

LIGA DE ENSINO DO RIO GRANDE DO NORTE
CENTRO UNIVERSITÁRIO DO RIO GRANDE DO NORTE
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RANIERE MOURA DE OLIVEIRA

TESTES DE VULNERABILIDADE DE SISTEMAS EM LOJAS VIRTUAIS

NATAL/RN

2016

RANIERE MOURA DE OLIVEIRA

TESTES DE VULNERABILIDADE DE SISTEMAS EM LOJAS VIRTUAIS

Trabalho de Conclusão de Curso,
apresentado ao Centro Universitário do
Rio Grande do Norte como requisito final
para a obtenção do título de Bacharel em
Sistemas de Informação.

Orientador: Prof. M.Sc. Ytalo Rosendo do
Amaral.

NATAL/RN

2016

Catálogo na Publicação – Biblioteca do UNI-RN
Setor de Processos Técnicos

Oliveira, Raniere Moura de.

Testes de vulnerabilidade de sistemas em lojas virtuais / Raniere Moura de Oliveira. – Natal, 2016.
60 f.

Orientador: Prof. M.Sc. Ytalo Rosendo do Amaral.

Monografia (Graduação em Sistemas de Informação) – Centro Universitário do Rio Grande do Norte.

1. Comércio – Monografia. 2. Falha – Monografia. 3. Fraudes – Monografia. 4. Lojas virtuais – Monografia. 5. Segurança – Monografia. 6. Vulnerabilidade – Monografia. I. Amaral, Ytalo Rosendo do. II. Título.

RN/UNI-RN/BC

CDU 004.4

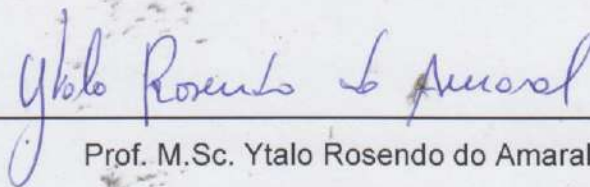
RANIERE MOURA DE OLIVEIRA

TESTES DE VULNERABILIDADE DE SISTEMAS EM LOJAS VIRTUAIS

Trabalho de Conclusão de Curso,
apresentado ao Centro Universitário do
Rio Grande do Norte como requisito final
para a obtenção do título de Bacharel em
Sistemas de Informação.

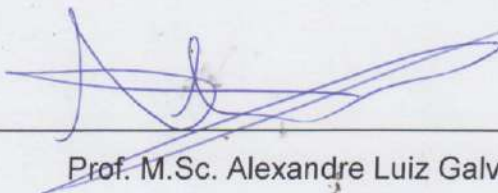
Aprovado em: ____/____/____.

BANCA EXAMINADORA



Prof. M.Sc. Ytalo Rosendo do Amaral

Orientador



Prof. M.Sc. Alexandre Luiz Galvão Damasceno

Membro

Prof. M.Sc. Maíra de Faria Barros Medeiros

Membro

Dedico este trabalho a minha mãe, que me concedeu a vida e jamais deixou de me apoiar e me incentivar nas escolhas que realizei até hoje.

AGRADECIMENTO

Agradeço infinitamente a Deus, pela força espiritual e por trilhar meus caminhos, fazendo-me chegar até aqui. Certamente, sem a sua presença em minha vida eu não teria capacidade para enfrentar os diferentes desafios que venci até então.

Agradeço aos meus familiares, que, mesmo distante, se fizeram presente no apoio e no incentivo em todos os anos que estive no Centro Universitário, principalmente nesta reta final. O estímulo de vocês foi fundamental para que eu não perdesse o foco em nenhum momento, durante o curso.

Agradeço imensamente ao Professor Ytalo Rosendo do Amaral, mestre que me guiou durante todo o desenvolvimento deste trabalho, orientando-me nos mais diversos ajustes em todas as etapas desta pesquisa.

Agradeço aos companheiros de trabalho, em especial a toda equipe da 5ª (quinta) Unidade do Serviço de Conformidade de Software da empresa DATAPREV (SCS5), que me auxiliaram, muito pacientemente, durante o desenvolvimento desse trabalho.

Agradeço a todos os amigos, principalmente aos que conviveram comigo ao longo desses quatro anos de graduação, etapa decisiva em minha vida.

Enfim, a todos que participaram de forma direta ou indireta, meus sinceros agradecimentos. Muito obrigado.

RESUMO

Em um mundo onde as pessoas estão cada vez mais conectadas à rede mundial de computadores, novas formas de comércio eletrônico têm surgido e se propagado rapidamente. No Brasil, estima-se que as lojas virtuais atuam com uma proporção de 80% desse variado mercado digital. Mesmo com essa representação massiva e as inegáveis vantagens desse mercado, alguns problemas podem ocorrer durante as transações virtuais, como fraudes e vazamento de informações pessoais, o que acarreta grande desconfiança de boa parte do crescente número de usuários da internet. Por esse motivo, despertou-se o interesse em analisar a segurança das lojas virtuais que mais vendem no país, testando a vulnerabilidade de seus sistemas a fim de encontrar possíveis falhas que possam servir como aberturas para ataques de hackers.

Palavras-chave: Comércio. Falha. Fraudes. Lojas virtuais. Segurança. Vulnerabilidade.

LISTA DE IMAGENS

Imagem 1 – Representação da Engenharia de Software	14
Imagem 2 – Representação da abordagem funcional	18
Imagem 3 – Princípios da Segurança da Informação	20
Imagem 4 – Exemplos de gráficos gerados pela E-bit.....	24
Imagem 5 – Representação referente à classificação das lojas	25
Imagem 6 – VVS – Configuração do proxy.....	28
Imagem 7 – VVS – Tela de análise	28
Imagem 8 – VVS – Análise de tráfego	29
Imagem 9 – Loja virtual Utilplast – Tela de login	32
Imagem 10 – Loja virtual Utilplast – Tela Minha Conta.....	32
Imagem 11 – Loja Submarino – Propriedades de segurança	33
Imagem 12 – Loja Paulinho Motos – Menu capacetes e sub menus.....	35
Imagem 13 – Loja Paulinho Motos – Escolha do produto.....	35
Imagem 14 – Loja Paulinho Motos – Carrinho de compras	37
Imagem 15 – Loja Utilplast – Tela inicial	38
Imagem 16 – Loja Utilplast – Resultado de busca por churrasqueira.....	38
Imagem 17 – Loja Utilplast – Minha conta	39
Imagem 18 – Resultado da análise de vulnerabilidades.....	41

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas
CPF – Cadastro de Pessoa Física
DOM – Document Object Model
HTML – Hyper Text Markup Language
HTTP – Hyper Text Transfer Protocol
HTTPS – Hyper Text Transfer Protocol Secure
IEEE – Institute of Electrical and Electronic Engineers
ISO – International Organization for Standardization
OWASP – The Open Web Application Security Project
RG – Registro Geral
SO – Sistema Operacional
SQL – Structured Query Language
URL – Uniform Resource Locator
VVS – Vega Vulnerability Scanner

SUMÁRIO

1 INTRODUÇÃO	11
1.1 OBJETIVOS	11
1.1.1 Objetivo Geral	11
1.1.2 Objetivos Específicos	12
2 REFERENCIAL TEÓRICO.....	13
2.1 HARDWARE E SOFTWARE.....	13
2.2 ENGENHARIA DE SOFTWARE	14
2.3 QUALIDADE DE SOFTWARE	14
2.4 TESTES DE SOFTWARE	15
2.5 ABORDAGENS SOBRE TESTES DE SOFTWARE	16
2.6 SEGURANÇA DA INFORMAÇÃO	18
3 METODOLOGIA	23
3.1 A ESCOLHA DAS LOJAS VIRTUAIS.....	23
3.2 A FERRAMENTA SCANNER.....	25
3.3 O BROWSER UTILIZADO	26
3.4 CONFIGURAÇÃO DA FERRAMENTA COM O BROWSER.....	26
3.5 TESTES FUNCIONAIS UTILIZADOS	29
4 RESULTADOS OBTIDOS	30
4.1 VULNERABILIDADES EM COMUM ENTRE OS SISTEMAS	30
4.1.1 Form Password Field Autocomplete Enable.....	30
4.1.2 HTML Authentication Over Unencrypted HTML	31
4.2 VULNERABILIDADES DA LOJA TOP-DIAMANTE (SUBMARINO)	32
4.2.1 E-mail Addresses Found	33
4.3 VULNERABILIDADES DA LOJA TOP-OURO (GIRAFÁ)	34
4.4 VULNERABILIDADES DA LOJA TOP-PRATA (PAULINHO MOTOS)	34
4.4.1 Local Filesystem Paths Found.....	34
4.4.2 Possible Source Code Disclosure	35
4.4.3 Integer Overflow	36
4.5 VULNERABILIDADES DA LOJA TOP-BRONZE (UTILPLAST)	37
4.5.1 Cross-site Scripting	37
4.5.2 Possible Credit Card Data, Social Insurance Number and Social Security Number Detected.....	39

4.5.3 SQL Injection	40
4.6 COMPARATIVO FINAL.....	40
5 CONCLUSÃO	42
REFERÊNCIAS.....	43
APÊNDICES	45

1 INTRODUÇÃO

Hoje em dia não há como imaginar a vida das pessoas sem o uso da internet. Ela, de fato, se tornou uma dependência mundial pelas inegáveis vantagens que traz aos seus usuários. As notícias se espalham de maneira absurdamente rápida, as pessoas facilmente se comunicam de qualquer lugar do planeta através das redes sociais, realizam transações bancárias sem sair do sofá da sala e movimentam um comércio eletrônico que aquece a economia de países no mundo inteiro.

Diversas empresas estão modificando a forma de vender, migrando suas lojas físicas para ambientes virtuais, onde é possível encontrar e estudar seus clientes potenciais mais facilmente. Entretanto, ao levar um comércio para o mundo virtual, deve-se lembrar de que lá, os perigos existentes no mundo real também estão presentes, como os riscos de vazamento de informações pessoais, fraudes, roubos de identidade, dentre outros, em uma complexidade ainda maior devido à crescente no número de usuários da rede mundial de computadores. As empresas necessitam investir fortemente em segurança da informação para garantir a integridade e a *confidencialidade dos dados de seus usuários*. Mas será que as empresas estão realmente disponibilizando sistemas seguros?

Este trabalho visa analisar a segurança da informação nas lojas virtuais, dentre as que mais vendem no mercado brasileiro, através da realização de testes de vulnerabilidade em seus sistemas, a fim de encontrar as prováveis debilidades existentes.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Ao observar a importância e a complexidade do tema segurança da informação, viu-se a necessidade de analisar a vulnerabilidade das lojas virtuais devido ao número crescente desse negócio no mercado brasileiro e da quantidade de usuários conectados à internet hoje em dia. Apesar do grande volume de vendas que acontece diariamente, esses usuários ainda possuem grande desconfiança em ter que disponibilizar seus dados pessoais na web.

1.1.2 Objetivos Específicos

Tomar conhecimento sobre as vulnerabilidades existentes nas lojas virtuais do nosso mercado é fundamental para que sejam compreendidos os riscos viventes.

Este trabalho tem como objetivos específicos:

- Mostrar as vulnerabilidades encontradas;
- Explicar de onde cada vulnerabilidade foi coletada;
- Tornar público em quais lojas essas vulnerabilidades foram encontradas.

2 REFERENCIAL TEÓRICO

Neste ponto será dado início às considerações sobre computação e engenharia de software, focalizando a qualidade de software e o entendimento sobre segurança da informação e testes de vulnerabilidade de sistemas.

2.1 HARDWARE E SOFTWARE

Pode-se entender como computador toda e qualquer máquina eletrônica capaz de realizar processamento de dados. Através de circuitos internos integrados, juntamente com outros componentes dispostos e interligados em sua arquitetura, denominada hardware (sua parte física), a máquina realiza cálculos matemáticos que possibilita a execução de uma grande quantidade de tarefas solicitadas pelo seu usuário. Essas solicitações se dão através de programas para computadores que, na verdade, são facilitadores da comunicação homem-máquina para realização de tarefas. Os softwares.

De maneira simples, software se define como sendo uma sequência de instruções escritas, através de regras sintáticas e semânticas a serem interpretadas pela máquina, com o objetivo de executar tarefas específicas (ÁVILA, 2013). Embora sejam utilizados em sua produção outros utensílios para definir como será a sua versão terminante, como por exemplo, os protótipos de tela que exemplificam como o design do software devem ficar, instruções escritas são realmente o seu resultado final. Também conhecido como a parte lógica de um computador, este fornece os dados necessários para o processamento e manipulação da informação.

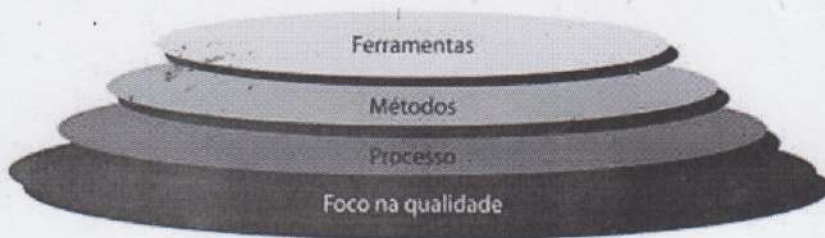
Em uma visão mais ampla, para Sommerville (2003), software é um conjunto de artefatos e não apenas o seu código fonte (instruções escritas). Pressman (2006) reforça esta afirmativa quando define que um software é um conjunto composto por algoritmos de computador, estruturas de dados e documentos. Um artefato de software é o resultado de uma ou um conjunto de atividades dentro do contexto do desenvolvimento de um sistema. Sendo assim a modelagem, a diagramação e os protótipos de tela utilizados durante o processo de desenvolvimento são exemplos de artefatos de software.

2.2 ENGENHARIA DE SOFTWARE

Segundo o IEEE (1993), Engenharia de software é a aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, operação e manutenção de software. Bauer apud Pressman (2011, p.39), diz que a “Engenharia de Software é a criação e o emprego de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em máquinas reais”.

Quando se deu início a trajetória do desenvolvimento de software em todo o mundo, em meados da década de 60, com ela também surgiu a engenharia que comanda todos os processos, métodos e ferramentas de sua produção: a engenharia de software. Esta nasce com o intuito de organizar e padronizar a forma na qual todas as empresas e instituições desenvolvedoras de software devem caminhar com sua linha de produção, visando melhorar o escopo do projeto e do produto, diminuir o tempo e o custo de sua produção e entregar softwares de qualidade dentro dos prazos estabelecidos aos seus clientes (PRESSMAN, 2011). A imagem 1 exibe a pirâmide representativa da engenharia de software, na qual mostram os três itens anteriormente citados, tendo como base sustentadora para todos eles, o foco na qualidade.

Imagem 1 – Representação da Engenharia de Software.



Fonte: Pressman, 2011.

2.3 QUALIDADE DE SOFTWARE

No final dos anos 80, houve-se o fortalecimento de uma abordagem acerca de qualidade, onde aponta a melhoria contínua dos processos de produção como sendo a única forma de garantir qualidade em produtos e/ou serviços. O reforço desta abordagem veio a concretizar o já estabelecido anteriormente pela engenharia

de software, que trabalha no controle de sua produção, por meio de processos, afim de eliminar os elementos geradores de não conformidades em seu produto, obtendo assim, qualidade de software.

Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais (PRESSMAN, 2006, p.58).

Entende-se como processo, um conjunto de atividades que recebem entradas, transforma-as em produtos e/ou serviços que atendem as necessidades reais dos usuários finais, com agregação de valor. Sommerville (2003, p. 7) refere processo de software como "um conjunto de atividades e resultados associados que geram um produto de software". Internamente, as atividades no âmbito do desenvolvimento de sistemas devem ser padronizadas e processos são utilizados para reger todas as etapas dessa produção.

Focada no processo padrão posto por cada empresa desenvolvedora de sistemas, a engenharia, em qualidade de software, busca verificar se o que está sendo criado pela mesma está em conformidade com o processo preestabelecido. Visa à garantia de qualidade, que consiste nas funções de auditar e descrever sobre os artefatos do software e o controle de qualidade, que envolve uma série de inspeções, revisões e testes usados ao longo do desenvolvimento para garantir que o produto de software satisfaça os requisitos estabelecidos, consumindo o mínimo de recurso da organização. Sommerville (2003) faz referência à garantia de qualidade como sendo o estabelecimento de processos que direcionam o desenvolvimento de sistemas para a alta qualidade. Ainda ele, sobre controle de qualidade, diz que este se dá com a aprovação na utilização dos processos que garantem esta alta qualidade.

2.4 TESTES DE SOFTWARE

Os testes de qualidade são realizados a todo o momento durante a produção e/ou manutenção do software como uma forma de assegurar que o que foi requisitado pelo cliente está sendo desenvolvido da forma correta. O processo de teste de software está voltado para a obtenção de qualidade no produto de software

que ao longo de sua criação, sofre mudanças à medida que avançam as atividades (RIOS; MOREIRA, 2013).

Teste de software é um processo de execução de um software para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. "Seu objetivo é revelar falhas em um produto para que as causas dessas falhas sejam apresentadas e identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final (DIAS NETO; 2008, p. 54).

A engenharia, em testes de software, busca os melhores meios de se verificar se o software em criação se adequa ao que o cliente espera dele. Para isso, ela cria e gerencia os diferentes tipos de testes que podem ser realizados ao longo da trajetória de produção. Estes tipos de testes podem ser executados em três abordagens: a abordagem estrutural, popularmente conhecida como testes caixa branca; a abordagem funcional, também conhecida como testes caixa preta ou simplesmente testes funcionais; ou ainda a abordagem mista, também conhecida como testes caixa cinza. Essas abordagens definem o interesse e a profundidade dos testes, de acordo com a necessidade do projeto (PIQUEIRO; ZANDRA, 2015).

2.5 ABORDAGENS SOBRE TESTES DE SOFTWARE

A abordagem estrutural ou teste caixa branca utiliza o conhecimento das estruturas internas do software para gerar e aplicar testes, de maneira a avaliar os caminhos lógicos da aplicação. Essa abordagem visa garantir que os caminhos lógicos do sistema foram testados pelo menos uma vez e para isso, o testador deve ter conhecimento sobre o código fonte da aplicação. São testadas nesta abordagem as estruturas internas de repetição, seleção e tomadas de decisão. Pressman (apud PIQUEIRO; ZANDRA, 2015, p. 13) afirma que: esta abordagem verifica se o código atende aos padrões da empresa e se não há erros de codificação.

A abordagem funcional ou testes caixa preta baseia-se em pré-condições e pós-condições, ou seja, o testador sabe exatamente o que ele irá inserir como dado de entrada e a saída esperada por parte do sistema. Nessa abordagem o testador não possui conhecimento sobre o código fonte da aplicação, tendo contato somente com a sua interface visual. Por este motivo é conhecido como "caixa preta". Sendo assim, o testador faz basicamente o papel do usuário final. Nessa abordagem o

teste se baseia no que foi especificado no levantamento de requisitos, testando as funcionalidades criadas através de um conjunto de situações de testes (PIQUEIRO; ZANDRA, 2015).

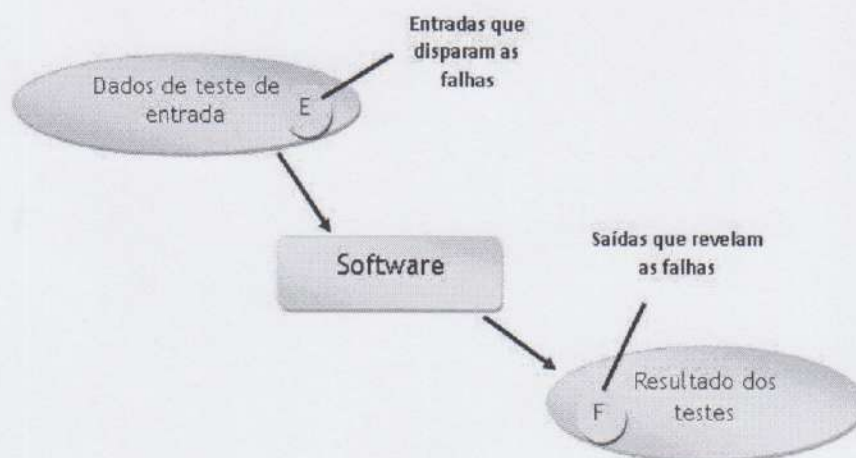
Esta abordagem busca verificar e encontrar erros nas seguintes categorias: erros de inicialização e término de operações no sistema; erros de comportamento; erros na interface do sistema; erros em funções incorretas. A respeito dos resultados dos testes funcionais, Pressman (apud DIAS NETO, 2008, p. 58) afirma que:

Haverá sucesso no teste se o resultado obtido for igual ao esperado. O componente de software a ser testado pode ser um método, uma função interna, um programa, um componente, um conjunto de programas e/ou componentes ou mesmo uma funcionalidade. A técnica de teste funcional é aplicável a todos os níveis de testes.

Há também uma nova abordagem que vem sendo cada vez mais utilizada pelos desenvolvedores de software, a abordagem mista. Também conhecida como testes caixa cinza, essa abordagem é um “meio termo” entre as duas abordagens anteriormente citadas. Se por um lado as abordagens estrutural e funcional se diferenciam principalmente pelo fato do testador ter ou não ter o conhecimento dos caminhos lógicos internos da aplicação (código fonte), na abordagem mista o testador possui o conhecimento do código fonte, mas realiza os testes caixa preta, podendo assim utilizar a engenharia reversa, ou seja, identificar um erro na parte externa do software, através da interface já codificada e indicar as possíveis soluções. Para Stefanelli (2012), a engenharia reversa consiste em examinar um produto já existente, tentando entender como o mesmo funciona e se comporta.

Embora essa cultura ainda receba certa resistência, principalmente em pequenas e médias empresas desenvolvedoras de software, os ambientes de testes vêm se tornando parte integrante dos seus processos de qualidade. Pressman (2011) afirma que essas atividades consomem cerca de 40% do tempo e do custo do desenvolvimento. Por este motivo, é comum a execução da abordagem funcional como principal forma de testar as aplicações em desenvolvimento, uma vez que esta abordagem se aplica na interface em que o usuário final irá atuar. Para Lestin (2008) essa é a única, rápida e mais realista maneira de retorno sobre o investimento. A imagem 2 exemplifica como funcionam os testes caixa preta, mostrando o momento de inserção dos dados por parte do testador, o processamento de dados no sistema e o disparo da falha.

Imagem 2 – Representação da abordagem funcional



Fonte: iTestes, 2013.

2.6 SEGURANÇA DA INFORMAÇÃO

A informação é um fator essencial para o sucesso ou fracasso de uma instituição. Ela é o principal ativo no mundo corporativo e é exatamente por ser essencial que ela traz consigo uma necessidade básica: segurança. A ABNT indica que:

A informação é um ativo que, como qualquer outro ativo importante, é essencial para os negócios de uma organização e consequentemente 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 crescente número e a uma grande variedade de ameaças e vulnerabilidades (ASSOCIAÇÃO..., 2005, p. 16).

A segurança da Informação está diretamente ligada à preservação de dados importantes para qualquer negócio, nas mais variadas maneiras possíveis e deve-se implementar uma forma de proteção antes que um incidente ocorra. Essa proteção, segundo Tavares, Carvalho e Botelho (2003), são medidas a serem seguidas que tornam o software capaz de inspecionar, detectar e reagir a ameaças, limitando e reduzindo assim seus impactos.

Tecnologicamente, embora possua predominantemente um caráter reativo, ou seja, verifica primeira a existência de um problema, por exemplo, uma invasão, para depois buscar a saída, como correção das vulnerabilidades, a segurança da informação, ainda assim deve possuir medidas para prevenir sobre os perigos que

rodeiam as aplicações. Sobre a prevenção de sistemas de informação, Tavares, Carvalho e Botelho (2003) ponderam que é um conjunto de medidas que tende a diminuir a possibilidade de consolidação dos riscos existentes.

Está cada vez mais necessário fazer uso de mecanismos que garantam a segurança dos dados de usuários nos softwares, principalmente em sistemas web. Durante o processo de produção de um software, testes de segurança devem ser realizados para garantir que o sistema criado possui um ótimo nível de seguridade, explorando suas possíveis vulnerabilidades. Basicamente, um teste de segurança consiste em verificar no sistema se os dados são acessados somente por pessoas autorizadas, de maneira segura e que guarde a responsabilidade dos acessos (ELIZA; LAGARES, 2013). Apesar disso, garantir que um sistema está 100% (cem por cento) seguro é arriscado.

Com o aumento significativo dos usuários de internet, as pessoas estão cada vez mais expostas a riscos de segurança. Acessar conteúdos impróprios, conversar com pessoas mal-intencionadas, invasão de privacidade e roubo de identidade são exemplos de perigos que rodeiam a web. Estes perigos se dão devido à existência de sistemas vulneráveis e de pessoas com conhecimentos avançados em informática que se aproveita dessas vulnerabilidades. A cerca disso, a cartilha de segurança para internet diz que:

Vulnerabilidade pode ser definida como uma condição que, quando explorada por um atacante, pode resultar em uma violação de segurança. Exemplos de vulnerabilidade são falhas no projeto, na implementação ou configuração de programas, serviços ou equipamentos de rede (COMITÊ..., 2012, p. 18).

A cartilha de segurança para internet (COMITÊ..., 2012, p. 34) indica ainda que “um ataque de exploração de vulnerabilidade ocorre quando o atacante, utilizando-se de uma vulnerabilidade, tenta executar ações maliciosas”. Esse atacante, denominado “hacker” pelas mídias populares como programas de televisão, rádios e jornais, tenta ferir os princípios da segurança da informação, na qual são ilustrados na imagem 3.

Imagem 3 – Princípios da Segurança da Informação



Fonte: Mod. Solutions, 2014.

Sobre disponibilidade, Tavares, Carvalho e Botelho (2003) indica que os dados de um negócio devem estar disponíveis, a qualquer momento, para pessoas autorizadas. Acrescentam ainda, sobre integridade, que a segurança da informação deve manter esses dados corretos, independente de quaisquer fatores, sejam humanos, físicos ou naturais e ainda completam o entendimento sobre esses princípios informando que os dados precisam se manter confidenciais, devendo o sistema fazer distinção para usuários não permitidos.

A palavra hacker deriva do verbo inglês "to hack", que significa "retalhar". Ela é utilizada para explicitar as pessoas que fazem uso de diversas "gambiarras" em computadores para tirar proveitos indevidos. Segundo Assunção (2002), a palavra hacker é utilizada para diferenciar o indivíduo que se dedica, com intensidade fora do normal, a conhecer profundamente determinados âmbitos da informática. Embora pareça incomum, é possível trabalhar de forma profissional com o conhecimento hacker, tentando encontrar aberturas para a execução de ações maliciosas nos sistemas criados pelas empresas desenvolvedoras, através da execução de testes funcionais, já citados anteriormente. Isso não significa dizer que somente um indivíduo com conhecimento hacker é capaz de trabalhar com segurança da informação, mas ter essa ciência já facilita, e muito, o caminho a seguir.

Dentre as mais variadas formas de explorar e atacar aos sistemas virtuais, algumas são praticadas e utilizadas com maior frequência pelos "hackers" em toda web. Para Assunção (2002), a ideia geral é tentar conhecer as estruturas internas de

um sistema através de suas possíveis vulnerabilidades. Caso esse objetivo não seja alcançado, diminui-se o nível de exploração, tentando obter sempre alguma informação valiosa que possa ser utilizada posteriormente pelo atacante.

A coleta de endereços eletrônicos, por exemplo, é um passo fundamental para os que desejam realizar ataques de forma massiva. Isso faz do e-mail algo essencial para os atacantes. Uma vulnerabilidade é constatada nessa classe quando o sistema permite que aplicações adicionais vasculhem o campo de informação de e-mail do seu usuário. Para a OWASP (2013), comunidade livre que foca seus estudos para a melhoria na segurança de aplicações, essa vulnerabilidade se dá através da ação de "Spam-bots", uma espécie de extrator de e-mails onde sua função é buscar em sistemas web, campos onde a estrutura de um e-mail é obrigatória. Sua missão é colher o endereço eletrônico dos usuários de sistemas web e gerar uma própria base de dados que, mais tarde, tornará possível a realização de ataques de forma massiva por parte de seus controladores: os "hackers".

Outra conhecida vulnerabilidade em sistemas informatizados é o Cross-site Scripting, ou simplesmente script cruzado. Essa forma de exploração se dá quando, na navegação web pode-se implementar uma subversão de controles de segurança na aplicação. Segundo a OWASP (2013), ataques dessa classe ocorrem quando uma falta de validação de entrada permite que usuários injetem código de script em um sistema alvo, de tal maneira que esse código seja executado no navegador de outro usuário que está acessando a mesma aplicação, ou seja, vulnerável, o sistema reconhece o script como nativo. Desta forma, o código malicioso contido no script pode manipular o documento HTML e alterar as características oficiais da página acessada bem como manipular ou adicionar funções ao sistema e executar ações dentro da aplicação sem que o usuário perceba.

Os exemplos de exploração de sistemas anteriormente citados são bastante utilizados, pois trazem consigo a possibilidade de uso dos dados coletados ou manipulados posteriormente. Porém, não permitem conhecer e manipular as informações sobre o negócio de um sistema alvo. Conhecida como SQL Injection, essa vulnerabilidade se dá em sistemas que "confiam" no que o usuário irá informar como dado de entrada. Dessa forma, pode-se enviar comandos para a base de dados da aplicação através dos campos de formulários ou URLs e assim conseguir manipular o negócio da organização. Sobre o SQL Injection, a OWASP indica que:

Vulnerabilidades desta natureza ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando de consulta. [...] Os dados enviados pelo atacante podem iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados (OWASP; 2013, p. 7).

SQL é uma linguagem de gerenciamento de banco de dados que permite ao desenvolvedor criar alocações e manipular informações pertencentes ao negócio do cliente (CARVALHO; PIRES, 2015). Conhecer essa linguagem é fundamental para que possa ser estruturado um banco de dados que atenda aos princípios da segurança da informação. Conforme Carvalho e Pires (2015), a vulnerabilidade de Injeção ocorre quando o sistema aceita que um usuário comum, que não tem permissão para realizar ações ao banco de dados da aplicação, através de campos de preenchimento, realize instruções de SQL. Isso possibilita que o atacante tenha privilégios especiais ao sistema, como por exemplo, consultar ao seu banco de dados, adicionar novas tabelas a até destruir as já existentes, ferindo totalmente os princípios da segurança da informação.

São várias as formas de tentar burlar os sistemas em busca de dados protegidos para adquirir informações privilegiadas. Além dos exemplos de vulnerabilidades anteriormente citadas, existe uma infinidade de outros tipos, algumas ainda nem catalogadas. Diversos ataques tentam se aproveitar de defeitos internos do sistema para explorar seus caminhos lógicos. Por exemplo, inserir dados propositalmente errados tentando gerar falhas na aplicação que mostrem os caminhos internos do software. Para a OWASP (2013), sistemas que contemplam esse tipo de vulnerabilidade trazem possibilidade de o atacante conhecer o desenho arquitetural da aplicação, onde, em um tratamento de exceção não realizado corretamente no momento do desenvolvimento, torna possível descobrir os caminhos internos do sistema. Isso acontece frequentemente em sistemas criados com linguagens de programação fracas que misturam código fonte com elementos de interface e design em uma mesma camada.

3 METODOLOGIA

3.1 A ESCOLHA DAS LOJAS VIRTUAIS

A análise se classifica como exploratória e descritiva por atuar na avaliação da vulnerabilidade de sistema em lojas virtuais do mercado brasileiro. Para isso foi necessário escolher, dentre tantas, quais lojas do comércio eletrônico nacional seriam alvo desta avaliação. Em seguida, definir quais os instrumentos e métodos utilizar.

O primeiro e mais importante passo foi decidir quais lojas virtuais seriam avaliadas. Através da internet, por meio de pesquisa, encontrou-se um site que é considerado uma referência nacional sobre o assunto, por ser mantenedor das mais variadas informações sobre o nosso comércio eletrônico: o site da E-bit. Nele, podem-se encontrar tendências, informações financeiras, balanços comerciais, hábitos e costumes, dentre outras informações que auxiliem os empresários e consumidores a enxergar como está e como andar o comércio virtual brasileiro. A E-bit é “uma empresa nacional que objetiva categorizar as lojas virtuais baseado na avaliação dos próprios clientes” (LOJA..., 2013). A imagem 4 possui exemplos de gráficos existentes no sistema E-bit e que são acessíveis aos interessados, informando em porcentagem (%) alguns dados que são relevantes ao comércio virtual como variações de preços, facilidade de compras, entregas no prazo, etc.

Imagem 4 – Exemplos de gráficos gerados pela E-bit



Fonte: Loja..., 2013.

O cliente, ao comprar em um e-commerce cadastrado na E-bit e ao receber seu produto em casa, responde a pequenas pesquisas que visam colher dados sobre sua satisfação no processo de realização da compra, cumprimento dos prazos de entrega e satisfação com o produto comprado. O resultado dessas pesquisas agregado à quantidade total de vendas anual gera uma espécie de ranking no qual é possível especificar quais as melhores lojas virtuais brasileiras. Anualmente, baseada nessa experiência de uso dos clientes, a empresa E-bit classifica as lojas virtuais em quatro categorias: diamante, ouro, prata e bronze.

Para a realização dessa análise, a escolha das lojas virtuais baseou-se nos resultados obtidos pela E-bit no ano de 2014, na qual a empresa divulgou as lojas que ganharam o prêmio de excelência em qualidade de comércio eletrônico em cada uma das categorias já mencionadas. Sendo assim, foram avaliados a vulnerabilidade nos sistemas, através de testes exploratórios (caixa preta) nas funcionalidades básicas em quatro lojas virtuais. As lojas top-diamante (submarino.com), top-ouro (girafa.com.br), top-prata (paulinhomotos.com.br) e top-bronze (utilplast.com.br). A próxima imagem (05) mostra como serão referenciadas as lojas, seguindo a respectiva sequência anteriormente citada.

Imagem 5 – Representação referente à classificação das lojas



Fonte: Loja..., 2013.

3.2 A FERRAMENTA SCANNER

Como instrumento de análise para a consumação dos testes nas lojas virtuais escolhidas utilizou-se a Vega Vulnerability Scanner 1.0 (VVS), uma ferramenta gratuita que atua na verificação de vulnerabilidades em sistemas web. A Vega nada mais é do que a eclipse (ferramenta já conhecida no universo do desenvolvimento de software) adaptada para a prática de avaliações em vulnerabilidade de sistemas. Com ela, é possível analisar as requisições efetuadas pelos usuários (request) bem como as respostas obtidas por parte do sistema (response). A ferramenta é de fácil instalação e está disponível no endereço <<https://subgraph.com/vega/>> para download gratuito.

Existem duas formas de utilização da ferramenta VVS para a realização dos testes: os modos automático e manual. No modo automático, partindo da URL index, ou seja, página principal da aplicação na web, a ferramenta realiza as possíveis requisições e observa as respostas por parte do sistema, avaliando assim o tráfego da informação nas duas vertentes. Este modo é ideal para ambientes de desenvolvimento de software onde se utiliza uma massa de dados específica, própria para este ambiente, não trabalhando com dados reais. Com a massa de dados exclusiva e controlada, rapidamente são obtidos os resultados através desse scanner. Porém, trabalhar com dados reais no modo automático é praticamente inviável devido a quantidade de requisições que seriam feitas ao sistema com uma mesma finalidade e o número de vulnerabilidades encontradas que se repetiriam ao longo de toda a análise. Por este motivo, a ferramenta VVS disponibiliza um modo

manual, que possibilita a realização de requisições livres por parte do testador, com o browser configurado juntamente a ela.

3.3 O BROWSER UTILIZADO

O browser escolhido para navegar na internet no momento da realização dos testes funcionais foi o Mozilla Firefox (v. 42.0). Embora a ferramenta Vega permita a utilização de qualquer browser existente no mercado, a escolha do Firefox se deu devido à facilidade de configuração com a mesma. O browser pode ser facilmente encontrado no endereço <<https://www.mozilla.org/pt-BR/firefox/new/>> para download gratuitamente.

3.4 CONFIGURAÇÃO DA FERRAMENTA COM O BROWSER

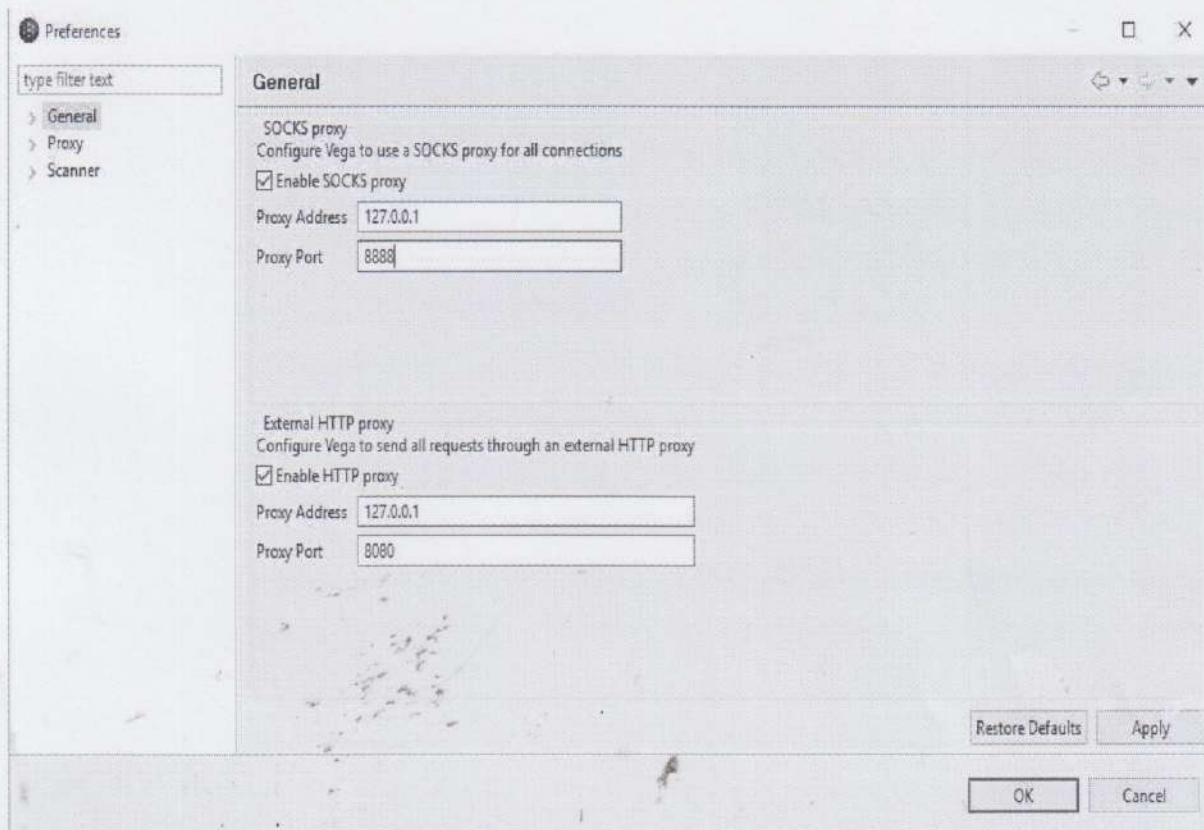
Como a realização da análise na qual se destina esse trabalho necessitou utilizar dados reais contidos nas lojas virtuais escolhidas, o modo manual da ferramenta Vega foi utilizado. Para o emprego do mesmo, foi necessário configurar manualmente o proxy da rede no browser juntamente com a ferramenta.

Para isso, na ferramenta VVS, utiliza-se o seu menu de opções para se dirigir até a aba de preferências para selecionar a opção geral (general). Em seguida, necessita-se habilitar as opções de proxy, tanto para a ferramenta (SOCKS) quanto para os protocolos HTTP, informando os parâmetros necessários. Dependendo do SO do computador no qual se executará os testes, será preciso utilizar seus recursos de configurações para saber qual o proxy atribuído à máquina. No Windows, o proxy pode ser verificado nas configurações gerais do browser utilizado para a análise, pois ao instalá-lo no computador, o proxy da máquina já será atribuído ao browser automaticamente.

Na ferramenta, como parâmetro de endereço proxy, tanto para a opção da ferramenta (SOCKS) quanto para a opção External HTTP, habilitadas anteriormente, coloca-se o endereço proxy da máquina de realização dos testes, como mostra as imagens 06 e 07 que exemplifica o processo que faz com que a ferramenta trabalhe de forma intermediária entre o browser e a internet. Todo o resto da configuração padrão já existente na ferramenta deverá ser mantido.

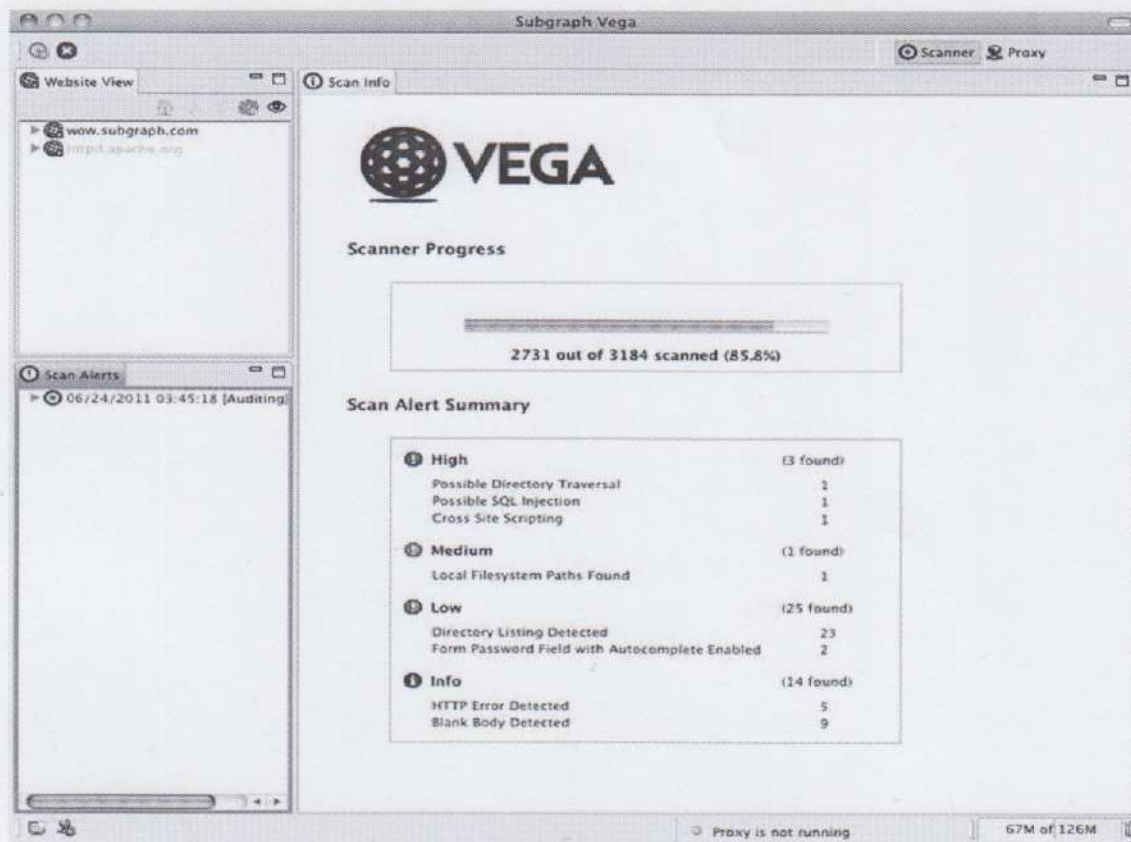
Cada requisição realizada pelo testador, antes de trafegar na internet e chegar ao servidor que irá respondê-lo, passa primeiro pelo scanner de vulnerabilidade. Da mesma forma as respostas por parte do sistema, antes de serem exibidas na tela do testador.

Imagem 6 – VVS – Configuração do proxy



Fonte: VVS, 2016.

Imagem 7 – VVS – Tela de análise



Fonte: VVS, 2016.

Através da VVS foi possível capturar todo tráfego de informações durante a navegação realizada nas quatro lojas virtuais. Configurada para analisar as solicitações (request) e as respostas por parte do sistema (response), bastou apenas inicializar o escaneamento e utilizar o browser de internet para os sistemas propostos. A imagem a seguir exibe a tela de análise de tráfego da ferramenta VVS, que manifesta o tempo de resposta do sistema, o método que a aplicação passa os dados online, o status das ações, dentre outras informações referentes as requisições e respostas por parte do sistema testado.

Imagem 8 – VVS – Análise de tráfego

ID	Host	Method	R...	Status	Length	Time...	Tags
0	http://www...	GET	/	200	104644	359	
1	http://www...	GET	/...	200	35597	188	
2	http://www...	GET	/...	200	133348	641	
3	http://www...	GET	/...	200	183849	390	
4	http://www...	POST	/...	405	369991	307	
5	http://www...	GET	/...	200	183153	219	
6	http://www...	POST	/...	405	369991	203	
7	http://www...	POST	/...	405	369991	323	
8	http://www...	GET	/...	401	381	156	
9	http://www...	POST	/...	405	369991	278	
10	http://www...	GET	/...	200	105083	390	
619	http://www...	GET	/	200	100669	140	
620	http://www...	GET	/...	200	34193	140	
621	http://www...	GET	/...	302	210	1631	
622	http://www...	GET	/...	200	35596	250	
623	http://www...	GET	/...	301	302	1313	
624	http://www...	GET	/...	301	318	254	
625	http://www...	POST	/...	405	366971	1205	
626	http://www...	GET	/	200	100669	140	

Request Response
GET / HTTP/1.1

Fonte: VVS, 2016.

3.5 TESTES FUNCIONAIS UTILIZADOS

Com a ferramenta VVS devidamente configurada, agregada a técnica de testes caixa-preta (testes funcionais) utilizando as funcionalidades básicas dos sistemas alvos, foi possível a realização da análise de vulnerabilidade nas lojas virtuais selecionadas. A abordagem funcional foi aplicada devido ao fato do não conhecimento do código fonte utilizado na criação das lojas virtuais. Sendo assim, os testes basearam-se em pré-condições e pós-condições mediante as requisições do testador aos sistemas.

Foram consideradas funcionalidades básicas nas lojas virtuais, todas as ações possíveis e em comum entre elas. A seguir, segue a lista destas ações:

- Manter Usuário (create, update, delete)
- Listar produtos
- Buscar produtos
- Comprar produto (carrinho de compras, realização da compra).

Ao inicializar a utilização do browser, bastou apenas ativar a ferramenta para que ela começasse a realizar a análise. Enquanto percorreu-se as funcionalidades básicas de cada sistema, a ferramenta coletou dos dados referente ao tráfego de informações entre usuário e servidor (request e response).

4 RESULTADOS OBTIDOS

A própria ferramenta classifica os resultados obtidos em três categorias de riscos de vulnerabilidade: alto risco (HIGH), risco médio (MEDIUM) e baixo risco (LOW). Ao todo, foram detectadas onze vulnerabilidades nos quatro sistemas virtuais analisados, sendo sete na categoria HIGH, dois na categoria MEDIUM e dois na categoria LOW. Serão apresentados a seguir os dados referentes às vulnerabilidades encontradas como a nomenclatura de referência indicada pela ferramenta, qual o lugar coletado e o que eles representam como potencial de ataque, por loja virtual. A classificação padrão das vulnerabilidades apresentada pela VVS foi mantida nessa análise e, ao seu final, uma tabela exibirá as vulnerabilidades existentes em cada loja virtual testada como caráter comparativo.

Vale ressaltar que a análise se baseia apenas no tráfego de informações verificado pela ferramenta VVS, não empregando nenhum tipo de técnica de invasão hacker, pois se utilizando desse conhecimento, qualquer verificação feita nas aplicações passa a ser considerado um ataque ao sistema, o que não é o propósito dessa análise.

4.1 VULNERABILIDADES EM COMUM ENTRE OS SISTEMAS

Durante a execução dos testes funcionais, foram coletadas vulnerabilidades em comum entre as lojas analisadas, ou seja, riscos iguais existentes nas aplicações virtuais. Foram coletados dois apontamentos nas quais a ferramenta classificou como sendo um na categoria LOW e ou outro na categoria HIGH.

4.1.1 Form Password Field Autocomplete Enable

Esse risco que está presente em todos os sistemas testados (quatro) na análise e foi apontada como uma vulnerabilidade de categoria LOW pela VVS, faz menção a alternativa de auto completar os dados referentes à autenticação dos usuários. Nas lojas virtuais, essa opção é criada a fim de agilizar o processo de identificação do usuário e de permanência de sessão na aplicação. No entanto, a análise da ferramenta aponta essa opção como uma vulnerabilidade por não saber exatamente a partir de qual computador o internauta se conectará ao sistema. Se a

partir de um computador doméstico, empresarial ou de PCs privados, como os de lan house e cyber coffee. Por esse motivo, essa tentativa de facilitar a vida do usuário torna-se um risco, já que outras pessoas podem fazer uso do mesmo computador e assim, com os dados salvos no sistema, pode-se ter acesso às informações pessoais.

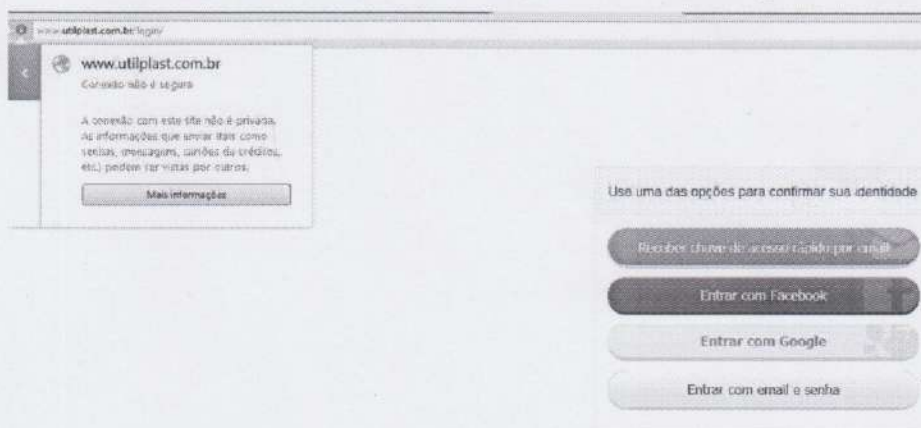
Para verificar essa opção da aplicação, bastou realizar o login, utilizando credenciais válidas, e em seguida, fechar a aba da página autenticada da loja virtual no navegador, de maneira que o browser não seja encerrado totalmente. Ao abrir uma nova aba no navegador e entrar novamente no sítio da loja, verificou-se a permanência da autenticação no sistema, o que, de acordo com a ferramenta, não deveria acontecer.

4.1.2 HTML Authentication Over Unencrypted HTML

Na tela de identificação de usuário constatou-se que o sistema não possui protocolo de transferência seguro. O HTTPS. A ferramenta Vega apontou e classificou como vulnerabilidade de categoria HIGH, pois dessa forma, um usuário autenticando-se no sistema através de simples protocolo HTTP, sua credencial fornecida pode estar sendo enviada como simples texto, sem nenhuma restrição, e assim, os dados tendem a ficar vulneráveis a espionagem. Uma conexão com protocolo de segurança HTTPS impediria a divulgação não autorizada das credenciais de autenticação e o usuário ainda poderia visualizar, dentre outras informações, os dados de sua conexão como o proprietário do domínio, a homologação, o certificado digital e o nível de criptografia utilizado na aplicação.

Esta vulnerabilidade está presente nas lojas top-prata (paulinhomotos.com.br) e top-bronze (utilplast.com.br). Como exemplo, as imagens 09 e 10 exibem as telas de identificação do sistema top-bronze na qual comprova a não existência de um protocolo de segurança HTTPS, tanto antes da autenticação como após o usuário já autenticado na aplicação.

Imagem 9 – Loja virtual Utilplast – Tela de login



Fonte: UTILPLAST, 2016.

Imagem 10 – Loja virtual Utilplast – Tela Minha Conta



Fonte: Utilplast, 2016.

4.2 VULNERABILIDADES DA LOJA TOP-DIAMANTE (SUBMARINO)

Neste ponto serão exibidas as vulnerabilidades presentes na loja virtual top-diamante (submarino.com.br). Ao todo foram coletados na aplicação dois registros de vulnerabilidades e ambos classificados pela ferramenta Vega na categoria LOW. Um deles refere-se à “Form Password Field Autocomplete Enable”, que está presente em todos os sistemas testados e citada anteriormente na sessão 4.1.1 deste trabalho.

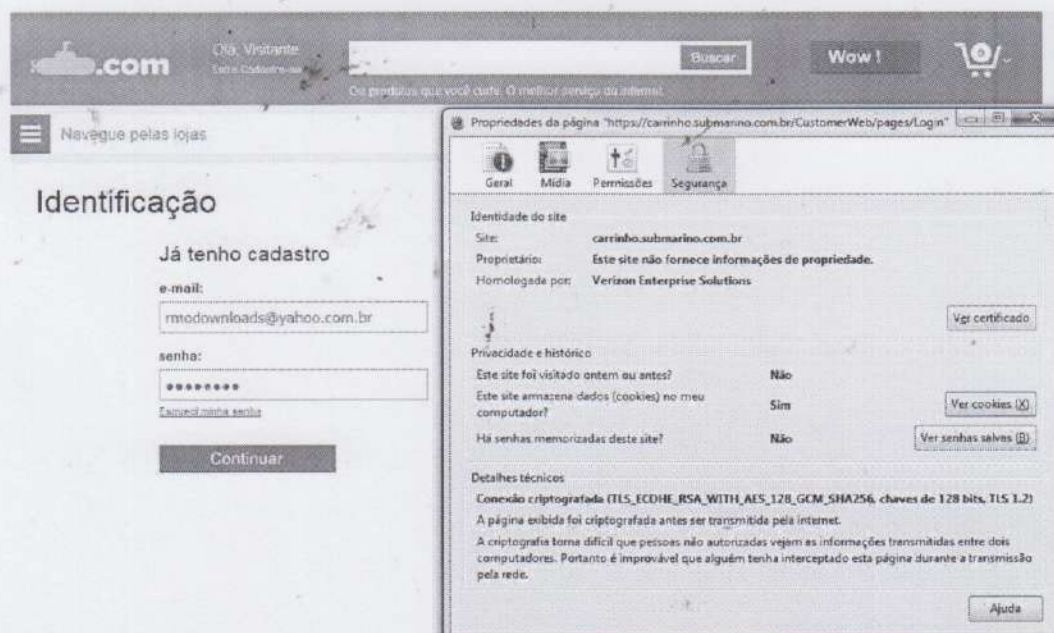
4.2.1 E-mail Addresses Found

Ao realizar o login em seu sistema, foi apontada essa vulnerabilidade no momento do envio dos dados referentes às credenciais do usuário ao servidor da aplicação (request), em sua autenticação. Para a ferramenta, a forma na qual os dados são submetidos ao sistema, de alguma maneira permite que um script adicional, rodando do lado do cliente, ou seja, no computador do próprio usuário, possa pegar essas informações.

Mostrando-se bastante criteriosa, a ferramenta indica que a aplicação deveria realizar uma checagem em seus campos de preenchimento para investigar se há a ação de scripts ou software de terceiros antes de submeter os dados ao sistema. Para realizar essa verificação, bastou ingressar no sitio top-diamante (Submarino), ir até a área "Olá Visitante" e clicar em "Entre" para informar as credenciais para autenticação e submetê-las ao sistema.

Apesar de ser apontada como uma vulnerabilidade, a loja virtual possui certificado digital e conexão segura HTTPS, o que garante certo nível de confiabilidade, uma vez que os seus usuários podem visualizar os dados de sua conexão, como nível de criptografia e o certificado digital, dentre outras informações, como ilustra a imagem 11.

Imagem 11 – Loja Submarino – Propriedades de segurança



4.3 VULNERABILIDADES DA LOJA TOP-OURO (GIRAFÁ)

Na loja virtual top-ouro (girafa.com.br) foi apontado pelo scanner de vulnerabilidades apenas uma única vulnerabilidade. Na categoria LOW, a "Form Password Field Autocomplete Enable" citada anteriormente como vulnerabilidade existente em todas as lojas virtuais testadas, faz referência à alternativa de auto complemento dos dados para autenticação e permanência de sessão dos usuários. Possuindo credenciais validadas no sistema, bastou apenas seguir os mesmos passos citados anteriormente no ponto 4.1.1 deste trabalho.

4.4 VULNERABILIDADES DA LOJA TOP-PRATA (PAULINHO MOTOS)

Neste ponto serão exibidas as vulnerabilidades encontradas na aplicação top-prata (paulinhomotos.com.br). Ao todo foram coletados cinco riscos classificados pela ferramenta VVS como sendo um na categoria LOW, dois na categoria MEDIUM e dois na categoria HIGH. As duas vulnerabilidades mencionadas anteriormente na sessão 4.1 deste trabalho, foram coletadas nessa loja virtual e também contabilizam para a sua análise.

4.4.1 Local Filesystem Paths Found

Categorizada como MEDIUM pela VVS, foi catalogado este risco quando, autenticado na aplicação, ao solicitar a inclusão de um produto no carrinho de compras da loja, a ferramenta detectou uma falha na resposta da solicitação que possivelmente pode exibir os caminhos do sistema de arquivos absolutos em formato digitalizado, ou seja, algo referente às estruturas internas do software que pode ser mostrado na tela do usuário final.

A aparição desses caminhos internos revela informações sobre o desenho arquitetural da aplicação e dessa forma, os dados tendem a se tornar visíveis. A divulgação dessas informações pode aumentar as chances de sucesso para quaisquer outros ataques. As imagens 12 e 13 mostram o caminho percorrido ao analisar a aplicação para a exibição da vulnerabilidade na ferramenta.

Imagem 12 – Loja Paulinho Motos – Menu capacetes e sub menus



Fonte: Paulinho Motos, 2016.

Imagem 13 – Loja Paulinho Motos – Escolha do produto



Fonte: Paulinho Motos, 2016.

4.4.2 Possible Source Code Disclosure

Derivado da mesma ação descrita na vulnerabilidade anterior e também classificada como MEDIUM, o scanner de vulnerabilidades detectou uma possibilidade de vazamento de código fonte do sistema. Para a ferramenta, esse risco existe porque possivelmente, na linguagem de programação na qual foi desenvolvida a aplicação pode-se misturar código nativo com o conteúdo estático

que se apresenta ao usuário final. Uma falha nessa combinação pode resultar no vazamento de informações importantes, uma vez que durante o desenvolvimento, além de utilizar os recursos da linguagem implementada, o programador escreve diversos comentários no código construído para facilitar na futura correção de erros nas funcionalidades da aplicação. Uma tag (marcação) HTML utilizada de maneira errada, por exemplo, pode mostrar essas informações na tela do usuário final.

Apesar de não ter acesso ao código fonte da aplicação, a ferramenta analisa o conteúdo que será exibido na tela do testador, que pode constatar tipos de linguagem de programação que utilizam a combinação de código fonte e elementos de interface, como PHP, JSP, Ruby, etc.

4.4.3 Integer Overflow

Após ser adicionado um produto ao carrinho de compras da loja, a ferramenta apontou este risco que se dá no espaço reservado para alocação dos resultados de operações aritméticas no sistema. Um erro na alocação dos resultados dessas operações pode disparar uma exceção na tela do usuário final, caso não seja tratado antecipadamente pelo desenvolvedor. Essa exceção pode acarretar na divulgação dos caminhos internos do sistema, como já visto anteriormente na vulnerabilidade "Local Filesystem Paths Found". A diferença é que a anterior é mais genérica e representa qualquer exceção que possa ser disparada pelo sistema, já essa, é uma, das várias exceções existentes que podem ser desfechadas e, por este motivo, foi classificada como sendo HIGH pela ferramenta.

A imagem 14 exibe a tela do carrinho de compras da loja virtual, contendo o produto adicionado. Nota-se que a tela do usuário é composta por diversos campos contendo números derivados de operações aritméticas.

Imagem 14 – Loja Paulinho Motos – Carrinho de compras



PRODUTO	ENTRADA	PREÇO	QUANTIDADE	TOTAL
 Capacete AGV Horizon Absolute BRANCO-PRATA 58	a calcular	R\$ 999,90 R\$ 999,90 / UN	 1 	R\$ 999,90

CUPOM DE DESCONTO

Subtotal R\$ 999,90

Entrega

Total R\$ 999,90

Fonte: Paulinho Motos, 2016.

4.5 VULNERABILIDADES DA LOJA TOP-BRONZE (UTILPLAST)

Neste tópico serão abordadas as vulnerabilidades deparadas na aplicação top-bronze (Utilplast). Ao todo foram achadas sete vulnerabilidades qualificadas pela ferramenta Vega como sendo, um na categoria LOW e seis na categoria HIGH. Contabilizam também para a sua análise as duas vulnerabilidades mencionadas na sessão 4.1 deste trabalho.

4.5.1 Cross-site Scripting

Durante a utilização da loja virtual, em dois momentos a ferramenta detectou o que pode ser uma abertura para a inserção de scripts a serem interpretados como código nativo pela aplicação, o que caracteriza uma vulnerabilidade. O impacto desse risco depende muito do sistema, já que o código malicioso contido em um script cruzado, dentre as várias ações possíveis, pode de alguma maneira manipular o DOM no HTML e assim alterar as características oficiais da página acessada. Por este motivo a VVS classificou o risco na categoria HIGH.

As imagens 15 e 16 exibem os dois momentos em que, navegando pelo sistema, foi apontado o risco desse ataque. A primeira imagem exhibe a tela de busca de produtos no sistema. Já a segunda mostra o resultado de uma busca realizada.

Imagem 15 – Loja Utilplast – Tela inicial



Fonte: Utilplast, 2016.

Imagem 16 – Loja Utilplast – Resultado de busca por churrasqueira



Fonte: Utilplast, 2016.

Considerando que a ação “busca de produtos” é frequentemente utilizada pelos usuários que navegam na loja virtual e, conseqüentemente suas respostas são apresentadas de uma mesma maneira, esse risco perpetua por toda a

aplicação. As buscas podem ser realizadas através do menu de navegação ou do campo "buscar produto" existente.

4.5.2 Possible Credit Card Data, Social Insurance Number and Social Security Number Detected

Registradas em um mesmo momento, essas três vulnerabilidades foram encontradas pela ferramenta VVS e apontadas como sendo da categoria HIGH. Faz menção à possibilidade de vazamento de informações pessoais como, dados de cartões de crédito, números de CPF e RG, dados previdenciários, dentre outros. Foi coletas no momento em que se tentava realizar alterações nos dados pessoais existentes no cadastro da loja, na opção "Minha conta".

A divulgação de dados não autorizados pode levar a fraudes ou roubos de identidade e isso deve ser investigado pelas equipes de desenvolvimento e manutenção do sistema, uma vez que esses dados correspondem a informações vindas do banco de dados da aplicação ou de alguma outra vulnerabilidade inesperada.

A imagem 17 mostra o momento em que foram capturadas as vulnerabilidades. Nota-se que na tela, não há informações referentes a cartões de crédito, mas a ferramenta exibe como vulnerabilidade essa possibilidade por ser da mesma natureza das outras duas citadas.

Imagem 17 – Loja Utilplast – Minha conta

Dados pessoais

Nome: Raniere Sobrenome: Moura

Apelido: RaniMoura CPF: [Empty]

Data de nascimento: [Empty] Sexo: ☒ Masculino ☐ Feminino

☐ Incluir dados da pessoa jurídica

Dados de contato

E-mail: rmodownloads@yahoo.com.br Telefone: 8888-8888

Telefone Comercial: 8888-8888

Salvar Cancelar

4.5.3 SQL Injection

Derivada da mesma ação anteriormente citada, ao concluir o processo de alteração de dados no sistema e submetê-los, a ferramenta detectou o que pode ser a possibilidade de rodar scripts SQL em campos do formulário de cadastro, sem tratamento adequado por parte da aplicação. Como qualquer pessoa que navega na internet pode realizar seu cadastro livremente na loja, tornando-se cliente, passando a ter acesso aos campos de alteração de sua conta, um usuário com conhecimentos avançados em informática pode utilizar-se de noções SQL para tentar colher informações privilegiadas sobre o negócio da empresa, ou até mesmo destruir as informações contidas no banco de dados da aplicação. Por este motivo, a ferramenta Vega classificou essa vulnerabilidade na categoria HIGH.

4.6 COMPARATIVO FINAL

Após a exposição das vulnerabilidades apresentadas pela ferramenta Vega Vulnerability Scanner 1.0 na análise realizada, uma planilha foi montada para exibir as vulnerabilidades existentes em cada uma das lojas virtuais testadas. A imagem 18 mostra os resultados obtidos e quantificados na planilha, onde estão presentes as lojas sob domínio Submarino, Girafa, Paulinho Motos e Utilplast, a nomenclatura padrão de acordo com a ferramenta para as vulnerabilidades encontradas e a marcação "X" para determinar a qual loja virtual cada vulnerabilidade pertence.

Imagem 18 – Resultado da análise de vulnerabilidades

VULNERABILIDADES	SUBMARINO	GIRAFÁ	PAULINHO MOTOS	UTILPLAST
E-MAIL ADDRESSES FOUND	X			
FORM PASSWORD FIELD AUTOCOMPLETE ENABLE	X	X	X	X
LOCAL FILESYSTEM PATHS FOUND			X	
POSSIBLE SOURCE CODE DISCLOSURE			X	
CROSS-SITE SCRIPTING				X
HTML AUTHENTICATION OVER UNENCRYPTED HTML			X	X
INTEGER OVERFLOW			X	
POSSIBLE CREDIT CARD DATA DETECTED				X
POSSIBLE SOCIAL INSURANCE NUMBER DETECTED				X
POSSIBLE SOCIAL SECURITY NUMBER DETECTED				X
SQL INJECTION				X
TOTAL	2	1	5	7

Fonte: VVS, 2016.

Como pode-se observar, loja Girafa apresentou-se com o menor índice de vulnerabilidades em seu sistema contabilizando apenas uma vulnerabilidade encontrada na classificação LOW (baixo risco), enquanto a loja Utilplast mostrou-se mais propícia a ataques com um total de sete vulnerabilidades, sendo seis delas da classificação HIGH (alto risco).

Vale lembrar que, para cada sistema testado pode haver equipes de manutenção de software que possam corrigir esses riscos apresentados e disponibilizando periodicamente novas versões das aplicações existentes. Isso leva a concluir que, no momento dessa análise, foram constatadas as vulnerabilidades acima citadas, porém, em um momento posterior a essa data, pode não haver mais os riscos apontados pela ferramenta Vega, devido as necessárias manutenções casuais.

5 CONCLUSÃO

Conforme abordado nesse trabalho, viu-se que, hoje em dia, a realização de testes em sistemas de informação é fundamental para tentar neutralizar os riscos em um projeto, no entanto sabe-se que nem sempre isso é possível. A segurança da informação na web é fator prioritário e deve ser avaliada com cautela, pois dentre os riscos a serem minimizados, os de maior importância são as vulnerabilidades de sistemas, já que mexem com todo o negócio do cliente.

A utilização da ferramenta Vega, no modo manual, juntamente com a abordagem funcional (testes caixa preta), possibilitou uma análise clara sobre as vulnerabilidades nos sistemas. Mostrou-se rápida, completa e adequada para esta finalidade.

Com base nos resultados obtidos na análise de vulnerabilidade realizada com a ferramenta Vega Vulnerability Scanner 1.0, pode-se concluir que, embora os riscos detectados não afetem prioritariamente os clientes do mercado eletrônico, alguns desses podem interferir no desempenho dos sistemas. No entanto, os apontamentos não comprometem, no ato, o processo de transação nas vendas pela internet, mas podem contribuir para que haja vazamento de informações pessoais em alguns casos.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 27002**: Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da informação. Rio de Janeiro: ABNT, 2005

ASSUNÇÃO, Marcos F. A. **O Guia do hacker brasileiro**. São Paulo: Visual Books, 2002.

ÁVILA, Ana Luiza. Engenharia de software: introdução à requisitos de software. **Revista Engenharia de Software**, ed. especial, 2013.

CARVALHO, Amaury; PIRES, Antônio. Google Hacking para Ataques SQL Injection. **Artigo da Faculdade de Tecnologia SENAI de Desenvolvimento Gerencial**, 2015.

COMITÊ GESTOR DA INTERNET NO BRASIL. Cartilha de Segurança para Internet. **CERT.BR**, v. 4, n. 2, 2012.

DIAS NETO, Arilo Cláudio. Introdução a Testes de Software. **Revista Engenharia de Software**, n. 1, 2008.

ELIZA, Renata ;LAGARES, Vivian. Teste de Segurança: agregando confiança ao software. **Revista Java Magazine**, v. 115, 2013.

LESTIN, Flávio. Processos de Testes de Software. **PUCRN - Faculdade de Informática**, Porto Alegre/RS, 2008. Disponível em: <<http://www.inf.pucrs.br>>. Acesso em: 21 out. 2015.

LOJA INTEGRADA. O que é o E-Bit e Como Configurá-lo em sua Loja. **Central de Suporte Loja integrada**, São Paulo, p. Única, out. 2013. Disponível em: <<https://suporte.lojaintegrada.com.br/pt-br/articles/200383344-O-que-%C3%A9-o-E-bit-e-como-configurar%C3%A1-lo-em-sua-loja>> Acesso em: 1 dez. 2015.

MÓDULO SOLUTIONS. O que é Segurança da Informação. **Página de Fundamentos sobre Segurança da Informação**, São Paulo, p. Única, ago., 2013. Disponível em: <<http://segurancadainformacao.modulo.com.br/seguranca-da-informacao>>. Acesso em: 24 dez. 2015.

PIQUEIRO, Thiago José; ZADRA, Augusto. Processo de Testes: Abordagem de Projeto de Testes. **Revista Pensar Tecnologia**, v. 4, 2015.

RIOS, Emerson; MOREIRA, Trayahú. **Testes de software**. 3. ed. Rio de Janeiro: Alta Books, 2013.

SOMMERVILLE, Ian. **Engenharia de Software**. 6.ed. São Paulo: Pearson Education do Brasil, 2003.

STEFANELLI, J. Eduardo. A Engenharia Reversa. **Revista do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo**, v. 1, 2012.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: McGraw-Hill, 2006.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional**. 7. ed. São Paulo: McGraw-Hill, 2011.

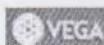
TAVARES, Pedro; CARVALHO, Hugo; BOTELHO, Catarina. **Segurança dos Sistemas de Informação**. Lisboa: Centro Atlântico, 2003.

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE). **IEEE93a**: Standard for Software Test Documentation. New York: IEEE Computer Society, 1993.

THE OPEN WEB APPLICATION SECURITY PROJECT (OWASP). **As 10 vulnerabilidades de segurança mais críticas em aplicações web**. 3. ed. [s.l.]: OWASP Foundation, 2013.

APÊNDICES

APÊNDICE A – RELATÓRIO DA FERRAMENTA VEGA VULNERABILITY SCANNER 1.0 (LOJA SUBMARINO)



EMAIL ADDRESSES FOUND

D-URL: <<https://carrinho.submarino.com.br/CustomerWeb/pages/Login>>

DISCUSSION:

User the addresses of system users, addresses inserted in user-supplied content, or third-party addresses embedded in components of the application (such as Javascript libraries). Automatically scraping websites is one way that spammers and phishers collect e-mail addresses for their distribution lists. It is recommended that e-mail addresses not be displayed on exposed parts of the web application, directly or indirectly.

IMPACT:

E-mail addresses exposed to the Internet will be scraped by spambots and added to spam lists.

E-mail addresses can also be used in targeted and phishing attacks.

E-mail addresses could be used to more accurately guess application usernames.

REMEDIATION:

If the are e-mail addresses are those of users, developers should investigate why they being output and try to remove or obfuscate them.

E-mail addresses embedded in user-supplied content should be filtered or obfuscated to prevent unintended disclosure.

It is common that e-mail addresses may be automatically included in third-party components, such as Javascript libraries.

Another possibility is that the server is automatically configured to include e-mail addresses. Tweaking configuration can usually remove these.



FORM PASSWORD FIELD WITH AUTOCOMPLETE ENABLE

D-URL: <<https://minhaconta.submarino.com.br/#/account/home>>

DISCUSSION:

Not set to off. This may result in some browsers storing values input by users locally, where they may be retrieved by third parties.

IMPACT:

A password value may be stored on the local filesystem of the client.

Locally stored passwords could be retrieved by other users or malicious code.

REMEDIATION:

The form declaration should have an autocomplete attribute with its value set to "off".

APÊNDICE B – RELATÓRIO DA FERRAMENTA VEGA VULNERABILITY SCANNER 1.0 (LOJA GIRAFÁ)



FORM PASSWORD FIELD WITH AUTOCOMPLETE ENABLE

D-URL: <<https://www.girafa.com.br/Login>>

DISCUSSION:

Not set to off. This may result in some browsers storing values input by users locally, where they may be retrieved by third parties.

IMPACT:

A password value may be stored on the local filesystem of the client. Locally stored passwords could be retrieved by other users or malicious code.

REMEDIATION:

The form declaration should have an autocomplete attribute with its value set to "off".

APÊNDICE C – RELATÓRIO DA FERRAMENTA VEGA VULNERABILITY SCANNER 1.0 (LOJA PAULINHO MOTOS)



FORM PASSWORD FIELD WITH AUTOCOMPLETE ENABLE

D-URL: <http://www.paulinhomotos.com.br/>

DISCUSSION:

Not set to off. This may result in some browsers storing values input by users locally, where they may be retrieved by third parties.

IMPACT:

A password value may be stored on the local filesystem of the client.

Locally stored passwords could be retrieved by other users or malicious code.

REMEDIATION:

The form declaration should have an autocomplete attribute with its value set to "off".



HTML AUTHENTICATION OVER UNENCRYPTED HTML

URL: <http://www.paulinhomotos.com.br/>

DISCUSSION:

HTTP. If a user were to authenticate to this resource over HTTP, the supplied

credentials would be sent in cleartext and be vulnerable to eavesdropping. HTTPS will prevent unauthorized disclosure of HTTP authentication credentials.

IMPACT:

Disclosure of credentials to network eavesdropper.

Unauthorized access.

REMEDIATION:

Resources protected by HTTP authentication Should only be accessible over HTTPS.

**INTEGER OVERFLOW**

D-URL: <<https://www.paulinhomotos.com.br/checkout/#/cart>>

DISCUSSION:

Occurs in programs written in languages such as C, the resulting behavior can have security implications. In these cases, unsigned integers will be reduced, wrapping back to a lower numeric value. The potential impact on security depends on how the integer value is used. If it is used as the size of a data buffer, forcing it to wrap to a lower value may result in bypassing of size checks, introducing possible buffer overflow conditions.

IMPACT:

Integer overflow errors can have a variety of impacts, depending on the context and the purpose of the integer.

Integers used to check the size of a data buffer, if reduced, can incorrectly

represent the total amount of data, resulting in a possible buffer overflow.

REMEDIATION:

The developer should investigate the error and determine if a vulnerability is presente.



LOCAL FILESYSTEM PATHS FOUND

D-URL: <http://www.paulinhomotos.com.br/capacete-agv-horizon-absolute-branco---prata/p>

DISCUSSION:

This information is sensitive, as it may reveal things about the server environment to an attacker. Knowing filesystem layout can increase the chances of success for blind attacks. Full system paths are very often found in error output. This output should never be sent to clients on production systems. It should be redirected to another output channel (such as an error log) for analysis by developers and system administrators.

IMPACT:

Vega has detected what may be absolute filesystem paths in scanned content.

Disclosure of these paths reveals information about the filesystem layout.

This information can be sensitive, its disclosure can increase the chances of success for other attacks.

REMEDIATION:

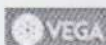
Absolute paths are often found in error output.

Both the system administrators and developers should be made aware, as the problem.

May be due to an application error or server misconfiguration.

Error output containing sensitive information such as absolute system paths should not be sent to remote clients on production servers.

This output should be sent to another output stream, such as an error log.

**POSSIBLE SOURCE CODE DISCLOSURE**

D-URL: <<http://www.paulinhomotos.com.br/capacete-agv-horizon-absolute-branco---prata/p>>

DISCUSSION:

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.

APÊNDICE D – RELATÓRIO DA FERRAMENTA VEGA VULNERABILITY SCANNER 1.0 (LOJA UTILPLAST)



FORM PASSWORD FIELD WITH AUTOCOMPLETE ENABLE

D-URL: <<http://www.utilplast.com.br/>>

DISCUSSION:

Not set to off. This may result in some browsers storing values input by users locally, where they may be retrieved by third parties.

IMPACT:

A password value may be stored on the local filesystem of the client.

Locally stored passwords could be retrieved by other users or malicious code.

REMEDIATION:

The form declaration should have an autocomplete attribute with its value set to "off".



CROSS SITE SCRIPTING (XSS)

D-URL: <<http://www.utilplast.com.br/fim-de-semana/para-o-churrasco/churrasqueiras-e-acessorios/?PS=15&O=OrderByNameASC>>

DISCUSSION:

Cross circumvented site-scripting .When a (XSS) browser visits class a page of vulnerabilities onawebsite, affecting scriptcode web originating applications inthe that website can domain result can insecurity access and controls manipulate implemented the DOM in (document browsers object model), a representation of the page and its properties in the browser. Script code from another website can not. This is known as the "same origin policy", a critical control in the browser security model. Cross-site scripting vulnerabilities occur when a lack of input validation permits users to inject script code into the target website such that it runs in the browser of another user who is visiting the same website. This would circumvent the browser same-origin policy because the browser has no way to distinguish authentic script code from inauthentic, apart from its origin.

IMPACT:

The precise impact depends greatly on the application.

XSS is generally a threat to web applications which have authenticated users or are otherwise security sensitive.

Malicious code may be able to manipulate the content of the site, changing its appearance and/or function for another user.

This includes modifying the behavior of the web application (such as redirecting forms, etc).

The code may also be able to perform actions within the application without user knowledge.

Script code can also obtain and retransmit cookie values if they haven't been set HttpOnly.

REMEDIATION:

The developer must identify how the untrustworthy data is being output to the client without adequate filtering.

There are various language/platform specific techniques for filtering untrustworthy data.

General rules for preventing XSS can be found in the recommended OWASP XSS Prevention Cheat Sheet (see references).



HTML AUTHENTICATION OVER UNENCRYPTED HTML

URL: <<http://www.utilplast.com.br/>>

DISCUSSION:

HTTP. If a user were to authenticate to this resource over HTTP, the supplied credentials would be sent in cleartext and be vulnerable to eavesdropping. HTTPS will prevent unauthorized disclosure of HTTP authentication credentials.

IMPACT:

Disclosure of credentials to network eavesdropper.

Unauthorized access.

REMEDIATION:

Resources protected by HTTP authentication Should only be accessible over HTTPS.



POSSIBLE CREDIT CARD DATA DETECTED

URL: <<http://www.utilplast.com.br/account/>>

DISCUSSION:

This should be reviewed carefully (which can be done by examining the server response).

IMPACT:

Vega has detected a string of digits that have some of the properties of credit card numbers.

Unauthorized disclosure of this information could lead to fraud, or identity theft Unauthorized disclosure could also lead to regulatory penalties.

REMEDIATION:

This should be investigated to identify the nature of the data matching the database, or the triggering of some unexpected vulnerability.

**POSSIBLE SOCIAL INSURANCE NUMBER DETECTED**

URL: <<http://www.utilplast.com.br/account/>>

DISCUSSION:

In scanned content. This could be a false positive against a pattern with the same characteristics.

IMPACT:

This should be investigated to identify the nature of the data matching the database, or the triggering of some unexpected vulnerability.

Unauthorized disclosure of this information could lead to fraud, or identity

theft.

Unauthorized disclosure could also lead to regulatory penalties.

REMEDIATION:

This should be investigated to identify the nature of the data matching the detection pattern. The root cause of a disclosure could be test data, a flat file database, or the triggering of some unexpected vulnerability.



POSSIBLE SOCIAL SECURITY NUMBER DETECTED

URL: <<http://www.utilplast.com.br/account/>>

DISCUSSION:

Scanned content. This could be a false positive against a pattern with the same characteristics.

IMPACT:

Unauthorized disclosure of this information could lead to fraud, or identity theft.

Unauthorized disclosure could also lead to regulatory penalties.

REMEDIATION:

This should be investigated to identify the nature of the data matching the detection pattern. The root cause of a disclosure could be test data, a flat file database, or the triggering of some unexpected vulnerability.



SQL INJECTION

URL: <http://www.utilplast.com.br/account/>

DISCUSSION:

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. 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.

IMPACT:

Vega has detected a possible SQL Injection Vulnerability.

These vulnerabilities can be exploited by remote attacks to gain unauthorized read or write access to the underlying database.

Exploitation of SQL injection vulnerabilities can also allow for attacks against the logic of the application.

Attackers may be able to obtain unauthorized access to the server hosting the database.

REMEDIATION:

The developer should review the request and response against the code to manually verify whether or not a vulnerability is present.

The best defense against SQL injection vulnerabilities is to use parameterized statements.

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.

Use of stored procedures can simplify complex queries and allow for tighter access control settings.

Object-relational mapping eliminates the need for SQL.