) 11.4.6 Controle de conexão de rede;
- f) 11.5.4 Uso de utilitários de sistema;
- g) 11.5.5 Desconexão de terminal por inatividade;
- h) 11.5.6 Limitação de horário de conexão;
- i) 11.6.1 Restrição de acesso à informação.

A ferramenta desenvolvida foi idealizada como sendo uma ferramenta web. Por isso foi realizada uma pesquisa de *frameworks* que se propõe a ajudar ou facilitar o desenvolvimento web. Um dos requisitos desejados é que ele validasse os dados entrados em formulários. Garantindo que os dados enviados passassem por verificações a fim de validar que eles são consistentes com os valores possíveis para este campo.

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

O software livre foi desenvolvido na linguagem de programação PHP, utilizando quando necessário a execução de comandos do *shell* BASH. O ambiente de desenvolvimento utilizado foi o Eclipse com o *plugin* PHPEclipse. Para a modelagem e manutenção do banco de dados foram utilizados os softwares DBDesigner e phpMyAdmin. A segurança da aplicação foi implementada observando os controles da norma como descrito no tópico 3.3.6.

A linguagem de programação PHP foi escolhida por ter uma ampla utilização em

servidores GNU/Linux juntamente com o servidor web Apache, por suportar Orientação a Objetos (OO) e não demandar de requisitos excessivos de memória ou processador, permitindo a execução da ferramenta com um bom desempenho sem comprometer o funcionamento normal do servidor em que está instalada.

Para o desenvolvimento deste trabalho, optou-se por utilizar um *framework* em PHP que auxiliasse o processo de implementação da ferramenta. Pallett (2006) disponibiliza um comparativo entre dez *frameworks* em PHP que pode ser visualizado na Figura 8. Neste comparativo, o *framework* com mais funcionalidades é o Zoop Group (2007). Com base neste critério começou-se a testá-lo para verificar a sua possível utilização no trabalho. A sua instalação e configuração foi feita, não com facilidade, houveram problemas na sua configuração que tardaram a sua utilização. Mas o motivo decisivo pela decisão de procurar um outro *framework* foi a dificuldade em encontrar documentação do mesmo, e poucos exemplos do seu funcionamento.

Framework	PHP4	PHP5	MVC ¹	Multiple DB's ²	ORM ³	DB Objects ⁴	Templates ⁵	Caching ⁶	Validation ⁷	Ajax ⁸	Auth Module ⁹	Modules ¹⁰
Zend Framework	-	✓	✓	✓	-	✓	-	✓	✓	-	-	✓
CakePHP	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	-
Symfony Project	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	-
Seagull Framework	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓
WACT	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	-
Prado	-	✓	-	✓	-	-	✓	✓	✓	✓	✓	✓
PHP on TRAX	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	-
ZooP Framework	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓
eZ Components	-	✓	-	✓	-	✓	✓	✓	✓	-	-	✓
CodeIgniter	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	✓

#1: Indicates whether the framework comes with inbuilt support for a Model-View-Controller setup.
 #2: Indicates whether the framework supports multiple databases without having to change anything.
 #3: Indicates whether the framework supports an object-record mapper, usually an implementation of ActiveRecord.
 #4: Indicates whether the framework includes other database objects, like a TableGateway.
 #5: Indicates whether the framework has an inbuilt template engine.
 #6: Indicates whether the framework includes a caching object or some way other way of caching.
 #7: Indicates whether the framework has an inbuilt validation or filtering component.
 #8: Indicates whether the framework comes with inbuilt support for Ajax.
 #9: Indicates whether the framework has an inbuilt module for handling user authentication.
 #10: Indicates whether the framework has other modules, like an RSS feed parser, PDF module or anything else (useful).

Fonte: Pallett (2006).

Figura 8 – Comparativo entre *frameworks* PHP

O próximo *framework* testado teve a sua instalação e configuração facilitada, e pesquisando na internet foi encontrado uma comunidade ativa utilizando e contribuindo com o seu desenvolvimento. O *framework* CakePHP foi utilizado como base para o desenvolvimento do software, facilitando a implementação da verificação dos dados transmitidos pelos formulários da aplicação, o desenvolvimento baseado no modelo de classes definido, e a possibilidade da separação em três camadas da aplicação: *Model*, *View* e *Controller* (MVC).

Foi utilizado para o desenvolvimento do trabalho a ferramenta `bake` do CakePHP, esta facilita o início do desenvolvimento através da geração automática das classes do modelo utilizado com base nas tabelas do banco de dados (*Model*), da geração do controle que é a lógica do negócio da aplicação (*Controller*), inserindo automaticamente código para a adição, alteração e deleção de objetos no banco. Permite também a criação automática do código de exibição na tela (*View*), com os controles apropriados dos campos.

Esta parte foi bastante útil para validar o modelo do banco de dados definido, já que a ferramenta `Bake` lê o modelo do banco como índices, relacionamentos, chaves estrangeiras para outras tabelas e os mapeia para o CakePHP criando um *scaffolding*, ou seja, páginas para cada tabela do banco, com funções básicas visualização e manutenção dos registros como `index()`, `add()`, `edit()` e `delete()`. Desta forma, criando o *scaffolding* já era possível testar a manutenção e o relacionamento entre as tabelas. O Quadro 1 mostra a tela inicial da ferramenta `Bake`.

```

Check27002.org:/var/www/check27002/cake/scripts# php bake.php

|_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
|_| |_| |_| |_| \_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_|
-----

Bake -app in /var/www/check27002/app (y/n)
[y] > y

Baking...
-----
Name: app
Path: /var/www/check27002/app
-----

[M]odel
[C]ontroller
[V]iew

What would you like to Bake? (M/V/C)
>

```

Quadro 1 – Ferramenta `Bake`

3.3.2 Implementação do software

O processo de planejar, modelar e construir a aplicação no cakePHP demanda o conhecimento de seus conceitos, padronização de campos, variáveis, nomes de tabelas, índices de tabelas, modelos, visões, etc., que são fundamentais para utilizar toda a facilidade e

capacidade que o *framework* proporciona, justificando assim a sua utilização, e transformando em vantagens para o trabalho as suas funcionalidades suportadas.

A programação para exibir em outro controle ao invés do código da chave estrangeira, a sua descrição, nome, ou informações combinadas como código e nome, foi realizada utilizando a função `myGenerateList()` (KOSCHUETZKI, 2007). A função é inserida na classe `AppModel`, de onde são extendidas as demais classes da aplicação, permitindo que em qualquer classe ela esteja disponível. Facilitando o desenvolvimento de seleção da chave estrangeira exibindo descrições, nomes ou campos combinados. Foi importante principalmente para o desenvolvimento da ferramenta, onde ao cadastrar as auditorias e verificações manuais, era possível exibir descrições além da chave estrangeira. Com o amadurecimento e estabilização da ferramenta as auditorias e verificações se tornaram cada vez mais automatizadas, diminuindo a seleção ou intervenção do auditor. Mas a função continua sendo utilizada facilitando a seleção de `Servidores` e `Controles`.

Para o processo de automaticamente verificar a segurança do servidor foi necessário dividi-lo em partes, como segue a ordem do desenvolvimento:

- a) a primeira parte foi o reconhecimento das configurações do servidor local, pegar isso automaticamente e salvá-las no banco de dados;
- b) depois, pegar as interfaces de rede do mesmo, através do comando `ifconfig`, quebrando e salvando da sua saída os campos relevantes para as posteriores verificações de segurança, identificando interfaces do tipo local ou *ethernet*;
- c) após isso, houve a necessidade de pegar as portas abertas, e os seus respectivos processos rodando na máquina, para isso foi necessário executar o comando `netstat`, a sua saída quebrar, verificar as linhas relevantes, e salvar estas informações no banco para também futuramente fazer as verificações de segurança.

Inicialmente a aplicação havia sido imaginada como executando localmente no servidor a ser auditado, reconhecendo automaticamente as suas configurações, mas se restringindo a fazer isso local. Com o desenvolvimento da ferramenta, verificou-se a possibilidade de implementar a auditoria remota através de uma conexão *Secure Shell* (SSH), o que facilita a auditoria em uma empresa que tem mais servidores GNU/Linux. Centralizando a instalação do software livre em uma única máquina e nos demais servidores a serem auditados seria necessário somente fazer a liberação do usuário e senha configurados na ferramenta permitindo a auditoria remota.

Desta forma, é acessado o servidor da auditoria, copiado o arquivo ou executado o comando necessário, e retornado ao servidor da ferramenta para executar o processamento da

informação obtida. O que pode ser considerado uma vantagem quando se tem um servidor em produção a ser auditado, pois a verificação dos controles, em alguns casos necessita de um grande processamento, o que poderia comprometer a performance de um servidor em produção com aplicações críticas para a empresa.

3.3.3 Implementação dos *scripts*

Neste tópico será demonstrado como foram implementados alguns dos *scripts* que verificam as recomendações dos controles no servidor.

Para o desenvolvimento dos controles aplicados ao servidor GNU/Linux não foi possível verificar a implementação feita por Gonçalves (2007) no seu trabalho. O livro BS7799 da Tática à Prática em servidores Linux (MELO, et al., 2006) relaciona a teoria da norma, e a prática, a aplicação dos controles da norma em servidores GNU/Linux. Com isso a implementação dos *scripts* de verificação dos controles pode ser equiparada nas recomendações de configuração de Melo et al. (2006).

A norma NBR ISO/IEC 27002 recomenda medidas de segurança através dos seus controles, Melo et al. (2006) sugere uma forma de implementação destas medidas em servidores GNU/Linux, e a ferramenta desenvolvida neste trabalho verifica se os controles da norma estão implementados em servidores GNU/Linux.

```

<?php
/*
 * Created on Set 27, 2007
 * To change the template for this generated file go to
 * Window - Preferences - PHPeclipse - PHP - Code Templates
 */
    // informar o nro do script
    array_push($this->txt_saida_auditoria, "Iniciando o Scrip xxxxx");
    // informar a descrição dele
    array_push($this->txt_saida_auditoria, "Descrição do Script xxxxx");

    // executar as verificações necessárias
    // ...
    // ...

    // para cada verificação ok encontrada atribuir o seguinte:
    //array_push($this->adequacao, 1);
    // ou para cada verificação não ok encontrada
    // array_push($this->adequacao, 0);

    array_push($this->txt_saida_auditoria, "Terminou a verificação xxx");
?>

```

Quadro 2 – *Script* de exemplo

No Quadro 2 é mostrado um *script* de exemplo, contendo a estrutura básica para a construção de novos *scripts*. No Apêndice C consta o manual para a personalização ou criação de novos *scripts*.

Identificou-se, já na construção dos *scripts*, que haveria situações em que a programação implementada não verificaria todas as possibilidades, deixando uma margem para ocorrer algum falso negativo na sua implementação, como pode ser visualizado no Quadro 3, onde um agendamento do crontab não seria reconhecido, mesmo contendo internamente o comando correto para se adequar ao controle.

```
Check27002.org:~# crontab -l
// ok
3 4 * * * /usr/sbin/ntpdate 200.135.24.8
// falso negativo
3 4 * * * /root/atualiza_horario.sh
```

Quadro 3 – Falso negativo nos *scripts*

Para entender melhor a ferramenta, é importante verificar o funcionamento de três dos principais *scripts*, os quais seguem nas próximas seções.

3.3.3.1 Script 10.10.3.php verificação dos arquivos de log

No Quadro 4 é exibido o *script* que realiza a verificação dos arquivos de *log*, onde o servidor executa o comando do BASH `fuser` para verificar se o arquivo `/sbin/syslog-ng` está sendo executado. Desta forma, se estiver executando ele vai pegar o arquivo de configuração específico para o sistema Syslog-NG, senão ele vai verificar o arquivo de configuração padrão do GNU/Linux, o Syslogd.

```

...
    array_push($this->txt_saida_auditoria, "Verificando a relação dos arquivos
de log");

    // verificando que sistema de logs está rodando
    unset($cat);
    unset($cat2);
    exec ("sudo -u check27002 fuser -v /sbin/syslog-ng", $cat2);
    if ( $cat2 ){
        //echo "Reconhecido o sistema de logs Syslog-NG";
        exec ("sudo -u check27002 cat /etc/syslog-ng/syslog-ng.conf", $cat);

        verificaSyslogNG($cat,$this->txt_saida_auditoria,$this->retorno);
    } else {
        exec ("sudo -u check27002 fuser -v /sbin/syslogd", $cat2);
        if ( $cat2 ){
            //echo "Reconhecido o sistema de logs Syslogd";
            exec ("sudo -u check27002 cat /etc/syslog.conf", $cat);
            verificaSyslogd($cat,$this->txt_saida_auditoria,$this-
>retorno);
        } else {
            array_push($this->txt_saida_auditoria, "Não foi Reconhecido o
sistema de logs utilizado no servidor");
            $this->retorno = '0';
        }
    }
}
...

```

Quadro 4 – Script 10.10.3.php verificação dos arquivos de log

3.3.3.2 Script 10.10.6.php sincronização dos relógios

No Quadro 5 é mostrado o *script* que realiza a verificação da sincronização do relógio no servidor. Primeiramente ele verifica no `crontab` do usuário `root`, para identificar se está agendada a execução dos comandos, e em seguida verifica no arquivo `/etc/rc.local` se consta para executar na inicialização da máquina os comandos `ntupdate` ou `ntpd -q`.

```

<?php
    array_push($this->txt_saida_auditoria, "Iniciando o Scrip 10.10.6");
    array_push($this->txt_saida_auditoria, "Sincronização dos relógios");
    $encontrou = false;
    array_push($this->txt_saida_auditoria, "Verificando a Sincronização dos
relógios no crontab do usuario root");
    exec ("sudo -u check27002 crontab -u root -l", $saidasudo);
    array_push($this->txt_saida_auditoria, "Varrendo o arquivo...");
    foreach ($saidasudo as $key => $linha){
        array_push($this->txt_saida_auditoria, "Pesquisando na linha
$key...");
        if (ereg ("^[^#]+(. *ntpd -q.*|. *ntpddate*)", $linha)) { //
pesquisando por expresao regular, se nao
// começa por comentario, e tenha ou o comando ntpd -q
// ou o comando ntpdate
            array_push($this->txt_saida_auditoria, "Opção de atualização
do relógio do servidor nos agendamentos OK: $linha, encontrada na linha $key");

            array_push($this->adequacao, 1);
            array_push($this->txt_saida_auditoria, "Continuando...");

            $encontrou = true;
            break;
        }
    }
    if (! $encontrou) { // se nao encontrou seta essa verificação com falsa
        array_push($this->adequacao, 0);
    }
    unset ($saidasudo);
    $encontrou = false;
    array_push($this->txt_saida_auditoria, "Verificando a Sincronização dos
relógios na inicialização do sistema");
    exec ("sudo -u check27002 cat /etc/rc.local", $saidasudo);
    array_push($this->txt_saida_auditoria, "Varrendo o arquivo...");
    foreach ($saidasudo as $key => $linha){
        array_push($this->txt_saida_auditoria, "Pesquisando na linha
$key...");
        if (ereg ("^[^#]+(. *ntpd -q.*|. *ntpddate*)", $linha)) { //
pesquisando por expresao regular, se não começa por comentario, e tenha ou o
comando ntpd -q ou o comando ntpdate
            array_push($this->txt_saida_auditoria, "Opção de atualização
do relógio do servidor na inicialização OK: $linha, encontrada na linha $key");
            array_push($this->adequacao, 1);
            array_push($this->txt_saida_auditoria, "Continuando...");

            $encontrou = true;
            break;
        }
    }
    if (! $encontrou) { // se nao encontrou seta essa verificação com falsa
        array_push($this->adequacao, 0);
    }
    array_push($this->txt_saida_auditoria, "passou");
?>

```

Quadro 5 – Script 10.10.6.php sincronização dos relógios

3.3.3.3 Script 11.4.6.php controle de conexões de rede

No Quadro 5 conta o *script* que realiza o controle das conexões de rede. Este *script* implementa várias checagens, que são citadas no quadro e inserido somente as expressões regulares que as verificam, estando a implementação completa disponível nos fontes do projeto. Executa checagens no arquivo de configuração do serviço SSH `/etc/ssh/sshd_config` para verificar a porta de acesso configurada, o IP, a possibilidade do usuário `root` fazer *login*, e verifica também a exibição de um *banner* com a política de segurança da organização.

```
<?php
/*
 * Created on Sep 22, 2007
 *
 * To change the template for this generated file go to
 * Window - Preferences - PHPeclipse - PHP - Code Templates
 */

    array_push($this->txt_saida_auditoria, "Iniciando o Scrip 11.4.6");
    array_push($this->txt_saida_auditoria, "Controle de conexão de
rede");
    exec ("sudo -u check27002 cat /etc/ssh/sshd_config", $cat);
...
    array_push($this->txt_saida_auditoria, "Verificando a configuração da
porta de acesso.");
if (ereg ("^ *Port +[^\d] *", $linha)) {
...
...
    // aqui começa outra verificação
    array_push($this->txt_saida_auditoria, "Verificando a configuração do IP
.");
if (ereg ("^ *ListenAddress +[^\.\0\.\0\.\0|\.:\:] *", $linha)) {
...
...
    // aqui começa outra verificação
    array_push($this->txt_saida_auditoria, "Verificando a possibilidade do
root fazer login.");
if (ereg ("^ *PermitRootLogin +no *", $linha)) {
...
...
    // aqui começa outra verificação
    array_push($this->txt_saida_auditoria, "Verificando a exibição de banner
com a política de segurança.");
if (ereg (".*[p|P]olítica.*|.*POLÍTICA.*|.*[s|S]egurança.*|.*SEGURANÇA.*",
$linha2)) {
...
...
    array_push($this->txt_saida_auditoria, "Terminou a verificação 11.4.6");
?>
```

Quadro 6 – Script 11.4.6.php controle de conexões de rede

3.3.4 Cálculo da adequação

O resultado do *script* é o percentual de adequação verificado no servidor auditado, para realizar este cálculo, é chamada a função `calculaAdequacao()` passando como parâmetro a variável `$itens` que é um array, contendo elementos booleanos, com 0 indicando uma não adequação encontrada, ou 1 indicando uma adequação. A função pega o número de 1s recebidos no array, proporcional ao número total de elementos recebidos no array gerando o percentual de adequação do *script*, salvando em `$this->int_resultado_adequacao`, como exibido no Quadro 7.

Vale definir que checagem no desenvolvimento deste trabalho significa divisões de uma verificação, ou sub-verificações, implementadas em um *script*. Como exemplo, no Quadro 5, na verificação do *script* 11.4.6, tem-se 4 checagens:

- a) porta de acesso configurada;
- b) endereço IP;
- c) possibilidade do usuário `root` fazer *login*;
- d) exibição de *banner* com a política de segurança da organização.

```
function calculaAdequacao ($itens ) {

    $msg = "Recebido " . count($itens) . " itens no total.";

    array_push($this->txt_saida_auditoria, $msg );
    $nrook = 0;
    if ($itens) // se não vier nenhum item ele não vai dividir por 0 ou null
    {
        foreach ( $itens as $item => $valor) {
            if ($valor == '1') {
                $nrook +=1;
            }
        }
        array_push($this->txt_saida_auditoria, "Imprimindo o nro de oks
    $nrook .");

        // colocando o resultado na variavel certa
        $this->int_resultado_adequacao = ($nrook * 100)/count($itens);
        array_push($this->txt_saida_auditoria, "Imprimindo a adequação do
    controle $this->int_resultado_adequacao %.");
    } else {
        array_push($this->txt_saida_auditoria, "Não recebeu nenhum parametro
    para calcular.");
        array_push($this->txt_saida_auditoria, "Saindo com resultado igual a
    zero.");
        $this->int_resultado_adequacao = 0;
    }
}
```

Quadro 7 – Calcula adequação

3.3.5 Considerações quanto à auditoria de sistema de informação

Com o objetivo de maximizar a eficácia e minimizar a interferência no processo de auditoria dos sistemas de informação, na norma o item 15.3 Considerações quanto à auditoria de sistemas de informação traz recomendações importantes para a realização de auditorias que são atendidas neste trabalho. No Quadro 8 seguem as diretrizes para a implementação do tópico 15.3.1 Controles de auditoria de sistemas de informação e a sua situação na implementação do sistema desenvolvido (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2005, p. 113).

Diretrizes para implementação	Situação no sistema
a) requisitos de auditoria sejam acordados com o nível apropriado da administração;	Não se aplica ao sistema por ser uma recomendação em nível de recursos humanos e não de software.
b) escopo da verificação seja acordado e controlado;	Atende. O escopo da auditoria são os controles implementados pelo software, estes são aceitos ao logar na ferramenta.
c) a verificação esteja limitada ao acesso somente para leitura de <i>software</i> e dados;	Atende. As informações são obtidas e verificadas dos servidores, mas nada é alterado. Esse é o motivo pelo qual a ferramenta não corrige os problemas encontrados.
d) outros acessos diferentes de apenas leitura sejam permitidos somente através de cópias isoladas dos arquivos do sistema, e sejam apagados ao final da auditoria, ou dada proteção apropriada quando existir uma obrigação para guardar tais arquivos como requisitos da documentação da auditoria;	Não se aplica. Todos os acesso foram programados como somente leitura.
e) recursos para execução da verificação sejam identificados explicitamente e tornados disponíveis;	Atende. Os requisitos para a instalação, configuração e execução da ferramenta constam na documentação de instalação disponível no Apêndice A.
f) requisitos para processamento adicional ou especial sejam identificados e acordados;	Atende. O requisito especial do <i>script</i> 11.6.1 que verifica em todo o sistema de arquivos permissões incorretas, demandando um alto nível de I/O nos discos está devidamente alertado no momento da instalação da ferramenta e no momento da execução do <i>script</i> .
g) todo acesso seja monitorado e registrado de forma a produzir uma trilha de referência; convém que o uso de trilhas de referência (<i>time stamped</i>) seja considerado para os sistemas ou dados críticos;	Atende. A aplicação gera log de todos os acessos feitos no sistema, bem ou mal sucedidos. Bem como registra todas as ações executadas pelos usuários.
	Atende. Os procedimentos estão documentados e podem ser consultados pelo sistema.

h) todos os procedimentos, requisitos e responsabilidades sejam documentados;	A documentação da auditoria está disponível na página principal; Requisitos de instalação estão documentados no manual de instalação; E as responsabilidades são aceitas ao acessar o sistema.
i) as pessoas que executem a auditoria sejam independentes das atividades auditadas.	Não se aplica ao sistema por ser uma recomendação em nível de recursos humanos e não de software.

Fonte: adaptado de Associação Brasileira de Normas Técnicas (2005).

Quadro 8 – Diretrizes para implementação do controle 15.3.1

Em seqüência o tópico 15.3.2 traz mais recomendações, como podem ser vistas no Quadro 9.

15.3.2 Proteção de ferramentas de auditoria de sistemas de informação

Controle

Convém que o acesso às ferramentas de auditoria de sistema de informação seja protegido, para prevenir qualquer possibilidade de uso impróprio ou comprometimento.

Diretrizes para implementação

Convém que acessos às ferramentas de auditoria de sistemas de informação, por exemplo, *software* ou arquivos de dados, sejam separados de sistemas em desenvolvimento e em operação e não sejam mantidos em fitas de biblioteca ou áreas de usuários, a menos que seja dado um nível apropriado de proteção adicional.

Informações adicionais

Quando terceiros estão envolvidos em uma auditoria, existe um risco de mau uso de ferramentas de auditoria por esses terceiros e da informação que está sendo acessada por este terceiro. Controles, tais como em 6.2.1 (para avaliar os riscos) e 9.1.2 (para restringir o acesso físico), podem ser considerados para contemplar este risco, e convém que quaisquer conseqüências, tais como trocar imediatamente as senhas reveladas para os auditores, sejam tomadas.

Fonte: adaptado de Associação Brasileira de Normas Técnicas (2005).

Quadro 9 – Diretrizes para implementação do controle 15.3.2

Os itens de segurança implementados na aplicação, visando atender a esta recomendação da norma, são relacionados na seção 3.3.6.

3.3.6 Segurança implementada na aplicação

Nesta seção são demonstradas as técnicas de segurança selecionadas e implementadas nesta ferramenta.

3.3.6.1 Captcha

Na página de *login* do sistema, além de pedir a identificação do usuário através do seu usuário e senha, o sistema solicita o captcha como exibido na Figura 9. Para essa implementação foi incluído ao projeto o componente PhpCaptcha (ELIOT, 2007), conforme documentação consultada de The Bakery Everything CakePHP (2007).



The image shows a web form for logging in. At the top, it says "Check27002 : Welcome". Below that, a red message reads "Deve ser feito o login para acessar o sistema" (Login must be performed to access the system). Underneath, it says "Please log in." The form includes two input fields: "Nome de Usuário:" (Username) and "Senha:" (Password). Below these is a captcha image showing the letters "GN OHP M" overlaid on a complex, scribbled background. There is a "Reload image" link below the captcha. At the bottom of the form is a "Captcha:" label and an input field for the user to enter the captcha text. A "login" button is located at the very bottom of the form.

Figura 9 – Página de *login* com captcha

Esta segurança é importante para a ferramenta evitando que o acesso aconteça por algum software e não pelo auditor. Reduzindo também a efetividade de ataques de força bruta na exploração de usuário e senha com permissão de acesso à ferramenta.

3.3.6.2 Autenticação

Conforme exibido na Figura 9 o sistema solicita a autenticação do usuário através do seu nome de usuário e senha. Estas informações são validadas na função *login* que pode ser vista no Quadro 10. Nesta função, se for enviado dados, primeiramente é chamado o método para verificar o captcha digitado, `validateCaptcha()`, em seguida é executado o método para pesquisar o usuário digitado. Após isso, é verificada se a senha confere, através da função `checkPassAes()`. Se a senha confere, é criada a seção para o usuário, contendo

principalmente as variáveis de seção user e perfil.

```
function login()
{
    $this->set('error', false);
    if ($this->data)
    {
        if ( $this->validateCaptcha( strtoupper($this->data['User']['userCode'])) ) {

            $results = $this->User->findByUsername($this->data['User']['username']);

            if ($results && $this->checkPassAes( $this->data['User']['senha'], $results))
            {
                $this->Session->write('user', $this->data['User']['username']);
                $this->Session->write('last_login', $results['User']['last_login']);
                $this->Session->write('perfil', $results['User']['perfil']);
                $results['User']['last_login'] = date("Y-m-d H:i:s");
                $this->User->save($results);
                $this->redirect('/users/index');
            } else {
                $this->set('error', true);
            }
        } else {
            $this->set('error', true);
            $this->Session->setFlash("Digite o código de verificação corretamente");
        }
    }
    $this->render();
}
```

Fonte: adaptado de IBM (2007).

Quadro 10 – Função de *login* na *Controller User*

3.3.6.3 Validação dos dados de entrada

Modelo da classe *User*, com a variável *validate* definindo os campos que são verificados para este modelo, como visto no Quadro 11.

```
<?php
class User extends AppModel {

    var $name = 'User';
    var $validate = array(
        'id' => VALID_NOT_EMPTY,
        'username' => '/^{6,40}$/',
        'senha' => '/^{6,40}$/',
        'email' => VALID_EMAIL,
        ...
    );
}
```

Quadro 11 – Validação dos dados de entrada no *Model User*

Com o CakePHP as definições de validação ocorrem no Model e na View, não necessitando fazer nenhuma programação na camada Controller. No Quadro 12 segue a codificação para exibir a mensagem de erro ao usuário se o campo informado estiver incorreto.

```

<h2>Novo Usuário</h2>
<form action="<?php echo $html->url('/users/add'); ?>" method="post">
<div class="required">
    <?php echo $form->labelTag('User/username', 'Username');?>
    <?php echo $html->input('User/username', array('size' =>
'60'));?>
    <?php echo $html->tagErrorMsg('User/username', 'Informe o Nome de
Usuário. ');?>
</div>
<div class="required">
    <?php echo $form->labelTag('User/senha', 'Senha');?>
    <?php echo $html->input('User/senha', array('size' => '60'));?>
    <?php echo $html->tagErrorMsg('User/senha', 'Informe a
Senha. ');?>
</div>
<div class="required">
    <?php echo $form->labelTag('User/email', 'Email');?>
    <?php echo $html->input('User/email', array('size' => '60'));?>
    <?php echo $html->tagErrorMsg('User/email', 'Informe o
Email. ');?>
</div>
<div class="submit">
    <?php echo $html->submit('Add');?>
</div>
</form>
<ul class="actions">
<li><?php echo $html->link('List Users', '/users/index')?></li>
</ul>

```

Quadro 12 – Validação dos dados de entrada na View User

3.3.6.4 Controle do processamento interno

A norma orienta que sejam incorporadas nas aplicações checagens de validação com o objetivo de detectar qualquer corrupção das informações, seja causada por algum erro ou por ação deliberada.

Dentre as recomendações da norma com relação às diretrizes para implantação puderam ser atendidas as relacionadas no Quadro 13.

Diretrizes para implementação	Situação no sistema
c) validação de dados de entrada gerados pelo sistema (ver 12.2.1);	Atende. Conforme demonstrado na seção 3.3.6.3.
d) verificações de integridade, autenticidade ou qualquer outra característica de segurança, de dados ou <i>softwares</i> transferidos, ou atualizados entre computadores centrais e remotos;	Atende. Implementado no reconhecimento remoto.
e) implementação de técnicas de consistência (<i>hash</i>) para registros e arquivos;	Atende. Utilizada na verificação dos <i>scripts</i> .

Fonte: adaptado de Associação Brasileira de Normas Técnicas (2005).

Quadro 13 – Diretrizes para implementação do controle 12.2.2

3.3.6.5 Validação de dados de saída

Com relação aos controles efetuados nos dados de saída, relacionados no item 12.2.4 da norma, puderam ser aplicados ao sistema os relacionados no Quadro 14.

Diretrizes para implementação	Situação no sistema
a) verificações de plausibilidade para testar se os dados de saída são razoáveis;	Atende. Na hora de gravar o resultado da auditoria.
b) controles envolvendo contagens de reconciliação para garantir o processamento de todos os dados;	Atende. Em cada verificação é gravada junto um campo com o hash do registro, e ao final da auditoria, é gravado um hash de todas as verificações, validando cada verificação individualmente e todas elas na auditoria.

Fonte: adaptado de Associação Brasileira de Normas Técnicas (2005, p. 87).

Quadro 14 – Diretrizes para implementação do controle 12.2.4

3.3.6.6 Criptografia

Segundo MySQL AB (2007, tradução nossa) *exploits* para os algoritmos md5 e sha1 se tornaram conhecidos [...] as funções `AES_ENCRYPT()` e `AES_DECRYPT()` podem ser

consideradas as mais seguras funções de criptografia disponíveis atualmente no MySQL.

A criptografia utilizada na ferramenta é o algoritmo *Advanced Encryption Standard* (AES) com chave de 128 bits. É utilizado na função de cadastro de usuário e na comparação da senha dos usuários `checkPassAes()`.

3.3.6.7 Controle de acesso ao código-fonte de programa

A norma recomenda que o acesso ao código fonte de programa seja restrito. Por se tratar de software livre esse item não se aplica, o acesso ao código fonte do programa está disponível livremente para *download* no *site* do projeto. A visão de segurança do software é, por se tratar de um software livre onde todos tem acesso aos fontes do programa, que os problemas de segurança serão encontrados e corrigidos mais rapidamente, diferente de softwares proprietários onde os eventuais problemas de segurança ficam ocultos. O diretório onde ele consta instalado no servidor, este sim tem restrições de acesso quanto à gravação e execução de arquivos pelos usuários.

3.3.6.8 Gestão de vulnerabilidades técnicas

O software implementa a sua gestão das vulnerabilidades técnicas através do menu *Atualizações*, disponível no menu do administrador. É executada cada vez que um usuário faz *login* no sistema, informando a versão atual e se existem atualizações disponíveis. Quando executada pelo administrador permite fazer a atualização do software baixando as atualizações do *site* da ferramenta, verificando a integridade dos arquivos baixados, descompactando num diretório de temporário para então colocar a nova versão em produção.

3.3.6.9 Trilha de verificação

Todos os acessos ao sistema são registrados, bem como todos os procedimentos executados durante o acesso. É gerado um *log* específico da aplicação de auditoria, que fica disponível para o administrador do sistema.

No Quadro 13 pode-se verificar a saída dos *logs* da aplicação sendo acessada pelo auditor.

```
...
2007-10-01 10:49:43 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/visualizaResultado
2007-10-01 10:52:55 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/visualizaResultado
2007-10-01 10:52:57 Error: Liberou o acesso para o usuário auditor acessar
a url servers/
2007-10-01 10:53:00 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/
2007-10-01 10:53:02 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/addAuditoria
2007-10-01 10:53:17 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/addAuditoria
2007-10-01 10:53:17 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/addAuditoria
2007-10-01 10:53:17 Error: Liberou o acesso para o usuário auditor acessar
a url auditorias/addAuditoria
...
```

Quadro 15 – Trilha de auditoria

3.3.7 Operacionalidade da implementação

Neste tópico é apresentado um estudo de caso demonstrando a operacionalidade da ferramenta para os usuários com perfil de auditor, conforme o diagrama de casos de uso da Figura 1.

A utilização da ferramenta depende dos procedimentos do administrador do sistema, que vai instalar e disponibilizar um acesso para o auditor através de um usuário e senha. Neste momento estaremos partindo da premissa que esses procedimentos já foram executados, e o acesso pelo auditor em questão é um acesso autorizado e autenticado. Informações com relação a instalação da ferramenta constam no Apêndice A.

O auditor vai acessar o servidor em que foi instalada a ferramenta, através do endereço disponibilizado pelo administrador. A primeira página a ser exibida é a página inicial com uma mensagem de boas vindas e informações sobre o sistema. Nesta página é exibida a documentação de como utilizar o sistema, contendo uma visão geral dos principais passos a serem seguidos, como demonstrado na Figura 10.

Check27002 : Welcome

Seje Bem Vindo a página inicial do Software Check 27002

O que é o projeto?

Um Software Livre com o objetivo de verificar a adequação de servidores GNU/Linux com a Norma de segurança ISO/IEC 27002.

Como utilizar o sistema?

1 - Para utilizar o sistema, primeiramente faça o Login
É necessário estar autorizado e autenticado para acessar o sistema neste servidor, qualquer dúvida entre em contato com o administrador

2 - O segundo passo é cadastrar o servidor através do reconhecimento automático que pode ser


- Cadastra o Servidor Local ou
- Cadastra um servido Remoto (em desenvolvimento).

Atenção, este procedimento demora alguns minutos interromper a sua execução pode causar o reconhecimento incompleto do servidor e assim o falso positivo ou falso negativo das verificações.

3 - Executar a auditoria

Figura 10 – Página inicial do software

Na página principal, verificando a documentação disponibilizada, o auditor vai ter em mente os procedimentos que deve executar para realizar a auditoria no servidor, e em que ordem precisam ser executados. O próximo passo é acessar a página de *login*, e autenticar no sistema com o usuário e senha repassados pelo administrador conforme a Figura 11.



Check27002 : Users - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://check27002/users/login

Check27002 : Welcome

Please log in.

Nome de Usuário:

Senha:

AOTORU

[Reload image](#)

Captcha:

login

Concluído

Figura 11 – Página de *login*

Tendo executado a autenticação no sistema com sucesso, o sistema vai identificar que o usuário em questão tem privilégios do perfil auditor, exibindo no menu somente os *links* que ele tem permissão de acesso, e ainda, para cada *link* acessado, o sistema confirma se o usuário tem permissão. Evitando que o usuário digitasse algum *link* que simplesmente não estivesse disponibilizado no seu menu, mas que quando acessado não verificasse permissão de acesso. Neste momento é exibida a página com informações do usuário autenticado, último acesso e qual o perfil de permissão ele possui, como demonstrado na Figura 12.



Figura 12 – Página autenticado

Após isso o auditor deve cadastrar o servidor a fim de fazer a auditoria, acessando o menu *Servidores*. Existem duas formas para fazer o cadastramento do servidor, a primeira manual, onde são inseridas todas as informações do servidor manualmente por quem está fazendo o cadastro, mas é usada apenas nas exceções em que o sistema não reconhece automaticamente o servidor, é acessada pelo *link* *Novo - Cadastrar Manualmente*. A outra forma, que é o procedimento padrão do sistema, em que o sistema verifica e salva as informações automaticamente é executada pelo *link* *Novo Servidor Local - Automático*, conforme exibido na Figura 13.



Figura 13 – Página cadastro de servidores

A opção *Novo Servidor Local - Automático*, faz com que seja executada a função para reconhecer e cadastrar o servidor em que a aplicação está rodando. São identificadas as seguintes distribuições de GNU/Linux: Debian, Fedora, Red Hat e Ubuntu.

O próximo passo é o procedimento mais importante do sistema, executar a auditoria. É acessado pelo menu Auditorias, conforme visualizado na Figura 14, em seguida Nova Auditoria.



Figura 14 – Página de auditorias

Neste momento o auditor vai verificar a segurança do servidor cadastrado com base nas recomendações dos controles da norma NBR ISO/IEC 27002. O sistema busca a relação de servidores e dos controles cadastrados, permite que o auditor selecione o servidor alvo da auditoria, informe uma descrição e selecione quais os controles vai utilizar para realizar a auditoria. A Figura 15 exibe esta página do sistema.

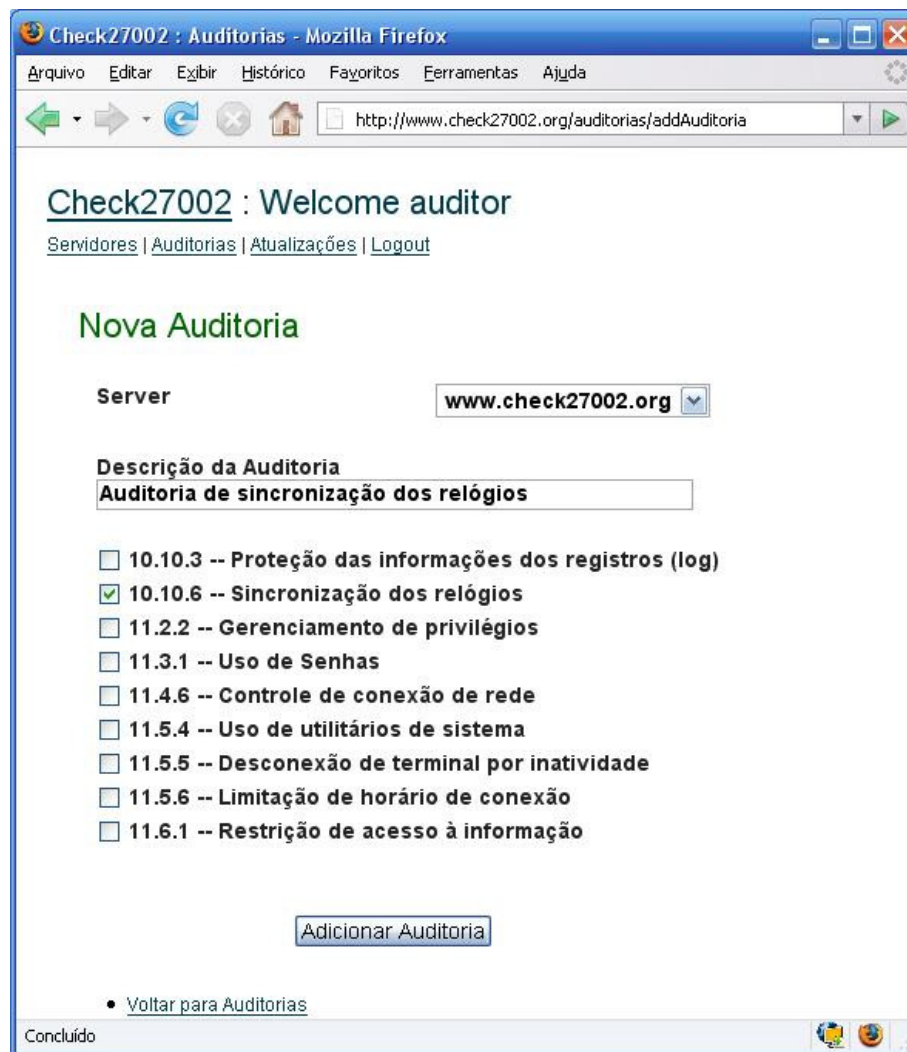


Figura 15 – Página nova auditoria

Com as informações corretamente selecionadas, o auditor clica em Adicionar Auditoria, para executar e salvar a auditoria no sistema.

Após a execução da auditoria, clicando em `view`, ao lado da auditoria, o auditor pode exibir o resultado da auditoria e das verificações relacionadas, como mostrado na Figura 16.

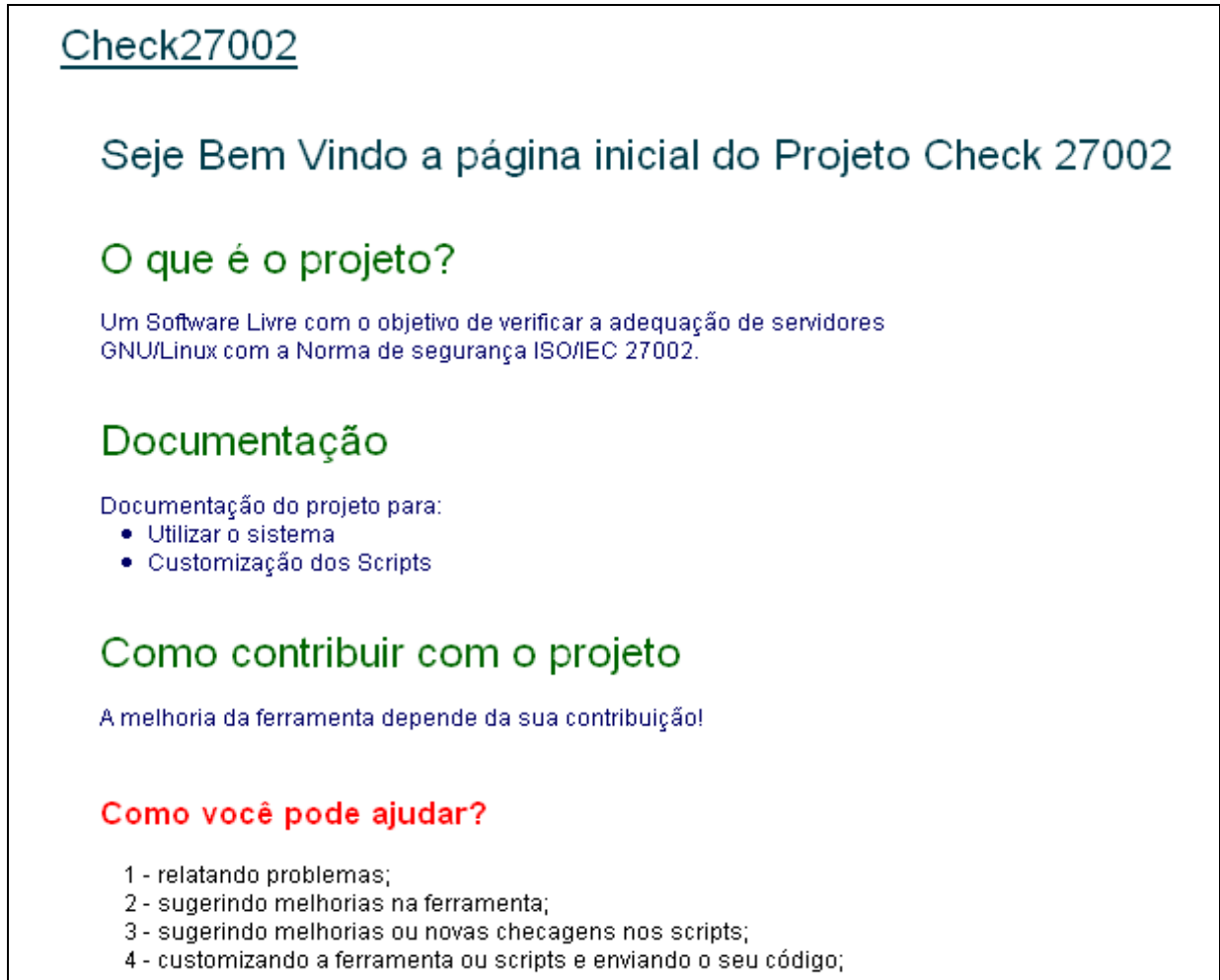
Visualiza Auditoria					
Id	33				
Usuário	auditor				
Servidor	turbo				
Desc. Auditoria	Auditoria completa				
Data	2007-10-30 14:34:13				
Verificações Relacionadas					
id	auditoria_id	Controle	Servidor	Adequação	Saída detalhada
153	33	10.10.3 - Proteção das informações dos registros (log)	turbo	0 %	Iniciando o Scrip 10.10.3 Pro ... Mostrar detalhes Ocultar
154	33	10.10.6 - Sincronização dos relógios	turbo	50 %	Iniciando o Scrip 10.10.6 Sin ... Mostrar detalhes Ocultar

Figura 16 – Página visualizando o resultado da auditoria

3.4 DESENVOLVIMENTO DO *WEBSITE* PARA O SOFTWARE LIVRE

Para a divulgação da ferramenta, foi importante construir um *website* onde o projeto de software livre ficasse hospedado. Para este desenvolvimento foi utilizado o CakePHP, com o mesmo *layout* utilizado na ferramenta, permitindo assim que o *site* ficasse com a mesma aparência da ferramenta. Neste *website* são disponibilizadas informações sobre o projeto, a documentação da ferramenta, documentação para personalização dos *scripts*, forma de contato e informações para quem deseja contribuir com o projeto.

Para facilitar o acesso à ferramenta foi registrado o domínio check27002.org através do *site* www.godaddy.com e hospedado em www.rapidvps.com. A Figura 17 exibe a página inicial do *site* do projeto.



The screenshot shows the homepage of the Check27002 project. At the top, the title 'Check27002' is underlined in blue. Below it, a large heading reads 'Seje Bem Vindo a página inicial do Projeto Check 27002'. A green heading asks 'O que é o projeto?'. The text below explains it is a free software for checking server compliance with the ISO/IEC 27002 standard. Another green heading is 'Documentação', followed by a list of documentation topics: 'Utilizar o sistema' and 'Customização dos Scripts'. A third green heading is 'Como contribuir com o projeto', with a note that improvement depends on user contribution. A red heading asks 'Como você pode ajudar?', followed by a numbered list of four ways to help: reporting problems, suggesting improvements, adding new checks, and customizing the tool or scripts.

Figura 17 – Página inicial do *site* do projeto

Com a exibição do *website* para o projeto de software livre, finaliza-se a parte de implementação, iniciando os resultados alcançados e discussão.

3.5 RESULTADOS E DISCUSSÃO

O módulo de reconhecimento automático foi o que mais tempo levou para a sua construção, não pela complexidade, mas porque foi o primeiro a ser desenvolvido, e com isso houve o aprendizado da ferramenta CakePHP. Procurou-se utilizar principalmente comandos do *shell* BASH para obter os dados dos servidores, desta forma testes iniciais nas diferentes distribuições, durante o desenvolvimento, já mostravam resultados equivalentes.

Inicialmente havia-se pensado em restringir o desenvolvimento da ferramenta à distribuição GNU/Linux Fedora, testando e garantindo o seu funcionamento nas versões 4, 5, 6 e 7. Optou-se pelo desenvolvimento e testes na última versão disponível, o Fedora Core 7, e por suportar também outras distribuições de GNU/Linux como Debian, Red Hat e Ubuntu. O que necessitou de mais tempo para os testes e adequação.

O comparativo da execução da ferramenta e dos *scripts* em distribuições GNU/Linux diferentes é mostrado no Quadro 16. Não surpreendentemente os resultados são os mesmos, se comportando da mesma forma para as distribuições suportadas.

	Debian	Fedora	RedHat	Slackware	Ubuntu
Scripts					
10.10.3	ok	ok	ok	não testado	ok
10.10.6	ok	ok	ok	não testado	ok
11.2.2	ok	ok	ok	não testado	ok
11.3.1	ok	ok	ok	não testado	ok
11.4.6	ok	ok	ok	não testado	ok
11.5.4	ok	ok	ok	não testado	ok
11.5.5	ok	ok	ok	não testado	ok
11.5.6	ok	ok	ok	não testado	ok
11.6.1	ok	ok	ok	não testado	ok
Ferramenta					
Instalação	ok	ok	ok	não testado	ok
Auditoria remota	ok	ok	ok	não testado	ok
Reconhecimento do servidor local	ok	ok	ok	não testado	ok
Reconhecimento do sistema operacional	ok	ok	ok	não implementado	ok

Quadro 16 – Comparativo da execução em distribuições GNU/Linux

A adequação dos controles ou *scripts* da norma inicialmente havia sido imaginada contendo um resultado como atende, ou não atende. No decorrer do trabalho, verificou-se a possibilidade de mais de uma checagem programada em um *script*. Assim se houvessem 2 checagens, uma atendesse e a outra não, seria incorreto relatar uma adequação ou inadequação completa. Com isso surgiu a necessidade de criar uma função para centralizar o cálculo da adequação dos *scripts*, considerando que o número de parâmetros que ela recebesse seria variável, pois nos *scripts* ocorriam números de checagens diferentes. Para isso foi implementada a função `calculaAdequação()` recebendo um array de zeros ou uns como demonstrado no tópico 3.3.4, que se mostrou suficiente.

A ferramenta foi projetada e implementada pensando na sua segurança. Não somente na verificação da segurança dos servidores, mas também na segurança da própria ferramenta. O desenvolvimento com estes requisitos se mostrou mais complexo do que simplesmente fazê-la funcionar. Durante a fase de implementação dos *scripts*, os testes e as alterações foram

mais trabalhosos, pois o software já implementava a comparação do *hash* dos arquivos. Desta forma depois de uma alteração no *script*, é necessário gerar o novo *hash*, salvá-lo no banco de dados para então executar o *script* novamente através de uma nova auditoria.

No ambiente GNU/Linux não existe apenas uma forma de realizar uma tarefa, então verificar um objetivo de segurança se torna mais difícil quando não se tem todos os caminhos que podem ter sido usados para fazê-lo. Como por exemplo, o agendamento de um controle no *crontab*, onde pode-se ter a string '*ntpdate*', representando o comando a ser localizado e isso significar para a ferramenta o atendimento do controle, como implementado no *script* 10.10.6. Ou ainda este comando poderia estar programado dentro de um outro *script* que agruparia outras verificações, neste caso resultando um falso negativo na verificação deste controle, como exibido no Quadro 3.

Para aperfeiçoar os *scripts* tentando diminuir este problema, não foi encontrado uma única solução, tendo sido utilizada a seguinte abordagem:

- a) se o usuário teve problemas com a execução da ferramenta ou de algum controle específico ele reporta como problema. Para isso percebeu-se a necessidade de um *site* ou e-mail para criar esta forma de comunicação;
- b) o usuário gostaria de sugerir alguma melhoria na ferramenta ou nos *scripts*. Um *site* também pareceu a forma mais apropriada para possibilitar que cada administrador relate como implementa o determinado controle da norma, e permita que esta forma seja também implementada nos *scripts*;
- c) permitir e facilitar a personalização da ferramenta, criação de novos ou alteração dos *scripts*.

Como discutido anteriormente se a dificuldade para personalização ocorreu para o próprio construtor da ferramenta poderia desencorajar outros colaboradores a fazerem as suas alterações no código fonte e assim contribuírem com a melhoria da ferramenta.

Incentivando a personalização da ferramenta, foi construído um manual da construção dos *scripts* que verificam os controles. Neste manual consta passo-a-passo os procedimentos necessários para o desenvolvimento dos *scripts*, a sua inclusão no sistema e a atualização ou inserção do *hash* de integridade. O Quadro 2 demonstra a estrutura básica dos *scripts* e o Apêndice C traz a documentação para a personalização, com o objetivo de facilitar o melhoramento dos atuais ou a criação de novos *scripts*.

Ao realizar os testes e o estudo de caso, verificou-se que a ferramenta é de fácil operabilidade para o usuário, necessitando de poucas interações para executar os procedimentos da auditoria. O que simplifica o trabalho do auditor e também diminui a sua

interferência sobre o resultado da ferramenta.

Nos testes realizados com a ferramenta verificou-se que o primeiro momento de avaliação das características e testes da ferramenta poderia ser facilitado. Primeiramente, como a ferramenta executa a maior parte das suas verificações baseadas em comandos do BASH, identificou-se a possibilidade de executar auditorias remotas utilizando para isso uma conexão segura através do SSH, o que foi testado com sucesso na auditoria remota. A hospedagem de um servidor específico para o projeto Check27002, permitiu que a ferramenta ficasse além de disponível para download através do seu código fonte, também disponível para demonstração através de um usuário que pode visualizar os procedimentos de uma auditoria e resultados existentes.

A configuração deste servidor permitiu também que num segundo momento possam ser executadas auditorias de segurança reais com a ferramenta Check27002, através do site oficial do projeto – www.check27002.org. Para isso seria necessário o cadastro de usuários, com o perfil de auditores, estes podem então cadastrar os seus servidores GNU/Linux na internet para o servidor do projeto execute estas auditorias remotamente nos seus servidores. O requisito para isso é apenas informar no site o servidor, usuário e senha para acesso com privilégios para acesso via SSH. Trazendo facilidade para o auditor sem ter que instalar ou configurar nenhuma ferramenta adicional possa realizar as auditorias da ferramenta segundo a norma NBR ISO/IEC 27002.

4 CONCLUSÕES

O estudo mais detalhado da norma de segurança NBR ISO/IEC 27002 e dos documentos de segurança para o GNU/Linux, permitiram uma melhor compreensão das recomendações da norma através dos seus controles. Este estudo foi importante por permitir o mapeamento dos controles que seriam implementados através dos *scripts* que verificam o servidor GNU/Linux.

O objetivo deste trabalho de construir um software livre para auditar e analisar a segurança de servidores GNU/Linux foi alcançado. Foi desenvolvido o sistema para efetuar a auditoria, bem como os *scripts* que implementam a verificação dos controles, tendo sido efetuadas algumas auditorias de teste que comprovaram o funcionamento correto da ferramenta.

Com o desenvolvimento deste trabalho, foi possível conhecer melhor a norma e com isso desenvolver a verificação dos seus controles para o sistema operacional GNU/Linux. O desenvolvimento destes controles não foi algo inédito, mas resultado de uma compilação de conhecimentos encontrados em livros, em especial BS7799 da tática à prática em servidores Linux (MELO ET AL., 2006), fóruns de discussão, *sites* como Linux Security.com (GUARDIAN DIGITAL, 2007), e outros mais, encontrados através de buscas na internet.

O *framework* CakePHP permitiu maior produtividade no desenvolvimento da aplicação como um todo. A ferramenta Bake, que é o gerador de código do CakePHP, apresentou problemas na geração de algumas *Controllers*, mas mesmo assim o resultado foi positivo, tendo se mostrada acertada a sua escolha.

Os *scripts* desenvolvidos podem não reconhecer todas as possibilidades que foram usadas para implementar as recomendações dos controles da norma. Pode ainda, ocorrer que o controle tenha sido atendido de outra forma. Neste sentido o detalhamento das suas verificações, disponível na visualização da auditoria, é de bastante utilidade. Mesmo assim, existe a possibilidade da ferramenta emitir um falso positivo ou falso negativo na execução dos *scripts*. Por isso, para possibilitar o melhoramento da ferramenta, foi criado um formulário eletrônico para o recebimento de problemas ou sugestões pelo *site*. Além de facilitar a personalização da ferramenta e dos *scripts* com a documentação construída.

Os requisitos de segurança foram prioridades quando levados em conta os problemas que podem ocorrer com um relatório de segurança fraudado. Para compensar os controles adicionais de segurança implementados, em contrapartida com a facilidade de personalização

da ferramenta, a construção do manual de personalização dos *scripts* disponível no Apêndice C mostrou-se suficiente.

O objetivo final foi adicionar e não dividir esforços com outros projetos de software livre existentes que também objetivam a segurança, mas com focos diferentes. Por exemplo, pode-se integrar ao trabalho desenvolvido o Nessus como analisador de vulnerabilidades, o Nmap como *portscan*, ou ainda o Snort como um *Network Intrusion Detect System* (NIDS), que são soluções consagradas e a implementação não pretende refazer. Também não é objetivo neste momento abordar como seria esta integração com estas ferramentas, mas deixa sugestões para futuras implementações.

Desenvolver software para a área de segurança não é uma tarefa fácil, principalmente considerando os problemas de segurança encontrados e explorados nas aplicações atuais. Deste modo, um software que analisa a segurança, deve ser projetado e desenvolvido já com os requisitos de segurança bem definidos, e freqüentemente repensado para que novas atualizações e melhorias possam ser implementadas.

A possibilidade de execução de auditorias reais através do site do projeto, mostra-se como uma possibilidade de divulgar e alavancar o projeto, já que permite fácil e rapidamente um usuário executar uma auditoria de segurança em um servidor GNU/Linux com base na norma NBR ISO/IEC 27002.

Espera-se que este trabalho possa ser útil como ferramenta adicional de segurança e na adequação à norma ISO/IEC 27002, mesmo que muitas melhorias ainda possam ser feitas. Este é o começo, o primeiro passo dado nesse projeto que pode avançar e melhorar com o conhecimento e contribuição de todos, que poderão estar trabalhando em conjunto numa comunidade de software livre.

4.1 EXTENSÕES

Nesta seção do trabalho são sugeridos trabalhos futuros, incluindo a possibilidade de extensão da ferramenta ou sua melhoria.

A qualquer momento é oportuna a revisão da implementação de segurança da ferramenta. Cada vez mais rápido são lançadas técnicas aprimoradas dos mecanismos de segurança, e assim, melhorando ou readequando a sua implementação.

A integração com outras ferramentas de segurança, interpretando o resultado delas

conforme a norma NBR ISO/IEC 27002 seria interessante. Agregar o conhecimento específico de outras ferramentas conhecidas como software livre, como por exemplo Nmap, Nessus e Snort, geraria um resultado bastante interessante agregando o *expertise* dessas ferramentas às recomendações da norma.

Ainda com relação à segurança da ferramenta, pode-se implementar o mecanismo de segurança de assinatura digital para conferir autenticidade e não repúdio aos fontes da ferramenta, à sua instalação no servidor pelo administrador do sistema, à execução e autoria das auditorias, e por fim, ao gerente quando visualizando o resultado destas auditorias.

Seria também interessante verificar a relação desta ferramenta com as outras normas da série 27000, possibilitando a evolução este trabalho de forma abranger as demais normas.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 27002**: tecnologia da informação. Rio de Janeiro, 2005. 120 p.

C & A SYSTEMS SECURITY. **COBRA**: security risk assessment, security risk analysis and ISO 17799 / BS7799. [S.l.], [2007]. Disponível em: <<http://www.riskworld.net/index.htm>>. Acesso em: 15 abr. 2007.

CAMPOS, André L. N. **Sistema de segurança da informação**: controlando os riscos. Florianópolis: Visual books. 2006. 180 p.

CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES DE SEGURANÇA NO BRASIL. **Estatísticas dos incidentes reportados ao CERT.br**. [S.l.], out. 2007. Disponível em: <<http://www.cert.br/stats/incidentes>>. Acesso em: 15 out. 2007.

DIAS, Cláudia. **Segurança e auditoria da tecnologia da informação**. Rio de Janeiro: Axcel, 2000. 218 p.

ELIOT, Ed. **Visual and audio PHP CAPTCHA generation class**. [S.l.], [2007]. Disponível em: <<http://www.ejliot.com/pages/2>>. Acesso em: 03 ago. 2007.

FENZI, Kevin; WRESKI, Dave. **Linux security howto**. [S.l.], [2007]. Disponível em: <<http://tldp.org/HOWTO/Security-HOWTO/>>. Acesso em: 16 abr. 2007.

FREE SOFTWARE FOUNDATION. **Bash**. Boston, 2007. Disponível em: <<http://www.gnu.org/software/bash>>. Acesso em: 19 abr. 2007.

GNU. **O que é o software livre?**: o projeto GNU e a fundação para o software livre (FSF). [S.l.], [2007]. Disponível em: <<http://www.gnu.org/philosophy/free-sw.pt.html>>. Acesso em: 17 abr. 2007.

GONÇALVES, L. R. de O. **Um modelo para verificação, homologação e certificação de aderência à norma nacional de segurança de informação – NBR-ISO/IEC-17799**. 2005. 133 f. Dissertação (Engenharia de Sistema e Computação) – COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro. Disponível em: <<http://www.ravel.ufrj.br/arquivosPublicacoes/tese-luisrodrigo.pdf>>. Acesso em: 20 mar. 2007.

IBM. **Cook up web sites fast with CakePHP, part 2**: Bake bigger and better with CakePHP. [S.l.], [2007]. Disponível em: <<https://www6.software.ibm.com/developerworks/education/os-php-cake2/os-php-cake2-a4.pdf>>. Acesso em: 03 mar. 2007.

KOSCHUETZKI, Tim. **How to put combined fields into CakePHP's model->generateList()**. [S.l.], ago. 2007. Disponível em: <<http://php-coding-practices.com/cakephp-specific/how-to-put-combined-fields-into-cakephps-model-generatelist>>. Acesso em: 10 out. 2007.

MELO, Sandro et al. **BS7799 da tática à prática em servidores linux**. Rio de Janeiro: Alta Books, 2006. 232 p.

MÓDULO. **Módulo Risk Manager**. [S.l.], [2007]. Disponível em: <<http://www.checkuptool.com.br/index.htm>>. Acesso em: 30 abr. 2007.

MYSQL AB. **MySQL 5.0 reference manual: 11.10.2 encryption and compression function**. [S.l.], [2007]. Disponível em: <<http://dev.mysql.com/doc/refman/5.0/en/encryption-functions.html>>. Acesso em: 05 set. 2007.

PALLETT, Dennis. **Framework comparison chart**. [S.l.], [2006]. Disponível em: <<http://www.phpit.net/demo/framework%20comparison/chart.php>>. Acesso em: 05 jun. 2007.

PEIXOTO, Mário César Pintaudi. **Engenharia social e segurança da informação na gestão corporativa**. Rio de Janeiro: Brasport, 2006. 132 p.

RED HAT. **Red Hat enterprise linux 4: security guide**. [S.l.], 2007. Disponível em: <<http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/security-guide>>. Acesso em: 15 abr. 2007.

ROSEMANN, D. **Software para avaliação da segurança da informação de uma empresa conforme a norma NBR ISO-IEC 17799**. 2002. 93 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.bc.furb.br/docs/MO/2002/266314_1_1.pdf>. Acesso em: 8 abr. 2007.

SÊMOLA, Marcos. **Gestão da segurança da informação: uma visão executiva**. Rio de Janeiro: Campus, 2003. 184 p.

SOARES, Luiz Fernando G.; LEMOS, Guido; COLCHER, Sérgio. **Redes de computadores: das LANs, MANs e WANs às redes ATM**. 2. ed. Rio de Janeiro: Campus, 1995. 705 p.

SOLUTION TI. **Auditoria de segurança**. [S.l.], [2007]. Disponível em: <<http://www.tisolution.com.br/auditoria.html>>. Acesso em: 12 nov. 2007.

THE BAKERY EVERYTHING CAKEPHP. **Captcha component with PhpCaptcha**. [S.l.], [2007]. Disponível em: <<http://bakery.cakephp.org/articles/view/captcha-component-with-phpcaptcha>>. Acesso em: 30 maio 2007.

ZOOP GROUP. **The zoop framework for php: zoop php framework**. [S.l.], [2007]. Disponível em: <<http://zoopframework.com>>. Acesso em: 11 abr. 2007.

APÊNDICE A – Manual de instalação do software Check 27002

Neste tópico consta o manual para a instalação da ferramenta na distribuição GNU/Linux Ubuntu versão 7.10. O procedimento inclui desde o *download* do arquivo até o cadastramento do acesso para o auditor, como pode ser visto no Quadro 17.

Instalação do Software

Requisitos para a instalação

Os seguintes programas e bibliotecas precisam estar instalados antes para então instalar o software Check27002:

- servidor web Apache com os módulos habilitados: sessions e mod_rewrite;
- PHP 4.3.2 ou superior (recomendamos a última versão disponível) com suporte aos seguintes módulos: pear, gd, mysql;
- banco de dados mysql;
- utilitários do sistema: zip, unzip e sha512sum

Comando na distro Ubuntu para instalar estes pacotes:

```
# apt-get update
# apt-get install apache2 mysql-server mysql-client-5.0 php5 php5-gd \
php5-mysql php5-pear php5-curl
```

Adicionar um usuario do sistema:

```
# adduser check27002
# passwd check27002
```

Baixar o arquivo de instalação de:

```
# http://jboss-dsv.furb.br/check27002_20071017.zip
```

Descompactar no diretório /var/www/check27002

```
# cd /var/www
# unzip check27002_(versao).zip
```

No MySQL:

```
criar banco e importar os dados:
mysql> create database check27002;
```

Dar permissão para o usuário acessar:

```
mysql> grant all on check27002.* to user27002@localhost identified by
'suasenhaforte';
mysql> flush privileges;
```

Importar os dados do arquivo:

```
$ cd /var/www/check27002
$ mysql -u user27002 -p check27002 < banco_check27002.sql
```

Editar o arquivo de conexão do CakePHP com o banco de dados, incluindo as informações que você cadastrou:

```
$ vim /var/www/check27002/app/config/database.php
```

Criar as tabelas de seção:

```
$ cd /var/www/check27002/cake/scripts
```

Configurar o diretório da aplicação no apache:

Adicionar a seguinte configuração no *site default*:

```
# vim /etc/apache2/sites-available/default
```

```
<Directory /var/www/check27002/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
    # This directive allows us to have apache2's default start page
    # in /apache2-default/, but still have / go to the right place
    RewriteBase /check27002/
</Directory>
```

Criar um link simbólico no *mods-enabled*:

```
# cd /etc/apache2/mods-enabled
# ln -s ../mods-available/rewrite.load rewrite.load
```

Restartar o Apache:

```
# /etc/init.d/apache2 restart
```

E testar o acesso:

```
http://ip_ou_nome_do_servidor/check27002
```

Neste momento a aplicação deve estar exibindo a página inicial.

Agora, é necessário configurar o acesso para o auditor. Isso pode ser realizado através da seguinte forma:

Acessar o mysql:

```
$mysql -u check27002 -p check27002
```

Cadastrar um novo usuário com o perfil de auditor:

```
mysql> insert into users \
(username,senha,email,first_name,last_name,perfil) \
values 'auditor_fernando', \
aes_encrypt('senhafortedoauditor','chaveAES'), \
'fernando@check27002.org', 'Fernando', 'Cugik', 'auditor');
```

Atenção, o campo 'chaveAES' é utilizado como chave para criptografar as senhas dos usuários com o algoritmo AES. Esta mesma string será utilizada na função `checkPassAes()` para criptografar a senha digitada e comparar com a armazenada. Para funcionar, esta chave deve ser salva no arquivo `/var/www/check27002/chaveAES.txt`

Lembrando também que é importante que esta senha seja passada de forma segura para o auditor.

Pronto, a aplicação está funcionando, e o acesso configurado.

APÊNDICE B – Detalhamento dos casos de uso do auditor

Neste tópico são detalhados os diagramas de caso de uso do Auditor, no Quadro 18 é exibida a descrição do caso de uso Efetua Login, no Quadro 19 o detalhamento do caso de uso Cadastra Servidor, no Quadro 20 Visualiza Resultado da Auditoria e no Quadro 21 Realiza Auditoria.

Efetua Login

Cenários

Login {Caminho Básico}.

1. O Auditor acessa a página principal e clica no *link* Login;
2. O servidor exibe a página de autenticação;
3. O Auditor informa usuário, senha, captcha e clica em Efetuar login;
4. O sistema valida a autenticação e exibe a página do usuário.

Login Incorreto {Alternativo}.

1. O Auditor acessa a página principal e clica no *link* Login;
2. O servidor exibe a página de autenticação;
3. O Auditor informa usuário ou senha incorretos, e clica em Efetuar *login*;
4. O sistema verifica os dados, como não ok exibe mensagem de aviso de aviso e volta para a página de autenticação.

Não reconhece captcha { Alternativo }.

1. O Auditor acessa qualquer página restrita do sistema;
2. O Sistema identifica que ele ainda não está autenticado, e exibe a página de autenticação;
3. O Auditor não reconhece a captcha, e clica no *link reload image*;
4. O Sistema gera e exibe um novo captcha;
3. O Auditor reconhece o captcha e informa corretamente usuário, senha e captcha;
4. O sistema valida a autenticação e exige a página do usuário.

Quadro 18 – Detalhamento do caso de uso Efetua Login

Cadastra Servidor

Cenários

Cadastro de servidor Local { Caminho Básico }.

1. O Auditor acessa o *link* Servidores;
2. O Sistema verifica se ele tem permissão de acesso e exibe a página de Servidores;
3. O Auditor opta por cadastrar um novo servidor local, e clica no *link* apropriado;
4. O sistema faz o reconhecimento do servidor, salvando as configurações da máquina, o sistema operacional, as suas interfaces de rede e os processos ativos no servidor, após isso exibe uma mensagem que o reconhecimento e o cadastro ocorreram com sucesso, e redireciona para a página inicial.

Cadastro de servidor remoto { Alternativo }.

1. O Auditor acessa o *link* Servidores;
2. O Sistema verifica se ele tem permissão de acesso e exibe a página de Servidores;
3. O Auditor opta por cadastrar um novo servidor remoto e clica no *link* apropriado;
4. O sistema exibe a página solicitando informações para acesso ao servidor remoto;
5. O Auditor informa os dados para o sistema acessar o servidor remoto e executar os comandos com o privilégio necessário;
6. O Sistema tenta se conectar no servidor com as informações recebidas, se conseguir conectar com sucesso, exibe mensagem e vai para o item 7, se não conseguir conectar exibe mensagem de aviso e retorna para o item 4.
7. O sistema faz o reconhecimento do servidor, salvando as configurações da máquina, o sistema operacional, as suas interfaces de rede e os processos ativos no servidor, após isso exibe uma mensagem que o reconhecimento e o cadastro ocorreram com sucesso, e redireciona para a página inicial.

Quadro 19 – Detalhamento do caso de uso Cadastra Servidor

Visualiza resultado da auditoria

Cenários

Visualiza resultado { Caminho Básico }.

1. O Auditor acessa o *link* Auditorias;
2. O sistema verifica se o Auditor tem privilégios para acessar e exibe a página de auditorias, relacionando todas as auditorias já executadas;
3. O Auditor escolhe qual auditoria deseja visualizar e clica no botão *View* ao lado direito dela;
4. O sistema verifica se o Auditor tem privilégios para acessar e exibe a auditoria específica, contendo todas as verificações executadas, e os seus resultados.

Quadro 20 – Detalhamento do caso de uso Visualiza resultado da auditoria

Realiza a auditoria

Cenários

Realiza a auditoria { Caminho Básico }.

1. O Auditor acessa os *links* Auditoria e Nova Auditoria;
2. O Sistema *exibe* a tela para a nova auditoria, permitindo selecionar um dos servidores cadastrados e disponibilizando os controles cadastrados como opção para verificação;
3. O Auditor informa a descrição da auditoria, seleciona o servidor alvo e os controles que deseja incluir para verificação nesta auditoria, clica em adicionar Auditoria.
4. O Sistema obtém a lista dos controles selecionados, executa a verificação de cada um destes controles no servidor alvo da auditoria, salvando as mensagens geradas pela verificação e registrando o resultado da verificação de cada controle. Ao final *exibe* mensagem que a auditoria ocorreu com sucesso e está disponível para consulta.

Servidor não cadastrado { Alternativo }.

1. O Auditor acessa os *links* Auditoria e Nova Auditoria;
2. O Sistema pesquisa os servidores cadastrados, se não houver nenhum *exibe* mensagem de aviso e redireciona para página de cadastro de servidores.

Verifica *hash* dos *scripts* { Alternativo }.

1. O Auditor acessa os *links* Auditoria e Nova Auditoria;
2. O Sistema *exibe* a tela para a nova auditoria, permitindo selecionar um dos servidores cadastrados e disponibilizando os controles cadastrados como opção para verificação;
3. O Auditor informa a descrição da auditoria, seleciona o servidor alvo e os controles que deseja verificar para fazer parte desta auditoria, clica em adicionar Auditoria.
4. Antes de executar a verificação do controle selecionado o sistema verifica se o respectivo script não está corrompido ou foi alterado através da comparação *hash* do arquivo, se houver diferença entre o *hash* salvo e o *hash* atual do arquivo, o sistema interrompe a execução do script atual, coloca 0% de adequação neste controle, relata o erro na saída detalhada da verificação e *exibe* mensagem sobre o problema encontrado.
4. O Sistema obtém a lista dos controles selecionados, executa a verificação de cada um destes controles no servidor alvo da auditoria, salvando as mensagens geradas pela verificação e registrando o resultado da verificação de cada controle. Ao final *exibe* mensagem que a auditoria ocorreu com sucesso e está disponível para análise.

APÊNDICE C – Manual para a personalização ou criação de novos *scripts*

Para facilitar o processo de personalização da ferramenta através ou da criação de novos *scripts*, ou de alteração dos existentes, foi criado este manual com os procedimentos necessários para efetuar esta tarefa, como consta no Quadro 22

Manual para a personalização ou criação de novos *scripts*

Para a Criação de novos *scripts*:

1 - Adicionar um novo controle:

Para adicionar um novo controle da norma o primeiro procedimento é cadastrar o controle no banco de dados, para isso é recomendado uma ferramenta como o phpMyAdmin, disponível em www.phpmyadmin.net .

- Acesse o banco de dados da ferramenta (por padrão é o check27002);
- Acesse a tabela controles;
- Insira um novo registro, informando:
 - o em `txt_nome_controle` o número do controle da norma;
 - o `txt_desc_controle` a descrição contida no controle da norma;
 - o `txt_obj_controle` o objetivo do controle.

2 - Copiar o *script* modelo para o número do controle que está sendo criado:

O *script* modelo contém a estrutura básica necessária para os *scripts*, para ser copiado e modificado.

Para copiar o *script*:

```
cd /var/www/check27002/scripts27002
cp -a /exemplo.php 11.2.2.php
```

3 - Implemente as verificações no *script* com a sua ferramenta de desenvolvimento preferida, ou simplesmente utilizando o vi.

4 - Quando tiver terminado o desenvolvimento do *script*, gere o hash do arquivo:

Para isso utilize o comando: `sha512sum nomedoarquivo`

5 - Salve o hash do arquivo na tabela de *scripts*:

Utilizando o phpmyadmin crie um novo registro, informando em:

- `controle_id` o código do controle cadastrado, deve ser consultado na tabela `controles`;
- `txt_script` contendo uma descrição do *script* cadastrado;
- `txt_file_script` o local no sistema de arquivos onde está localizado o arquivo do *script*, a partir do diretório de instalação da ferramenta, normalmente similar a `scripts27002/11.2.2.php`;
- `txt_hash_script` o hash code do arquivo, gerado pelo comando `sha512sum`.

Quadro 22 – Manual para a personalização ou criação de novos *scripts*