UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE CIÊNCIA DA INFORMAÇÃO

Helton Ricardo Santos

DESAFIOS DA SEGURANÇA DA INFORMAÇÃO EM GERENCIADORES DE CONTEÚDO WEB: VULNERABILIDADES E AMEAÇAS

Belo Horizonte
2015

Helton Ricardo Santos

DESAFIOS DA SEGURANÇA DA INFORMAÇÃO EM GERENCIADORES DE CONTEÚDO WEB: VULNERABILIDADES E AMEAÇAS

Monografia apresentada ao programa de Especialização do Núcleo de Informação Tecnológica e Gerencial – NITEG, no curso de Gestão Estratégica da Informação da Escola de Ciência da Informação da Universidade Federal de Minas Gerais como requisito para obtenção do certificado de Especialista em Gestão Estratégica da Informação.

Orientador: Prof. Eduardo Ribeiro Felipe

Belo Horizonte
Maio 2015

AGRADECIMENTOS

Agradeço a Deus por ter me dado o dom da vida fazendo com que cada um de nós que aqui estamos pudéssemos participar da vida feliz Dele e por ter me dado uma esposa tão especial a quem faço meu segundo agradecimento.

A você, Regiane, por acreditar em mim e me incentivar na realização deste trabalho contribuindo em vários momentos com questionamentos que me fizeram deslanchar no desenvolvimento deste.

A minha avó Maria que também ajudou para que eu me formasse, e conseguisse chegar e vencer mais esta etapa na vida.

Aos meus sogros e cunhados que tornaram as coisas mais fáceis, principalmente nos momentos difíceis que passamos juntos.

Aos meus pais que sempre acreditaram em mim e, mesmo separados, acredito estar junto deles, senão nesta vida, na feliz vida do céu.

Ao meu orientador, Eduardo Felipe, por também acreditar em mim, contribuir e apostar neste trabalho sendo tão solícito e atencioso.

Ao professor Maurício Almeida que contribuiu também para este trabalho e para o sucesso deste.

"O que o Senhor fala, o Senhor faz.
O meu Deus é o Deus do impossível."
(Pe. José Augusto, 2006)

RESUMO

O estudo tem como objetivo discutir o impacto do desenvolvimento e ou uso de *plugins* nos principais sistemas de gerenciamento de conteúdo utilizados na Internet. Para fazer isso, nós nos concentramos em Wordpress e sobre a segurança das informações contidas nos sistemas acima mencionados. Além disso, foi discutido como esses *plugins* podem transformar um seguro gerenciador de conteúdo web em um que é vulnerável a vários vetores de ataques, permitindo, assim, o roubo de informações.

Palavras-chave: CMS, Gerenciadores de conteúdo, Web, Plugins, Segurança, Informação, Vulnerabilidades, Ameaças.

ABSTRACT

The study aims to discuss the impact of development and or use of plugins in the main content management systems used on the Internet. To do so, we focused on Wordpress and on the security of the information contained in the aforementioned systems. Also, it was discussed how these plugins can turn a safe web content manager into one that is vulnerable to various vectors of attacks, enabling, thus, information theft.

Keywords: CMS, Content Management System, Web, Plugins, Security, Information, vulnerabilities, threats.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BSI	*Bundesamt für Sicherheit in der Informationstechnik*
CEO	*Chief Executive Officer*
CERT.br	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CMS	*Content Management System*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transport Protocol*
ISO	*International Organization for Standardization*
LDAP	*Lightweight Directory Access Protocol*
SQL	*Structured Query Language*
TI	Tecnologia da Informação
URL	*Uniform Resource Locator*

SUMÁRIO

1. INTRODUÇÃO

No momento em que estamos vivendo a informação passou a ter valor imenso para as empresas, independente do seu porte, pois esta pode chegar a representar conhecimento e consequentemente vantagem competitiva, visto que as empresas necessitam ter maior agilidade e rapidez aliadas ao menor custo e segurança para manter seus sites atualizados.

Desde o final da década de 90 as empresas e os profissionais utilizam ferramentas indispensáveis para conseguir gerir o conteúdo de seus sites dessa forma, são os sistemas gerenciadores de conteúdo ou *CMS* (em inglês, *Content Management Systems*). Alguns dos mais conhecidos CMS utilizados via web são: Joomla!, Wordpress, Drupal, Plone, entre outros.

Neste contexto é imprescindível tomarmos decisões rápidas, independente do local onde estamos, e nisto os gerenciadores de conteúdos nos ajudam muito bem. Além disto, ao utilizar um gerenciador de conteúdos conseguimos controlar e deliberar melhor as funções para as várias pessoas que colaboram no processo de construção e manutenção das informações, aumentando a agilidade na tomada de decisões.

A tarefa de ter um canal de informações através de um site ou blog na internet é atualmente muito mais fácil que há alguns anos atrás, porém devemos estar atentos a alguns pontos que não podemos deixar de lado e outros que não podemos descuidar ao longo do processo de gestão da informação. Um dos pontos principais é a ferramenta gerenciadora, pela qual perpassa outros pontos como: a empresa ou profissional que irá desenvolver o canal, os *plugins* que serão utilizados e a segurança da ferramenta.

Assim, ao decidir ter um canal de informações na internet, é preciso ter em mente quais são as necessidades a se atender – que tipo de conteúdo será disponibilizado, como e quem gerenciará este conteúdo, que ferramentas serão necessárias para gerenciar este conteúdo e quais os papéis ou funções que os gestores assumirão – para identificarmos o melhor gerenciador para estas necessidades, pois do ponto de vista técnico, desde que o problema consiga ser mapeado computacionalmente, é possível ter um sistema para resolver o problema.

Nos últimos anos percebemos um crescimento na geração de conhecimento nas organizações e podemos afirmar que este crescimento é proporcionado pelas informações geradas a partir de seus clientes e ou colaboradores através dos vários dispositivos, atualmente em sua grande maioria os móveis, que se conectam a internet.

Assim como as Tecnologias da Informação se desenvolveram muito nas últimas décadas e o volume de informação gerada pelos usuários cresceu, a necessidade de

segurança desta informação se torna cada vez maior (VAZ, 2007, p.38), principalmente se associada ao fator competitivo mencionado anteriormente.

Pensando neste desenvolvimento e no volume e velocidade destas informações percebemos que é de grande importância para uma organização não depender estritamente de pessoal técnico especializado em tecnologia da informação, desenvolvimento de sites e computação para gerenciar o conteúdo disponibilizado na web gerado por seus colaboradores e ou clientes tornando as tarefas mais simples e automatizadas (COBB, 2013).

De acordo com o site *Internet World Stats* em novembro de 2015 em torno de 46,1% da população mundial possuía acesso à rede mundial de computadores e o crescimento desta população com acesso à internet entre os anos de 2000 e 2015 foi de 826.9% (WORLD, 2015).

Em paralelo com o crescimento do número de pessoas tendo acesso à internet, houve também um aumento de pessoas produzindo informações. Vivenciamos o despertar da mídia independente, o surgimento e ascensão dos blogs, aplicativos e sites de venda criados e administrados por pessoas autônomas e, muitas vezes, sem formação específica, o que nem sempre altera a qualidade, segurança e funcionalidade do mesmo.

A necessidade de permanecer alerta em relação à forma e veracidade que as informações chegam a quem interesse é evidente em ambos os contextos, sejam grandes empresas ou não. Foi então que começaram a pensar, segundo Lapa (2004, p.4), que “o que faltava a essas empresas pontocom era apenas o principal aspecto de uma organização: a capacidade de gestão de informação.”

Empresas consagradas montaram equipes de colaboradores para desenvolver e manter seu espaço na rede, podendo alcançar um público maior, ganhando assim novos clientes, satisfeitos tanto pelo acesso ao produto quanto pela facilidade de interagir com tais corporações e, muitas vezes, o fato do custo ser menor que o encontrado em lojas físicas.

Um bom exemplo disto foi o que aconteceu com as indústrias automobilísticas. Segundo Lapa (2004, p.5) “algumas montadoras colocam na mão dos clientes as tarefas de montar o automóvel, realizar o pedido, entre outros. E num modelo tradicional, seria o canal de distribuição o responsável por essas tarefas.”

Pensando que, entre outras coisas, o pedido realizado pelo cliente é feito pela internet, seu cadastro contendo dados pessoais e financeiros será recebido e armazenado pela empresa. Caso a empresa não se atenha a importância da segurança com que estas informações são gerenciadas, podem acarretar sérios problemas a si própria e a seu cliente.

O exemplo citado acima é apenas um entre vários quando pensamos na necessidade de termos uma segurança da informação gerada, transmitida e guardada. É

necessário pensar na segurança desde o início do desenvolvimento na plataforma a ser utilizada, para que não se tenha surpresas negativas posteriormente.

Uma organização deve se preocupar com o desenvolvimento do seu site por profissionais especializados em qualquer que seja o CMS utilizado, pois a codificação feita para suprir alguma necessidade de função adicional a ser exercida pelo gerenciador pode comprometer a segurança de todo o núcleo seguro dele e colocar em risco toda a informação gerenciada pelo sistema.

O Relatório Anual de Segurança da Cisco[1] nos diz que “todas as organizações devem estar preocupadas em encontrar o equilíbrio certo de confiança, transparência e privacidade, pois muito está em jogo”. (CISCO, 2014, p.4) “Tradução nossa”.

E ainda que:

> Embora as tendências como computação em nuvem e mobilidade estão reduzindo a visibilidade e aumentando a complexidade da segurança, as organizações ainda devem abraçá-las, porque elas são fundamentais para a sua vantagem competitiva e sucesso do negócio. (CISCO, 2014, p.9) “Tradução nossa”.

Pensando que a segurança da informação é algo imprescindível diante da quantidade e da importância da mesma gerada e disponibilizada na web e por desenvolver vários trabalhos utilizando alguns destes gerenciadores de conteúdo, nasceu então a necessidade de pesquisar sobre o cuidado, em relação à segurança, ao desenvolver novas funcionalidades utilizando determinado CMS.

Visto todo este cenário, percebemos assim que esta é uma questão que deve ser considerada, pois a negligência pode levar uma organização a ter sérios problemas quanto a roubo de informação, perda da vantagem competitiva e ou informações erradas sendo divulgadas em seu próprio site. Este trabalho visa, então, discutir sobre a segurança da informação no desenvolvimento de *plugins* de sites utilizando os CMS que trabalham no ambiente web.

1.1. OBJETIVOS

O presente trabalho tem por objetivo geral discutir, com ênfase no gerenciador Wordpress, as ameaças e vulnerabilidades que podem existir em vários gerenciadores de conteúdo utilizados na *web* devido ao uso e desenvolvimento de novas funcionalidades através de *plugins*.

Para tal precisamos conhecer um pouco mais sobre os gerenciadores, como podem ser estruturados, que ameaças podem comprometer o sistema, como podem se

[1] Empresa americana sediada na Califórnia, fundada em 1984 pelo casal Len Bosack e Sandy Lerner que oferece soluções para redes e comunicações e é uma grande referência da área.

tornar vulneráveis, em que aspectos de segurança podemos investir para conseguirmos maior tranquilidade.

Este estudo limita-se a discutir os aspectos de segurança da informação relacionados ao desenvolvimento da plataforma que vai gerenciar o conteúdo na *web* e não contempla a segurança física das máquinas, equipamentos, rede e pessoas e nem a segurança do dispositivo daquele que acessa a informação via internet.

1.2. JUSTIFICATIVA

Visto que atualmente as informações assumiram um papel muito importante na competitividade das empresas e que estas mesmas informações sem a segurança devida podem fazer com que uma grande empresa perca todo o seu potencial competitivo diante de seus concorrentes, o presente trabalho se justifica pela grande contribuição que pode trazer para aqueles que desconhecem as vulnerabilidades de CMS e ou estão iniciando seus trabalhos com desenvolvimento neste meio tecnológico.

A segurança da informação é algo imprescindível diante da quantidade e da importância desta que é gerada e disponibilizada na web. A necessidade de discutir sobre o cuidado, com foco na segurança, ao desenvolver novas funcionalidades utilizando determinado CMS surgiu após perceber, em certos momentos, que existiam lacunas no conhecimento de algumas pessoas e por desenvolver vários trabalhos utilizando alguns destes gerenciadores de conteúdo.

Os CMS têm papel importante na gestão de sites da *web* atualmente, pois a informação muda muito rapidamente e os CMS proporcionam agilidade e rapidez na tomada de decisão, além de diminuir os custos e aumentar a segurança ao disponibilizar as informações na *web*. Como o trabalho foi desenvolvido em cima de gerenciadores de conteúdo de código aberto, com certeza será muito importante para vários usuários, pois são gerenciadores amplamente utilizados e com poucos trabalhos acadêmicos sobre este assunto específico.

Desta forma, o capítulo 2 fala sobre os assuntos relacionados ao referencial teórico do trabalho: a gestão de conteúdo e os sistemas, aprofunda um pouco mais nos sistemas de gerenciamento de conteúdo e fala também de segurança da informação.

O capítulo 3 fala como foi a metodologia do trabalho, quais foram os passos para a execução desta pesquisa.

O capítulo 4 fala a respeito daquilo que está relacionado ao CMS e à segurança dele como: os *plugins* e as vulnerabilidades e ameaças que podem comprometer os gerenciadores, os assuntos deste capítulo formam a parte principal deste trabalho.

O capítulo 5 fala sobre as considerações finais desta pesquisa e aponta sugestões para trabalhos futuros.

2. REFERENCIAL TEÓRICO

Neste contexto apresentado é muito importante para as empresas aprenderem a fazer a gestão do conhecimento que existe dentro do seu próprio ambiente, sejam através de seus documentos, colaboradores, gestores, fornecedores e também de seus clientes, pois toda informação proveniente deste ambiente pode proporcionar a ela conhecer melhor: o mercado, a necessidade de seus consumidores, os desafios que deverá enfrentar entre o que a empresa conhece e o que não e para onde deverão seguir os rumos de seu negócio.

A gestão do conhecimento

> Visa favorecer a organização por meio de seu próprio conhecimento adquirido e desenvolvimento e a partir do conhecimento colhido no ambiente externo (experiência de concorrentes, influências culturais, inovações tecnológicas, etc.). Essa gestão se preocupa com as condições organizacionais, localização, geração e partilha do conhecimento, e das ferramentas a serem utilizadas na comunicação e organização de determinado conteúdo. (REBOUÇAS, 2014, *on-line*).

Assim como nos diz Rebouças (2014, *on-line*) não é garantido que uma organização que saiba muito terá um melhor nível de competitividade, ela precisa gerenciar bem seu conhecimento que além de precisar de suportes, irá necessitar também: de gestão, de processo de armazenamento, de cuidado na guarda e segurança das informações, gerenciamento e canais para sua disseminação.

É neste ponto que a gestão do conteúdo, através dos sistemas gerenciadores de conteúdo, irá favorecer a gestão do conhecimento e serão estes, gestão de conteúdo e sistemas gerenciadores de conteúdo, os nossos assuntos da próxima seção.

2.1. GESTÃO DE CONTEÚDO E OS SISTEMAS

Lemos (2000, *on-line*) nos explica em seu artigo *Conteúdo: quem faz, como faz* que o termo conteúdo passou a ser muito utilizado, mas que não podemos encará-lo como mais um modismo no mundo das tecnologias da informação, pois o termo tem significado diferente da palavra informação. Esta é mais próxima de um conjunto de dados estáticos e em contrapartida a palavra conteúdo traz em si certo valor, coerente, consistente e fundamentado em uma argumentação sólida como, por exemplo, quando mencionamos que alguém disse ‘algo com conteúdo’.

Sabemos, portanto, que todo o conteúdo colocado em um site não é somente um conjunto de informações qualquer, mas contém valor agregado a ele que foi pensado e trabalhado pelos profissionais que o disponibilizaram. Sendo assim esses profissionais precisam entender mais do negócio que o site trata do que da área técnica especializada em tecnologia da informação, desenvolvimento de sites e computação. “O conteúdo é o

mecanismo de alimentação para todos os processos de negócio. E sempre foi.” (MOORE, 2001) “Tradução nossa”.

Segundo Lapa (2004, p.37) “no âmbito da tecnologia da informação, o gerenciamento de conteúdo pode ser visto como um dos maiores desafios, principalmente para empresas que utilizam internet, intranet e extranet em seus processos de negócio.” O mesmo autor nos afirma também que “o motivo dessa afirmação é fundamentado no argumento de que o volume de conteúdo publicado cresce explosivamente.” (LAPA, 2004, p.37).

> [...] As TI – principalmente a Internet –, ao mesmo tempo que têm um papel fundamental no processo de geração da informação, desempenham também uma função primordial na criação das condições de tratamento e interpretação da informação. A Internet e as tecnologias que a cercam, alteraram significativamente, portanto, os estoques e os fluxos de informação disponíveis e a maneira como a informação é consumida, digerida e discernida.
>
> A Internet vem alterando drástica e irreversivelmente as relações contemporâneas entre homem/informação/conhecimento porque, ao mesmo tempo em que propicia a expansão quase infinita dos estoques e fluxos de informação, demanda ferramentas e processos que auxiliem o acesso, o tratamento e a interpretação da informação.
>
> Uma das principais implicações da explosão informacional produzida pela Internet para o mundo dos negócios incide principalmente sobre a questão da administração, da gestão dos fluxos e dos estoques de informação online e em tempo real. E mais importante: essas mudanças informacionais afetam o processo de decisão empresarial. (LEMOS, 1998, *on-line*).

Desta maneira no final dos anos 90 surgiram ferramentas que ajudaram esses profissionais e empresas a disponibilizarem o conteúdo de seus sites de uma forma mais rápida sem depender no momento deste trabalho dos profissionais técnicos e, além disto, estas ferramentas ajudaram: a atender a demanda que surgia para criação de novos sites, na integração de aplicações corporativas via internet e melhoraram cada vez mais, refinando, o conteúdo de outros sites que já existiam. (LAPA, 2004. p.37).

Atualmente essas ferramentas são o que denominamos de Sistemas de Gerenciamento de Conteúdo (do inglês, *Content Management Systems* – CMS) e é sobre este assunto que vamos aprofundar um pouco mais na próxima seção.

2.1.1. Sistemas de gerenciamento de conteúdo

Um sistema gerenciador de conteúdo de acordo com Quinn e Gardner-Madras “[...] é um pacote de software que permite que você construa um site que pode ser rápido e facilmente atualizado por seus funcionários não técnicos. Estes sistemas de código aberto são criados e apoiados por uma comunidade de desenvolvedores.” (2010, p.5) “Tradução nossa”.

Além disto:

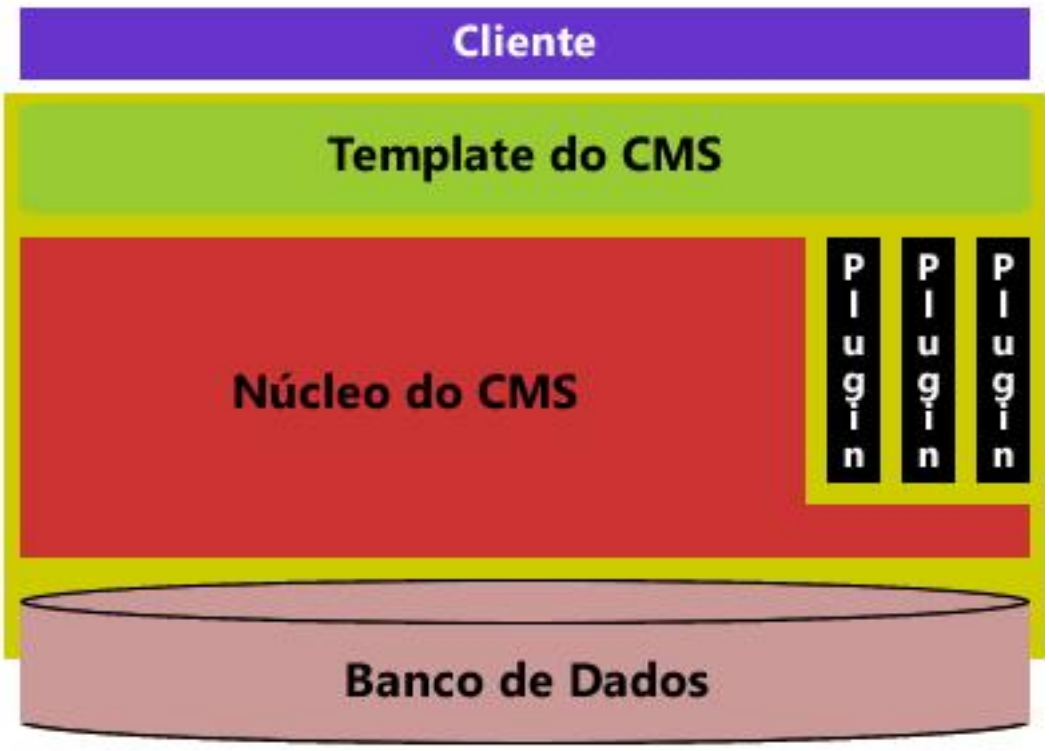
> A ideia básica por trás de um CMS é a de separar o gerenciamento do conteúdo do design gráfico das páginas que apresentam o conteúdo. O design das páginas que apresentam os conteúdos são colocados em arquivos chamados moldes (templates), enquanto o conteúdo é armazenado em banco de dados ou arquivos separados. Quando um usuário solicita uma página, as partes são combinadas para produzirem a página HTML padrão. A página resultante pode incluir conteúdos de diferentes fontes. (BAX; PEREIRA, 2000)

Ainda segundo estes mesmos autores:

> O CMS deve permitir que os próprios colaboradores, no papel de autores, criem seus conteúdos sem necessidade de intermediários, utilizando os diversos programas disponíveis. Em seguida, estes conteúdos são armazenados em repositórios centralizados para serem tratados (gerenciados, padronizados, formatados e publicados no website) através do CMS. O CMS deve gerir também as revisões, atualizações e o controle de acesso, garantindo confiabilidade ao que será publicado e segurança quanto à propriedade e a autoria dos conteúdos. (BAX; PEREIRA, 2000)

Sendo assim, um CMS para ser considerado como tal deve garantir algumas funcionalidades essenciais e para garantir a execução destas funcionalidades e algumas outras necessárias ao funcionamento do sistema cada gerenciador tem em seu núcleo um conjunto de funções que podem ser utilizadas pelas outras partes do CMS como os *plugins* ou *templates*. Em linhas gerais, um CMS pode ser dividido em quatro principais partes estruturais que seguem:

Figura 1: Partes que se dividem um CMS



FONTE: Elaborado pelo autor.

- **Núcleo ou *core* do CMS:** é a parte central do sistema, onde se encontram todas as principais funções para funcionamento do sistema e para interações ou extensão para que outras funcionalidades possam ser anexadas ao gerenciador, é a parte que geralmente é mantida pela comunidade de desenvolvedores quando são de código aberto, encontra-se em constante evolução para garantir segurança, agilidade e compatibilidade com as novas tecnologias;
- ***Plugins* ou extensões (componentes ou módulos):** conjunto de arquivos que possuem funções específicas codificadas em arquivos para a execução de determinadas tarefas que o sistema necessita fazer, mas que não fazem parte do núcleo de funções do CMS. Geralmente esses *plugins* ou extensões são desenvolvidos por terceiros e é neste momento que temos de ter cuidado com o tipo de funcionalidade e a qualidade de codificação da extensão que estamos colocando dentro do nosso próprio sistema, pois um *plugin* pode comprometer todo o sistema e informações contidas nele;
- **Modelo ou *template*:** conjunto de arquivos que apresentarão o conteúdo do sistema conforme o *layout* escolhido. É responsável pela parte visual do site, cores, tamanhos, fontes, seções ou áreas de conteúdo e também pode conter algumas funcionalidades diferentes estendidas do núcleo do CMS fazendo com que o sistema fique um pouco mais completo em relação ao que se deseja que ele execute;
- **Banco de dados:** local que na maioria das vezes está externo ao CMS, podendo ser alocado remotamente em relação à máquina que se encontra o sistema, mas que dependendo do CMS pode ser alocado dentro da estrutura dele. É onde são guardados todos os conteúdos, toda a parte informacional, tudo o que é possível armazenar dentro de um banco de dados hoje, de acordo com o tipo de banco de dados compatível com o CMS do sistema.

Na atualidade basicamente todos os CMS mundialmente conhecidos e mantidos podem ser divididos nessas quatro partes. Logo abaixo veremos um mapa com os CMS mais utilizados em vários países, entre eles, os que vamos citar algumas de suas características ao longo deste trabalho.

O mapa elaborado por Gelbmann (2012) da W3Techs nos mostra quais os sistemas de gerenciamento de conteúdo mais populares no mundo. Como podemos perceber na Figura 2, desde 2012 o Wordpress já possuía o primeiro lugar em vários lugares do mundo, principalmente nos países do Norte e América do Sul, Europa e Oceania, muitos países da Ásia, entre eles Rússia e Índia, porém poucos países da África.

Constatamos que em seguida temos o Joomla que domina um bom número de países da África como, por exemplo, Argélia, Marrocos e Nigéria, vários países da América Central e do Sul, como Venezuela, Equador e Cuba, Grécia e Bósnia na Europa e

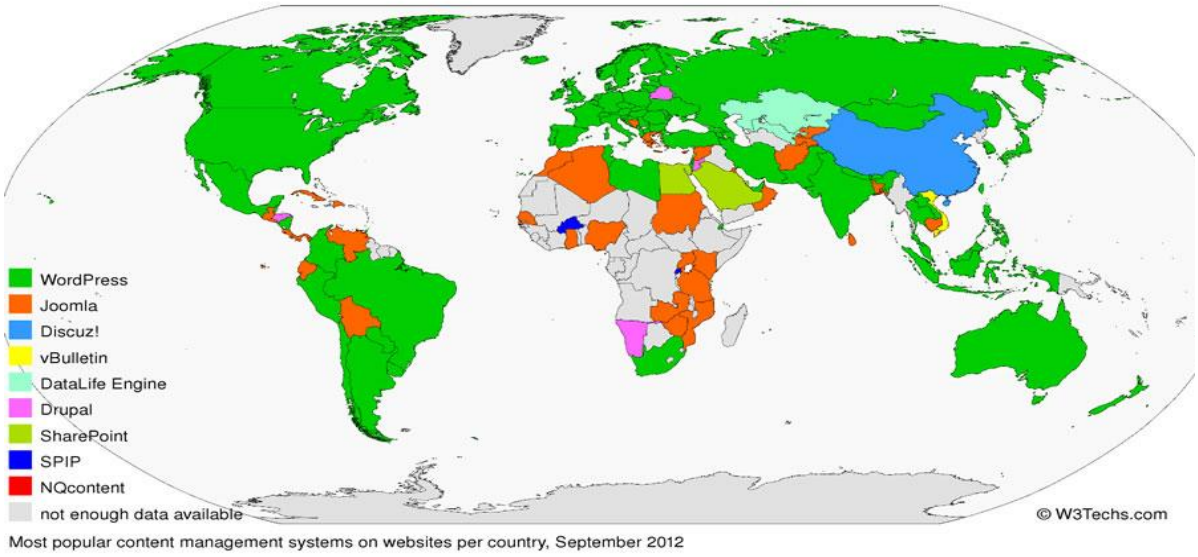
Afeganistão dentre outros países da Ásia. O Discuz! é um sistema chinês não muito conhecido porém de bastante expressão dentro de seu mercado doméstico, com 49% de participação.

O vBulletin é muito popular em países de língua árabe e somente alcança o primeiro lugar no Vietnã. O DataLife Engine é um sistema russo que ocupa o terceiro lugar em seu país de origem mas alcança o primeiro lugar no Cazaquistão e Uzbequistão.

O Drupal é considerado um dos três maiores sistemas, associado ao Wordpress e ao Joomla, porém alcança o primeiro lugar, de acordo com o mapa, somente em alguns países como Belarus, Jordânia, Honduras e Namíbia. O Drupal é o CMS mais popular em sites com domínios .edu e .gov.

O Share Point é o mais utilizado na Arábia Saudita, Egito, Catar e Líbano, e é bastante utilizado entre os sites com domínio .mil, porém não aparecem com destaque na Figura 2. Os dois últimos CMS que aparecem na figura são mais exóticos, o francês SPIP não domina na França, onde se encontra na quinta posição, mas aparece na figura dominando Ruanda e Burkina Faso e o CMS NQcontent que aparece na posição 84 da pesquisa global de gerenciadores e aparece como mais popular em Chipre.

Figura 2: Sistemas gerenciadores de conteúdo mais populares em sites web por países.



FONTE: Gelbmann, 2012. “Tradução nossa”.

Além destes, outros dois sistemas que são expressivos, mas surpreendentemente não aparecem no mapa são: o Blogger que ocupa a quinta posição global e é o segundo sistema mais popular em vários países como na Índia e o Typo3 que ocupa o sexto lugar geral no ranking de CMS e, além disto, é muito popular em países de idioma alemão, mas que fica atrás do Wordpress na Alemanha, Áustria e Suíça.

2.2. SEGURANÇA DA INFORMAÇÃO

Segundo ABNT (2005) “Segurança da informação é a proteção da informação de vários tipos de ameaças para garantir a continuidade do negócio, minimizar o risco ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio.”

Assim, a ABNT também nos diz que:

> A segurança da informação é importante para os negócios, tanto do setor público como do setor privado, e para proteger as infra-estruturas críticas. Em ambos os setores, a função da segurança da informação é viabilizar os negócios como o governo eletrônico (*e-gov*) ou o comércio eletrônico (*e-business*), e evitar ou reduzir os riscos relevantes. (ABNT, 2005)

Podemos dizer que uma informação está segura quando está livre de ser divulgada, modificada ou destruída devido a algum acesso intencional ou não autorizado (FINNE, 2000). Assim, definiremos a seguir as três principais dimensões que compõem a segurança da informação: confidencialidade, integridade, e disponibilidade.

Confidencialidade quer dizer que a informação está livre de qualquer acesso não autorizado (SCHULTZ et al., 2001), seja do interior da organização ou a partir de fora dela (LEE et al., 2004; WANG et al., 1996.). “Manter a confidencialidade pressupõe assegurar que as pessoas não tomem conhecimento de informações, de forma acidental ou proposital, sem que possuam autorização para tal procedimento.” (BRASIL. Tribunal de Contas da União, 2007, p.26)

Integridade da informação quer dizer que esta não foi modificada ou falsificada e está compatível com seu estado original (SHIH; WEN, 2003) e livre de erros (WANG et al., 1995, 1996; BOVEE et al., 2001). “Sinaliza [...] a conformidade dos dados transmitidos pelo emissor com os recebidos pelo destinatário. A manutenção da integridade pressupõe a garantia de não violação dos dados com o intuito de alteração, gravação ou exclusão, seja ela acidental ou proposital.” (BRASIL. Tribunal de Contas da União, 2007, p.26)

Disponibilidade é quando a informação está acessível de qualquer lugar necessário para a utilização e consumo desta. Quanto maior a disponibilidade da informação, mais vulnerável pode se tornar o sistema, do contrário, quanto menor mais afeta a disponibilidade da informação (FINNE, 2000; SCHULTZ et al., 2001; SHIH; WEN, 2003; FLOWERDAY; VON SOLMS, 2005). Sendo assim: “Manter a disponibilidade de informações pressupõe garantir a prestação contínua do serviço, sem interrupções no fornecimento de informações para quem de direito.” (BRASIL. Tribunal de Contas da União, 2007, p.26).

Se conseguirmos garantir essas três dimensões em todo o ciclo de vida da informação, ou seja, ao gerar, guardar, transmitir e também no seu descarte podemos dizer que conseguimos garantir a segurança desta informação. E isto fora do ambiente virtual nos parece ser mais fácil, pois a via de acesso a esta informação é somente uma, a física.

Oposto a isto, como podemos garantir a segurança da informação através das dimensões citadas, pensar que um software é totalmente seguro é errado. No estudo *A Consumers Guide to Content Management Systems for Nonprofits* realizado em 2014 pela Idealware, Andrei, Quinn e Pope nos diz que em relação à segurança:

> Todo software, por natureza, tem vulnerabilidades, então quando você está comprando um CMS, não é uma simples questão de um sistema seguro contra um inseguro — o CMS ideal é aquele com o menor número de vulnerabilidades identificadas e que resolve estas vulnerabilidades mais rapidamente. (ANDREI; QUINN; POPE, 2014, p.17). “Tradução nossa”.

Geralmente os softwares de código aberto têm como mantenedores de suas versões uma comunidade de usuários e o “Quão eficaz que a comunidade é em seu trabalho é uma consideração importante ao escolher um CMS de código aberto.” (GARDNER-MADRAS; QUINN, 2010, p.8) “tradução nossa”. Assim, quanto mais ativa é a comunidade de desenvolvedores que mantém o núcleo do CMS atualizado e livre de risco, melhor será o gerenciador quanto a sua segurança.

Os softwares estão em constante evolução – por conseqüência também da evolução das linguagens de programação – e é por isso que a cada nova versão de um CMS, além de trazer novos recursos e inovação tecnológica para competir com seus concorrentes em velocidade de processamento e melhores ferramentas do ponto de vista destas serem facilitadoras do trabalho cotidiano de seus usuários, os gerenciadores também podem trazer consigo algumas vulnerabilidades, falhas ou criar incompatibilidades com outros softwares gerando inconsistências na comunicação entre ambos.

De acordo com o conceito existente na Cartilha de Segurança para Internet do CERT.br “vulnerabilidade é definida como uma falha no projeto, implementação ou configuração de um software ou sistema operacional que, quando explorada por um atacante, resulta na violação da segurança de um computador” (CERT.BR, 2006, p.7) ou sistema podendo ocasionar o acesso, roubo, alteração, dentre outras possibilidades que o mesmo possa permitir que o atacante faça.

Além disso, em alguns casos, de acordo com o CERT.br, muitas dessas vulnerabilidades podem ser exploradas remotamente, ou seja, qualquer pessoa que tenha acesso à Internet poderá explorar tal vulnerabilidade obtendo acesso não autorizado ao computador ou sistema vulnerável se localizando em qualquer lugar do mundo, desde que o computador ou sistema atacado seja acessível pela internet ou pela mesma rede do atacante.

Como estamos falando de segurança da informação com foco no desenvolvimento de *plugins* para sites gerenciados por CMS não podíamos deixar de citar que os gerenciadores de conteúdos possuem *plugins* que trabalham diretamente na

segurança deles, porém eles podem não ser capazes de fechar todas as brechas e colocar o sistema livre de todas as vulnerabilidades existentes em outros *plugins* e temas.

Um estudo feito por Sreedharan (2014) verificou quatro dos principais *plugins* de segurança do Wordpress, escolhidos com base nas suas popularidades nos fóruns do Wordpress e na classificação deles no repositório do gerenciador quanto à segurança, ou seja, a capacidade destes *plugins* barrarem um ataque proveniente de outro *plugin* que esteja vulnerável.

Um detalhe importante que o estudo aponta é que “quase todos os *plugin*s de segurança do Wordpress focam principalmente em segurança do *login*” (SREEDHARAN, 2014, *on-line*) “tradução nossa”, os chamados *login* por força bruta (*Brute Force*), porém estes tipos de ataques são minoria em relação a outros.

Para o experimento realizado por Sreedharan (2014) os *plugins* escolhidos foram testados um a um separadamente sendo configurados de acordo com as suas características e as informações disponibilizadas pela equipe de desenvolvimento nas páginas dos respectivos *plugins* no repositório do Wordpress.

O resultado do estudo vem corroborar o que foi dito anteriormente: que os *plugins* não conseguem fechar todas as brechas, pois descobriu que alguns determinados tipos de vulnerabilidades que, teoricamente deveriam ser barradas por tais *plugins*, não puderam ser bloqueadas com a utilização destes, vejamos os resultados:

Quadro 1: *Plugins* de segurança contra vulnerabilidades recentes em outros *plugins*

Plugins de segurança **Vulnerabilidades (Plugin)**	**Bullet Proof Security (BPS)**	**Wordfence Security**	**iThemes Security**	**All In One WP Security**
Remote Code Execution (Timthumb)	Não bloqueia	Não bloqueia	Não bloqueia	Não bloqueia
Remote Code Execution via comentários (WP Super Cache e W3 Total Cache)	Não bloqueia	Não bloqueia	Não bloqueia	Não bloqueia
SQL Injection (Firestorm Real Estate)	Bloqueia ativando .htaccess do plugin e por GET, por POST não.	Não bloqueia	Filtro URL longa deve ser ativado, porém facilmente burlado.	Não bloqueia
SQL Injection usando requisições HTTP POST (Custom Contact Forms)	Não bloqueia	Não bloqueia	Não bloqueia	Não bloqueia

FONTE: Elaborado pelo autor.

Assim, vimos que quase todos os ataques verificados pelo estudo obtiveram sucesso através dos *plugins* de segurança escolhidos, porém não podemos descartar todos estes tipos de *plugins* e dizer que não funcionam, pois muitos deles ajudam a diminuir os riscos relacionados a algumas vulnerabilidades, principalmente àquelas que já são

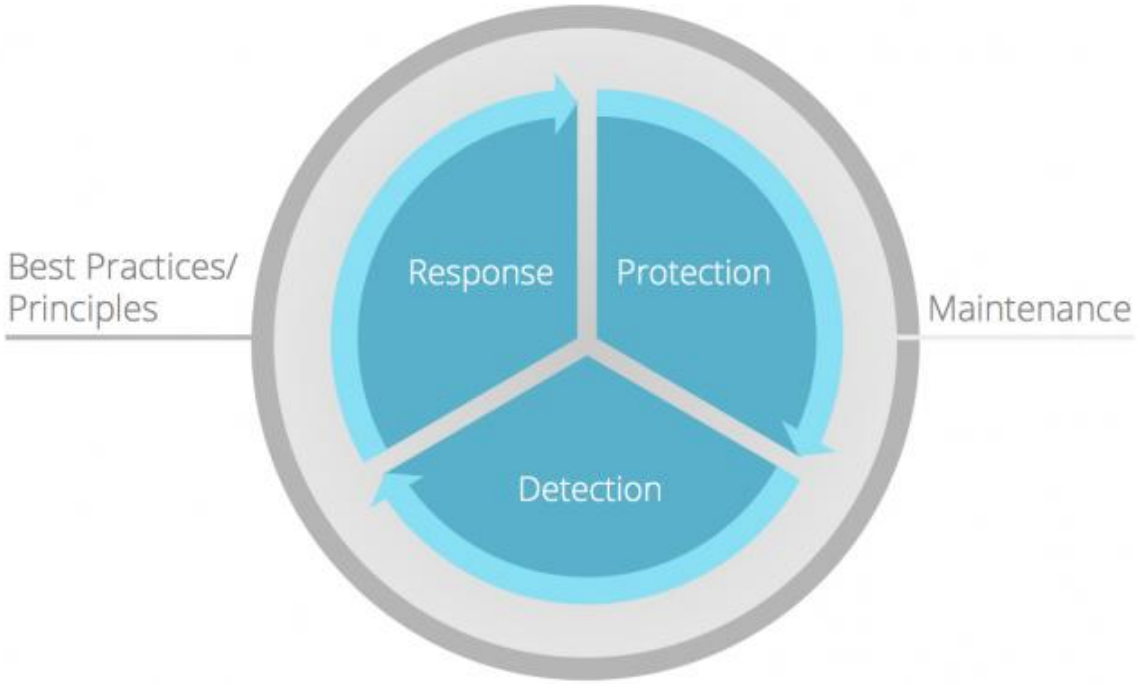
conhecidas, que se não for assim podem tornar a manutenção de segurança muito onerosa e com custos de tempo e/ou dinheiro muito elevados.

Além disto, muitas empresas investem no desenvolvimento de *plugins* para todos os tipos de funcionalidades e não somente aos relacionados à segurança. Fazendo uma busca rápida no repositório de *plugins* do Wordpress encontramos, em vários deles, empresas que desenvolvem e possuem versões gratuitas que oferecem muitos recursos e versões pagas que disponibilizam outras funções que complementam os recursos da versão gratuita pela qual estas empresas podem obter o lucro pretendido.

Um dos exemplos de empresas que investem em segurança treinando usuários, desenvolvendo *plugins* e dando consultoria a respeito de vulnerabilidades existentes e recentemente descobertas em *plugins* e sistemas gerenciadores de conteúdo é a Sucuri (sucuri.net).

Em um artigo publicado por Perez (2014), co-fundador e CEO da empresa Sucuri, nos mostra como os *plugins* de segurança podem ser divididos conforme suas características mais marcantes e baseado no ciclo comum de segurança da informação que é definido pela proteção, detecção e resposta. Na Figura 3 podemos ver o ciclo comum de segurança da informação incrementado por Perez (2014) para ser aplicado em sites da web.

Figura 3: Ciclo de segurança da informação em sites web



FONTE: Perez, 2014.

A segurança da informação em um site começa a partir do limite dos princípios e boas práticas de programação para web. Desde o início do desenvolvimento de um trabalho de programação de um site, independente da plataforma de desenvolvimento que será utilizada, devemos nos preocupar com os padrões de desenvolvimento, pois o simples não

cumprimento de uma das regras de boas práticas podem fazer com que uma organização inteira sofra os efeitos negativos disto.

Algumas regras de boas práticas de programação para sistemas podem também ser aplicadas ao desenvolvimento de sites e ou *plugins*, pois tratam as questões relacionadas ao desenvolvimento e manutenção de uma forma mais generalizada. Sommerville (2011) nos fala sobre as diretrizes de programação confiável ou também chamadas de boas práticas de programação que podemos numerar oito principais delas:

1. Limitar a visibilidade das informações de um sistema – Consiste em apresentar as informações apenas àqueles que necessitam dela para executar suas tarefas. Isto se aplica não somente aos usuários finais, mas também dentro do código desenvolvido, por exemplo, para um *plugin*, em que determinada função precisa ter acesso a apenas um atributo do objeto e não todos os atributos do objeto.

2. Verificar e validar todas as entradas de dados – Todos os sistemas, sejam CMS ou *plugin*, precisam de uma entrada de dados através do usuário, de um sensor ou coletando de outro sistema para executarem suas ações. Neste momento, muitas vulnerabilidades podem obter sucesso, pois caso estes dados de entrada não sejam verificados e validados de acordo com o tipo de entrada que é necessária, uma das vulnerabilidades possíveis, que vamos discutir na seção 4.2, poderá ser explorada por um usuário malicioso ou até mesmo um erro poderá ser gerado por quem fornece a entrada, sem a intenção para tal.

3. Tratar todas as exceções que ocorrerem - Algum evento inesperado ou erro pode acontecer enquanto o programa está em execução, a isto chamamos exceção. Estas podem ser causadas por hardware ou software e devem ser gerenciadas para não causar falhas de sistema. "Dessa forma, fornece-se um grau de tolerância a defeitos - o programa detecta os defeitos e pode tomar medidas para se recuperar deles." (SOMMERVILLE, 2011, p.250). Em vários momentos é possível continuar a operação após tratar a exceção, visto que muitos desses eventos inesperados ou erros são transitórios.

4. Minimizar o uso de construções de código propensas a erro – A maioria das falhas de um programa provém de erros humanos. De acordo com Sommerville (2011, p.250), no fim da década de 1960 algumas construções de código ficaram evidentes que induziam os programadores a erros em relação a outras. Ao desenvolver sistemas críticos em relação a segurança, alguns padrões proíbem que estas construções sejam usadas, porém em vários outros momentos estas técnicas são muito úteis e possuem casos específicos para serem utilizadas, podendo ser controladas dentro de tipos abstratos de dados ou objetos.

5. Utilizar recursos de reiniciação do sistema - Em sistemas de comércio eletrônico, somente após todas as etapas e finalização da compra é que o banco de dados

do sistema é atualizado. Problemas podem ocorrer e caso a quantidade do produto escolhido por um usuário seja debitada do estoque e a compra não seja efetuada por causa de algum evento inesperado, uma inconsistência será gerada no banco. Em outros casos, em que a execução da operação acontece em longos períodos de tempo, um evento desses tem grandes chances de ocorrer e causar a perda de tudo que já foi executado. Para todos esses casos é necessário guardar o estado da execução ou dos dados que estão sendo utilizados para que o programa possa reiniciar do ponto em que parou sem causar outros problemas ao sistema.

6. Fazer verificações em tamanhos de vetores - Algumas linguagens de programação sempre verificam os limites de um vetor ao acessar posições deste para gravar, ler ou apagar, porém outras acessam vetores linearmente a partir do início deste calculado o deslocamento em relação ao índice que será acessado, mesmo que esta área da memória faça parte ou não do vetor. Assim surgem as vulnerabilidades, pois conseguirá ler, alterar ou apagar informações que não fazem parte do vetor. Mesmo nestas linguagens de programação que não possuem verificações dos limites de vetor, isto pode ser garantido pela codificação de vetores através de um tipo abstrato de dados.

7. Ativar timeouts ao chamar componentes externos - Em alguns sistemas mais complexos e de funcionamento distribuído, seus componentes podem funcionar em computadores diferentes e desta forma se comunicam através da rede. Um componente pode ficar esperando a resposta de outro indefinidamente para concluir uma ação solicitada por um usuário, caracterizando assim uma falha silenciosa. Ao chamar o componente externo e ativar um timeout, evitamos a ocorrência desta falha, pois ao terminar o tempo, o sistema assume que houve uma falha e retorna a execução para o componente que chamou, o qual poderá informar o usuário sobre o ocorrido permitindo que o usuário decida o que fazer ou executar alguma outra tarefa.

8. Dar nomes às constantes que representam valores reais - Vários programas utilizam valores do mundo real que não são alterados enquanto o programa está em execução, chamados constantes. Esses valores geralmente são utilizados em vários locais do programa, em várias funções e um erro de digitação pode fazer com que algo esteja errado em seu programa, mas será difícil de ser detectado. Além disto, quando este valor for alterado, se o seu programa possuir um local onde se encontram todas as constantes utilizadas por ele, será muito mais fácil atualizar o novo valor e existirá menores chances de um erro ocorrer, pois ao alterar neste local, o valor da constante será alterado em todos os lugares em que é chamada pelo nome.

Assim como nos preocupamos com o desenvolvimento através destas oito regras, não podemos nos descuidar da manutenção deste site ou *plugin*, pois está totalmente relacionada à segurança das informações. Um descuido, uma manutenção não

efetuada ou postergada pode também comprometer as informações gerenciadas pelo sistema e levar a organização a arcar com graves prejuízos como perda de competitividade ou credibilidade devido ao roubo de informações.

A manutenção de segurança do site perpassa pelas três características do ciclo comum de segurança da informação como vimos na Figura 3. Apenas executando todo o ciclo de segurança da informação é que conseguimos obter o máximo de aproveitamento das ferramentas de segurança que possuímos e, mesmo assim, ainda corremos o sério risco de ter o site invadido por causa de outras vulnerabilidades, pois nunca podemos dizer que estamos 100% protegidos.

Ainda de acordo com Perez (2014) raramente vamos ter um *plugin* de segurança que trabalhe bem em todos os aspectos do ciclo de segurança e para que o usuário consiga estabelecer melhor qual o tipo de *plugin* ele necessita para seu site, baseando-se neste ciclo de segurança da informação, estes foram divididos em quatro grandes categorias de *plugins* de segurança: Prevenção, Detecção, Auditoria e Utilidades, que serão definidas abaixo.

Prevenção - Estes *plugins* trabalham no perímetro do seu site prevenindo que as vulnerabilidades conhecidas sejam exploradas. Um dispositivo muito difundido e muito utilizado por quem é da área de TI (Tecnologia da Informação) é o *firewall* para site, mas estes não são muito eficientes e possuem limitações na camada de aplicação, pois um ataque tem de acertar o servidor para obter uma resposta de bloqueio como vimos na Quadro 1.

Detecção - Para lidar com os problemas não conhecidos, pois a proteção não é 100% segura, existem os *plugins* de detecção. “Os atacantes têm a capacidade de encontrar novos pontos de entrada (ou seja, identificar novos vetores de ataque)” (PEREZ, 2014, *on-line*). Assim, os *plugins* desta categoria atuarão quando algum ataque ultrapassar o perímetro da prevenção do seu site. Eles utilizam diferentes mecanismos, como a verificação de integridade de arquivos e presença de *malware* entre outros, para detectar que há algum problema no site.

Auditoria - Talvez seja o que a maioria das pessoas não gostaria de realizar, pois temos hoje inúmeros *plugins* disponíveis para automatizar quase todo o nosso trabalho e quando falamos em acompanhar sistemática e ativamente as atividades realizadas no site pode ser incoerente. Esses *plugins* fornecem ferramentas que coletam informações sobre tudo o que é feito no site e disponibiliza de alguma forma para que seja analisado pelo administrador do site. Isto pode ser mais eficiente que vários *plugins* de outros tipos de categorias.

Utilidades - Nesta categoria se encaixam aqueles *plugins* que fazem de tudo um pouco. São os que mais possuem exemplares e os que todos querem utilizar, pois facilitam muito o trabalho de manutenção de segurança. Podem ser comparados com caixas de

ferramentas e por isso podem ser exaustivos em opções de configuração, podendo ser também muito eficientes para administradores ativos.

Assim, saber que tipo de *plugin* utilizar vai depender do perfil de segurança do seu site e do objetivo quanto à administração da segurança dele a que se quer ter. De acordo com o artigo de Perez (2014) essas categorias foram feitas voltadas para os *plugins* do gerenciador Wordpress, porém podem ser aplicadas aos outros vários CMS, pois possuem as mesmas características de proteção para sites visto que as vulnerabilidades e ameaças enfrentadas são as mesmas.

E *plugins*, vulnerabilidades e ameaças são os assuntos que vamos tratar no Capítulo 4 deste trabalho, antes, no próximo capítulo vamos falar sobre a metodologia utilizada.

3. METODOLOGIA

Este trabalho pode ser classificado quanto a sua natureza como uma pesquisa aplicada, pois tem por objetivo gerar conhecimento para ser aplicado na prática, para solucionar ou tentar minimizar um problema específico, como é o caso de vulnerabilidades que podem ocorrer nos gerenciadores de conteúdo através dos *plugins* utilizados de terceiros ou desenvolvidos pelos próprios programadores do site.

No início deste trabalho várias fontes relacionadas ao tema foram selecionadas e lidas. A escolha do tema deste trabalho foi feita com foco na discussão deste problema citado e pela percepção de certa lacuna no conhecimento de algumas pessoas relacionadas ao processo de desenvolvimento de sites em gerenciadores de conteúdo utilizando plugins.

Quanto ao procedimento, a metodologia deste trabalho é classificada como documental, pois foram utilizados alguns estudos, relatórios, livros e documentos técnicos relacionados ao assunto proposto como material. A pesquisa documental segue a mesma linha da pesquisa bibliográfica, porém aquela possui fontes mais diversificadas e não se atém somente a livros e artigos científicos localizados em bibliotecas.

Foram selecionados alguns trabalhos principais que tratam: dos tipos de vulnerabilidades e ameaças existentes em sistemas que funcionam via web; da ocorrência destas em vários tipos de sistemas, inclusive os gerenciadores de conteúdo e; do local onde estas vulnerabilidades ocorrem dentro dos gerenciadores de conteúdo. Além destes trabalhos, foram consultados sites, blogs e relatórios de empresas e ou profissionais que trabalham na área de segurança.

Conforme a escolha destes trabalhos e de acordo com Gil (2002, p.41) esta pesquisa tem objetivo exploratório, pois envolveu pesquisa bibliográfica e complementou, ilustrando com alguns exemplos que “estimulem a compreensão” (SELLTIZ et al., 1967, p.63 citado por GIL, 2002, p.41) e que estão relacionados às principais vulnerabilidades escolhidas de acordo com o material utilizado.

Sendo assim, dentro destes trabalhos utilizados são apontadas tendências de ocorrência de algumas vulnerabilidades e ameaças que são corroboradas por outros trabalhos e dentre estas foram selecionados três tipos, por serem os mais frequentes em alguns destes estudos, para trazermos alguns exemplos de cada um deles para entendermos melhor como isto acontece.

Para o entendimento deste problema foi escolhido uma abordagem com características mais qualitativas, pois não se preocupa em quantificar numericamente as informações para serem analisadas utilizando técnicas estatísticas e sim tenta compreender o fenômeno como um todo, analisando a dinâmica das relações sociais (GERHARDT; SILVEIRA, 2009, p.32).

Por fim, fizemos um apanhado das melhores técnicas para desenvolvimento de sites em gerenciadores de conteúdos diversos e como mantê-los funcionais e livres de grandes riscos devido à falta de atualizações e comprometimento da estrutura pelas vulnerabilidades e ameaças já conhecidas.

4. CMS E SEGURANÇA

Este trabalho buscou enfatizar como os *plugins* podem levar um CMS que possui um núcleo muito seguro a se tornar vulnerável e, por consequência, fazer com que uma organização tenha prejuízos por causa de roubo de informações, divulgação de informações erradas em seus sites dentre outros problemas e para isto vamos conhecer um pouco mais sobre estes componentes estruturais dos CMS na próxima seção.

4.1. PLUGINS

Alguns gerenciadores de conteúdo trabalham com *plugins*, outros com componentes, outros com extensões ou *add-on* e ainda existem alguns outros que trabalham com conjuntos de dois ou três destes itens, porém no seguimento deste trabalho vamos tratar todos estes itens como sinônimos de *plugins* para facilitar nosso entendimento e melhor comunicação visto que nem todas as pessoas interessadas são da área de computação.

De acordo com as boas práticas de programação em CMS quando vamos acrescentar novas funções ao nosso site, colocamos a codificação destas novas funções dentro de um *template* ou *plugin*, pois o núcleo do CMS estará em constante atualização pela comunidade que o mantém e caso acrescentemos estas funções dentro dos arquivos do núcleo todas as funções seriam perdidas quando o CMS fosse atualizado.

Quando desenvolvedores querem disponibilizar uma nova funcionalidade para outros desenvolvedores de outros sites, e ela não está relacionada a um novo *layout*, utilizamos *plugins* para que eles tenham como baixar os arquivos do *plugin* e instalar em seus sites sem alterações no *layout*. Atualmente encontramos repositórios oficiais dos gerenciadores de conteúdo com uma grande quantidade de *plugins* para baixar e sobre isto vamos falar mais adiante.

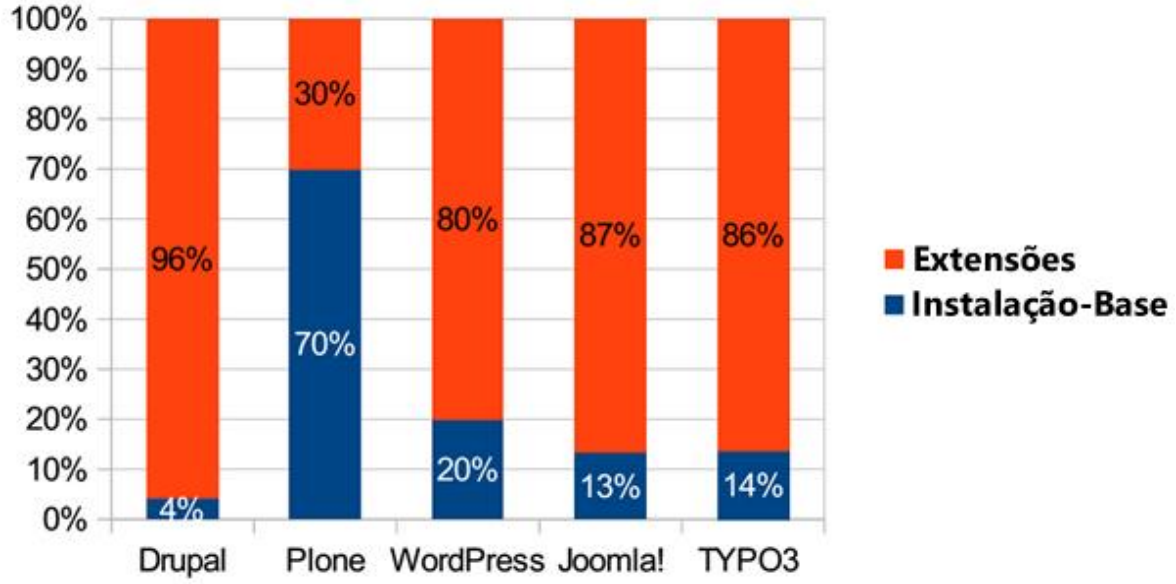
Sendo assim, ocorrerá uma grande chance de já existir um componente desenvolvido por terceiros que cumpra a função necessária. Logo, a probabilidade deste *plugin* existente ser utilizado é muito grande e somente necessite fazer algumas configurações específicas para o site que irá utilizá-lo. Visto isso, podemos afirmar que os *plugins* formam a parte que mais pode tornar o sistema vulnerável, pois estamos inserindo códigos que não conhecemos dentro do nosso sistema.

Estudos demonstram que a maior parte das vulnerabilidades de um site que utiliza um gerenciador de conteúdos *web* acontece através dos *plugins* que este utiliza. Assim como o estudo feito pelo Serviço Federal de Segurança em Tecnologia da Informação da Alemanha (do alemão, *Bundesamt für Sicherheit in der Informationstechnik* – BSI), que

vem corroborar a afirmação anterior, podemos ver que dentre os CMS escolhidos como objeto do estudo, a maior parte das vulnerabilidades ocorre nos *plugins* que o sistema utiliza.

O gráfico abaixo, retirado do estudo citado, com base nos dados coletados entre 2010 e 2012, nos mostra como é grande a contribuição dos *plugins* para a geração das vulnerabilidades quando comparadas àquelas geradas pelo núcleo do sistema. No gráfico fica evidente a diferença entre a contribuição de cada um deles.

Gráfico 1: Vulnerabilidades entre extensões e núcleo dos principais CMS da Web.



FONTE: BREITENSTROM, 2013, p. 67. “Tradução e adaptação nossa”.

Podemos perceber que a maioria das vulnerabilidades encontradas nos gerenciadores do estudo citado acontece nas extensões destes e assim podemos afirmar que as extensões possuem maior probabilidade de se tornarem vulneráveis quando comparadas com o núcleo do sistema.

Sendo assim, como podemos ter certeza que um componente que estenda alguma funcionalidade ao seu site possa ser muito útil e traga muitos benefícios ao longo do desenvolvimento, utilização e manutenção, em contra partida caso este *plugin* tenha falhas, o desenvolvimento e, em um grau mais crítico, a utilização do seu sistema pode estar com a integridade completamente comprometida.

> O uso generalizado de plugins, que são projetados para estender a funcionalidade de um CMS e para alimentar vídeos, animações e jogos, também está provando ser um benefício para meliantes que procuram obter acesso não autorizado a plataformas como WordPress e Joomla. Muitos dos comprometimentos do CMS observados por pesquisadores da Cisco em 2013 podem ser rastreados até plugins escritos na linguagem de script web PHP que foram mal projetados e sem pensar em segurança. (CISCO, 2014, p.54). “Tradução nossa”.

Cada um dos vários CMS utilizados na internet possui uma enorme quantidade de *plugins*/extensões/módulos e podemos encontrar locais específicos, chamados

repositórios, para baixar os *plugins* necessários ao seu site. Geralmente estes locais são mantidos pela comunidade que dá suporte ao núcleo do gerenciador, o que é mais seguro, e é onde podemos buscar e saber mais detalhes sobre as funcionalidades destes *plugins*.

O quadro seguinte nos aponta alguns dados relacionados aos repositórios de cada um dos CMS que constam no estudo citado anteriormente. Dentro desses repositórios cada plugin possui páginas com informações específicas sobre ele. Nos respectivos endereços indicados no quadro conseguimos pesquisar *plugins* com diversas funcionalidades:

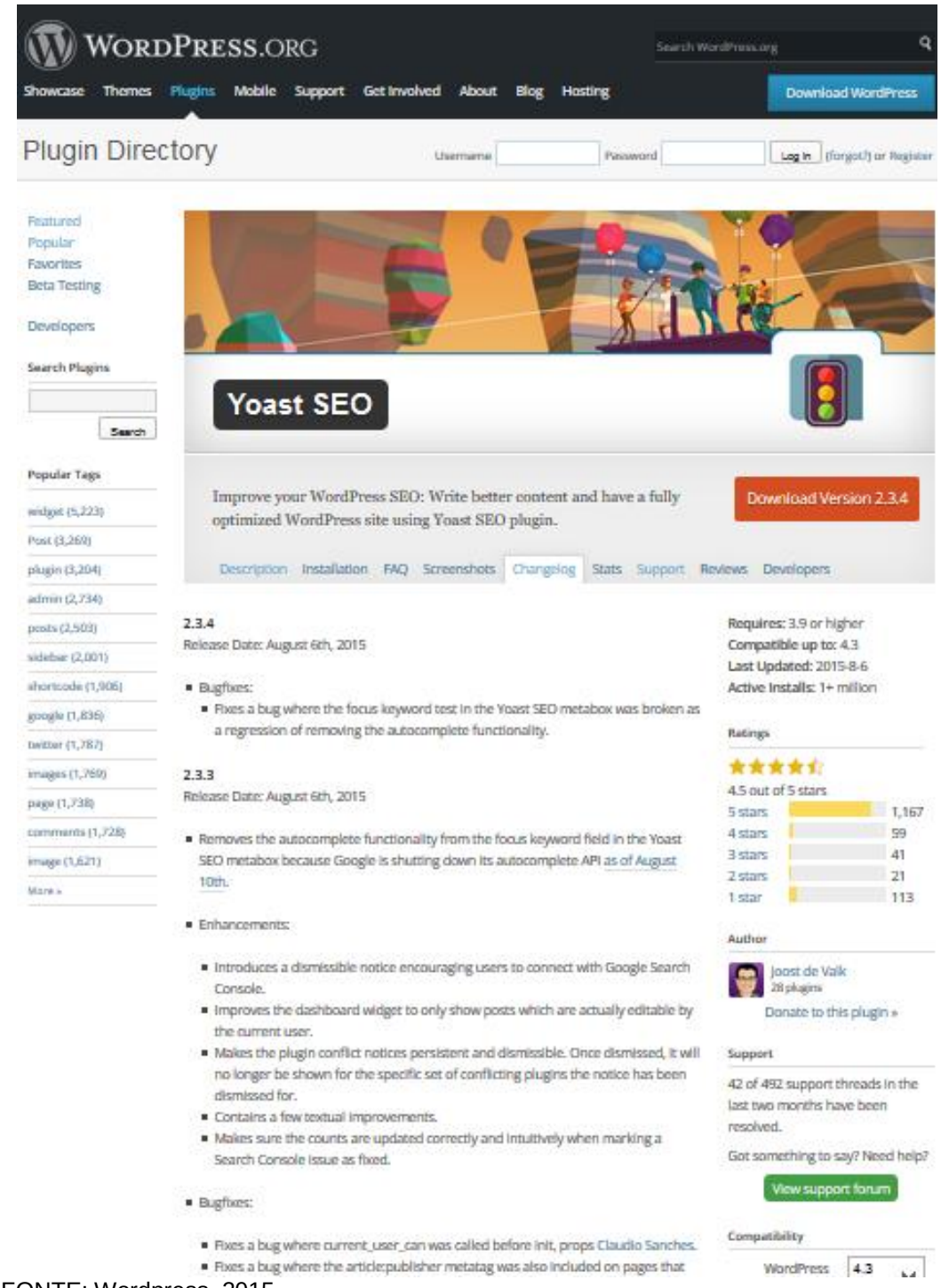
Quadro 2: Quantidade de extensões de cada gerenciador e endereço para download

Gerenciador	**Número de módulos/ extensões/ plugins[1]**	**Endereço para download**
Drupal	16.487	https://www.drupal.org/project/project_module
Plone	203	https://plone.org/products
Wordpress	35.899	https://wordpress.org/plugins/
Joomla!	8.628	http://extensions.joomla.org/
TYPO3	1331	http://typo3.org/extensions/repository/
(1) Dados coletados nos respectivos sites em 05 de fevereiro de 2015 às 17:40:00		

FONTE: Elaborado pelo autor.

É comum encontrar dentro das páginas de cada *plugin* nestes repositórios áreas que possuem comentários de usuários sobre as constatações a respeito do *plugin* e outras com listas de erros reportados pelos usuários e o estado em que o erro se encontra, caso tenha sido ou não solucionado. A imagem a seguir ilustra uma página de *plugin* no repositório do Wordpress.

Figura 4: Página de *plugin* no repositório do Wordpress



FONTE: Wordpress, 2015.

Estes *plugins* geralmente se integram ao gerenciador por meio de uma interface formada por funções abrigadas pelo núcleo e desta forma podem interferir no funcionamento de acordo com uma determinada ação executada dentro do gerenciador, antes ou depois desta, ou funcionando como um filtro para mostrar ou não uma determinada informação. Existem desde extensões mais simples, até outras que modificam totalmente a

funcionalidade principal de seu site deixando o totalmente diferente daquilo que o núcleo dele propõe.

Como exemplo, temos os *plugins* de redes sociais que geralmente são mais simples, mas temos também os módulos de comércio eletrônico em que o site passa de um *blog* que gerencia postagens de seus usuários para um site que gerencia compras de usuários, página de produtos, carrinho de compras e ainda se integra com plataformas de pagamento *on-line* com notificações enviadas por correio eletrônico para o gestor e o cliente logo que uma compra é efetuada.

Em resumo, os *plugins* têm um papel muito importante para conseguirmos produzir sites úteis sem demandar muito tempo atualmente. Conscientes do cuidado que devemos ter na produção de sites através de *plugins* já existentes, temos de reconhecer que são indispensáveis para elaboração de qualquer site hoje por mais simples que sejam as funcionalidades necessárias e devemos balancear a quantidade destes *plugins*, pois isto além de acelerar o desenvolvimento pode também onerar muito a manutenção do sistema.

4.2. VULNERABILIDADES E AMEAÇAS

Muitos são os tipos de vulnerabilidades e ameaças que podem permitir que um atacante obtenha acesso indevido a uma informação. A cada uma dessas vulnerabilidades podemos chamar de vetores de ataque. Esses vetores são ameaças comuns e podem comprometer sites e todos os sistemas que funcionam via web, tornando-se assim uma vulnerabilidade do sistema.

Atualmente existem estudos que demonstram quais são os principais tipos de ameaças de acordo com a quantidade de ocorrências em um dado período de tempo e isto se torna muito útil, pois “tem como objetivo a sensibilização sobre segurança em aplicações através da identificação de alguns dos riscos mais críticos enfrentados pelas organizações.” (OWASP, 2013b, p.3)

Como apresentado na Tabela 1 podemos ver as ocorrências de treze tipos diferentes de vulnerabilidades reportadas mundialmente ao longo dos últimos cinco anos em vários tipos de sistemas e não somente em gerenciadores de conteúdo. Tais dados foram agrupados e disponibilizados pelo site CVE Details, um dicionário de vulnerabilidades de segurança da informação publicamente conhecidas.

Definiremos abaixo algumas destas vulnerabilidades citadas na tabela:

DOS (*Denial of service*) - Um método de ataque em que um especialista malicioso (*cracker*) tenta prejudicar a disponibilidade de um sistema de TI por excesso de carga. São simulados vários acessos ao sistema ao mesmo tempo, fazendo com que este fique sobrecarregado.

Code Execution – Esta vulnerabilidade permite um atacante executar um código remotamente com o intuito de realizar outras tarefas, abrir portas para outras possibilidades de ataques.

Overflow – Geralmente esta vulnerabilidades aparece associada a outras, pois através do “vazamento” (*overflow*) de uma determinada área da memória de um computador – que seria o enchimento desta área com dados aleatórios até ter acesso a outras áreas -, outros ataques podem acontecer se executados por alguém que conheça os detalhes de como uma memória funciona.

Memory corruption – Ocorre quando o local de memória utilizado por um programa é alterado sem intenção, devido a um erro de programação do software.

Bypass something – Esta vulnerabilidade ocorre quando o atacante ultrapassa, ou seja, consegue burlar alguma barreira ou mecanismo de segurança e ou autenticação.

Gain Information – Permite ao atacante obter informações como chaves de sessão, dados de servidores e até informações sensíveis, o que pode levar a outros tipos de ataques.

Gain Privileges – Permite ao atacante obter privilégios, como acesso a sistemas e bancos de dados com usuários que possuem maiores direitos de acesso e permissão para execução de funções os quais o próprio usuário não tem.

Directory Traversal – Permite que atacantes remotos possam ler, alterar ou apagar arquivos locais arbitrários através de uma mudança de diretório no parâmetro controlador da página que está acessando.

File Inclusion – Esta é uma das vulnerabilidades mais comuns em sites da web e permite que o atacante inclua um arquivo dentro do servidor através da falha de um script que não verifica a entrada do arquivo e que pode causar desde a saída do conteúdo dos arquivos que lá se encontram até o roubo e manipulação de dados ou execução de códigos no servidor ou cliente, levando a outros ataques.

Muitos dos ataques que aparecem nesta tabela fazem parte de mais de um tipo de vetor de ataque, ou seja, são de mais de um tipo diferente de vulnerabilidade, podendo alguns deles ser de três ou mais tipos e assim se tornando muito complexos.

Tabela 1: Tipos de vulnerabilidades ocorridas na Web reportadas nos últimos 5 anos

Ano	Total de Vulnerabilidades	Vulnerabilidades					
		Dos	Code Execution	Overflow	Memory Corruption	Sql Injection	Bypass something
2010	4651	1102	1714	677	342	520	234
2011	4155	1221	1334	770	351	294	197
2012	5297	1425	1457	844	423	242	343
2013	5191	1453	1186	859	366	155	350
2014	7946	1597	1573	848	420	304	457
Total	**27240**	**6798**	**7264**	**3998**	**1902**	**1515**	**1581**

Ano	Vulnerabilidades							Total de exploits
	XSS	Http Response Splitting	CSRF	Gain Information	Gain Privileges	Directory Traversal	File Inclusion	
2010	605	8	86	282	237	275	73	1469
2011	467	7	58	409	206	108	17	564
2012	758	13	166	389	250	122	14	615
2013	650	7	123	510	274	110	1	202
2014	1107	12	264	2104	239	204	2	388
Total	**3587**	**47**	**697**	**3694**	**1206**	**819**	**107**	**3238**

FONTE: CVE Details, 2015, *on-line*. “Tradução e adaptação nossa”.

A *Open Web Application Security Project* (OWASP), “uma comunidade aberta dedicada a capacitar as organizações a desenvolver, adquirir e manter aplicações confiáveis” (OWASP, 2013b, p.3) elabora um estudo de tempos em tempos, através da participação de várias empresas, chamado *OWASP Top 10* no qual são elencadas as 10 principais vulnerabilidades encontradas nos dados disponibilizados por estas empresas em um dado período.

O estudo feito pela OWASP (2013b, p.7) elenca a seguir quais eram as principais vulnerabilidades encontradas naquele ano:

1. Injeção - Pode ocorrer injeção de SQL, de SO (Sistema operacional) e de LDAP onde dados não confiáveis são enviados como consultas aos respectivos interpretadores podendo resultar na execução de comandos não desejados ou acesso a informações não autorizadas.

2. Quebra de autenticação e gerenciamento de sessão - Utilizadas para autenticar e gerenciar a sessão do usuário, as funções são codificadas de forma incorreta

possibilitando o acesso ou comprometimento de senhas, chaves e *tokens* de acesso permitindo que o atacante assuma a identidade de outro usuário.

3. *Cross-Site Scripting* (XSS) - Ocorre quando uma aplicação recebe dados não confiáveis e os envia ao navegador do usuário, sem validar ou filtrar aquilo que é necessário, permitindo que os atacantes possam executar algum código malicioso assumindo a sessão do usuário, desfigurando sites e ou redirecionando o usuário para sites maliciosos.

4. Referência insegura e direta a objetos - Quando o programador permite que um objeto possa ser acessado diretamente através do acesso a algum dado, sem a verificação de controle de acesso permitindo que os atacantes possam manipular essas referências acessando funções ou atributos do objeto sem autorização.

5. Configuração incorreta de segurança - Para se obter segurança a configuração deve ser definida, implementada e mantida na aplicação, frameworks, servidor de aplicação, servidor web, banco de dados e plataforma, pois em geral a configuração padrão é insegura. Isto implica também em todos esses softwares que trabalham em conjunto estarem atualizados, principalmente quando as atualizações disponibilizadas forem de segurança.

6. Exposição de dados sensíveis - Os dados sensíveis como números de cartões de crédito e *IDs* fiscais e credenciais exigem proteção extra quando armazenadas e trafegadas, além das precauções especiais quando enviadas pelo navegador, pois os atacantes podem roubar ou modificar esses dados com o intuito de realizar fraudes, roubos ou outros crimes.

7. Falta de função para controle de nível de acesso - Muitas aplicações verificam os direitos de acesso em nível de função, antes de disponibilizar a funcionalidade na interface do usuário, porém se esquecem de verificar os mesmos controles ao chamar a função para executar a tarefa, permitindo que as requisições possam ser forjadas e as funções executadas por um usuário não autorizado.

8. *Cross-Site Request Forgery* (CSRF) - Esta falha permite que o navegador do usuário através de uma aplicação vulnerável possa criar requisições em outra aplicação que o usuário possa estar autenticado, podendo realizar qualquer tarefa como se fosse o próprio usuário e ou até mesmo assumindo a sessão do usuário através de informações de *cookies* ou outra informação de autenticação.

9. Utilização de Componentes Vulneráveis Conhecidos - Bibliotecas, *frameworks*, *templates*, *plugins* e outros componentes vulneráveis são executados com os mesmos direitos de acesso da aplicação e isto pode ser muito danoso, ocasionando roubo de dados, e ou comprometimento do servidor. Os CMS são muito visados por causa destas

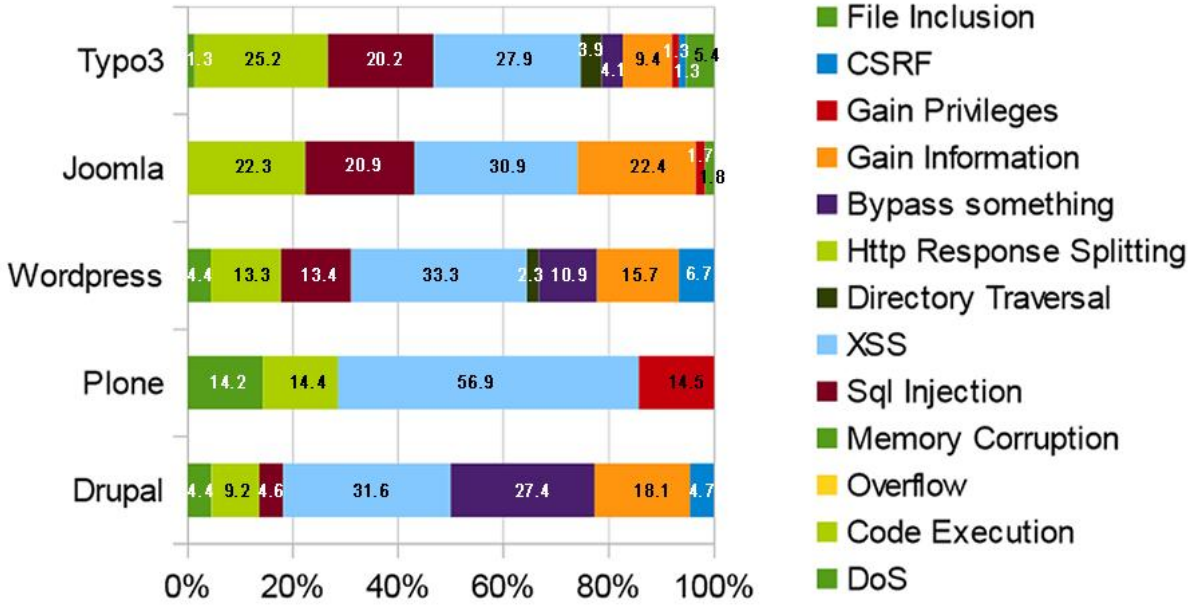
vulnerabilidades conhecidas, pois vários sites se utilizam da ferramenta e os ataques terão mais sucesso.

10. Redirecionamentos e Encaminhamentos Inválidos - A aplicação, ao criar um redirecionamento ou encaminhamento, utiliza dados não confiáveis para determinar a página de destino sem uma validação adequada, assim os atacantes podem direcionar os usuários para sites de *phishing* ou *malware*, ou ainda utilizar esses encaminhamentos para acessar páginas não autorizadas.

Outro estudo, citado anteriormente, feito pelo Serviço Federal de Segurança em Tecnologia da Informação da Alemanha (BSI) nos mostra quais os principais tipos de vulnerabilidades ocorreram entre os anos de 2010 e 2012 nos cinco principais gerenciadores de conteúdo utilizados na internet: Drupal, Plone, Wordpress, Joomla e Typo3. Nele fica evidente que dentre os vetores identificados, alguns corroboram a pesquisa feita pela OWASP.

O Gráfico 2, retirado do estudo em questão, nos mostra a quantidade média em que as porcentagens dessas vulnerabilidades ocorreram nestes CMS e ao analisarmos este gráfico podemos apontar tendências na ocorrência de alguns vetores.

Gráfico 2: Porcentagem de vulnerabilidades nos principais CMS entre 2010-2012



FONTE: BREITENSTROM, 2013, p. 66. “Adaptação nossa”.

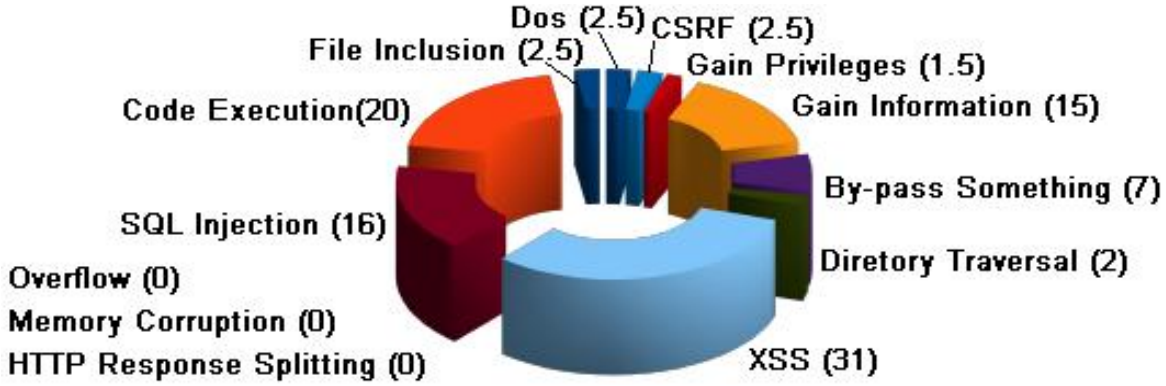
Assim, dentre as tendências de ocorrências que podemos apontar na análise deste gráfico, segundo este mesmo estudo,encontra-se a afirmação de que as três vulnerabilidades mais comuns entre estes CMS são: XSS (*Cross-site scripting*), Code Execution e SQL Injection onde:

> [...] cross-site scripting (XSS) é considerado uma das vulnerabilidades mais comuns [...]. No entanto, a execução de código [...], e injeção de SQL [...]

> são também muito comuns. Injeção de SQL tem estado na lista de vulnerabilidades *web* OWASP Top 10 há anos e informações sobre estratégias de mitigação de tais vulnerabilidades são amplamente disponíveis. (WUEEST, 2013, *on-line*) “Tradução nossa”.

O estudo ainda nos mostra que ao somarmos as vulnerabilidades encontradas nos dados destes gerenciadores, separadas por tipo, podemos corroborar nossa afirmação sobre as três vulnerabilidades mais comuns. Vejamos o gráfico a seguir:

Gráfico 3: Médias dos CMS quanto aos tipos de vulnerabilidades.



FONTE: BREITENSTROM, 2013, p. 66. “Adaptação nossa”

Visto isso, vamos entender melhor como podem acontecer na prática estas três vulnerabilidades mais comuns. Para ilustrar o trabalho, descrevemos abaixo alguns exemplos destes três principais vetores que aparecem nos dados destes estudos citados anteriormente, alguns relacionados às duas maiores redes sociais da atualidade, Facebook e Twitter.

Wueest (2010) nos conta que em 21 de setembro do mesmo ano uma vulnerabilidade do tipo *cross-site scripting* (XSS) começou a ser publicamente conhecida e explorada por usuários da rede social Twitter, considerado na atualidade uma das maiores redes sociais e com a característica de possuir postagens pequenas, objetivas e relacionadas a assuntos que estão sendo discutidos naquele momento.

O sistema de postagem da rede social não verificava quando um usuário postava um código na linguagem *javascript* que poderia ser executado logo que o mouse de quem acessava o conteúdo passava por cima de um *link*, não necessitando ser clicado. A própria rede social informou que um mês antes havia resolvido este problema, mas que devido a uma atualização no código a vulnerabilidade reapareceu.

Como vários usuários passaram a utilizar esta falha para criar efeitos em suas postagens sem fins maliciosos, ela começou a ficar conhecida e logo os *hackers* começaram a se aproveitar disto para divulgar códigos de *script* escondidos em postagens pelas quais infectavam usuários.

Em um curto período de tempo os ataques começaram a acontecer e os usuários não tinham como ser responsabilizados por serem infectados, pois não dependia

de uma ação deles visto que até os usuários mais experientes em tecnologia colocam o ponteiro do mouse sobre um *link* desconhecido para saber qual é o domínio para onde ele será redirecionado e esta simples ação já disparava o código malicioso utilizado pelos *hackers*.

O autor nos diz também que “este é um dos principais problemas com tais ataques XSS - é difícil dar boas sugestões de proteção para que um usuário possa facilmente seguir.” (WUEEST, 2010, *on-line*) “Tradução nossa”. E nem usando uma extensão de navegadores que bloqueia a execução de códigos irá deixá-lo totalmente protegido de tais ataques, já que o redirecionamento para sites maliciosos é feito pelo próprio domínio original da rede social.

> Ataques XSS nas redes sociais acontecem e podem ser mais do que apenas um incômodo, redirecionando os usuários a sites maliciosos ou extraindo informações privadas de contas de usuário. O primeiro passo para se proteger contra esses ataques é ter consciência deles. (WUEEST, 2010, *on-line*) “Tradução nossa”.

Neste ataque os *hackers* além de utilizarem um código para redirecionar o usuário a um site malicioso também o utilizaram para repassar a mensagem de uma conta para outra, fazendo com que outros usuários pudessem ser afetados. O próprio autor de um código *worm* XSS destes disse ter visto mais de 200.000 mensagens de usuários infectados com sua criação.

A maioria dos ataques identificados na época não foi tão devastadora, mas por mais simples que pareçam ser não podemos desprezá-los, pois um ataque XSS pode levar a outros tipos de vulnerabilidades podendo aumentar a seriedade do problema.

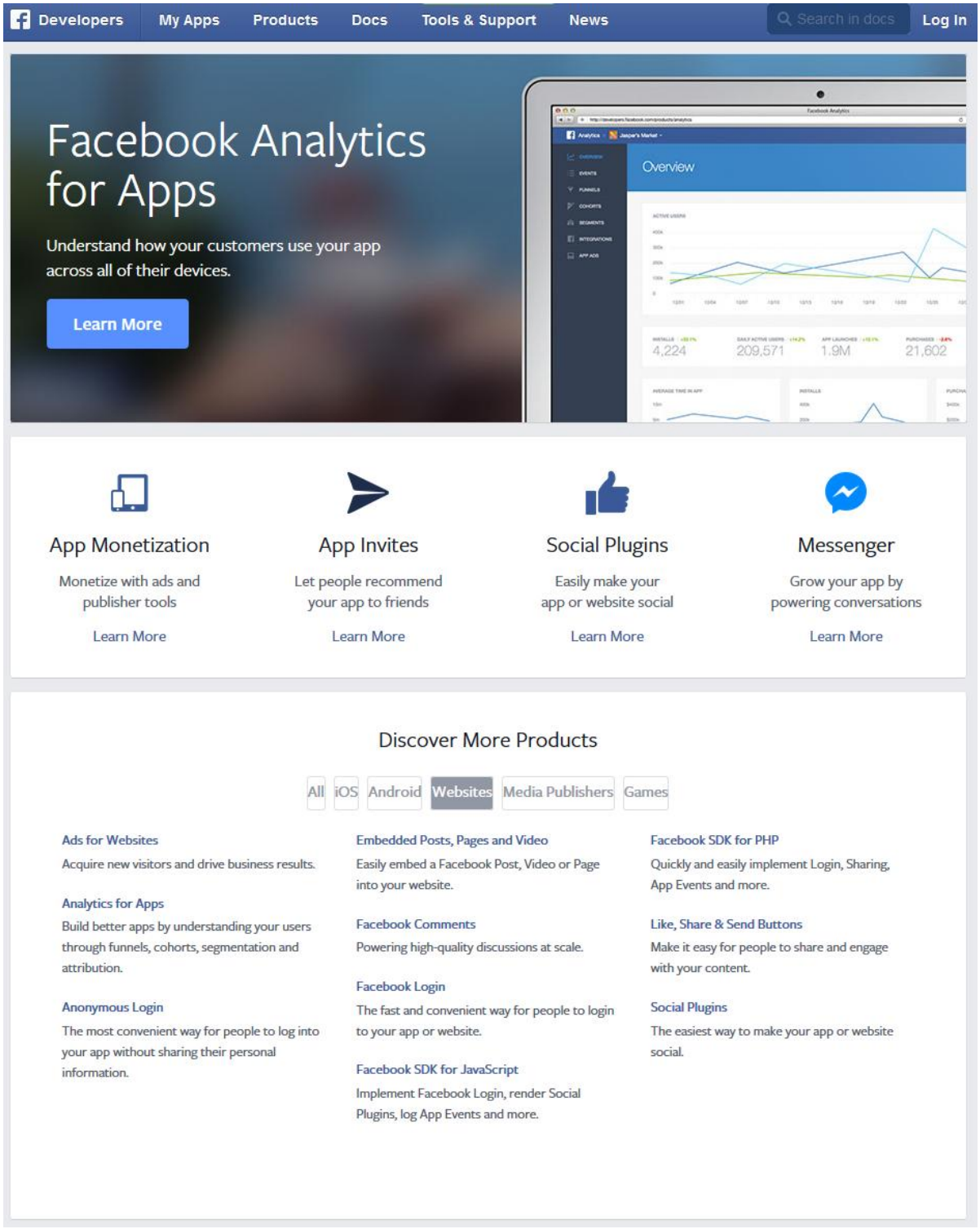
Em outro artigo publicado por Wueest (2011), ele nos conta a respeito de uma nova e, até 21 de março daquele mesmo ano, não corrigida vulnerabilidade *cross-site scripting* (XSS) na rede social Facebook que estava sendo amplamente utilizada por diferentes grupos, especialmente da Indonésia onde foram detectadas milhares de mensagens infectadas, para postar mensagens no mural de outros usuários sem o conhecimento deles.

A vulnerabilidade existia na versão móvel da API (*Application Programming Interface*) do Facebook, uma interface de programação com funções para permitir que outros softwares e ou aplicativos consigam se comunicar com a rede social, que não fazia algumas verificações de segurança necessárias.

Assim, a API permitia que o site incluísse um malicioso elemento *iframe* – elemento da linguagem HTML que abre uma página dentro de outra sem o usuário perceber – que continha um código na linguagem *javascript* ou usasse o valor "*refresh*" do atributo *http-equiv*, também do HTML, para redirecionar a navegação para a página que contivesse

o código *javascript* que efetuaria a postagem da mensagem arbitrária no mural do usuário que estivesse autenticado no Facebook.

Figura 5: Interface de programação de aplicações (API) do Facebook



FONTE: Facebook, 2015

Como isto era feito usando a API de integração do Facebook, qualquer outro site poderia incluir este código dentro de suas próprias páginas e um usuário ao visitá-la,

estando autenticado no Facebook, teria mensagens destas postadas em seu mural. Mesmo que a conta do Facebook estivesse com a opção de segurança (SSL - *Security Socket Layer*) ativada o ataque iria obter sucesso, pois não depende do protocolo de acesso e sim de uma chamada a um recurso disponibilizado pela própria API.

Assim, neste caso específico, uma extensão, por exemplo, de navegador web que bloqueia *worms* de ataques XSS como este é capaz de detectar o site que está infectado não permitindo que o usuário prossiga com o acesso sem antes aceitar e estar ciente do problema.

Outra vulnerabilidade, relacionada ao vetor *code execution*, descoberta pelo *firewall* de sites da empresa Sucuri em uma extensão de *e-commerce* chamada Hikashop desenvolvida para o CMS Joomla permitia que comandos pudessem ser executados remotamente nos sites que utilizavam versões abaixo da 2.3.2. Um *hacker*, conhecendo a vulnerabilidade, poderia executar esses códigos remotamente no site que permitisse ler algum arquivo de configuração, modificar arquivos e ou inserir algum *malware* no site.

Como o *plugin* trabalha com *e-commerce*, os usuários em algum momento precisariam se registrar no site e então surgia o problema, pois a chamada de uma função sensível neste momento fazia com que o site ficasse vulnerável ao ataque, pois não verificava os dados do usuário e executava a ação diretamente. Assim, aqueles que conheciam a vulnerabilidade poderiam facilmente manipular dados disponíveis no contexto da aplicação.

Montpas nos diz sobre o problema detectado que:

> Esses tipos de ataques são altamente dependentes das classes disponíveis para o atacante quando unserialize() analisa a carga útil. Nós, naturalmente, pensamos que poderia ser uma boa idéia verificar se algo de ruim poderia ou não ser feito usando as classes de Joomla! 3.*, e percebemos que existe. Com isso, fomos capazes de transformar o problema de injeção de objeto em uma **vulnerabilidade de execução remota de código**, o que nos permite executar comandos no site remoto. (MONTPAS, 2014, *on-line*).

O trecho acima, extraído do artigo de divulgação da vulnerabilidade no site da empresa, fala claramente da função chamada *unserialize* da linguagem PHP que foi utilizada para explorar a vulnerabilidade e o tipo de vulnerabilidade que se caracterizava. A Figura 6 nos mostra quais foram as mudanças efetuadas na nova versão do *plugin* para a retirada da vulnerabilidade do código.

Figura 6: Vulnerabilidade em *plugin* de *e-commerce* do Joomla


In the file ***administrator/components/com_hikashop/classes/user.php***, replace the line:

```
$vars = urlencode(base64_encode(serialize(array('passwd'=>$this->registerData->password,'username'=>$this->registerData->username)
)));
```

with:

```
$vars = urlencode(base64_encode(json_encode(array('passwd'=>$this->registerData->password,'username'=>$this->registerData->usernam
e))));
```

And in the file ***components/com_hikashop/controllers/checkout.php***, replace the line:

```
$infos = unserialize(base64_decode($infos));
```

with:

```
$infos = json_decode(base64_decode($infos),true);
```

FONTE: Hikashop, 2014

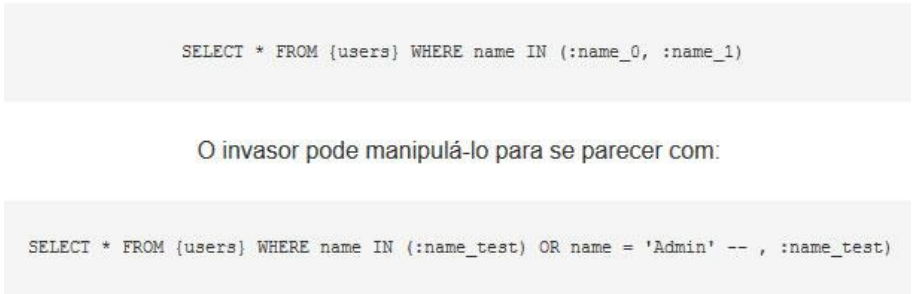
Após a divulgação da vulnerabilidade, os desenvolvedores da extensão lançaram uma atualização dentro de poucas horas permitindo assim que os usuários pudessem atualizar as extensões de seus sites e resolvessem o problema. A substituição da função foi algo simples que poderia ter se tornado um grande problema.

Outra falha, relacionada ao vetor SQL Injection, descoberta em setembro de 2014 pela equipe da SektionEins – uma empresa com sede em Colônia na Alemanha especializada em descobrir vulnerabilidades e falhas de segurança em web e aplicações móveis (SEKTIONEINS, 2015) – foi classificada como uma vulnerabilidade crítica no núcleo do Drupal versão 7.x que logo foi comunicada à equipe de desenvolvimento do CMS.

A vulnerabilidade foi considerada como sendo de nível crítico, pois podia ser explorada por qualquer pessoa remotamente e não era necessário estar autenticado no sistema. E como o Drupal utiliza persistência de dados de objetos - uma maneira mais simples e eficaz de trabalhar com a gravação e recuperação dos dados entre o banco e os objetos que estão na memória enquanto o sistema trabalha - isto se tornou mais crítico.

Assim, em uma determinada consulta para recuperação de dados do banco, por exemplo, o usuário mal intencionado conseguiria manipular estes dados conforme seus interesses para obter outros dados ou, também, modificar dados dentro do banco de dados. O exemplo abaixo retirado do artigo de Dede (2014, *on-line*) nos mostra a dimensão do problema:

Figura 7: Exemplo de injeção de SQL no CMS Drupal



```
SELECT * FROM {users} WHERE name IN (:name_0, :name_1)
```

O invasor pode manipulá-lo para se parecer com:

```
SELECT * FROM {users} WHERE name IN (:name_test) OR name = 'Admin' -- , :name_test)
```

FONTE: Dede, 2014.

Uma pequena consulta de dados no banco pode ser encadeada por uma série de comandos para executar tarefas distintas e podendo resultar no recolhimento de informações preciosas para concorrentes ou outros que tenham algum interesse.

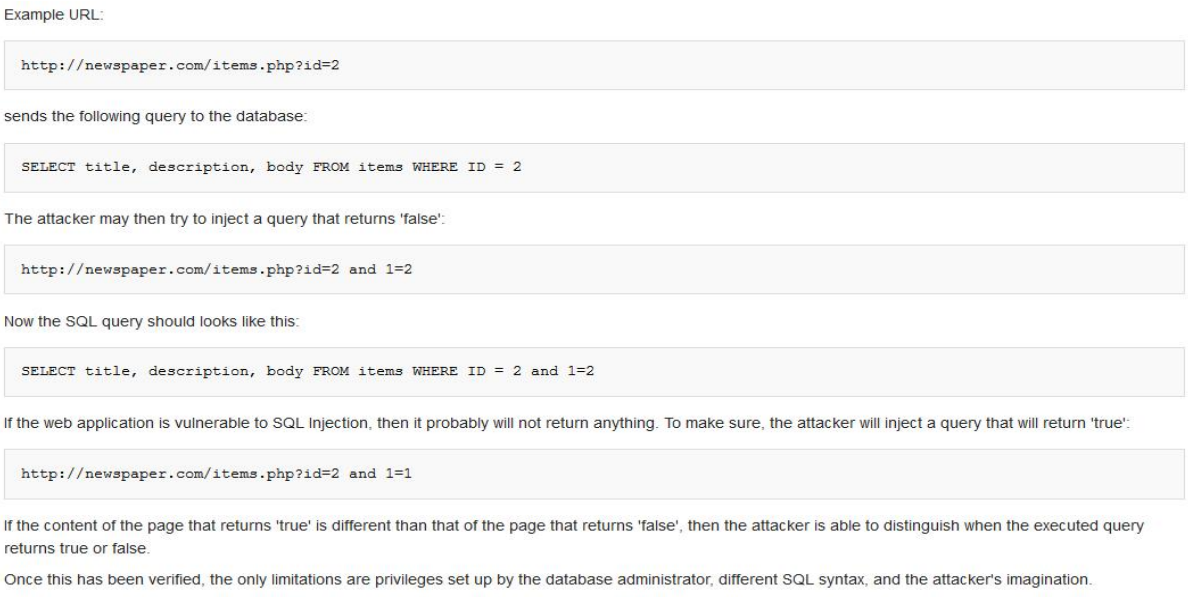
Outra falha descoberta em 10 de março de 2015 por Ryan Dewhurst – pesquisador em segurança e co-desenvolvedor do *scanner* de vulnerabilidades WPScan – ocorrida em um plugin, em geral, muito popular de SEO (*Search Engine Optimization*), utilizado para otimizar as buscas realizadas na internet relacionadas ao site, e também muito popular do Wordpress, onde mais de 1 milhão de sites ativos o utilizava está relacionada ao tipo de vulnerabilidade SQL Injection, chamada injeção de SQL às cegas (Blind SQL Injection).

Neste tipo de vetor o atacante tem de ir testando alguns valores através das consultas SQL e obtendo respostas verdadeiras ou falsas para descobrir aos poucos informações sensíveis relacionadas ao banco de dados.Esta vulnerabilidade é chamada de injeção de SQL às cegas, pois os dados resultantes da consulta SQL não são enviados de volta para o atacante, porém a resposta que uma aplicação vulnerável retorna tem como ser verificada.

Assim, quando a consulta enviada deve gerar um erro ou obter um resultado negativo e o conteúdo da página retornada é diferente de quando a resposta deve ser verdadeira, – por exemplo, a página não carrega totalmente – então o atacante pode julgar se o sistema é vulnerável ou não e, conforme as próximas respostas, saber se aquilo que está sendo testado é verdadeiro ou não e utilizar disto para descobrir outros detalhes.

Um simples exemplo citado por OWASP em sua página que trata deste assunto é o que vemos na Figura 8.

Figura 8: Exemplo de injeção de SQL às cegas testando retorno *true* ou *false*



Example URL:

```
http://newspaper.com/items.php?id=2
```

sends the following query to the database:

```
SELECT title, description, body FROM items WHERE ID = 2
```

The attacker may then try to inject a query that returns 'false':

```
http://newspaper.com/items.php?id=2 and 1=2
```

Now the SQL query should looks like this:

```
SELECT title, description, body FROM items WHERE ID = 2 and 1=2
```

If the web application is vulnerable to SQL Injection, then it probably will not return anything. To make sure, the attacker will inject a query that will return 'true':

```
http://newspaper.com/items.php?id=2 and 1=1
```

If the content of the page that returns 'true' is different than that of the page that returns 'false', then the attacker is able to distinguish when the executed query returns true or false.

Once this has been verified, the only limitations are privileges set up by the database administrator, different SQL syntax, and the attacker's imagination.

FONTE: OWASP, 2013a.

O atacante envia uma consulta que ele sabe que irá retornar um valor verdadeiro e depois envia uma consulta que ele sabe que o retorno será falso para testar o sistema. Caso ele consiga distinguir o conteúdo retornado das duas consultas, ele conseguirá distinguir outras consultas de acordo com o retorno e saber se o que ele envia é verdadeiro ou falso, podendo testar inúmeros valores de acordo com a sua imaginação e podendo automatizar este processo para obter melhores resultados.

A exploração desta falha exige que o usuário esteja autenticado, logo um ataque desses se utilizaria de outra vulnerabilidade chamada *cross-site request forgery* (CSRF) onde um usuário seria enganado através de um link malicioso e o atacante conseguiria executar os testes no banco de dados.

> Um ataque CSRF envolve forçar o navegador de um usuário a executar uma ação não autorizada em um site de terceiros quando o usuário visitar uma página controlada pelo invasor. Os sites devem implementar mecanismos especiais de proteção para evitar tais ataques. (PCWORLD, 2015, *on-line*).

Constantin (2015) nos diz em seu artigo que a empresa Yoast, desenvolvedora do *plugin* em questão, solucionou este problema logo no dia posterior liberando uma nova versão gratuita do Wordpress SEO e também uma nova versão da variante comercial do *plugin*.

Assim, vimos que estas ameaças são comuns a vários sistemas, sejam nos núcleos dos CMS, nos *plugins* ou sistemas com núcleos diferentes dos CMS, como os de redes sociais, por exemplo: Facebook ou Twitter.

Percebemos que muitas podem ser as possibilidades de vetores de ataques a sites e sistemas web e que quanto mais bem preparada estiver a equipe de desenvolvimento do *plugin* ou CMS, melhor será a resposta frente aos desafios e ameaças externas, proporcionando maior segurança aos seus usuários e suas informações.

5. CONSIDERAÇÕES FINAIS

Percebe-se que depois da definição do CMS a ser utilizado, a escolha dos *plugins* é algo que deve ser feito cuidadosamente, pois como vimos neste trabalho os *plugins* são partes estruturais críticas, do ponto de vista da segurança, porque eles possuem um potencial muito grande para se tornarem vulneráveis e algumas variáveis os tornam complexos, como serem desenvolvidos por terceiros e conseguirem interferir em todo o funcionamento do CMS através de funções disponibilizadas pelo núcleo.

Desta forma, não podemos nos descuidar da segurança dos *plugins* e do CMS, mantendo-os sempre atualizados, principalmente se as novas versões possuírem atualizações de segurança, além de escolher bem quais os *plugins* utilizar no site, através da análise de comentários de usuários, histórico de atualizações e problemas reportados nas páginas do próprio *plugin* em seus respectivos repositórios e outros sites da internet que falam sobre o assunto.

Caso a escolha dos *plugins* não seja sua responsabilidade, mas sim a contratação de um profissional ou empresa que irá desenvolver seu canal de informações, você terá de se resguardar de alguns problemas sérios que podem surgir e garantir que tenham conhecimento suficiente para desenvolver um produto que atenda às suas necessidades.

Para se resguardar destes problemas futuros, caso contrate uma empresa ou profissional, o portfólio somado a outros detalhes – como a garantia de desenvolvimento, o processo de desenvolvimento bem elaborado que inclua protótipos, a validação de *layout*, o acompanhamento técnico durante um período de adaptação e o treinamento de usuários – são essenciais.

Um ponto importante, e que a maioria dos interessados em produzir sites ou sistemas em um CMS esquece, é sobre a manutenção destes. Todo sistema deve ter um profissional ou empresa responsável pela execução da manutenção dele, pois tanto as linguagens de programação quanto os próprios CMS estão em constante atualização e isto pode gerar uma necessidade de manutenção no site que sempre deve ser feita, principalmente quando está relacionada à segurança como vimos ao longo do trabalho.

Em relação à atualização é importante ressaltar que ao realizá-la é preciso estar atento a alguns detalhes como: ao utilizar *template* e ou *plugins* gratuitos, se informar se as versões atualizadas continuam gratuitas ou não para saber como proceder; além disto, seu site pode ficar desfigurado devido à diferença de versões dos CMS, *plugins* e ou *template*, sendo, portanto, necessário conhecer e pesquisar a respeito dos mesmos antes de atualizar.

Outro importante ponto a se considerar é que antes de realizar alguma modificação, seja esta a instalação de um novo *plugin* ou alguma manutenção, devemos

fazer testes em outra cópia, idêntica ao site que está em execução, pois qualquer alteração pode ser danosa e fazer com que problemas sejam gerados e estes podem chegar a fazer com que o site fique fora do ar.

Concluímos que todas as questões tratadas neste trabalho são de grande importância para nos fazer refletir a cerca da segurança ao desenvolvermos ou utilizarmos *plugins* em CMS na Web, infelizmente sabemos que no dia-a-dia nem sempre se coloca em prática todas estas questões que aqui foram apresentadas, mas precisamos repensar tal prática, pois vimos quantos riscos estamos correndo e expondo os usuários.

Portanto, segurança é essencial desde o desenvolvimento, passando pela instalação até a manutenção do CMS e todos os seus *plugins*. E, finalizo deixando como referência e contribuição no desenvolvimento de outros trabalhos na área como: a segurança física do sistema, rede e servidor de hospedagem, bem como a segurança do núcleo dos gerenciadores de conteúdo.

BIBLIOGRAFIA

ABNT - Associação Brasileira de Normas Técnicas. **NBR ISO/IEC 17799: Tecnologia da informação - Técnicas de segurança - Código de prática para a gestão da segurança da informação**. Rio de Janeiro, 2005.

ANDREI, K.; QUINN, L.; POPE, E. **A Consumers Guide to Content Management Systems for Nonprofits**. Idealware, 2014. 168p. (Relatório Técnico). Disponível em: <http://www.idealware.org/sites/idealware.org/files/IDEALWARE_CMS_2014MARCH25.pdf> . Acesso em: 19 nov. 2014.

BAX, M. P.; PEREIRA, J. C. I. **Introdução à Gestão de Conteúdos**. Revista Gestão & Tecnologia, v. 1, n. 1, p. 1-11, 2002. Disponível em: <http://www.spell.org.br/documentos/ver/24392/introducao-a-gestao-de-conteudos>. Acesso em: 26 nov. 2014.

BOVEE, M.; SRIVASTAVA, R. P.; MAK, B. **A conceptual framework and belief-function approach to assessing overall information quality**. International Journal of Intelligent Systems, v. 18, n. 1, p. 51-74, 2003. Disponível em: <http://dx.doi.org/10.1002/int.10074>. Acesso em: 10 set. 2014.

BRASIL. Ministério da Ciência e Tecnologia. **Tecnologia da informação: Programa Brasileiro da Qualidade e Produtividade em Software**. 4.ed. rev. e ampl. Brasília: Secretaria de Política de Informática, 2006. 408p.

BRASIL. Tribunal de Contas de União. **Boas práticas em segurança da informação**. 2.ed. Brasília: TCU, Secretaria de Fiscalização de Tecnologia da Informação, 2007. 70p. Disponível em: <http://portal2.tcu.gov.br/portal/pls/portal/docs/2059162.PDF>. Acesso em: 10 fev. 2014.

BREITENSTROM, C. et al. 2013. **Sicherheitsstudie Content Management Systeme (CMS)**. Bonn, Germany, Bundesamt Für Sicherheit in Der Informationstechnik (BSI). v.1, 111p. (Relatório Técnico). Disponível em: <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/CMS/Studie_CMS.pdf?__blob=publicationFile>. Acesso em: 28 jan. 2014.

CERT.BR - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. **Cartilha de Segurança para Internet**. 2.ed. São Paulo: Comitê Gestor da Internet no Brasil, 2012. 140p.

CISCO. 2013. **Cisco Annual Security Report**. San Jose, CA, USA. 80p. (Relatório Técnico). Disponível em: <http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2013_ASR.pdf>. Acesso em: 21 jan. 2014.

CISCO. 2014. **Cisco Annual Security Report**. San Jose, CA, USA. 81p. (Relatório Técnico). Disponível em: <http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf>. Acesso em: 19 nov. 2014.

DEDE, D. **Vulnerabilidade Altamente Crítica de Injeção de SQL corrigido no Núcleo do Drupal**. Sucuri Blog, 2014. Disponível em:

<https://blog.sucuri.net/portugues/2014/10/15/vulnerabilidade-altamente-critica-de-injecao-de-sql-corrigido-no-nucleo-do-drupal.html>. Acesso em: 25 jun. 2015.

FACEBOOK Developers. 2015. Facebook. Disponível em: <https://developers.facebook.com/>. Acesso em: 5 jul. 2015.

FINNE, T. **Information Systems Risk Management: Key Concepts and Business Processes**. Computers and Security, v. 19, n. 3, p. 234-242, 2000.

FLOWERDAY, S.; VON SOLMS, R. **Real-time information integrity = system integrity + data integrity + continuous assurances**. Computers & Security, v. 24, n. 8, p. 604-613, 2005. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404805001458>. Acesso em: 10 jun. 2015.

GARDNER-MADRAS, H.; QUINN, L. 2010. **Comparing Open Source Content Management Systems: Wordpress, Joomla, Drupal and Plone**. Idealware. 82p. (Relatório Técnico). Disponível em: <http://www.idealware.org/sites/idealware.org/files/idealware_os_cms_2010_1.pdf>. Acesso em: 19 nov. 2014.

GELBMANN, M. **Most popular content management systems by country**. 03 set. 2012. Site W3Techs. Disponível em: <http://w3techs.com/blog/entry/most_popular_content_management_systems_by_country>. Acesso em: 22 jun. 2015.

GERHARDT, T. E.; SILVEIRA, D. T. (Orgs.). **Métodos de pesquisa**. Porto Alegre: Editora da UFRGS, 2009. 120 p. Disponível em: <http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf>. Acesso em: 29 nov. 2015.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4.ed. São Paulo: Atlas, 2002.

LEE, Y. W. et al. **Process-Embedded Data Integrity**. Journal of Database Management, Hershey, PA, USA**,** v. 15, n. 1, p. 87-103, 2004. Disponível em: <http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdm.2004010104>.

LEMOS, P. **A economia da informação e do conhecimento e as TI**. Revista de Informação e Tecnologia, n. 2, 1998. Disponível em: <http://www.ccuec.unicamp.br/revista/infotec/economia/economia2-1.html>. Acesso em: 26 nov. 2014.

LEMOS, P. **Conteúdo: quem faz, como faz**. Revista de Informação e Tecnologia, n. 8, 2000. Disponível em: <http://www.ccuec.unicamp.br/revista/infotec/economia/economia8-1.html>. Acesso em: 24 nov. 2014.

MACULAN, B. C. M. S. **Manual de normalização: padronização de documentos acadêmicos do NITEG/UFMG e do PPGCI/UFMG**. Belo Horizonte: UFMG, 2.ed. atual. e rev. 2011.

MONTPAS, M.-A. **Consultoria de Segurança – Extensão Hikashop para Joomla!**. 24 set. 2014. Blog Sucuri. Disponível em: <https://blog.sucuri.net/portugues/2014/09/24/consultoria-de-seguranca-extensao-hikashop-para-joomla.html>. Acesso em: 25 jul. 2015.

MOORE, A. **Content, the Once and Future King**. 1 mai. 2001. Site KM World. Disponível em: <http://www.kmworld.com/Articles/White-Paper/Article/Content-the-Once-and-Future-King-Andy-Moore-8603.aspx>. Acesso em: 05 jan. 2015.

OWASP - Open Web Application Security Project. **Blind SQL Injection**. 26 ago. 2013a. Site OWASP. Disponível em: <https://www.owasp.org/index.php/Blind_SQL_Injection>. Acesso em: 7 jul. 2015.

OWASP - Open Web Application Security Project. **OWASP Top 10 - 2013: Os dez riscos de segurança mais críticos em aplicações web**. Tradução de Carlos Serrão et al. Bel Air, Maryland, USA, 2013b. 23 p. Título original: OWASP Top 10 - 2013: The Ten Most Critical Web Application Security Risks. Disponível em: <http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2013_Brazilian_Portuguese.pdf>. Acesso em: 14 abr. 2014.

PCWORLD/EUA. **Falha de plug-in coloca em perigo mais de um milhão de sites do WordPress**. 12 mar. 2015. Site IDG NOW. Disponível em: <http://idgnow.com.br/internet/2015/03/12/falha-de-plug-in-coloca-em-perigo-mais-de-um-milhao-de-sites-do-wordpress/>. Acesso em: 5 jun. 2015.

PEREZ, T. **Entendendo o Ecossistema dos Plugins de Segurança do WordPress**. 2014. Sucuri Blog. Disponível em: <https://blog.sucuri.net/portugues/2014/09/16/entendendo-o-ecossistema-dos-plugins-de-seguranca-do-wordpress.html>. Acesso em: 19 jun. 2015.

REBOUÇAS, F. **O que é Gestão do Conhecimento?**. 24 jan. 2014. Blog Sociedade Brasileira de Gestão do Conhecimento. Disponível em: <http://www.sbgc.org.br/sbgc/blog/que-e-gestao-do-conhecimento>. Acesso em: 10 nov. 2015.

SCHULTZ, E. E. et al. **Usability and Security An Appraisal of Usability Issues in Information Security Methods**. Computer and Security, v. 20, n. 7, p. 620-634, 2001. Disponível em: <http://www.sciencedirect.com/science/article/pii/S016740480100712X>. Acesso em: 10 jun. 2015.

SECURITY Issue for HikaShop 2.3.2 and below and for HikaMarket 1.4.2 and 1.4.3. 2014. Site HikaShop. Disponível em: <http://www.hikashop.com/index.php?option=com_content&view=article&id=269>. Acesso em: 25 jul. 2015.

SEKTIONEINS. [About SektionEins]. 2015. Disponível em: <https://www.sektioneins.de/stories/company.html>. Acesso em: 09 ago. 2015.

SELLTIZ, Claire et al. **Métodos de pesquisa nas relações sociais**. São Paulo: Herder, 1967.

SHIH, S. C.; WEN, H. J. **Building E-Enterprise Security: A Business View**. Information Systems Security, v. 12, n. 4, p. 41-49, 2003. Disponível em: <http://dx.doi.org/10.1201/1086/43648.12.4.20030901/77305.8>. Acesso em: 10 set. 2014.

SOMMERVILLE, I. **Engenharia de Software**. 9.ed. São Paulo: Pearson Prentice Hall, 2011.

SREEDHARAN, S. **Does your WordPress Security Plugin really secure your site?**. 25 ago. 2014. Blogvault. Disponível em: <https://blogvault.net/does-wordpress-security-plugin-secure-your-site/>. Acesso em: 19 jun. 2015.

VAZ, A. **Segurança da Informação, Protecção da Privacidade e dos Dados Pessoais**. Nação e Defesa, Lisboa, n. 117, p. 35-63, 2007. Disponível em: <http://www.idn.gov.pt/publicacoes/consulta/NeD/NeD117/NeD117.pdf>. Acesso em: 01 jul. 2014.

VULNERABILITIES By Type. 2015. Site CVE Details. Disponível em: <http://www.cvedetails.com/vulnerabilities-by-types.php>. Acesso em: 3 jul. 2015.

WANG, R. Y.; REDDY, M. P.; KON, H. B. **Toward quality data: an attribute-based approach**. Decision Support System, Cambridge, MA, USA, v. 13, n. 3-4, p. 349-372, mar. 1995. Disponível em: <http://www.sciencedirect.com/science/article/pii/0167923693E0050N>. Acesso em: 12 mai. 2015.

WANG, R. Y.; STRONG, D. M. **Beyond accuracy: what data quality means to data consumers**. Journal of Management Information Systems, Armonk, NY, USA, v. 12, n. 4, p. 5-33, 1996. Acesso em: 10 abr. 2015.

WORLD Internet Users Statistics Usage and World Population Stats. 2015. Site Internet World Stats. Disponível em: <http://www.internetworldstats.com/stats.htm>. Acesso em: 5 dez. 2015.

WUEEST, C. **XSSING the line**. 2010. Symantec Connect Blogs. Disponível em: <http://www.symantec.com/connect/blogs/xssing-line>. Acesso em: 11 jun. 2015.

WUEEST, C. **NEW XSS Facebook Worm Allows Automatic Wall Posts**. 2011. Symantec Connect Blogs. Disponível em: <http://www.symantec.com/connect/blogs/new-xss-facebook-worm-allows-automatic-wall-posts>. Acesso em: 11 jun. 2015.

WUEEST, C. **THE RISK with Content Management Systems**. 2013. Symantec Connect Blogs. Disponível em: <http://www.symantec.com/connect/blogs/risk-content-management-systems>. Acesso em: 28 jan. 2014.

REFERÊNCIAS CONSULTADAS

BRAGA, M. **Gestão de risco: Ameaça vs vulnerabilidade na segurança da informática**. 27 mar. 2011. Blog Mateus Braga. Categoria Segurança em Geral. Disponível em: <https://0xmateusbraga.wordpress.com/2011/03/27/gestao-de-risco-ameaca-e-vulnerabilidades-na-seguranca/>. Acesso em: 13 mai. 2014.

FEDERAL Office for Information Security. **Study: A Penetration Testing Model**. 2013. Site BSI (Bundesamt Für Sicherheit in Der Informationstechnik). Disponível em: <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile>. Acesso em: 13 jan. 2015.

CHAGAS, F.; CARVALHO, C. L. de; SILVA, J. C. da. Dez. 2008. **Um estudo sobre os sistemas de gerenciamento de conteúdo de código aberto**. Goiás, Universidade Federal de Goiás, Instituto de Informática. 19p. (Relatório Técnico). Disponível em: <http://www.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_002-08.pdf>. Acesso em: 23 out. 2014.

CONSTANTIN, L. **Over a million WordPress websites at risk because of flaw in popular SEO plug-in**. 12 mar. 2015. Site PCWorld. Disponível em: <http://www.pcworld.com/article/2896156/over-a-million-wordpress-websites-at-risk-because-of-flaw-in-popular-seo-plugin.html>. Acesso em: 5 jun. 2015.

DAMIANO, A. L. **As fraudes no Internet Banking e sua evolução para o Social Banking**. 2013. 107f. Dissertação (Mestrado em Economia, Organizações e Gestão do Conhecimento) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18157/tde-12092013-094137/>. Acesso em: 14 abr. 2014.

DERN, D. P. **Joomla or Drupal: Which CMS handles security best?**. 2011. Disponível em: <http://www.itworld.com/security/157395/joomla-or-drupal-which-cms-handles-security-best>. Acesso em: 28 jan. 2014.

FRANKK, D. **Security Issues in Drupal Content Management System**. 17 mai. 2013. Site Examiner. Categorias: Tech, Gadgets & Tech e Internet. Disponível em: <http://www.examiner.com/article/security-issues-drupal-content-management-system>. Acesso em: 28 jan. 2014.

ITAHRIOUAN, Z. et al. **Validated CMS: Towards New Generation of Web Content Management Systems on Web 2.0**. International Journal of Information Technology and Computer Science, v. 4, n. 12, p. 40, 2012. Disponível em: <http://www.mecs-press.org/ijitcs/ijitcs-v4-n12/IJITCS-V4-N12-4.pdf>. Acesso em: 01 jul. 2014.

MARCIANO, J. L.; LIMA-MARQUES, M. **O enfoque social da segurança da informação Social approach concerning information security**. Ciência da Informação, v. 35, n. 3, p. 89-98, 2006. Disponível em: <http://www.scielo.br/pdf/ci/v35n3/v35n3a09.pdf>. Acesso em: 20 nov. 2013.

MCKEEVER, S. **Understanding Web content management systems: evolution, lifecycle and market**. Industrial Management & Data Systems, v. 103, n. 9, p. 686-692, 2003. Disponível em: <http://www-emeraldinsight-com.ez27.periodicos.capes.gov.br/journals.htm?issn=0263-

5577&volume=103&issue=9&articleid=850167&show=pdf&PHPSESSID=pgn18kjauf6vfp3an1lchs4nm5>. Acesso em: 01 jul. 2014.

SILVA NETTO, A. da; SILVEIRA, M. A. P. da. **Gestão da segurança da informação: fatores que influenciam sua adoção em pequenas e médias empresas**. JISTEM J.Inf.Syst. Technol. Manag. (Online), São Paulo, v. 4, n. 3, p. 375-397, 2007. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752007000300007&lng=en&nrm=iso>. Acesso em: 24 nov. 2014.

OLIVEIRA, P. Q. J. de; SOUZA, R. R. **Gerenciamento de portais corporativos com o uso de Scrum**. 2010. 40f. Monografia (Pós-Graduação em Gestão Estratégica da Informação) – Escola de Ciência da Informação, Universidade Federal de Minas Gerais, Belo Horizonte, 2010.

PARFITT, T. **The Future of CMS 5 Trends To Watch**. 2012. Site Netcel. Disponível em: <http://www.netcel.com/Resources/Insights/White-papers/The-future-of-cms-5-trends-to-watch/>. Acesso em: 27 jun. 2014.

PASCOAL, L. B.; BAX, M. P. **Gestão de conteúdo com softwares livres**. 2007. 30f. Monografia (Pós-graduação em Ciência da Informação) - Escola de Ciência da Informação, Universidade Federal de Minas Gerais, Belo Horizonte, 2007.

PINTER, D. **Kentico CMS Security White Paper**. 2011. Site Kentico. Disponível em: <http://www.kentico.com/downloads/Kentico-CMS_Security-White-Paper.pdf>. Acesso em: 28 jan. 2014.

POWELL, B. **Securing Your CMS Website**. 2013. Site About. Seção About Tech. Disponível em: <http://cms.about.com/od/maintain-your-cms-website/a/Securing-Your-Cms-Website.htm>. Acesso em: 28 jan. 2014.

Site PwC Brasil. [Pesquisa Global de Segurança da Informação 2014: Uma defesa ultrapassada]. 2014. Disponível em: <http://www.pwc.com.br/pt/publicacoes/servicos/assets/consultoria-negocios/pesq-seg-info-2014.pdf>. Acesso em: 26 maio 2015.

RAMON, J. D. C.; FELIPE, E. R. **Sistema de gerenciamento de conteúdo na Web - CMS**. 2011. 51f. Monografia (Pós-graduação em Arquitetura e Organização da Informação) - Escola de Ciência da Informação, Universidade Federal de Minas Gerais, Belo Horizonte, 2011.

SANTOS, B. J. **Análise do CMS (Content Management System) Moodle quanto à segurança da informação: Estudo de caso da escola de ensino a distância professor Benedito**. 2013. 97f. Monografia (Pós-graduação em Segurança da Informação) - Universidade Nove de Julho – UNINOVE, São Paulo, 2013.

SHTEIMAN, B. **Why CMS platforms are breeding security vulnerabilities**. Network Security, v. 2014, n. 1, p. 7-9, jan.2014. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1353485814700066>. Acesso em: 24 jul. 2014.

SILVA, B. P. R. A. D. **Planeamento e implementação de um sistema de gestão da segurança da informação**. 2011. 145f. Dissertação (Mestrado em Ciência da Informação) – Faculdade de Engenharia, Universidade do Porto, Porto, Portugal, 2011.

WHITMAN, M. E. **Enemy at the gate: threats to information security**. Communications of the ACM, New York, NY, USA, v. 46, n. 8, p. 91-95, ago. 2003.

WHITMAN, M. E.; MATTORD, H. J. **The Enemy at the Gates II: The Enemy Within**. In: 15th COLLOQUIUM FOR INFORMATION SYSTEMS SECURITY EDUCATION, 2011, Fairborn, Ohio. Disponível em: <http://www.cisse.info/resources/archives/category/16-papers?download=192:26-2011>. Acesso em: 24 nov. 2014.

PHP Coding Standards. 2014. Site Wordpress.org. Disponível em: <https://make.wordpress.org/core/handbook/best-practices/coding-standards/php/>. Acesso em: 12 ago. 2015.

GLOSSÁRIO

Firewall – “Muro de fogo”, dispositivo que barra a passagem daquilo que não está de acordo com regras pré-estabelecidas.

Hackers – Pessoas com grandes habilidades em programação de computadores interessadas em descobrir vulnerabilidades e meios de entrar em sites ou sistemas com algum intuito, são chamados de *crackers* quando esse intuito é maléfico.

HTML – *Hyper Text Markup Language* – Linguagem de marcação base para o funcionamento das páginas web, os navegadores lêem suas marcações para exibir o conteúdo disponibilizado e também criar os links para navegação entre as páginas.

Javascript – Linguagem de programação utilizada em páginas web, principalmente, para validar o preenchimento de formulários e para prover interação do usuário com a página. Geralmente funciona no navegador do usuário, porém hoje pode funcionar dentro de um servidor web específico como uma tecnologia mais avançada.

Layout – Parte gráfica de um projeto, no caso de um site, o *layout* está relacionado aos elementos: textos, imagens, cores, marcas, ícones e toda parte estética do projeto. O profissional responsável neste caso é o *Web Designer*.

LDAP – *Lightweight Directory Access Protocol* – Protocolo para acesso rápido a registros guardados em estrutura de diretórios organizados de forma hierárquica.

Link – Ou *hyperlink* é uma imagem ou texto que contem uma referência para outro local no mesmo documento ou outro, podendo ser em outro site que ao ser clicado o usuário é redirecionado para este local.

Script – É uma sequencia de comandos escrita em linguagem de programação e executada por algum outro sistema ou, mais especificamente, um interpretador.

SQL – *Structured Query Language* – Linguagem de Consulta Estruturada utilizada para comunicação com os principais bancos de dados.

SSL – *Security Socket Layer* – Camada de segurança utilizada para trafegar dados criptografados, não permitindo a leitura destes dados por qualquer pessoa que consiga obtê-los.

CEO – *Chief Executive Officer* – Em português é comparado com nosso Diretor Executivo, a maior autoridade na hierarquia operacional da empresa.

Malware – *Malicious software* – Programa desenvolvido com o intuito de se infiltrar em um computador para executar ações danosas, alterações e ou roubo de informações.