



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Auditoria em Segurança da Informação no Ambiente Uniriotec

Daniel Possolo Gomes Dezerto Castanha

**Orientador**

Sidney Cunha de Lucena

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2014

Auditoria em Segurança da Informação no Ambiente Uniriotec

Daniel Possolo Gomes Dezerto Castanha

Projeto de Graduação apresentado à Escola de  
Informática Aplicada da Universidade Federal do  
Estado do Rio de Janeiro (UNIRIO) para obtenção do  
título de Bacharel em Sistemas de Informação.

Aprovada por:

---

Prof. Sidney Cunha de Lucena (UNIRIO)

---

Prof. Morganna Carmem Diniz (UNIRIO)

---

Prof. José Carlos Albuquerque (CEFET –RJ)

RIO DE JANEIRO, RJ – BRASIL.

JUNHO DE 2014

## **Agradecimentos**

Essa monografia é o resultado de uma jornada árdua de quatro anos de graduação. Não seria possível a sua realização sem a colaboração das pessoas abaixo, por isso, gostaria de prestar os meus sinceros agradecimentos.

Em primeira instância, gostaria de agradecer a Deus por ter me encaminhado para a jornada certa de graduação.

Ao Decano, Luiz Amâncio de Sousa Junior, por ter confiado no trabalho e autorizado a realização das etapas do mesmo.

Ao meu professor orientador, Sidney Cunha de Lucena, pela dedicação e presteza durante todo o projeto.

Ao professor Maximiliano Martins Farias, por toda orientação e ideias durante o projeto.

Aos professores membros da banca examinadora, Morganna Carmem Diniz e José Carlos Albuquerque, por aceitarem o convite e disposição em avaliar e contribuir com o projeto.

A minha família e minha namorada Julia, que sempre me apoiaram em todas as minhas decisões e torceram pelo meu sucesso nesta caminhada.

Aos amigos, Enrique, Leonardo, Rafael, Thiago e Túlio pelos trabalhos em conjunto em disciplinas, momentos de confraternização vividos e pelo apoio que me prestaram durante a realização deste trabalho.

Por fim, gostaria de agradecer a todos que participaram deste trabalho de alguma forma e que me apoiaram durante esta caminhada.

## **RESUMO**

A segurança da informação é uma área que está em bastante evidência atualmente, visto os acontecimentos relacionados a espionagem e coleta de informações confidenciais, sendo essencial em qualquer ambiente tecnológico. Este trabalho tem como foco um nicho específico da segurança da informação, a área de auditoria que utiliza como métrica um teste de invasão. Tal nicho é necessário para que as organizações avaliem o nível de segurança que seus sistemas e serviços possuem, bem como os recursos humanos que ali realizam suas funções. Neste trabalho será apresentado o desenvolvimento de uma auditoria em segurança da informação através da execução de tarefas de um teste de invasão, aplicadas em alguns sistemas e aplicações que estão presentes no ambiente Uniriotec bem como nos recursos humanos do departamento. A análise dos resultados obtidos do teste de invasão juntamente com a análise da segurança do ambiente como um todo, se propõe a demonstrar o quão suscetível a ataques cibernéticos a Instituição se encontra e quais as recomendações para que os riscos encontrados sejam mitigados.

**Palavras-chave:** Segurança, Riscos, Teste de Invasão.

## **ABSTRACT**

Information security is an area that is currently in evidence, since the events related to spying and collecting confidential information, and is essential in any technological environment. This work focuses on a specific niche of information security, the area of audit that uses as a metric a penetration test. This niche is necessary for organizations to assess the security level of their own systems and services, as well as human resources to perform their duties there. This report will present the development of an information security audit by executing tasks of a penetration test, applied in some systems and applications that are present in Uniriotec environment as well as human resources department. The analysis of the results of the invasion assay along with analysis of the safety of the environment as a whole, aims to demonstrate how susceptible to cyber attacks the institution is and what recommendations found that risks are mitigated.

**Keywords:** Security, Risks, Penetration Test.

## Índice

1. Introdução.....	12
2. Fundamentos Teóricos .....	15
2.1 O Teste de Invasão .....	15
2.1.2 Tipos .....	16
2.2 Vulnerabilidades.....	17
2.2.1 Definição .....	17
2.2.2 Principais Vulnerabilidades de aplicações <i>Web</i> .....	17
2.2.2.1 <i>SQL Injection</i> .....	17
2.2.2.2 <i>Cross-Site-Scripting</i> .....	18
2.2.3 Classificação das Vulnerabilidades .....	20
2.2.3.1 Vulnerabilidades Críticas .....	20
2.2.3.2 Vulnerabilidades Altas .....	20
2.2.3.3 Vulnerabilidades Médias .....	21
2.2.3.4 Vulnerabilidades Baixas .....	21
2.3 Métricas para o Teste de Invasão .....	21
2.3.1 Varredura em Busca de Vulnerabilidades .....	21
2.3.2 Engenharia Social.....	22
2.3.2.1 Vírus disseminados por e-mail .....	22
2.3.2.2 <i>Phishing</i> .....	22
2.4 As Normas ISO/IEC 27001 e 27002 .....	23
3. Infraestrutura Técnica, Ferramentas e Arquitetura. ....	25
3.1 A Infraestrutura em Detalhes.....	25
3.2 Ferramentas .....	26
3.2.1 Varreduras de vulnerabilidades <i>Web</i> .....	26
3.2.2 Varreduras de vulnerabilidades em servidores.....	28

3.3 Arquitetura do Teste de Invasão .....	30
4. Análise dos Resultados e Recomendações .....	34
4.1 Vulnerabilidades <i>Web</i> .....	34
4.1.1 Descoberta de Informações .....	34
4.1.2 Página do Moodle Uniriodb .....	35
4.1.2.1 Vulnerabilidades Altas .....	36
4.1.2.2 Vulnerabilidades Médias .....	39
4.1.2.3 Vulnerabilidades Baixas .....	44
4.1.3 Página do Moodle Uniriodb2 .....	46
4.1.3.1 Vulnerabilidade Alta .....	47
4.1.3.2 Vulnerabilidades Médias .....	48
4.1.3.3 Vulnerabilidades Baixas .....	50
4.1.4 Página <i>Web</i> do BSI .....	50
4.1.4.1 Vulnerabilidade Alta .....	51
4.1.4.2 Vulnerabilidades Média e Baixa .....	53
4.2 Vulnerabilidades em Servidores .....	53
4.3 Engenharia Social .....	57
4.4 Recomendações .....	59
4.4.1 Vulnerabilidades Encontradas .....	59
4.4.2 Baseadas nos controles da ISO/IEC 27001 e 27002 .....	60
4.4.3 Gerais .....	63
5. Conclusão .....	64



## **Índice de Tabelas**

Tabela 1: Janela vulnerabilidades <i>Web</i> .....	30
Tabela 2: Organização varredura vulnerabilidades em servidores.....	31
Tabela 3: Vulnerabilidades dos servidores.....	57

## Índice de Figuras

Figura 1: <i>SQL Injection</i> .....	18
Figura 2: Teste do <i>Cross-Site-Scripting</i> .....	19
Figura 3: Constatação do <i>Cross-Site-Scripting</i> .....	20
Figura 4: Topologia.....	26
Figura 5: WhatWeb .....	27
Figura 6: Vega.....	28
Figura 7: Nmap.....	29
Figura 8: Nessus.....	30
Figura 9: Modelo do e-mail.....	32
Figura 10: Mensagem exibida ao usuário que acessou o link .....	33
Figura 11: Descoberta de Informações.....	35
Figura 12: Índice de vulnerabilidades encontradas em uniriadb.uniriotec.br .....	36
Figura 13: Índice de Vulnerabilidades Altas.....	37
Figura 14: Vulnerabilidade senha em conexão sem criptografia .....	38
Figura 15: Possibilidade de <i>SQL Injection</i> .....	38
Figura 16: Informações <i>SQL Injection</i> .....	39
Figura 17: Índice vulnerabilidades médias em uniriadb.uniriotec.br .....	40
Figura 18: Informações da Vulnerabilidade HTTP TRACE.....	41
Figura 19: Diretórios locais encontrados .....	42
Figura 20: Informações da Vulnerabilidade <i>Local File System Path</i> .....	42
Figura 21: Locais das assinaturas encontradas.....	43
Figura 22: Informações sobre a Vulnerabilidade <i>Source Code Disclosure</i> .....	44
Figura 23: Índice de Vulnerabilidades Baixas .....	45
Figura 24: Código HTML exibindo o index do diretório.....	45

Figura 25: Informações sobre a Vulnerabilidade <i>Form Password Field Autocomplete Enabled</i> .....	46
Figura 26: Índice de Vulnerabilidades Uniriodb2.....	47
Figura 27: Tela de <i>login</i> da aplicação Moodle Uniriodb2 .....	48
Figura 28: Captura do <i>Username</i> e Senha.....	48
Figura 29: Índice Vulnerabilidades Médias Uniriodb2.....	49
Figura 30: Vulnerabilidades baixas Uniriodb2 .....	50
Figura 31: Índice de vulnerabilidades da página do Bsi .....	51
Figura 32: Barra de exibição do Governo Federal .....	51
Figura 33: Focos de possíveis XSS .....	52
Figura 34: Informações sobre a vulnerabilidade XSS.....	53
Figura 35: Índice de vulnerabilidades média e baixa da página do Bsi .....	53
Figura 36: Nmap para a aplicação Moodle Uniriodb .....	54
Figura 37: Nmap para a aplicação Moodle Uniriodb2.....	55
Figura 38: Elevação de privilégio .....	56
Figura 39: Status do <i>Phishing</i> .....	58
Figura 40: E-mail de <i>Phishing</i> não lido .....	58
Figura 41: Exposição dos Equipamentos .....	62

## 1. Introdução

O acesso as tecnologias da informação aumenta cada dia que passa segundo os autores Giavaroto e Santos (GIAVAROTO; SANTOS, 2013). Esse fato, alinhado com as novas tendências tecnológicas, proporciona um mundo de facilidades digitais aos seus usuários, onde dados importantes circulam.

Paralelamente ao desenvolvimento dessas “facilidades” digitais, a área de segurança da informação também caminha em constante evolução com o objetivo de manter seguros e proteger os dados circulantes nesses contextos. As normas ISO/IEC 27001 e 27002 foram desenvolvidas justamente para direcionar uma organização em relação à área de segurança da informação e têm por objetivo implementar toda uma estrutura e controles de segurança da informação numa organização.

Com a vasta quantidade de informações importantes que circulam através dos meios digitais e estão concentradas em ambientes tecnológicos, existem aproveitadores que caminham silenciosamente com o objetivo de roubar esses dados para utilizá-los em benefício próprio. Esses são intitulados de *blackhat* (GIAVAROTO; SANTOS, 2013). As táticas de invasão, utilizadas por estes indivíduos, se tornam cada vez mais robustas e complicadas de serem contidas. Com

isso, os sistemas que disponibilizam, manipulam e armazenam informações devem estar devidamente protegidos contra as ameaças de segurança.

O comportamento humano também é outro fator que pode comprometer a segurança das informações de um determinado ambiente. Os mecanismos de proteção evoluem constantemente, mas em contrapartida, a conscientização das pessoas ao utilizar a tecnologia e os riscos envolvidos em seu uso ainda é um ponto que pode representar uma grande oportunidade para um indivíduo mal intencionado.

Uma das maneiras de se constatar o nível atual de segurança em algum ambiente tecnológico, é através da realização de auditorias internas, que são altamente recomendadas pela norma ISO/IEC 27001. Uma das diversas métricas para a realização de auditoria interna em segurança da informação é por meio de um teste de invasão, que consiste numa tentativa de intrusão controlada e devidamente documentada ao ambiente computacional da organização.

O autor Engebretson (ENGEBRETSON, 2011) define que a realização do teste possibilita descobertas de falhas de segurança e vulnerabilidades existentes nos sistemas de informação e nos recursos humanos do ambiente - que poderiam ser exploradas por indivíduos mal intencionados – e estas podem ser solucionadas antes que danos possam ser causados provenientes de algum tipo de ataque.

Diante deste panorama, este trabalho se propõe a testar e avaliar os sistemas e os recursos humanos disponíveis no ambiente Uniriotec, através da execução de algumas fases que constituem um Teste de Invasão e avaliação da utilização de controles pertencentes as normas ISO/IEC 27001 e 27002. Desta maneira, foi possível chegar aos resultados de quão protegido está o ambiente e quais as recomendações para torná-lo mais seguro.

Este trabalho está organizado em quatro etapas que acompanham o desenvolvimento do mesmo: a apresentação dos conceitos teóricos que baseiam o trabalho, a execução dos testes, a análise dos resultados obtidos e a realização de recomendações para a segurança do ambiente.

O Capítulo 2 apresenta os fundamentos teóricos com base para a realização do projeto. Inicialmente, serão detalhadas algumas definições de teste de invasão, como os principais tipos e quais serão utilizados neste trabalho. Também serão apresentados os principais ataques com exploração de vulnerabilidades nas aplicações *web*, as métricas utilizadas para a classificação das vulnerabilidades encontradas durante a realização das tarefas e a definição mais detalhada das normas ISO/IEC 27001 e 27002.

No Capítulo 3, a infraestrutura do teste é apresentada juntamente com as ferramentas utilizadas para a realização do projeto. O quarto capítulo expõe os resultados obtidos durante a execução das tarefas propostas e as recomendações gerais de segurança, com base nos controles estabelecidos pelas ISO/IEC 27001 e 27002 e nos resultados obtidos das etapas executadas do teste de invasão. No quinto capítulo são apresentadas as considerações finais sobre o trabalho, que envolvem os principais resultados obtidos e possibilidades para a realização de trabalhos futuros.

## 2.

# Fundamentos Teóricos

Neste capítulo, são definidos os conceitos específicos de um teste de invasão para melhor compreensão deste trabalho. Primeiramente, serão definidos o conceito de Teste de Invasão e seus tipos. Posteriormente, serão apresentados os tipos de escaneamento de vulnerabilidades, os principais ataques relacionados a aplicações *web*, a métrica utilizada para a avaliação das vulnerabilidades encontradas nos sistemas de informação e as normas utilizadas para gerar algumas recomendações de segurança.

### 2.1 O Teste de Invasão

Nesta seção serão definidos conceitos presentes no contexto de um teste de invasão.

Um teste de invasão “trata-se de um método para testar e descobrir vulnerabilidades em uma rede ou sistemas operacionais [...] insere métodos de avaliação de segurança em um sistema de computador ou rede” (GIAVAROTO; SANTOS, 2013, p. 19). Existem cinco fases que compõem um teste de invasão, são elas: (i) coleta de informações do alvo, (ii) varreduras de sistema, (iii) ganho de acesso ao sistema, (iv) manutenção do acesso ao sistema e (v) retirada de evidências.

A coleta de informação é a fase onde nada pode ser descartado. Segundo Engbretson, as informações coletadas são cruciais para o sucesso na exploração e ganho de acesso ao sistema, ou seja, quanto mais informações a respeito do sistema a ser auditado, maior a probabilidade de uma execução bem sucedida

(ENGBRETSON, 2011). As varreduras de sistema se caracterizam pela utilização de ferramentas que descobrem as vulnerabilidades que possam ser exploradas nos sistemas.

A fase de ganho de acesso consiste na exploração das vulnerabilidades encontradas pela fase anterior, de modo que o ganho de acesso ao sistema é efetivado. Após o ganho de acesso ao sistema, é necessário que esse acesso seja mantido com o objetivo de ser semelhante a um ataque real, então técnicas são utilizadas para que esse acesso não seja detectado pelas ferramentas de segurança. É um ponto importante para verificar se essas ferramentas instaladas no ambiente estão funcionando corretamente.

A última fase consiste na eliminação dos rastros da invasão. Como tais ações cometidas são consideradas crimes virtuais, os atacantes utilizam planejamentos adequados para eliminar os rastros. O teste de invasão também utilizará os mesmos conceitos, embora seja autorizado a executar as ações dos atacantes.

### **2.1.2 Tipos**

Existem três tipos de testes de invasão, são eles: (i) *Whitebox*, (ii) *Blackbox* e (iii) *Graybox*. O primeiro fornece aos auditores o conhecimento completo da infraestrutura a ser testada e simula um ataque interno realizado por alguém que conhece o ambiente, tem o objetivo de avaliar a segurança da empresa em geral (ENGBRETSON, 2011, p.12). O segundo teste não fornece nenhum tipo de informação a respeito da infraestrutura do ambiente, por isso é conhecido como teste “às cegas”. Simula um ataque externo ao ambiente. No terceiro tipo, o auditor tem conhecimento prévio da infraestrutura, como o conjunto de credenciais da rede. Simula o ataque de um colaborador ou um prestador de serviço.

Para este projeto, foi escolhida a metodologia *Whitebox*, para simular o ataque de uma pessoa que conheça o ambiente e que, por algum motivo, tem o objetivo de prejudicar o funcionamento das atividades do Centro de Ciências e Tecnologia da



Universidade Federal do Estado do Rio de Janeiro (Unirio) ou de subtrair dados importantes.

## **2.2 Vulnerabilidades**

Esta seção tem por objetivo: apresentar o conceito de vulnerabilidade; como funcionam as ferramentas que realizam a varredura nos sistemas em busca de vulnerabilidades; e a métrica para a avaliação das vulnerabilidades encontradas durante o teste. Serão definidas as principais vulnerabilidades voltadas às aplicações *web*. A seção não abordará as principais vulnerabilidades de Sistemas Operacionais, nem de aplicações específicas visto que estas variam de versão para versão.

### **2.2.1 Definição**

Segundo Stroparo, vulnerabilidade em computação representa uma brecha num sistema computacional. Esta brecha pode ser utilizada pelos atacantes como um ponto de exploração para ter acesso ao sistema ou serviço vulnerável (STROPARO, 2010).

### **2.2.2 Principais Vulnerabilidades de aplicações *Web***

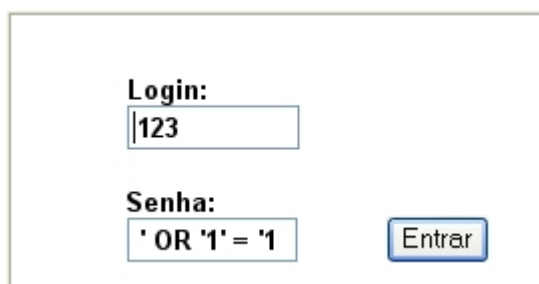
#### **2.2.2.1 SQL *Injection***

Um ataque baseado na vulnerabilidade *SQL Injection* consiste na alteração proposital e maliciosa de comandos SQL com o objetivo de manipular os comandos originais esperados pela aplicação, essas entradas, que são fornecidas pelo usuário, são utilizadas para a execução de instruções arbitrárias e não autorizadas. De modo geral, “o que a injeção de SQL faz é justamente usar a capacidade da linguagem SQL para modificar a consulta existente em um campo” (DIÓGENES, YURI; MAUSER,

DANIEL, 2013, p.124). Ao utilizar essa técnica de exploração, um atacante pode ter acesso a informações confidenciais, manipular dados e até mesmo executar comandos mais rebuscados através da API (*Application Programming Interface*) do Sistema de Gerenciamento de Banco de Dados (SGBD).

A maioria destes ataques tem origem de redes não conectadas diretamente ao SGBD, são originados a partir de redes conectadas aos servidores de aplicação. Com isso, este ataque explora falhas que não são de responsabilidade diretamente do SGBD, mas sim da validação ineficiente dos parâmetros informados pelo usuário e que são utilizados pela aplicação para compor seus comandos SQL.

A Figura 1 mostra um clássico exemplo de *SQL Injection* na autenticação de usuário. O problema ocorre devido à autenticação ser validada internamente pela aplicação com a seguinte consulta SQL: *SELECT \* FROM usuarios WHERE login='campo\_login' AND senha='campo\_senha'*.



Interface de login com os seguintes campos e valores:

- Login:** 123
- Senha:** " OR "1" = "1"
- Botão:** Entrar

Figura 1: *SQL Injection*

Normalmente, ao inserir o *login* e a senha, a aplicação montaria uma consulta SQL para verificar no banco de dados se o usuário e a senha são correspondentes. Neste caso, os parâmetros passados resultarão na seguinte instrução: *SELECT \* FROM usuários WHERE login='123' AND senha='' or '1' = '1'*. Pode ser observado que o comando passado na senha possibilita que, independente do *login* e senha informados, a condição sempre seja verdadeira, permitindo assim, o acesso de um usuário não autorizado à aplicação.

A mitigação de uma vulnerabilidade de *SQL Injection* deve acontecer no desenvolvimento da aplicação, onde deve ser restringido o tratamento dos parâmetros de entrada para consultas SQL no SGBD da aplicação.

#### 2.2.2.2 *Cross-Site-Scripting*

Um ataque baseado na vulnerabilidade *Cross-Site-Scripting* (XSS) também é um tipo de inserção de código. Tecnicamente, o parâmetro de entrada do usuário é apresentado integralmente pelo navegador, como no caso de um código *JavaScript*, que passa a ser interpretado como parte da aplicação *web* legítima. De um modo geral, o atacante injeta códigos em um campo texto de uma página confiável para o usuário, para que sejam executados no momento que a página seja acessada. Após a execução destes códigos, “podem ser executadas rotinas para envio, remoção ou alteração de dados” (DIÓGENES, YURI; MAUSER, DANIEL, 2013, p.99).

Geralmente, os atacantes procuram páginas que possuam campos de entrada de dados para pesquisas no site ou em campos utilizados para a realização de comentários. Dessa forma, eles podem testar se o site está vulnerável ou não ao XSS. A Figura 2 demonstra o teste realizado pelo atacante para verificar se a página possui esta vulnerabilidade. Neste caso, o indivíduo consegue ler o cookie da vítima pela linguagem *JavaScript* (exemplo: `alert(document.cookie)`) e enviar o seu conteúdo para o site do atacante visando um posterior sequestro de sessão.



Figura 2: Teste do *Cross-Site-Scripting*

A Figura 3 representa a execução deste script inserido pelo atacante, onde uma janela aparece demonstrando a *Session Id* do usuário que executa o código.



Figura 3: Constatação do *Cross-Site-Scripting*

## 2.2.3 Classificação das Vulnerabilidades

### 2.2.3.1 Vulnerabilidades Críticas

Vulnerabilidades as quais permitiram o comprometimento lógico da segurança do sistema/ativo avaliado. Na maioria dos casos, a exploração desta vulnerabilidade permitirá ao atacante acesso administrativo ao sistema afetado. Neste caso, a segurança do sistema ou da rede pode ser totalmente comprometida.

### 2.2.3.2 Vulnerabilidades Altas

Incluem vulnerabilidades as quais podem permitir o comprometimento lógico da segurança do sistema/ativo avaliado, porém não foi possível sua validação devido ao risco de ocorrência de indisponibilidade do sistema em uma possível tentativa de exploração ou devido à dependência de outras condições para que a exploração se materialize.

#### **2.2.3.3 Vulnerabilidades Médias**

Incluem vulnerabilidades que podem permitir o acesso indireto a dados e arquivos de configuração do sistema afetado, possibilitar ataques de força bruta para acesso indevido, dentre outras. Vulnerabilidades associadas a configurações padrões de sistemas que exponham informações sensíveis ou que causem exclusivamente negação de serviço também são classificadas como de nível médio.

#### **2.2.3.4 Vulnerabilidades Baixas**

Vulnerabilidades de nível baixo podem permitir que um atacante obtivesse estatísticas de sistema, listas de contas de usuários, versões de serviços de rede e outras informações que normalmente são utilizadas para a construção de ataques mais sofisticados.

### **2.3 Métricas para o Teste de Invasão**

Esta seção apresentará as métricas utilizadas para a realização do teste de invasão.

#### **2.3.1 Varredura em Busca de Vulnerabilidades**

Considerada como uma etapa de um teste de invasão, uma varredura em busca de vulnerabilidades consiste na realização de uma pesquisa em um determinado alvo para realizar testes para verificar se existem brechas de segurança nos sistemas computacionais analisados.

### **2.3.2 Engenharia Social**

Denomina-se engenharia social as práticas utilizadas para obter acesso a informações importantes ou sigilosas em organizações ou sistemas por meio da exploração da fragilidade e confiança dos recursos humanos. Para realizar essas práticas, o indivíduo mal intencionado utiliza técnicas, como se passar por outra pessoa ou empresa, e consegue explorar pontos sensíveis dos recursos humanos da organização que ainda não foram treinados contra estes tipos de ataques.

O autor Filho considera um aspecto relevante para o sucesso das práticas de engenharia social: a inconsciência dos indivíduos sobre o valor da informação que eles têm sob poder, portanto, não tem preocupação em protegê-la (FILHO, 2004).

A maioria das técnicas que são utilizadas em ataques de engenharia social, têm o objetivo de coletar informações privilegiadas e podem ser executadas em qualquer meio de comunicação. Duas dessas técnicas são: os Vírus que se espalham por e-mail, e o *Phishing*.

#### **2.3.2.1 Vírus disseminados por e-mail**

Essa técnica consiste na disseminação de vírus por e-mail. A ideia do atacante é despertar a curiosidade de usuário sob a forma de um e-mail, fazendo com que ele realize o download do anexo e instale o vírus em seu dispositivo.

#### **2.3.2.2 *Phishing***

O termo *Phishing*, do inglês *fishing* que quer dizer pesca, foi assim denominado pelo fato dos atacantes jogarem “iscas” e tentarem “pescar” informações de usuários desavisados. É um dos tipos de ataques de engenharia social mais comuns de ser utilizado. Existem dois tipos de *Phishing*, são eles: *Blind Phishing* e *Spear-Phishing*.

No tipo *Blind Phishing*, geralmente, os atacantes estão interessados nas informações bancárias da vítima e disparam diversos e-mails em massa. Como exemplo, a página de um banco pode ser clonada e hospedada em algum endereço da internet muito parecido com o verdadeiro. Assim, o atacante envia um e-mail utilizando um layout parecido com o do banco, avisando ao cliente que a conta dele está prestes a ser bloqueada e que para desbloqueá-la, o cliente deve entrar no endereço falso especificado no corpo do e-mail, e inserir os seus dados. Dessa forma, o atacante consegue capturar os dados bancários dos clientes.

O tipo *Spear-Phishing*, como sugerido pela tradução “pesca com arpão”, é mais direcionado a alvos específicos e previamente estudados. Segundo Karasinski, é um ataque mais rebuscado do que o normal, pelo fato de ter um alvo específico (KARASINSKI, 2011).

## **2.4 As Normas ISO/IEC 27001 e 27002**

Esta seção apresenta os conceitos das normas ISO/IEC 27001 e 27002, utilizadas como base para recomendar boas práticas de segurança da informação.

Cada série da ISO 27000 foi construída com um determinado foco, a 27001, tem por objetivo a implantação dos alicerces de segurança da informação em uma determinada organização. A 27002, implementa controles para manter a segurança da informação de uma organização.

A norma ISO/IEC 27001 é um guia para a gestão em segurança da informação, ou seja, ela define como administrar um Sistema de Gestão de Segurança da Informação (SGSI). Segundo Kosutic, esse sistema de gestão significa que a segurança da informação deve ser planejada, implementada, monitorada, analisada e aperfeiçoada (KOSUTIC, 2010).

A ISO/IEC 27002 apresenta o código de prática para a gestão da segurança da informação. Esta norma estabelece diretrizes e princípios gerais para que uma organização tenha uma gestão em segurança da informação de qualidade, também serve como um guia prático para desenvolver os procedimentos de segurança da informação da organização.

As duas normas são aplicadas em conjunto, visto que, sem os controles detalhados da ISO/IEC 27002, os controles propostos no anexo A da ISO/IEC 27001 não poderiam ser implementados. Sem a estrutura do SGSI, fornecida pela norma 27001, as recomendações da norma 27002 não teriam um impacto real na organização, visto que a alta administração não aprovaria a implementação de determinados controles, sem estarem baseados em alguma estrutura para a gestão da segurança da informação (KOSUTIC, 2010).



### **3. Infraestrutura Técnica, Ferramentas e Arquitetura.**

Neste capítulo, a infraestrutura do teste será especificada. Para cada etapa, serão demonstradas as ferramentas que foram utilizadas para a realização dos testes e chegar aos resultados finais. Também será explicitada a arquitetura das tarefas de cada etapa do trabalho.

#### **3.1 A Infraestrutura em Detalhes**

Esta seção apresentará o a infraestrutura do teste de invasão no ambiente Uniriotec. Será apresentado um diagrama que demonstra a disposição do servidor de testes na rede.

Para a realização dos testes, foi utilizado o sistema operacional Kali. Esta distribuição Linux foi escolhida porque é uma versão que possui diversas ferramentas para testes de invasão, é a mais atualizada e por isso, possui bastante quantidade de conteúdo virtual disponível. Foi criada uma máquina virtual, com cinco gigabytes de memória RAM e quarenta gigabytes de disco, num servidor VMware ESXi, pertencente ao projeto de monitoria da disciplina Redes 1.

A Figura 4 apresenta a topologia de rede que foi construída para a realização das etapas do teste de invasão. Os demais elementos de rede não foram representados para evitar a exposição completa da rede interna da Instituição.

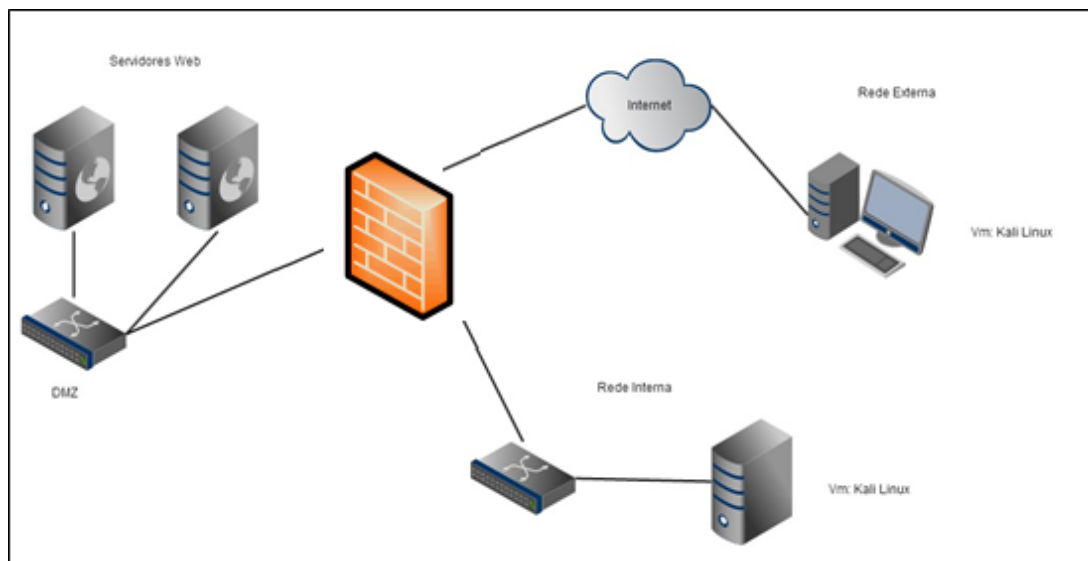


Figura 4: Topologia

## 3.2 Ferramentas

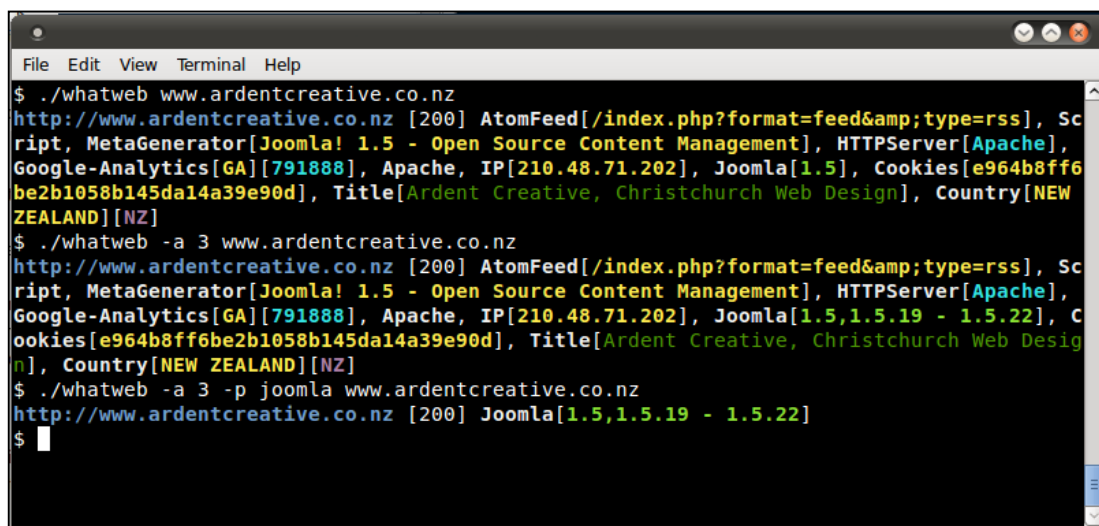
Esta seção realiza uma breve descrição de cada ferramenta utilizada para a execução das tarefas propostas neste trabalho.

### 3.2.1 Varreduras de vulnerabilidades *Web*

Para a busca de vulnerabilidades nas aplicações web, pertencentes ao ambiente Uniriotec, serão utilizadas as seguintes ferramentas: Whatweb e Vega.

A Whatweb é uma ferramenta de código aberto que é utilizada para obter informações sobre as tecnologias *web* que estão presentes no *website* analisado. Ela é capaz de reconhecer plataformas de *blog*, bibliotecas de *JavaScript*, servidores *web* e erros de SQL. Ela tem uma importante função de identificar o que existe por traz do *website* “escaneado”, logo, um atacante pode utilizar estes dados como forma de

exploração de vulnerabilidade. A Figura 5 mostra as informações que podem ser obtidas através do escaneamento da página.



```
File Edit View Terminal Help
$ ./whatweb www.ardentcreative.co.nz
http://www.ardentcreative.co.nz [200] AtomFeed[/index.php?format=feed&type=rss], Script, MetaGenerator[Joomla! 1.5 - Open Source Content Management], HTTPServer[Apache], Google-Analytics[GA][791888], Apache, IP[210.48.71.202], Joomla[1.5], Cookies[e964b8ff6be2b1058b145da14a39e90d], Title[Ardent Creative, Christchurch Web Design], Country[NEW ZEALAND][NZ]
$ ./whatweb -a 3 www.ardentcreative.co.nz
http://www.ardentcreative.co.nz [200] AtomFeed[/index.php?format=feed&type=rss], Script, MetaGenerator[Joomla! 1.5 - Open Source Content Management], HTTPServer[Apache], Google-Analytics[GA][791888], Apache, IP[210.48.71.202], Joomla[1.5,1.5.19 - 1.5.22], Cookies[e964b8ff6be2b1058b145da14a39e90d], Title[Ardent Creative, Christchurch Web Design], Country[NEW ZEALAND][NZ]
$ ./whatweb -a 3 -p joomla www.ardentcreative.co.nz
http://www.ardentcreative.co.nz [200] Joomla[1.5,1.5.19 - 1.5.22]
$
```

Figura 5: WhatWeb

A ferramenta de código aberto Vega tem o propósito de “escanear” e testar a segurança das aplicações *web*. Assim, consegue encontrar e validar vulnerabilidades como *SQL Injection*, *XSS* e informações sensíveis exibidas. Após a descoberta das falhas de segurança, a ferramenta indica o que deve ser feito para que estas sejam corrigidas. Com isso, é uma ferramenta indispensável na busca por vulnerabilidades nas aplicações *web*. A Figura 6 mostra a console da ferramenta.

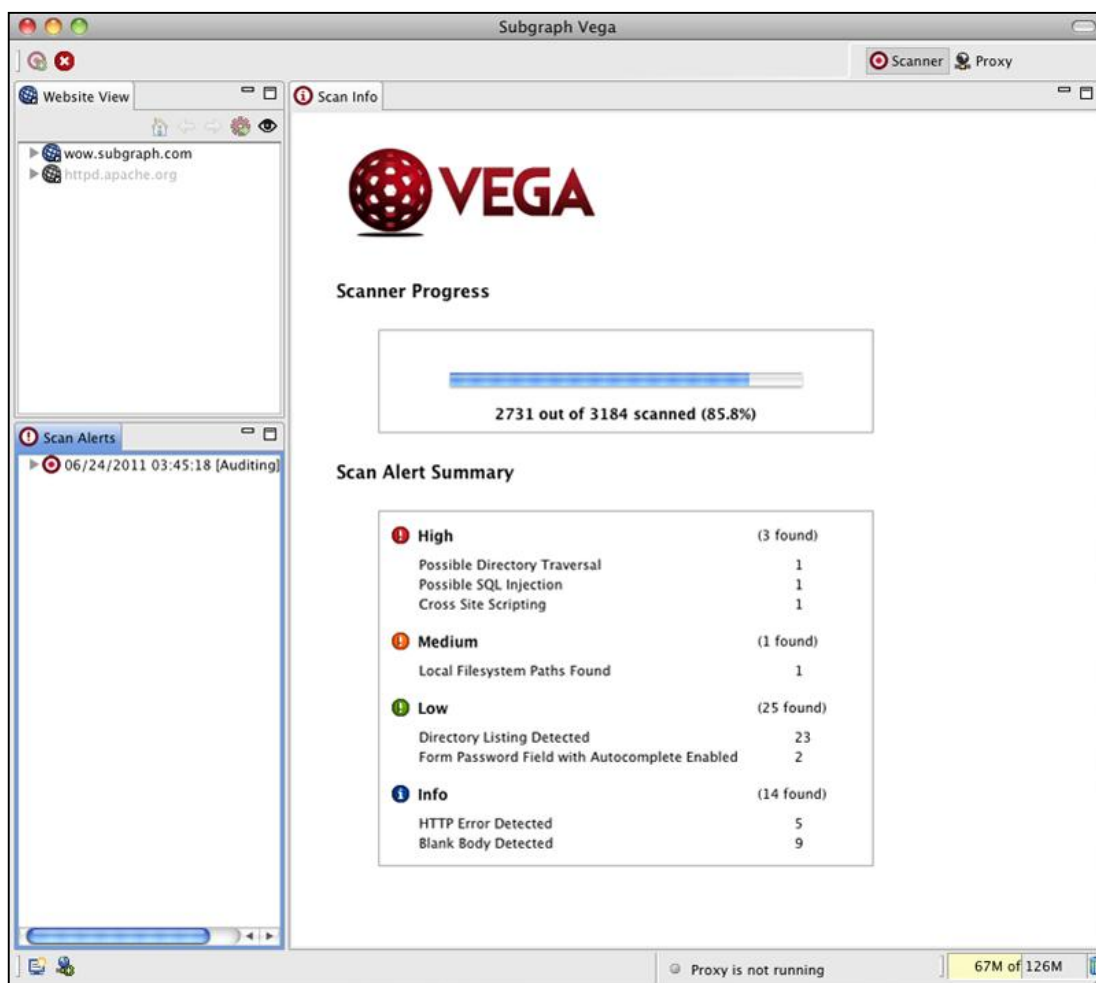


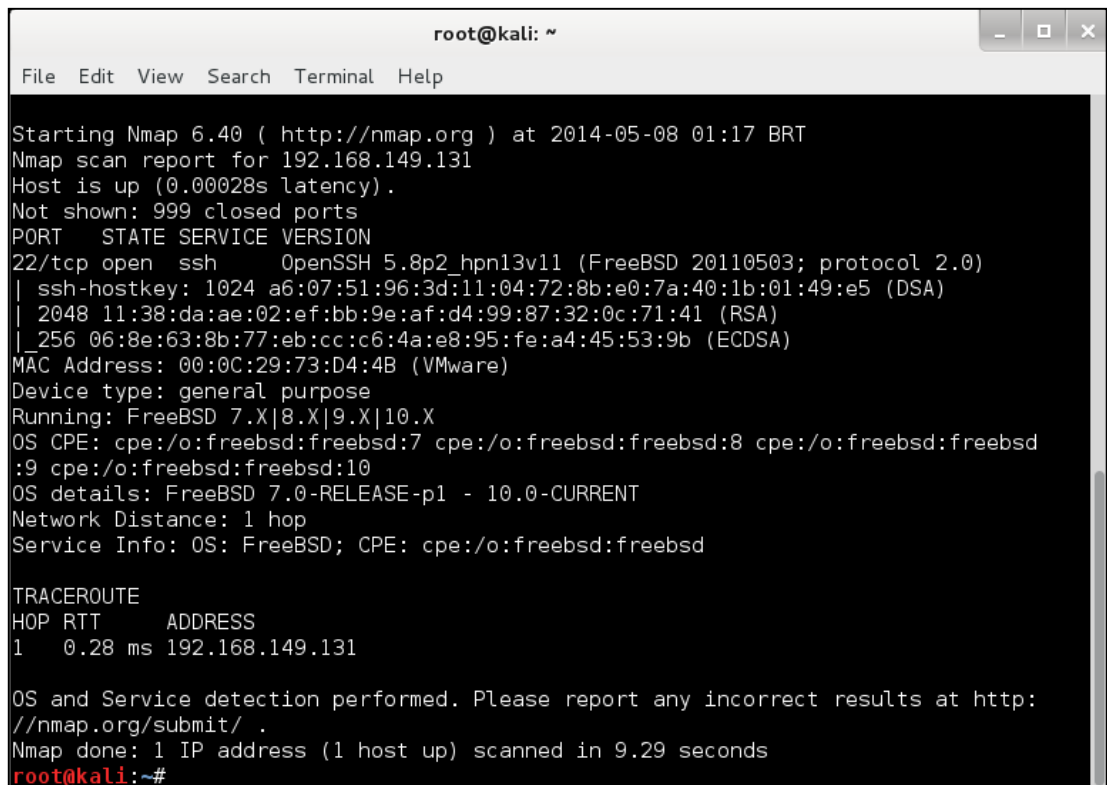
Figura 6: Vega

### 3.2.2 Varreduras de vulnerabilidades em servidores

Para o mapeamento da rede e descoberta de sistemas operacionais, será utilizada a ferramenta Nmap. Para a varredura de vulnerabilidades nos servidores, será utilizada a ferramenta Nessus.

O Nmap é um *scanner* com múltiplas funções, podendo ser usado para descobrir recursos em uma rede de computadores, criando assim um "mapa" da rede. É uma ferramenta que pode ser utilizada tanto do ponto de vista do administrador de

redes, quanto o de um indivíduo mal intencionado. O Nmap é capaz de descobrir remotamente, o sistema operacional, *uptime*, software utilizado para executar um serviço, entre outros detalhes. A Figura 7 mostra a uma varredura realizada pelo Nmap.



```
root@kali: ~  
File Edit View Search Terminal Help  
Starting Nmap 6.40 ( http://nmap.org ) at 2014-05-08 01:17 BRT  
Nmap scan report for 192.168.149.131  
Host is up (0.00028s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 5.8p2_hpn13v11 (FreeBSD 20110503; protocol 2.0)  
| ssh-hostkey: 1024 a6:07:51:96:3d:11:04:72:8b:e0:7a:40:1b:01:49:e5 (DSA)  
| 2048 11:38:da:ae:02:ef:bb:9e:af:d4:99:87:32:0c:71:41 (RSA)  
| 256 06:8e:63:8b:77:eb:cc:c6:4a:e8:95:fe:a4:45:53:9b (ECDSA)  
MAC Address: 00:0C:29:73:D4:4B (VMware)  
Device type: general purpose  
Running: FreeBSD 7.X|8.X|9.X|10.X  
OS CPE: cpe:/o:freebsd:freebsd:7 cpe:/o:freebsd:freebsd:8 cpe:/o:freebsd:freebsd  
:9 cpe:/o:freebsd:freebsd:10  
OS details: FreeBSD 7.0-RELEASE-p1 - 10.0-CURRENT  
Network Distance: 1 hop  
Service Info: OS: FreeBSD; CPE: cpe:/o:freebsd:freebsd  
  
TRACEROUTE  
HOP RTT      ADDRESS  
1    0.28 ms  192.168.149.131  
  
OS and Service detection performed. Please report any incorrect results at http:  
//nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 9.29 seconds  
root@kali:~#
```

Figura 7: Nmap

O Nessus é um *scanner* de vulnerabilidades para a identificação e categorização de vulnerabilidades de sistema operacional, serviços de rede e aplicativos. Realiza diversas verificações de segurança remotamente e possui uma base de soluções para as falhas de segurança apontadas. A Figura 8 mostra a console do Nessus.

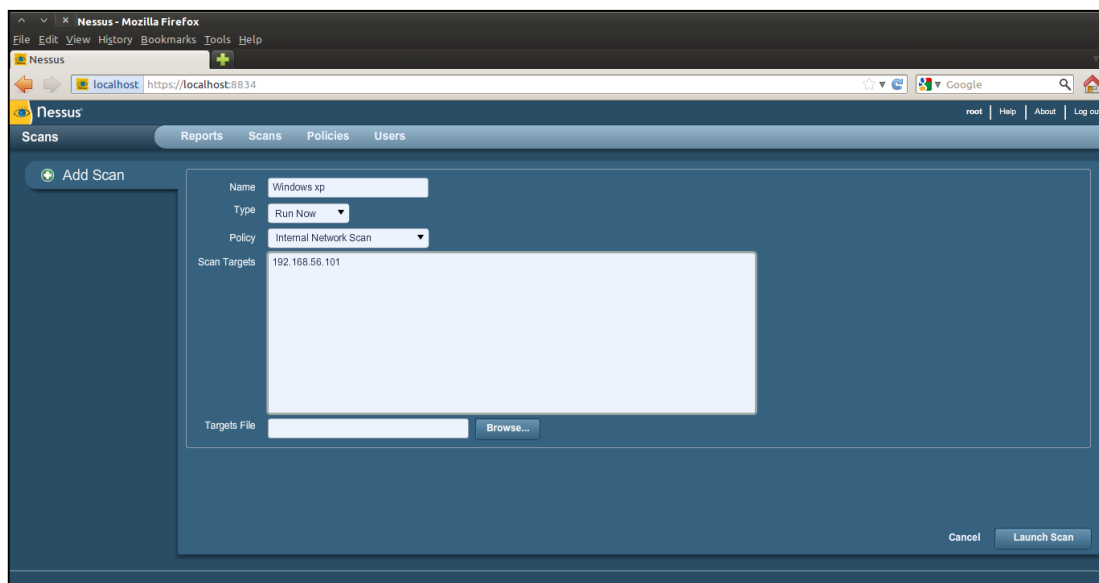


Figura 8: Nessus

### 3.3 Arquitetura do Teste de Invasão

Esta seção apresenta a organização geral das etapas do teste de invasão realizado.

O escaneamento de vulnerabilidades *web* foi composto pelos *websites* que são de propriedade da Uniriotec, sendo eles: *bsi.uniriotec.br*, *uniriodb.uniriotec.br*, *uniriodb2.uniriotec.br*. A Tabela 1 demonstra a janela em que os escaneamentos foram executados.

Páginas Web	Janela	Localização
uniriodb2.uniriotec.br	18/04/2014 - 08:00h até 09:30h	Remoto
uniriodb.uniriotec.br		
bsi.uniriotec.br		

Tabela 1: Janela vulnerabilidades Web

A varredura em busca de vulnerabilidades foi composta pelos servidores que hospedam o Moodle Uniriodb (também hospeda a página web *bsi.uniriotec.br*) e

Moodle Uniriodb2, que terão os ip's fictícios 127.0.0.1 e 127.0.0.2, respectivamente. Os ip's internos não serão representados visto à não exposição de dados da rede interna da Uniriotec. Essas máquinas foram escolhidas por estarem presentes na zona desmilitarizada (DMZ), logo, um atacante externo conseguiria enxergá-los e provavelmente, tentaria explorar as vulnerabilidades desses servidores. A Tabela 2 mostra a organização dos testes.

Servidores	Janela de Execução	Localização
127.0.0.1	28/05/2014 -20:00h até 21:30h	Acesso Físico
127.0.0.2		

Tabela 2: Organização varredura vulnerabilidades em servidores

No teste de Engenharia Social, foi executado um *Phishing*, através de um e-mail, enviado aos funcionários da secretaria objetivando que eles entrassem em um determinado link para que sejam alertados dos perigos dos ataques de Engenharia Social. Foi enviado um e-mail que tinha como remetente uma empresa inexistente de segurança da informação, que estaria prestando um serviço de auditoria à Unirio. Nesta mensagem foi comentada a mudança na política de senhas e foram passadas algumas recomendações para a construção de uma senha forte, utilizando caracteres especiais, letras maiúsculas e minúsculas e números.

Ao final da mensagem, um link foi disponibilizado para que o usuário acesse, caracterizando o *Phishing*. O link redirecionava o usuário para um formulário da Google que informa ao usuário a realização teste de Engenharia Social que foi realizado para um Projeto de Graduação. Algumas recomendações de segurança foram listadas com o objetivo de treinar aos funcionários a perceberem este tipo de ataque.

As Figuras 9 e 10 mostram, respectivamente, o modelo utilizado para o envio das mensagens e a mensagem exibida para os usuários que acessaram o link.



## Política de Senhas

Prezado Usuário,

A empresa HETÁ Security foi responsável pela realização de uma Auditoria em Segurança da Informação na UNIRIO, compreendida pelo período de 05/04/2014 até 05/05/2014.

Como parte deste trabalho, as senhas dos usuários que possuem acesso à plataforma SIE foram testadas, com o objetivo de estabelecer uma nova política de senhas seguras.

A nova política de senhas consiste em:

- Senhas com no mínimo 8 caracteres;
- Pelo menos um caractere especial, como: #,\$,%,@,!
- Utilização de letras maiúsculas e minúsculas;
- Utilização de números;

Solicitamos que entre no site abaixo o mais rápido possível e troque a sua senha.

[Clique aqui](#) para alterar a sua senha.

Atenciosamente,

Jão Carlos Faria de Melo

Analista de Segurança.

Figura 9: Modelo do e-mail



## Teste de Engenharia Social

Prezado Usuário,

Esta página refere-se à um teste de Engenharia Social, que foi executado como parte do projeto final de graduação do aluno Daniel Possolo, sob orientação do professor Sidney Lucena.

A realização deste teste foi devidamente autorizada pelo Decano Luiz Amâncio de Sousa Júnior, via assinatura do Termo de Autorização para a execução de Testes de Invasão.

Aproveitando a oportunidade, será divulgado um breve guia para a introdução e prevenção contra este tipo de ataque.

A Engenharia Social é uma prática utilizada para obter acesso a informações importantes ou sigilosas em organizações ou sistemas por meio da enganação ou exploração da confiança das pessoas. Para isso, o golpista pode se passar por outra pessoa, assumir outra personalidade, fingir que é um profissional de determinada área, etc. É uma forma de entrar em organizações que não necessita da força bruta ou de falhas em máquinas.

Existem diversas técnicas que podem ser utilizadas num ataque de Engenharia Social. Para esse teste foi utilizada a técnica de Phishing.

O Phishing (pronuncia-se "fishing") é um tipo de roubo de identidade online. Ele usa e-mail e websites fraudulentos que são concebidos para roubar seus dados ou informações pessoais, como números de cartão de crédito, senhas, dados de contas ou outras informações.

Para evitar esse tipo de ataque, as seguintes recomendações devem ser seguidas:

- Não se sirva de links contidos numa mensagem de e-mail para carregar uma página Web. Em vez disso, digite o URL no seu navegador Web;
- Verifique o certificado de segurança do site antes de inserir informações pessoais ou financeiras em um site.
- Não digite informações pessoais ou financeiras em janelas pop-up.
- Mantenha o software de seu computador protegido com as atualizações de segurança mais recentes.
- Ao receber mensagens com anexos, não abra os arquivos ou execute programas sem antes passar por um programa antivírus. Na dúvida sobre a veracidade da mensagem recebida, não abra.
- Desconfie das mensagens de correio eletrônico onde você não conhece o remetente. Leia atentamente a mensagem. Geralmente, e-mails falsos contêm diversos erros gramaticais e de ortografia. É importante observar que o remetente não deve ser usado para garantir a veracidade da mensagem, uma vez que pode ser forjado pelo fraudador.
- Desconfie de e-mails com ofertas extraordinárias que circulam pela Internet.
- Cuidado com downloads, especialmente os arquivos com extensões ".exe", ".zip" e ".scr".
- Crie senhas difíceis de serem descobertas.

Muito obrigado pela atenção,

Daniel Possolo Gomes D. Castanha.

Figura 10: Mensagem exibida ao usuário que acessou o link

## **4. Análise dos Resultados e Recomendações**

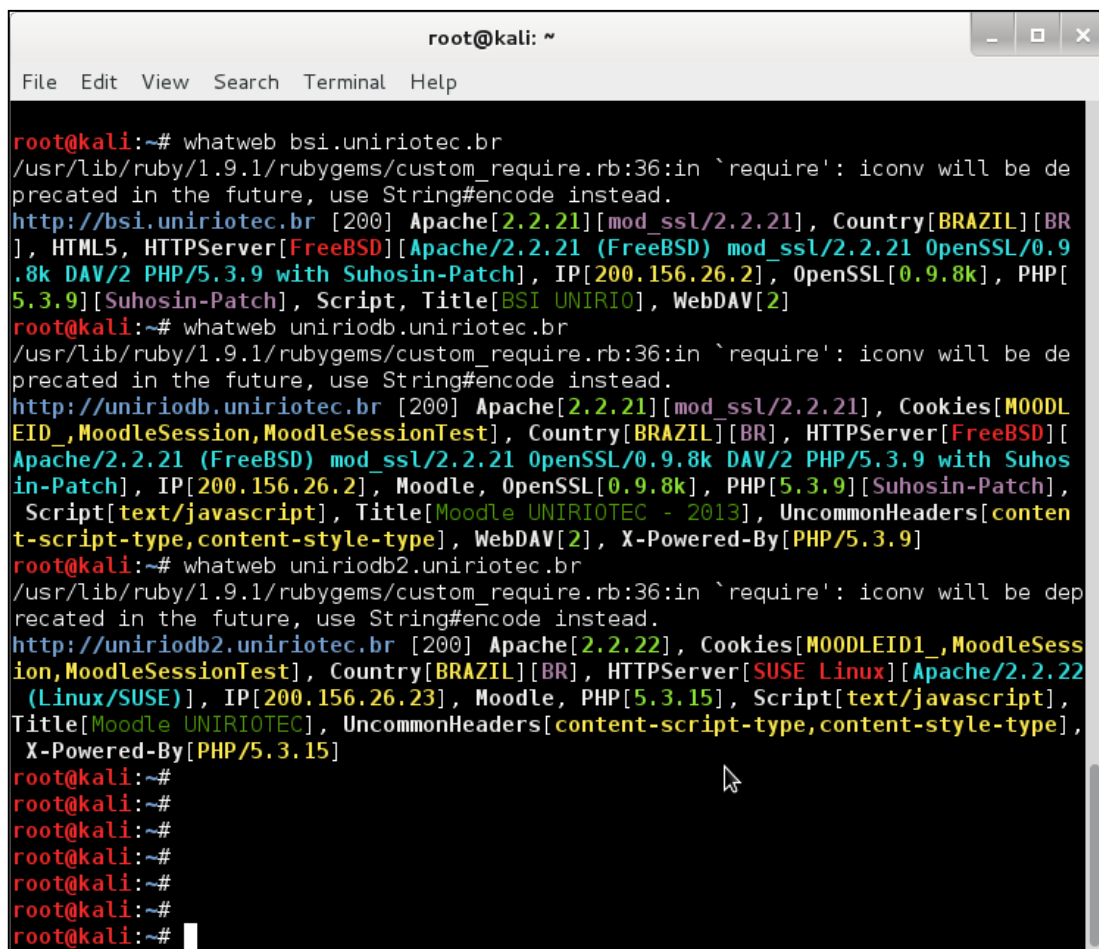
Este capítulo analisa os resultados obtidos através das tarefas propostas que compõe o teste de invasão realizado no ambiente Uniriotec. Também são apresentadas tanto as recomendações baseadas no resultado do teste de invasão, quanto as recomendações baseadas na ISO/IEC 27002.

### **4.1 Vulnerabilidades *Web***

Esta seção apresentará a análise dos resultados obtidos pelas varreduras em busca de vulnerabilidades nas aplicações *Web* do ambiente Uniriotec.

#### **4.1.1 Descoberta de Informações**

Através da ferramenta WhatWeb, algumas informações a respeito das aplicações *web* foram descobertas. Em um ataque real, este artefato seria utilizado para descobrir, por exemplo, a versão do servidor, o sistema operacional que roda a aplicação, a versão do OpenSSL utilizado (tendo em vista a vulnerabilidade *HeartBleed*, descoberta em Abril de 2014). De posse dessas informações, o indivíduo mal intencionado poderia explorar as vulnerabilidades específicas a estes serviços e ser mais eficiente em seu ataque. A Figura 11 mostra os resultados obtidos para os três sites analisados durante este trabalho.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# whatweb bsi.uniriotec.br  
/usr/lib/ruby/1.9.1/rubygems/custom_require.rb:36:in `require': iconv will be de  
precated in the future, use String#encode instead.  
http://bsi.uniriotec.br [200] Apache[2.2.21][mod_ssl/2.2.21], Country[BRAZIL][BR  
, HTML5, HTTPServer[FreeBSD][Apache/2.2.21 (FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9  
.8k DAV/2 PHP/5.3.9 with Suhosin-Patch], IP[200.156.26.2], OpenSSL[0.9.8k], PHP[  
5.3.9][Suhosin-Patch], Script, Title[BSI UNIRIO], WebDAV[2]  
root@kali:~# whatweb uniriodb.uniriotec.br  
/usr/lib/ruby/1.9.1/rubygems/custom_require.rb:36:in `require': iconv will be de  
precated in the future, use String#encode instead.  
http://uniriodb.uniriotec.br [200] Apache[2.2.21][mod_ssl/2.2.21], Cookies[MOODL  
EID_,MoodleSession,MoodleSessionTest], Country[BRAZIL][BR], HTTPServer[FreeBSD][  
Apache/2.2.21 (FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8k DAV/2 PHP/5.3.9 with Suhos  
in-Patch], IP[200.156.26.2], Moodle, OpenSSL[0.9.8k], PHP[5.3.9][Suhosin-Patch],  
Script[text/javascript], Title[Moodle UNIRIOTEC - 2013], UncommonHeaders[conten  
t-script-type,content-style-type], WebDAV[2], X-Powered-By[PHP/5.3.9]  
root@kali:~# whatweb uniriodb2.uniriotec.br  
/usr/lib/ruby/1.9.1/rubygems/custom_require.rb:36:in `require': iconv will be dep  
recated in the future, use String#encode instead.  
http://uniriodb2.uniriotec.br [200] Apache[2.2.22], Cookies[MOODLEID1_,MoodleSess  
ion,MoodleSessionTest], Country[BRAZIL][BR], HTTPServer[SUSE Linux][Apache/2.2.22  
(Linux/SUSE)], IP[200.156.26.23], Moodle, PHP[5.3.15], Script[text/javascript],  
Title[Moodle UNIRIOTEC], UncommonHeaders[content-script-type,content-style-type],  
X-Powered-By[PHP/5.3.15]  
root@kali:~#  
root@kali:~#  
root@kali:~#  
root@kali:~#  
root@kali:~#  
root@kali:~#  
root@kali:~#
```

Figura 11: Descoberta de Informações

#### 4.1.2 Página do Moodle Uniriodb

Este serviço, ainda disponível para os alunos na Internet, com o endereço *uniriodb.uniriotec.br*, apresentou oito vulnerabilidades consideradas altas, dezessete médias e vinte e três baixas, conforme a Figura 12 abaixo.




 <b>High</b>		(8 found)
Cleartext Password over HTTP	2	
SQL Injection	6	
 <b>Medium</b>		(17 found)
Local Filesystem Paths Found	12	
HTTP Trace Support Detected	1	
Possible Source Code Disclosure	4	
 <b>Low</b>		(23 found)
Directory Listing Detected	21	
Form Password Field with Autocomplete Enabled	2	

Figura 12: Índice de vulnerabilidades encontradas em *uniriodb.uniriotec.br*

#### 4.1.2.1 Vulnerabilidades Altas

As vulnerabilidades encontradas dizem respeito a não criptografia das senhas para autenticação na ferramenta e a seis possibilidades *SQL Injection* encontradas. A Figura 13 exibe em quais diretórios do site foram encontradas as vulnerabilidades.

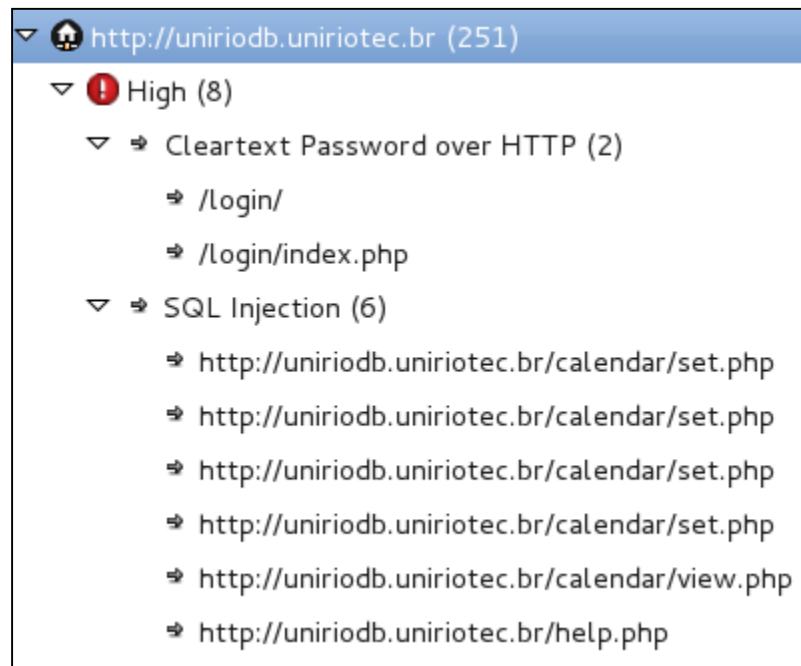


Figura 13: Índice de Vulnerabilidades Altas

A ferramenta verificou a inexistência de uma conexão criptografada no momento em que o usuário se autentica na aplicação. Dessa forma, um atacante que estiver interceptando os pacotes dessa conexão, pode coletar a senha de acesso sem maiores dificuldades. A Figura 14 exibe as informações geradas pela ferramenta.

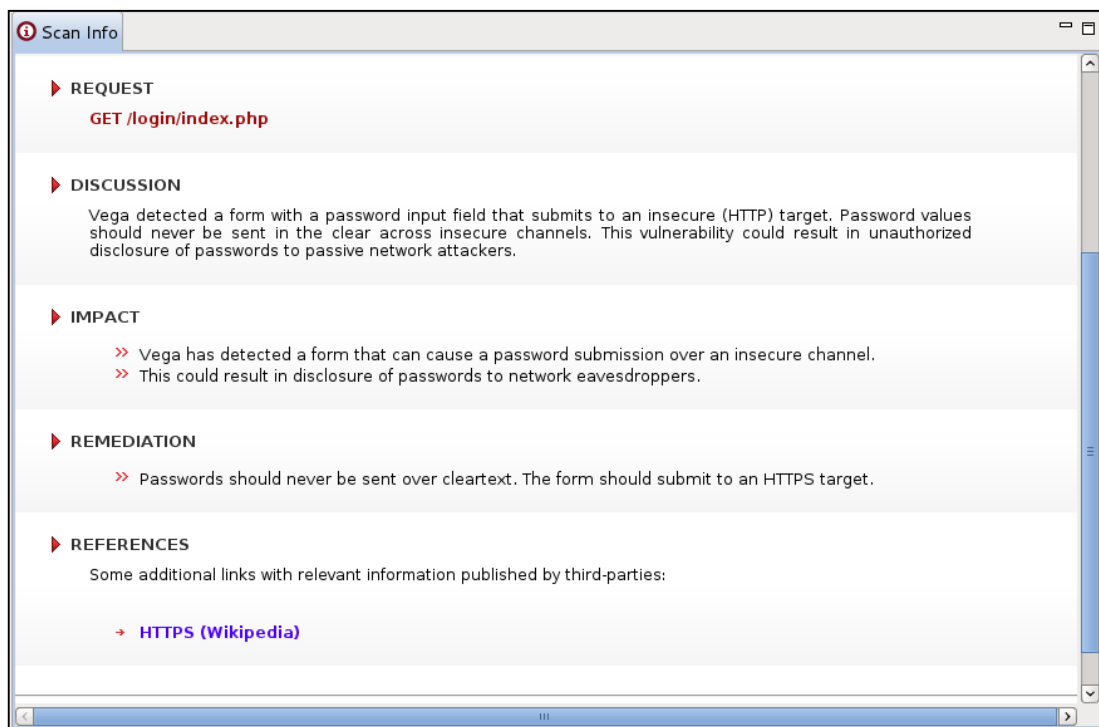


Figura 14: Vulnerabilidade senha em conexão sem criptografia

Também foi detectada a possibilidade de utilização de um ataque baseado em *SQL Injection*, na seção do calendário da aplicação onde uma mensagem é exibida alegando a falta do parâmetro *FROM*, conforme exibido na Figura 15.

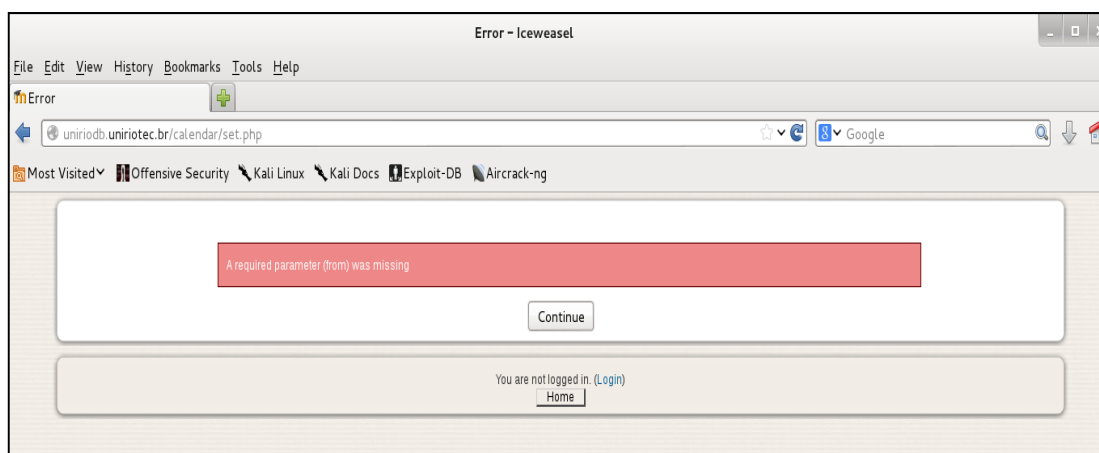


Figura 15: Possibilidade de *SQL Injection*

A Figura 16 apresenta as informações da vulnerabilidade originadas pela ferramenta.

► REQUEST

GET /calendar/set.php?var=showglobal&from=upcoming&id=0"&cal\_d=18&cal\_m=4&cal\_y=2014

► RESOURCE CONTENT

```
<meta http-equiv="refresh" content="0; url=" /><script type="text/javascript">
//
location.replace('http://uniriodb.uniriotec.br/calendar/');
//]]&gt;
&lt;/script&gt;</pre>
<p>► DISCUSSION</p>
<p>Vega has detected a possible SQL injection vulnerability. These vulnerabilities are present when externally-supplied input is used to construct a SQL query. If precautions are not taken, the externally-supplied input (usually a GET or POST parameter) can modify the query string such that it performs unintended actions. These actions include gaining unauthorized read or write access to the data stored in the database, as well as modifying the logic of the application.</p>
<p>► IMPACT</p>
<ul>
<li>» Vega has detected a possible SQL injection vulnerability.</li>
<li>» These vulnerabilities can be exploited by remote attackers to gain unauthorized read or write access to the underlying database.</li>
<li>» Exploitation of SQL injection vulnerabilities can also allow for attacks against the logic of the application.</li>
<li>» Attackers may be able to obtain unauthorized access to the server hosting the database.</li>
</ul>
<p>► REMEDIATION</p>
<ul>
<li>» The developer should review the request and response against the code to manually verify whether or not a vulnerability is present.</li>
<li>» The best defense against SQL injection vulnerabilities is to use parameterized statements.</li>
<li>» Sanitizing input can prevent these vulnerabilities. Variables of string types should be filtered for escape characters, and numeric types should be checked to ensure that they are valid.</li>
<li>» Use of stored procedures can simplify complex queries and allow for tighter access control settings.</li>
<li>» Configuring database access controls can limit the impact of exploited vulnerabilities. This is a mitigating strategy that can be employed in environments where the code is not modifiable.</li>
<li>» Object-relational mapping eliminates the need for SQL.</li>
</ul>
<p>► REMEDIATION</p>
<ul>
<li>» The developer should review the request and response against the code to manually verify whether or not a vulnerability is present.</li>
<li>» The best defense against SQL injection vulnerabilities is to use parameterized statements.</li>
<li>» Sanitizing input can prevent these vulnerabilities. Variables of string types should be filtered for escape characters, and numeric types should be checked to ensure that they are valid.</li>
<li>» Use of stored procedures can simplify complex queries and allow for tighter access control settings.</li>
<li>» Configuring database access controls can limit the impact of exploited vulnerabilities. This is a mitigating strategy that can be employed in environments where the code is not modifiable.</li>
<li>» Object-relational mapping eliminates the need for SQL.</li>
</ul>
<p>► REFERENCES</p>
<p>Some additional links with relevant information published by third-parties:</p>
<ul>
<li>→ <a href="#">SQL Injection (Wikipedia)</a></li>
<li>→ <a href="#">mysql_real_escape_string() (PHP Manual)</a></li>
<li>→ <a href="#">SQL Injection (Rails security guide)</a></li>
<li>→ <a href="#">How To: Protect from SQL Injection in ASP.NET (MSDN)</a></li>
<li>→ <a href="#">Dynamic SQL and SQL Injection (Raul Garcia's blog)</a></li>
<li>→ <a href="#">SQL Injection Prevention Cheat Sheet (OWASP)</a></li>
</ul>
</div>
<div data-bbox="380 584 696 602" data-label="Caption">
<p>Figura 16: Informações <i>SQL Injection</i></p>
</div>
<div data-bbox="194 651 477 668" data-label="Section-Header">
<h4>4.1.2.2 Vulnerabilidades Médias</h4>
</div>
<div data-bbox="194 719 884 867" data-label="Text">
<p>Foram encontradas três tipos de vulnerabilidades altas, a primeira tem a ver com o método HTTP TRACE habilitado no servidor Apache. A segunda significa que a ferramenta encontrou possíveis endereços locais do servidor que hospeda a aplicação <i>web</i>, já a terceira tem a ver com a exposição de assinaturas que correspondem ao código fonte da aplicação, conforme apresentado na Figura 17.</p>
</div>
<div data-bbox="851 952 884 969" data-label="Page-Footer">
<p>39</p>
</div>
```

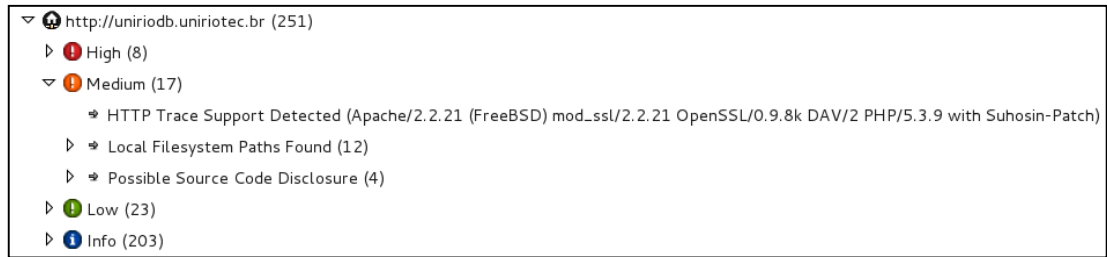


Figura 17: Índice vulnerabilidades médias em uniriodb.uniriotec.br

O método HTTP TRACE faz com que a requisição HTTP seja ecoada na resposta para que o cliente consiga saber o que os servidores intermediários estão alterando em seu pedido. Mesmo que o cliente utilize a *flag* `HttpOnly` em suas requisições, permitindo que o cookie só trafegue em requisições HTTP (visando a prevenção de ataques do tipo *Cross-Site-Scripting*), o indivíduo mal intencionado pode realizar o ataque *Cross-Site-Tracing*. Esta variante do XSS consegue entre outras explorações, obter o conteúdo de um *cookie* com a *flag*, se o método TRACE estiver habilitado, visto que é possível descobrir o que foi enviado na requisição, inclusive os cookies. A Figura 18 demonstra as informações da vulnerabilidade emitidas pela ferramenta.



▶ REQUEST

TRACE /http:

▶ RESOURCE CONTENT

```
TRACE /http: HTTP/1.1
SQUEEMISH: OSS1FR4GE
Accept-Encoding: gzip,deflate
Host: uniriodb.uniriotec.br
Connection: Keep-Alive
User-Agent: UserAgent
Cookie: MOODLEID_%25ED%25C3%251C%25B7d; MoodleSession=vrpjf8h6d83soontr0f1lo5v5; MoodleSessionTest=bjgfbAj7HH
Cookie2: $Version=1
```

▶ DISCUSSION

HTTP TRACE is an HTTP method that requests that the server echo the TRACE request back to the client. This includes headers that were sent along with the request. Support for HTTP TRACE can be abused in scenarios where a cross-site scripting vulnerability has been found, but cannot be exploited to retrieve cookie values because the target cookies are set with the HttpOnly flag. The HttpOnly flag instructs browsers not to permit access to the cookie by Javascript. If a cross-site scripting vulnerability is found, but the session cookie is set HttpOnly, support for HTTP TRACE will open an opportunity for cookie theft. An attacker can use the cross-site scripting vulnerability to have the target user's browser issue a TRACE request to the server via XMLHttpRequest (or a similar function) and then retrieve the cookie from the response, which will contain the request that was sent by the browser, including cookies.

▶ IMPACT

- » Allowing HTTP TRACE can permit cross-site tracing.
- » Attackers may be able to use cross-site tracing with cross-site scripting retrieve the value of HttpOnly cookies.

▶ REMEDIATION

- » For Apache based servers, the TraceEnable directive can be used to disable support for HTTP TRACE.
- » For IIS based servers, the EnableTraceMethod registry setting controls support for HTTP TRACE..

▶ EXTERNAL REMEDIATION GUIDELINES

Below are some links to third-party guidelines, tutorials and other documentation that may be useful in understanding and/or addressing this finding.

- [IBM HTTP Server: Disabling the HTTP TRACE method](#)
- [Apache 2: TraceEnable Directive](#)
- [Windows Server 2012: WWW Service Registry Entries - EnableTraceMethod](#)
- [W2K3 Server: WWW Service Registry Entries - Enable TraceMethod](#)

▶ REFERENCES

Some additional links with relevant information published by third-parties:

- [Wikipedia: Cross-site Tracing](#)
- [CERT: Web servers enable HTTP TRACE method by default](#)
- [OWASP: Cross Site Tracing](#)
- [W3C: RFC 2616 - HTTP 1.1 - 9.8: TRACE](#)

Figura 18: Informações da Vulnerabilidade HTTP TRACE

A vulnerabilidade *Local File System Paths Found*, descreve a possível exposição de diretórios do servidor de arquivos que hospeda a aplicação, essa informação é considerada sensível, visto que aumentará as chances de sucesso do atacante, pois estará de posse de mais informações sobre o alvo. As Figuras 19 e 20 demonstram os possíveis diretórios encontrados e as informações da vulnerabilidade divulgadas pela ferramenta, respectivamente.

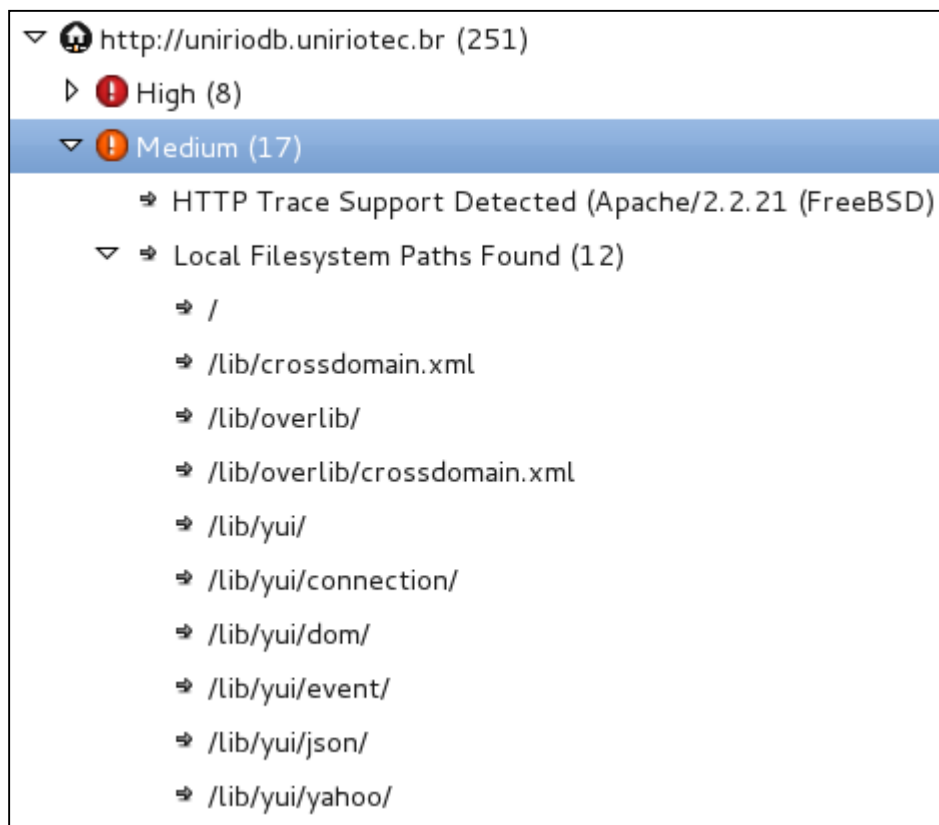


Figura 19: Diretórios locais encontrados

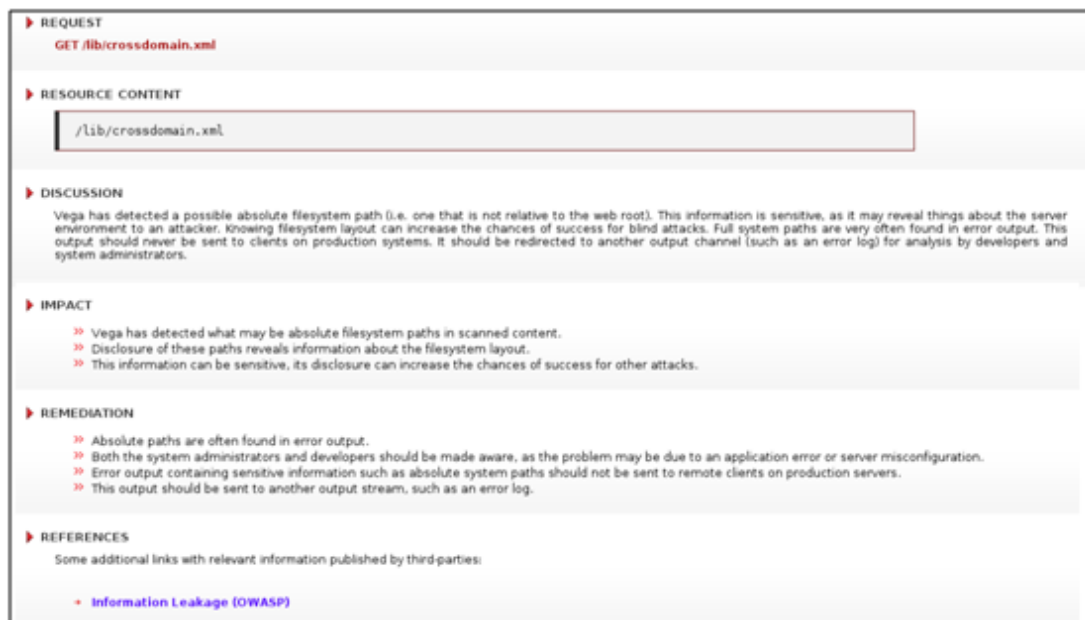


Figura 20: Informações da Vulnerabilidade *Local File System Path*

A última vulnerabilidade média encontrada consiste na exibição de assinaturas que possivelmente indicam a linguagem utilizada para a confecção do código fonte da página. Tal informação também é considerada sensível, visto que aumenta ainda mais o leque de informações coletadas pelo atacante, aumentando a sua possibilidade de sucesso. As Figuras 21 e 22 exibem, respectivamente, os locais onde essas possíveis assinaturas da linguagem PHP foram encontradas e as informações da vulnerabilidade emitidas pela ferramenta, incluindo o trecho do código que contém a assinatura PHP.

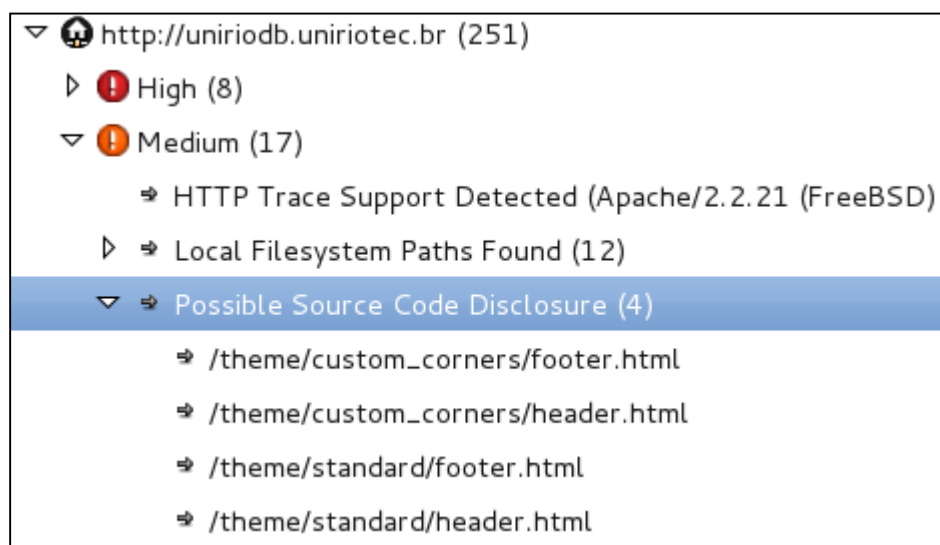


Figura 21: Locais das assinaturas encontradas

▶ REQUEST

GET /theme/standard/footer.html

▶ RESOURCE CONTENT

Possible PHP code:

<?php  
  
print\_container\_end(); // content container  
print\_container\_start(false, '', 'footer');  
  
echo '<p class="helplink">';  
echo page\_doc\_link(get\_string('moodledocslink'));  
echo '</p>';  
  
if (\$navigation and false) { ?>

▶ DISCUSSION

Vega has detected fragments of text that match signatures of application source code. Application source code unintentionally visible to remote clients can be a security vulnerability. This can occur in applications using technologies such as PHP and JSP, which allow for code to be mixed with static presentation content. For example, in-line code is sometimes commented using HTML comments, resulting in it being transmitted to remote clients. For an attacker, source code can reveal information about the nature of the application, such as its design or the use of third-party components. Sometimes sensitive information, such as a database connection string, can be included in source code.

▶ IMPACT

>> Could result in disclosure of sensitive information to attackers.

>> Source code fragments can include information about the design/structure of the application, including use of third-party components.

>> This information may not otherwise be easily known by an adversary.

>> Sometimes source code also contains highly sensitive information, such as passwords (database connection strings).

▶ REMEDIATION

>> The developer should verify that the output detected by Vega is in fact application source code.

>> The cause should be determined, and the material removed or prevented from being output.

▶ REFERENCES

Some additional links with relevant information published by third-parties:

→ [Information Leakage \(OWASP\)](#)

→ [CWE-540: Information Exposure through Source Code \(Mitre\)](#)

→ [Information Leakage \(WASC\)](#)

Figura 22: Informações sobre a Vulnerabilidade *Source Code Disclosure*

### 4.1.2.3 Vulnerabilidades Baixas

Foram encontradas dois tipos de vulnerabilidades baixa, a primeira se refere à exibição do diretório no código HTML da página e a segunda refere-se à utilização do recurso de auto completar palavras no campo de senhas. O índice das vulnerabilidades é apresentado na Figura 23.

44

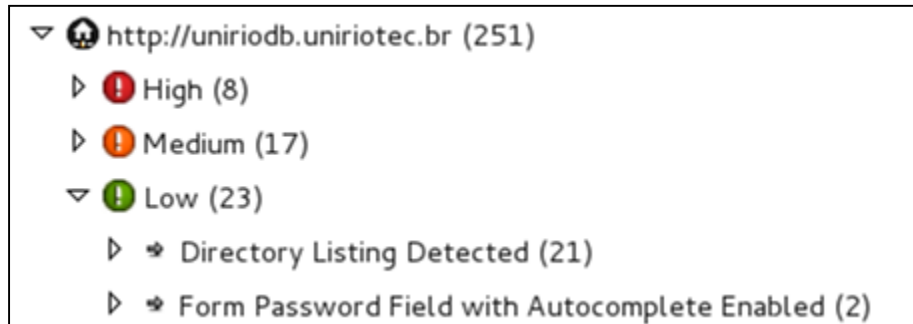


Figura 23: Índice de Vulnerabilidades Baixas

A exposição do índice de cada diretório nos códigos fonte da aplicação aumenta o grau de conhecimento do atacante sobre o sistema alvo, conforme relatado anteriormente. A Figura 21 apresenta o trecho do código HTML que contém o endereço do diretório pai do módulo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head>
    <title>Index of /lib/overlib</title>
  </head>
  <body>
    <h1>Index of /lib/overlib</h1>
    <ul><li><a href="/lib/"> Parent Directory</a></li>
    <li><a href="overlib.js"> overlib.js</a></li>
    <li><a href="overlib_anchor.js"> overlib_anchor.js</a></li>
    <li><a href="overlib_centerpopup.js"> overlib_centerpopup.js</a></li>
    <li><a href="ov...
```

Figura 24: Código HTML exibindo o índice do diretório

O recurso de auto completar habilitado é considerado como uma ameaça de segurança, pois se um indivíduo mal intencionado tem acesso ao computador da vítima, pode descobrir a senha do usuário mais facilmente. As informações sobre a vulnerabilidade baixa estão contidas na Figura 24.

<p>► REQUEST</p> <p>GET /login/index.php</p>
<p>► DISCUSSION</p> <p>Vega detected a form that included a password input field. The autocomplete attribute was not set to off. This may result in some browsers storing values input by users locally, where they may be retrieved by third parties.</p>
<p>► IMPACT</p> <ul style="list-style-type: none"> <li>&gt;&gt; A password value may be stored on the local filesystem of the client.</li> <li>&gt;&gt; Locally stored passwords could be retrieved by other users or malicious code.</li> </ul>
<p>► REMEDIATION</p> <ul style="list-style-type: none"> <li>&gt;&gt; The form declaration should have an autocomplete attribute with its value set to "off".</li> </ul>
<p>► REFERENCES</p> <p>Some additional links with relevant information published by third-parties:</p> <ul style="list-style-type: none"> <li>→ <a href="#">AUTOCOMPLETE attribute (MSDN)</a></li> <li>→ <a href="#">Using Autocomplete in HTML Forms (MSDN)</a></li> </ul>

Figura 25: Informações sobre a Vulnerabilidade *Form Password Field Autocomplete Enabled*

#### 4.1.3 Página do Moodle Uniriodb2

Esta seção apresenta a análise das vulnerabilidades encontradas na aplicação *web* Uniriodb2, endereçada em *uniriodb2.uniriotec.br*. Todas as vulnerabilidades encontradas estão indicadas na Figura 26.

<b>High</b>		(2 found)
Cleartext Password over HTTP	2	
<b>Medium</b>		(113 found)
Local Filesystem Paths Found	105	
HTTP Trace Support Detected	1	
Possible Source Code Disclosure	7	
<b>Low</b>		(140 found)
Directory Listing Detected	138	
Form Password Field with Autocomplete Enabled	2	

Figura 26: Índice de Vulnerabilidades Uniriodb2

#### 4.1.3.1 Vulnerabilidade Alta

A ferramenta encontrou apenas uma vulnerabilidade alta. Mesmo atualizando a versão da aplicação, as configurações sobre a criptografia durante a autenticação do usuário continuaram as mesmas, conforme exibido na Figura 27. A senha continua a ser transferida em “claro” e qualquer indivíduo que esteja capturando pacotes durante essa conexão, vai conseguir coletar a senha, conforme demonstrado na Figura 28.

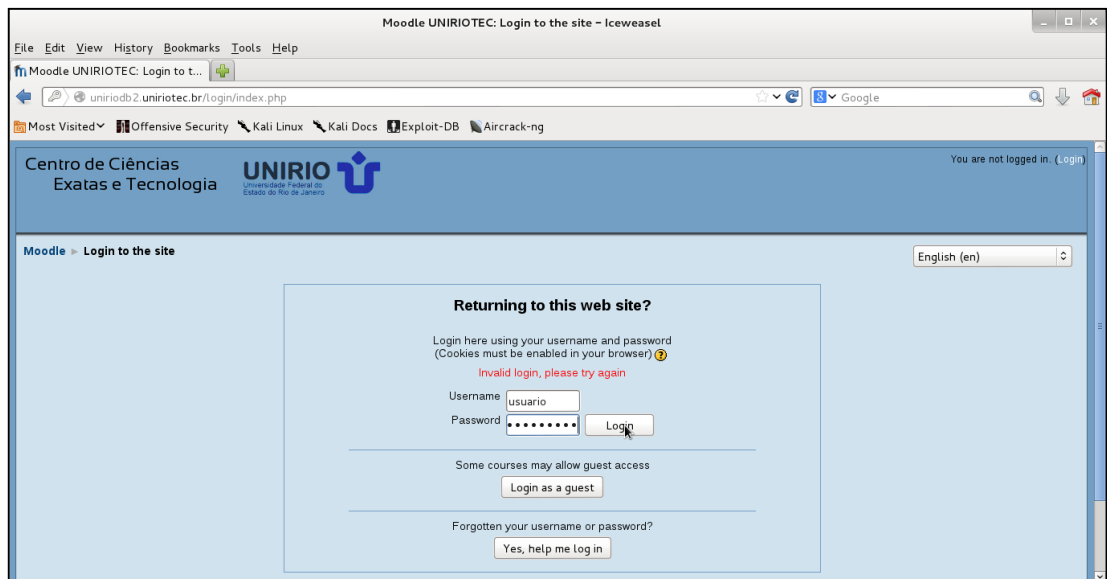


Figura 27: Tela de *login* da aplicação Moodle Uniriodb2

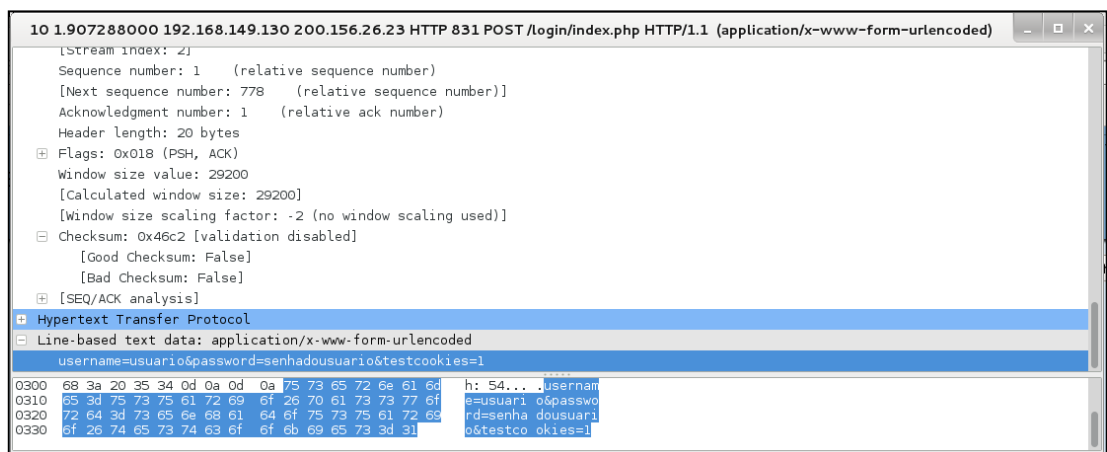


Figura 28: Captura do *Username* e *Senha*

A Figura 14, citada anteriormente, demonstra as mesmas informações que foram geradas pela ferramenta para a vulnerabilidade acima citada.

#### 4.1.3.2 Vulnerabilidades Médias

As mesmas três vulnerabilidades médias encontradas na varredura da aplicação Uniriodb, foram encontradas em Uniriodb2. A utilização do método HTTP TRACE,



exibição de diretórios do servidor de arquivos e exposição de assinaturas da linguagem de programação utilizada para criar a página. A Figura 29 exibe as vulnerabilidades médias encontradas.

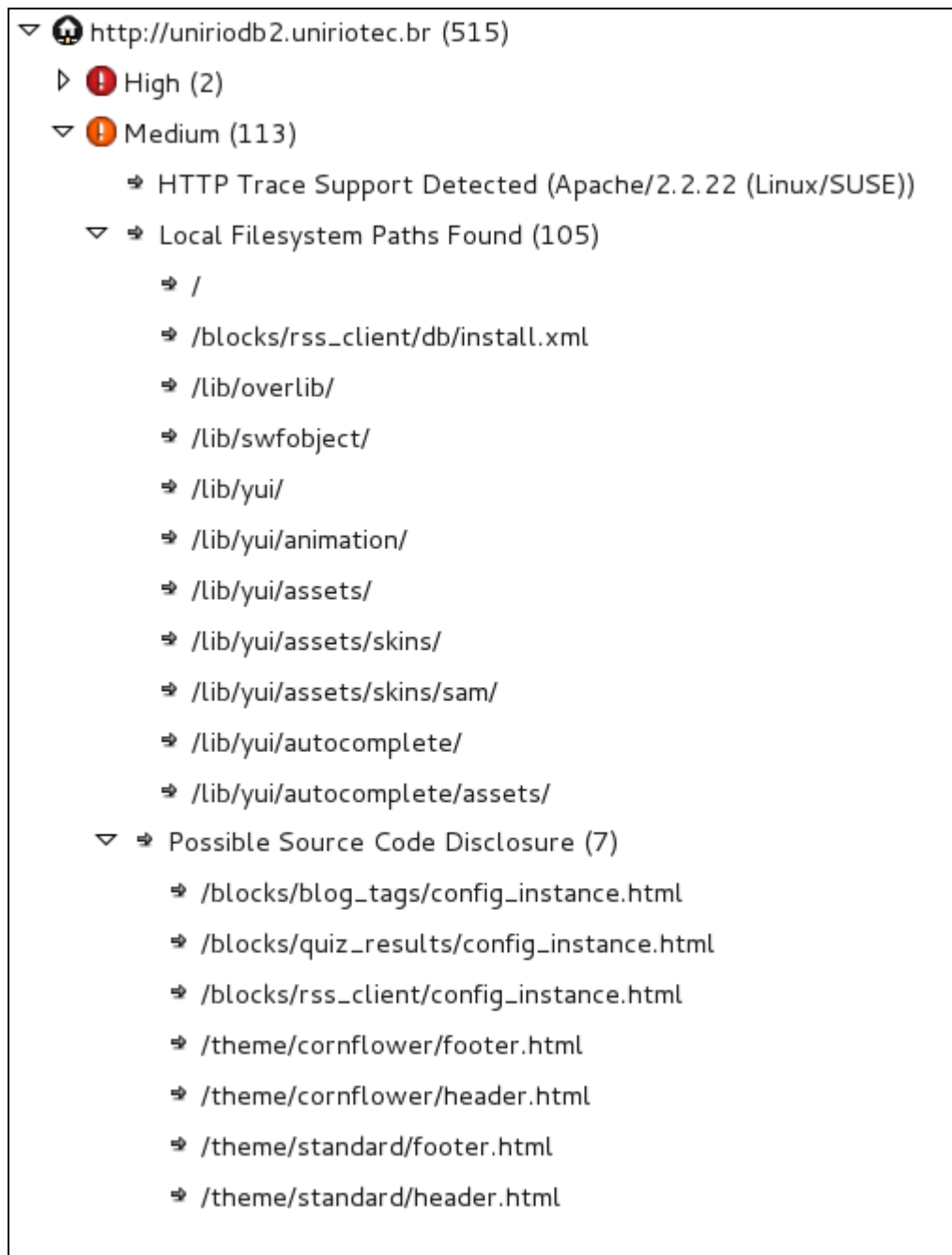


Figura 29: Índice Vulnerabilidades Médias Uniriodb2

Algumas diferenças podem ser notadas em relação à aplicação Uniriodb, a começar pela versão do Apache, que está mais atualizada, e da distribuição Sistema Operacional (SO) da aplicação. Também foram identificados endereços diferentes de diretórios, fato que pode ter haver com a mudança de distribuição do SO.

#### 4.1.3.3 Vulnerabilidades Baixas

A ferramenta indicou as mesmas duas vulnerabilidades baixas da aplicação Uniriodb, explicitadas anteriormente, conforme demonstrado pela Figura 30.

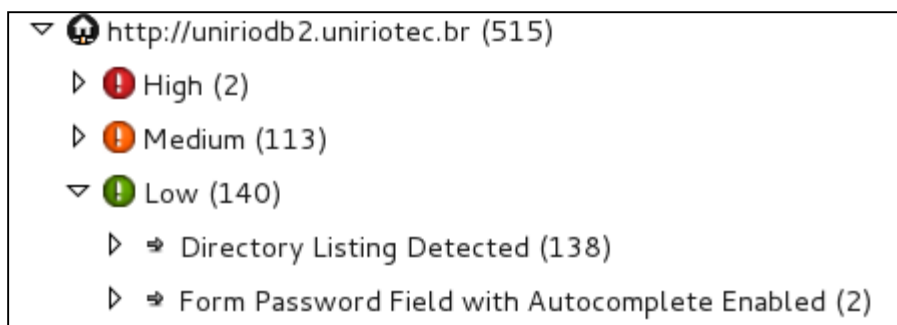


Figura 30: Vulnerabilidades baixas Uniriodb2

#### 4.1.4 Página Web do BSI

Esta seção apresenta os pontos vulneráveis da página *web* do curso Bacharelado em Sistemas de Informação da Unirio, encontrada pelo endereço: *bsi.uniriotec.br*. A ferramenta encontrou uma possível execução de XSS, o método HTTP TRACE habilitado e a listagem de índice de diretório, conforme a Figura 31 demonstra abaixo.

<b>High</b>	(104 found)
Cross-Site Script Include	104
<b>Medium</b>	(1 found)
HTTP Trace Support Detected	1
<b>Low</b>	(1 found)
Directory Listing Detected	1

Figura 31: Índice de vulnerabilidades da página do Bsi

#### 4.1.4.1 Vulnerabilidade Alta

A possível presença de um ataque *Cross-Site-Scripting* foi detectada pela ferramenta visto o código *JavaScript* que insere a barra de exibição do Governo Federal em cada diretório da página, apresentada na Figura 32. A existência de um código *JavaScript* na página de outro domínio, pode caracterizar um ataque de XSS, no entanto, esse caso pode ser considerado um falso positivo, visto que o código é proveniente de um domínio confiável.



Figura 32: Barra de exibição do Governo Federal

Como a barra de exibição está presente em todos os diretórios da página, a ferramenta marcou todos como possíveis focos de XSS, uma parte desses endereços

é exibida na Figura 33. A Figura 34 mostra as informações geradas pela ferramenta à respeito da vulnerabilidade, é válido notar o endereço de onde o código se encontra.

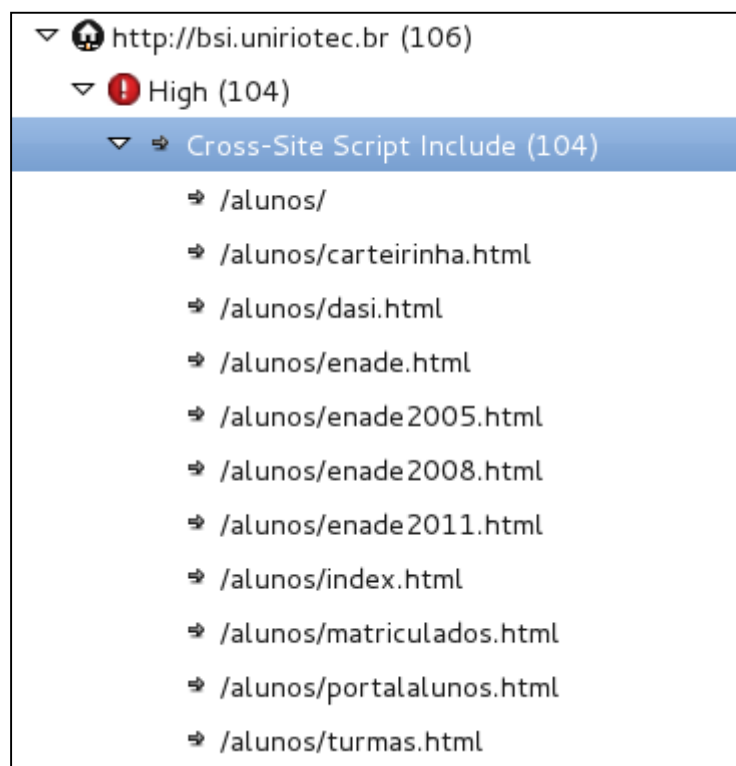


Figura 33: Focos de possíveis XSS

▶ REQUEST  
GET /alunos/

▶ RESOURCE CONTENT

```
Local domain: bsi.uniriotec.br
Script source: http://barra.brasil.gov.br/barra.js
```

▶ DISCUSSION

Vega detected that content on a server is including Javascript content from an unrelated domain. When this script code is fetched by a user browser and loaded into the DOM, it will have complete control over the DOM, bypassing the protection offered by the same-origin policy. Even if the source of the script code is trusted by the website operator, malicious code could be introduced if the server is ever compromised. It is strongly recommended that sensitive applications host all included Javascript locally.

▶ IMPACT

- >> Vega has detected that script code is being included from an unrelated domain.
- >> This gives the operator of the server where the code originates control over the DOM, and the web application .
- >> Even if the source is trusted, there are implications if the website hosting the script code is ever compromised.

▶ REMEDIATION

- >> Servers should host their own Javascript, especially for critical applications.

▶ REFERENCES

Some additional links with relevant information published by third-parties:

- [Wikipedia: Same-Origin Policy](#)

Figura 34: Informações sobre a vulnerabilidade XSS

#### 4.1.4.2 Vulnerabilidades Média e Baixa

Foram encontradas as mesmas vulnerabilidades média e baixa da aplicação Uniriodb, fato justificável visto que as duas aplicações *web* são hospedadas no mesmo servidor. A Figura 35 exibe as vulnerabilidades.

▼ http://bsi.uniriotec.br (106)

- ▶ High (104)
- ▼ Medium
  - ✦ HTTP Trace Support Detected (Apache/2.2.21 (FreeBSD) mod\_ssl/2.2.21 OpenSSL/0.9.8k DAV/2 PHP/5.3.9 with Suhosin-Patch)
- ▼ Low
  - ✦ Directory Listing Detected (/imagens/)

Figura 35: Índice de vulnerabilidades média e baixa da página do Bsi

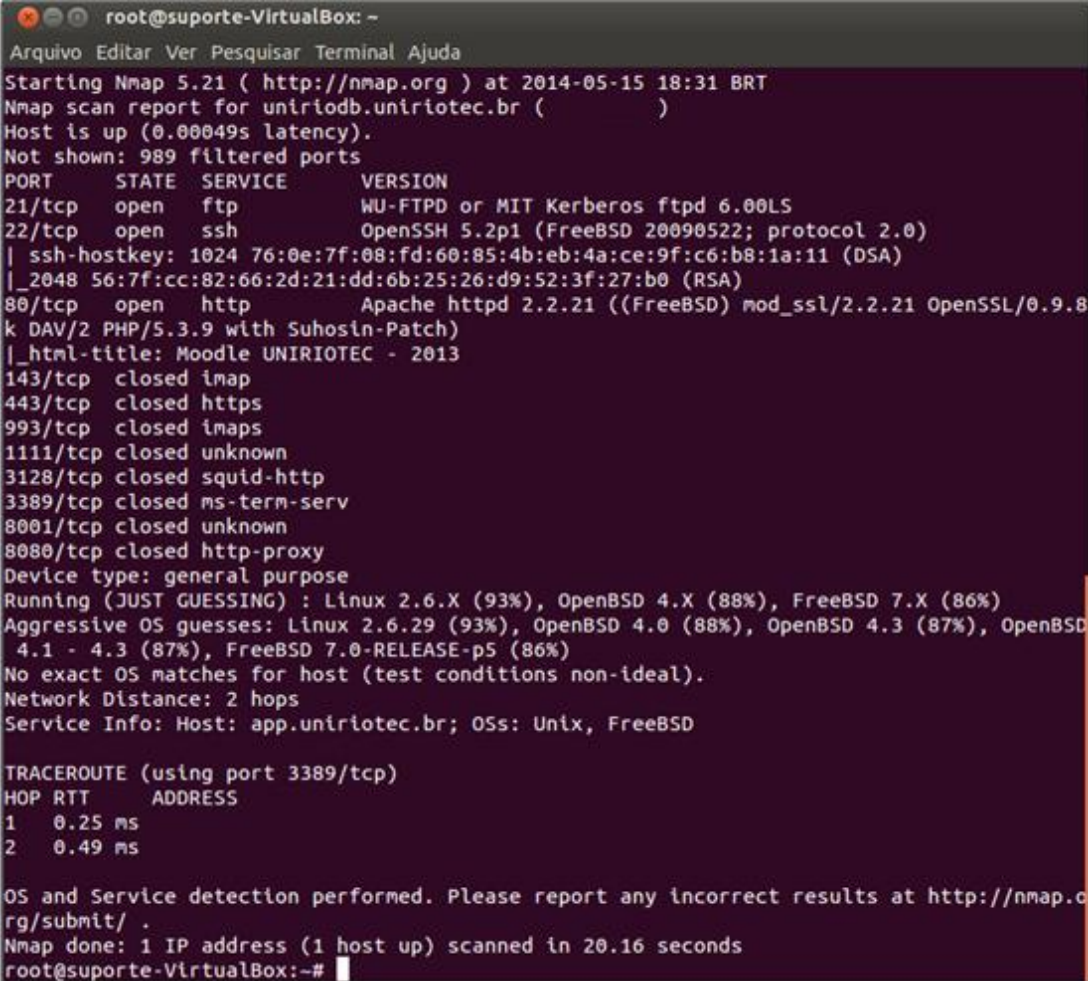
## 4.2 Vulnerabilidades em Servidores

Esta seção descreve a análise das vulnerabilidades descobertas nos servidores alvos deste teste.

Para um atacante, conforme dito em diversas vezes anteriormente, quanto mais informações reunidas sobre os sistemas alvos melhor. Sendo assim, antes de executar

as varreduras em busca de falhas de segurança, foi realizada a descoberta, através da ferramenta Nmap, dos sistemas operacionais dos servidores em questão, conforme as Figuras 36 e 37.

Esse escaneamento foi realizado a partir de uma máquina virtual com o sistema operacional Ubuntu, instalada numa máquina física do laboratório três. A instalação da ferramenta só foi possível de ser realizada pela elevação de privilégio do usuário *labccet* para o usuário *root*, conforme a Figura 38.



```
root@suporte-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
Starting Nmap 5.21 ( http://nmap.org ) at 2014-05-15 18:31 BRT
Nmap scan report for uniriodb.uniriotec.br ( )
Host is up (0.00049s latency).
Not shown: 989 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          WU-FTPD or MIT Kerberos ftpd 6.00LS
22/tcp    open  ssh          OpenSSH 5.2p1 (FreeBSD 20090522; protocol 2.0)
| ssh-hostkey: 1024 76:0e:7f:08:fd:60:85:4b:eb:4a:ce:9f:c6:b8:1a:11 (DSA)
|_ 2048 56:7f:cc:82:66:2d:21:dd:6b:25:26:d9:52:3f:27:b0 (RSA)
80/tcp    open  http         Apache httpd 2.2.21 ((FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8
k DAV/2 PHP/5.3.9 with Suhosin-Patch)
|_html-title: Moodle UNIRIOTEC - 2013
143/tcp   closed imap
443/tcp   closed https
993/tcp   closed imaps
1111/tcp  closed unknown
3128/tcp  closed squid-http
3389/tcp  closed ms-term-serv
8001/tcp  closed unknown
8080/tcp  closed http-proxy
Device type: general purpose
Running (JUST GUESSING) : Linux 2.6.X (93%), OpenBSD 4.X (88%), FreeBSD 7.X (86%)
Aggressive OS guesses: Linux 2.6.29 (93%), OpenBSD 4.0 (88%), OpenBSD 4.3 (87%), OpenBSD
4.1 - 4.3 (87%), FreeBSD 7.0-RELEASE-p5 (86%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: app.uniriotec.br; OSs: Unix, FreeBSD

TRACEROUTE (using port 3389/tcp)
HOP RTT ADDRESS
1 0.25 ms
2 0.49 ms

OS and Service detection performed. Please report any incorrect results at http://nmap.o
rg/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.16 seconds
root@suporte-VirtualBox:~#
```

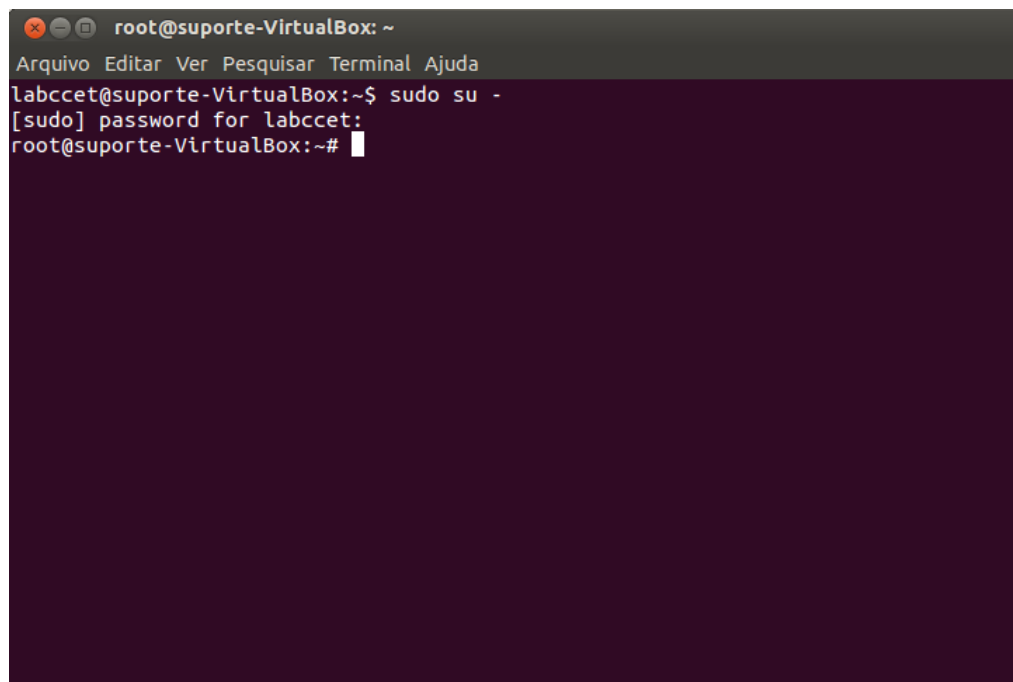
Figura 36: Nmap para a aplicação Moodle Uniriodb

```
root@suporte-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
Starting Nmap 5.21 ( http://nmap.org ) at 2014-05-15 18:29 BRT
Nmap scan report for uniriadb2.uniriotec.br ( )
Host is up (0.00065s latency).
Not shown: 989 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  tcpwrapped
22/tcp    open  ssh          OpenSSH 6.0 (protocol 2.0)
| ssh-hostkey: 1024 dc:e5:05:60:63:4a:b5:e8:50:6c:1f:61:85:76:9d:e7 (DSA)
|_ 2048 fe:ae:ec:d2:ed:32:19:54:70:78:92:e8:89:a9:58:82 (RSA)
80/tcp    open  http         Apache httpd 2.2.22 ((Linux/SUSE))
|_html-title: Moodle UNIRIOTEC
143/tcp   closed imap
443/tcp   closed https
993/tcp   closed imaps
1111/tcp  closed unknown
3128/tcp  closed squid-http
3389/tcp  closed ms-term-serv
8001/tcp  closed unknown
8080/tcp  closed http-proxy
Device type: general purpose
Running (JUST GUESSING) : Linux 2.6.X (95%), OpenBSD 4.X (90%), FreeBSD 7.X|6.X (87%)
Aggressive OS guesses: Linux 2.6.29 (95%), OpenBSD 4.0 (90%), OpenBSD 4.3 (90%),
FreeBSD 7.0-RELEASE-p5 (87%), FreeBSD 6.2-RELEASE (86%), OpenBSD 4.1 - 4.3 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using port 3389/tcp)
HOP RTT ADDRESS
1 0.47 ms
2 0.63 ms

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submt/ .
Nmap done: 1 IP address (1 host up) scanned in 15.82 seconds
```

Figura 37: Nmap para a aplicação Moodle Uniriadb2



```
root@suporte-VirtualBox: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
labccet@suporte-VirtualBox:~$ sudo su -  
[sudo] password for labccet:  
root@suporte-VirtualBox:~#
```

Figura 38: Elevação de privilégio

Com o conhecimento dos sistemas operacionais dos servidores analisados, foi possível refinar a busca por vulnerabilidades com a utilização de *plugins* específicos para sistemas Unix FreeBSD e OpenSuse, diminuindo assim, a sobrecarga no servidor bem como os resultados falsos positivos.

A ferramenta diagnosticou diversos pontos vulneráveis nos dois servidores analisados, mas cabe salientar, se a busca por vulnerabilidades fosse autenticada - a ferramenta iria se autenticar com uma conta de acesso privilegiado ao sistema – este número poderia ser muito maior, visto que seriam analisadas outras partes do sistema operacional.

A Tabela 3, especificada abaixo, apresenta as vulnerabilidades para cada servidor encontradas ordenadas pela sua classificação.



Vulnerabilidade	Classificação	Servidor(es)	Constatação	Correção
Unsupported Unix Operating System	Crítica	127.0.0.1	O servidor está rodando um sistema operacional obsoleto.	Atualizar o FreeBSD para alguma dessas versões 10.0 / 9.2 / 9.1 / 8.4.
OpenSSL Heartbeat Information Disclosure (Heartbleed)	Alta	127.0.0.2	O servidor é afetado com a vulnerabilidade Heartbleed.	Atualizar a versão do OpenSSL para 1.0.1g ou mais atual. Alternativamente, recompilar o o OpenSSL com a flag '-DOPENSSL_NO_HEARTBEATS' para desabilitar a funcionalidade que gera a vulnerabilidade.
SSL Certificate Cannot Be Trusted	Média	127.0.0.2	O certificado SSL não é confiável.	Comprar ou gerar um certificado próprio para este serviço.
SSL Medium Strength Cipher Suites Supported	Média	127.0.0.2	O serviço suporta algoritmos de criptografia de média complexidade.	Reconfigurar o serviço para utilizar criptografia de alta complexidade.
SSL Self-Signed Certificate	Média	127.0.0.2	O certificado do serviço não é reconhecido.	Comprar ou gerar um certificado próprio para este serviço.
HTTP TRACE / TRACK Methods Allowed	Média	127.0.0.1 / 127.0.0.2	O métodos Http Trace/ Track estão habilitados.	Desabilitar os métodos.
Apache HTTP Server httpOnly Cookie Information Disclosure	Média	127.0.0.1	O servidor expõe informações que podem comprometer o HttpOnly Cookie.	Atualizar a versão do Apache para 2.0.65 / 2.2.22 ou superior.
LDAP NULL BASE Search Access	Média	127.0.0.1	O serviço permite requisições LDAP utilizando bases nulas ou vazias.	Desabilitar queries baseadas em valores nulos ou vazios.
SSH Server CBC Mode Ciphers Enabled	Baixa	127.0.0.1 / 127.0.0.2	O serviço SSH do servidor está configurado com o Cipher Block Chaining.	Desabilitar o CBC e habilitar o CTR ou GSM módulos de criptografia. Referências: <a href="http://www.freebsd.org/cgi/man.cgi?query=ssh_config&amp;sektion=5">http://www.freebsd.org/cgi/man.cgi?query=ssh_config&amp;sektion=5</a> , <a href="http://en.opensuse.org/SDB:Configure_openSSH">http://en.opensuse.org/SDB:Configure_openSSH</a>
SSH Weak MAC Algorithms Enabled	Baixa	127.0.0.1 / 127.0.0.2	O serviço SSH está configurado para aceitar os algoritmos: MD5 e 96-bit MAC.	Desabilitar os algoritmos. Referências: <a href="http://www.freebsd.org/cgi/man.cgi?query=ssh_config&amp;sektion=5">http://www.freebsd.org/cgi/man.cgi?query=ssh_config&amp;sektion=5</a> , <a href="http://en.opensuse.org/SDB:Configure_openSSH">http://en.opensuse.org/SDB:Configure_openSSH</a>
FTP Supports Clear Text Authentication	Baixa	127.0.0.1	As credenciais de autenticação podem ser interceptadas.	Utilizar o SFTP (Módulo do SSH) ou FTPS (FTP sobre SSL/TLS).

Tabela 3: Vulnerabilidades dos servidores

### 4.3 Engenharia Social

Esta seção apresenta os resultados obtidos através da utilização da técnica de *Phishing* no teste de engenharia social.

A mensagem caracterizada como *Phishing*, foi enviada para o e-mail de cada funcionário que supostamente realiza tarefas envolvidas com a administração e manutenção da situação acadêmica dos alunos do Centro de Ciências Exatas e Tecnologia da Universidade Federal do Estado do Rio de Janeiro.

Após o envio das mensagens, a ferramenta alertou que uma mensagem não foi entregue devido à inexistência de um endereço de e-mail de um determinado funcionário. Sendo assim, a mensagem foi enviada novamente para este funcionário, considerando o endereço de e-mail correto.

Foi considerado um tempo de espera de quarenta e oito horas para a análise dos resultados. Com o fim do período de espera, foi constatado que nenhum funcionário acessou o link enviado no e-mail de *Phishing*, conforme demonstrado nas Figuras 39 e 40 abaixo.



Figura 39: Status do *Phishing*

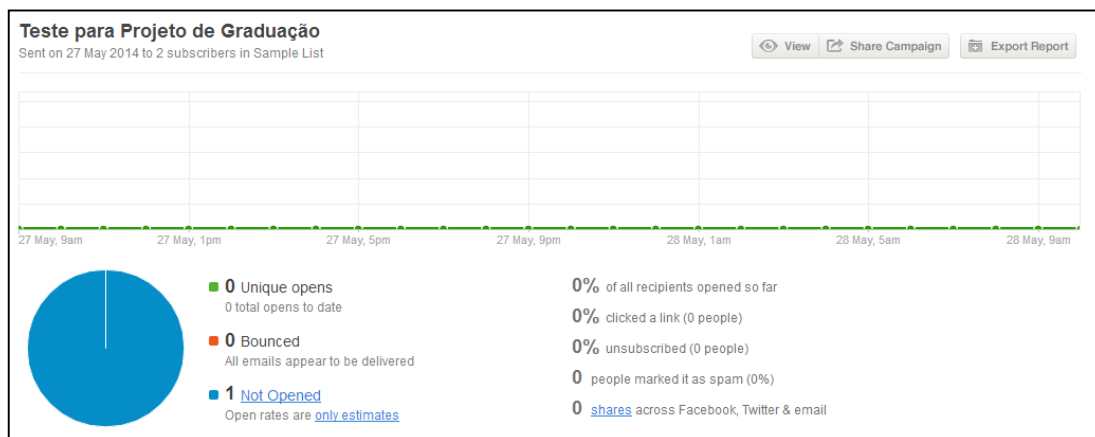


Figura 40: E-mail de *Phishing* não lido

Os resultados obtidos nessa etapa do trabalho foram considerados satisfatórios, pois os funcionários que abriram a mensagem se atentaram que esta não era verdadeira e não acessaram o link disponibilizado.

## 4.4 Recomendações

Esta seção apresenta as recomendações para a as correções das vulnerabilidades encontradas e baseadas nos controles definidos pelas normas ISO/IEC 27001 e 27002, e recomendações gerais de segurança para o ambiente.

### 4.4.1 Vulnerabilidades Encontradas

As vulnerabilidades encontradas nas aplicações *web* devem ser corrigidas na ordem de acordo com o seu nível de criticidade. Com isso, as vulnerabilidades altas devem ser corrigidas primeiramente, em seguida as médias e logo após, as baixas.

Como a aplicação Moodle Uniriodb não está sendo mais utilizada pelos professores e alunos do Centro de Ciências Exatas e tecnologia, é sugerido que a aplicação seja retirada do ar, a fim de eliminar as vulnerabilidades de *SQL Injection* e autenticação em sem conexão criptografada. Para a aplicação Moodle Uniriodb2, um estudo para utilização do protocolo HTTPS para a autenticação dos usuários pode ser realizado para eliminar a vulnerabilidade de autenticação mencionada acima. Conforme explicado anteriormente o item encontrado como possibilidade de ataque de *Cross-Site-Scripting* na página *web* do Bsi foi considerado um falso positivo e portanto, não há nenhuma manutenção a ser realizada.

Para a vulnerabilidade média do método HTTP TRACE habilitado, encontrada em todas as aplicações, sugere-se que este método seja desabilitado nos servidores, afim de eliminar a vulnerabilidade. A outra vulnerabilidade média encontrada, que diz respeito à exibição de diretórios locais do sistema de arquivos da aplicação, como a plataforma Moodle é de código aberto, sugere-se um novo ciclo de desenvolvimento para a correção da falha.

Para a correção da vulnerabilidade de exibição de índice de diretório no código da página, também é sugerida uma revisão no desenvolvimento da mesma. A

opção de auto completar no campo de senhas deve ser desabilitada na aplicação Uniriodb2, para que esse ponto seja corrigido.

As vulnerabilidades encontradas nos servidores devem ser tratadas em ordem sua ordem de criticidade, a resolução dos pontos críticos e altos deve ser realizada o mais breve possível, visto o alto grau de exposição que estas falhas de segurança representam. Para as vulnerabilidades médias referentes à certificados SSL, sugere-se uma análise mais detalhada sobre a utilização de certificados digitais. Deve ser verificada se realmente há utilização de consultas LDAP no servidor 127.0.0.1, caso não haja, o ponto deverá ser desconsiderado. As demais vulnerabilidades médias deverão ser corrigidas de acordo com o indicado pela ferramenta.

Os pontos baixos encontrados relacionados ao protocolo SSH, podem ser corrigidos facilmente nos arquivos de configuração de cada servidor. Para a vulnerabilidade de transferência via FTP, deve ser avaliada se este servidor recebe ou envia arquivos utilizando este protocolo.

#### **4.4.2 Baseadas nos controles da ISO/IEC 27001 e 27002**

De acordo com a norma ISO/IEC 27001 no ponto Auditorias internas do SGSI (6), a realização de uma auditoria em segurança da informação contribui positivamente para a gestão de segurança de um ambiente e deve ser realizada periodicamente. O programa de cada auditoria deve ser planejado, considerando o estado e importância dos processos e sistemas a serem auditados. Tendo em vista a utilização das recomendações desta norma, sugere-se que este tipo de trabalho seja realizado anualmente.

O ambiente de tecnologia da Uniriotec apresenta um grau interessante de práticas de segurança da informação e foram ser notados o cumprimento de alguns controles pertencentes a ISO/IEC 27002. Foi notado o ponto de Controles de entrada física (9.1.2), visto a restrição para acesso às salas de pesquisa e de servidores somente a pessoas autorizadas. A utilização de uma ferramenta de Antivírus nas máquinas dos laboratórios contempla a Proteção contra códigos maliciosos (10.4.1). Foi verificada

a Segregação de redes (11.4.5), onde há a separação das redes dos laboratórios, zona desmilitarizada (DMZ), das máquinas de pesquisa. O Isolamento de sistemas sensíveis (11.6.2) também foi notado visto que os sistemas expostos para a internet estão localizados apartados da rede interna.

No entanto, também foram verificadas algumas brechas de segurança que podem ser resolvidas ao implementar algumas recomendações desta norma.

O primeiro controle a ser analisado é a Política de segurança da informação (5). O departamento de Ciências Exatas e Tecnologia não possui uma política de segurança definida, e esse fato pode ser prejudicial à Instituição visto à grande gama de ameaças, sejam virtuais ou físicas, existentes. Convém a criação de um documento, baseado nas diretrizes especificadas pela norma ISO/IEC 27002, que declare o comprometimento da direção e estabeleça o enfoque da organização para gerenciar a segurança da informação.

O segundo controle é o de Segurança de Equipamentos (9.2), que tem como objetivo de impedir perdas, danos, furto ou roubo e interrupção das atividades da organização. Existe um *rack* no laboratório três, conforme a Figura 41, que tem diversos equipamentos de rede, este fica aberto e com a sua fonte de energia ligada a um estabilizador que pode ser desligado por qualquer pessoa que esteja no local, ou seja, fica totalmente exposto às características acima citadas. Dessa forma, se faz necessário um estudo para o isolamento dos equipamentos para que fiquem seguros.



Figura 41: Exposição dos Equipamentos

O terceiro controle que não é cumprido em parte, é o Gerenciamento da segurança em redes (10.6), que tem por objetivo garantir a proteção das informações e a proteção da infraestrutura. Não existem ferramentas implementadas para garantir a proteção das informações e da infraestrutura, excetuando-se o *firewall*. Existem diversas ferramentas de código aberto disponíveis para contribuir com o cumprimento do item acima citado. Por exemplo, para monitorar, alertar e bloquear comportamentos maliciosos na rede existe a ferramenta Snort, que funciona como um “farejador” em busca dessas anomalias. Outro controle que poderia ser feito seria a implantação de um filtro de conteúdo para o acesso à internet, isso impediria o acesso à sites com conteúdos maliciosos ou impróprios para a Instituição.

A quarta recomendação que não seguida é a Gestão de vulnerabilidades técnicas (12.6). Como a Instituição não possui uma política de segurança da informação, não existe um procedimento para realizar a gestão das vulnerabilidades dos sistemas do ambiente Uniriotec. O ideal é realizar uma varredura em busca de vulnerabilidades

trimestralmente em sistemas já em produção, e antes dos sistemas em homologação entrarem em produção.

O quinto controle não cumprido, é a Gestão de incidentes em segurança da informação (13). A não utilização desse ponto também deve-se ao fato da inexistência de uma política de segurança da informação que especifique o que se deve fazer antes, durante e depois de um incidente.

Apesar do resultado positivo no teste de engenharia social, sugere-se, em comprimento ao controle de Conscientização, educação e treinamento em segurança da informação (8.2.2), a confecção de uma cartilha contendo as principais orientações relacionadas à segurança da informação, como por exemplo, como se evitar um ataque de engenharia social.

#### **4.4.3 Gerais**

A elevação de privilégio do usuário *labccet* para o usuário *root* com a mesma senha utilizada para o *login* no sistema Ubuntu das máquinas virtuais, representa uma possibilidade para instalação de ferramentas de exploração de vulnerabilidades, mapeamento de redes e servidores no ambiente. Esse risco pode ser mitigado, caso a senha seja trocada por uma credencial mais forte.

Sugere-se como leitura pela equipe de Tecnologia da Informação do departamento, as instruções normativas de Segurança da Informação publicadas pelo Departamento de Segurança da Informação e Comunicações do Gabinete de Segurança Institucional da Presidência da República (Mais informações em: <http://dsic.planalto.gov.br/legislacao/dsic/23-dsic/legislacao/52-instrucoes-normativas>), visto que contém instruções relevantes sobre a Segurança da Informação em órgãos e entidades da Administração Pública Federal.

## 5. Conclusão

O teste de invasão, utilizado como métrica na auditoria de segurança da informação, realizado no ambiente Uniriotec foi composto por duas vertentes: a busca por vulnerabilidades em servidores e aplicações *web* e a realização de um ataque de engenharia social. Os resultados coletados de ambas vertentes serviram como base para a construção de uma percepção do nível de segurança da informação atual do Centro de Ciências Exatas e Tecnologia.

As correções para as vulnerabilidades encontradas devem ser realizadas o quanto antes para que o ambiente computacional Uniriotec fique com um nível de segurança satisfatório em suas aplicações e sistemas.

O teste de engenharia social teve um resultado considerado positivo, visto que nenhum dos colaboradores que abriram o e-mail falso clicou no endereço malicioso que o redirecionava para outra página. Esse fato é de grande importância, pois mesmo sem receberem nenhum tipo de treinamento contra este tipo de ataque, os colaboradores tiveram a sensibilidade de examinar o conteúdo do e-mail para verificar a sua autenticidade.

Tendo em vista os resultados obtidos através do Teste de Invasão, o ambiente computacional Uniriotec, incluindo os funcionários que ali desempenham suas tarefas, pode ser considerado com um nível intermediário de segurança, desde que as vulnerabilidades técnicas encontradas sejam corrigidas e as recomendações embasadas nas normas ISO/IEC 27001 e 27002 sejam seguidas.

De um modo geral, o trabalho atingiu os objetivos esperados, apresentando um panorama atual da segurança da informação no ambiente Uniriotec. Também é interessante ressaltar que este estudo possui certas limitações, visto que o escopo dos servidores e aplicações testados teve que ser reduzido, já que os testes tiveram que ser executados num curto intervalo de tempo.

Sugestões para trabalhos futuros envolvendo este estudo incluem a realização de outro teste de invasão, utilizando diferentes métodos para que sejam gerados



outros tipos de resultados. Também pode ser sugerido um estudo para a construção e implementação de política de segurança da informação do Centro de Ciências Exatas e Tecnologia.

## Referências Bibliográficas

- GIAVAROTO, SÍLVIO; SANTOS, GERSON. Backtrack Linux Auditoria e Teste De Invasão em Redes de Computadores. Rio de Janeiro: Editora Ciência Moderna Ltda.,2013.
- DIÓGENES, YURI; MAUSER, DANIEL. Certificação Security +: da prática para o exame syo-301. 2. Ed. Rio de Janeiro: Novaterra Editora e Distribuidora Ltda, 2013.
- ENGBRETSON, PATRICK. The Basics of hacking and penetration Testing: Ethical hacking and penetration Testing Made Easy. Elsevier Inc. Waltham, 2011.
- STROPARO, Elder. Invasões e as Vulnerabilidade de (BD). Jan, 2010. Disponível em:<<http://elderstroparo.blogspot.com.br/2010/01/invasoes-e-vulnerabilidade.html>>. Acesso em: 26 mai. 2014.
- FILHO, Antônio. Entendendo e Evitando a Engenharia Social: Protegendo Sistemas e Informações. Dez, 2004. Disponível em:<<http://www.espacoacademico.com.br/043/43amsf.htm>>. Acesso em: 26 mai. 2014.
- KARASINSKI, Lucas. Como identificar um ataque por *Phishing*. Ago, 2011. Disponível em: < <http://www.tecmundo.com.br/antivirus/12110-como-identificar-um-ataque-por-Phishing.htm>>. Acesso em: 26 mai. 2014.
- KOSUTIC, Dejan. Semelhanças e diferenças entre a ISO 27001 e a ISO 27002. Dez, 2010. Disponível em: < <http://blog.iso27001standard.com/pt-br/2010/12/19/iso-27001-vs-iso-27002-4/>>. Acesso em: 23 mai. 2014.