Primeiros passos para o Desenvolvimento Seguro de Aplicações *Web*

Lucas Souza Montanheiro[1], Ana Maria Martins Carvalho[1]

[1]Núcleo de Computação – Instituto Federal Goiano – Campus Morrinhos (IF Goiano)
Caixa Postal 92 – 75.650-000 – Morrinhos – GO – Brazil

lucasmontanheiro10@gmail.com, ana.carvalho@ifgoiano.edu.br

Abstract. *The lack of knowledge of systems development teams regarding information security and the lack of attention of programmers at the time of developing a software project can lead to several flaws that can be exploited by malicious agents. Some managers forget the importance of software security that is developed by their team, which is a big mistake. The purpose of this article is to expose technology course students to ways to develop secure systems throughout the project life cycle, from design to delivery to the end customer.*

Resumo. *A falta de conhecimento das equipes de desenvolvimento de sistemas a respeito de segurança da informação e a falta de atenção de programadores no momento do desenvolvimento de um projeto de software podem ocasionar várias falhas, que podem ser exploradas por agentes maliciosos. Alguns gestores se esquecem da importância da segurança dos softwares que são desenvolvidos por sua equipe, o que é um grande erro. A proposta deste artigo é expor a alunos de cursos de tecnologia maneiras de desenvolver sistemas seguros em todo ciclo de vida do projeto, desde a concepção até a sua entrega ao cliente final.*

1. Introdução

A negligência da segurança durante o desenvolvimento de aplicações geram falhas, que podem ser usadas para obter e alterar informações, que na sua grande maioria são sigilosas. Quando indivíduos mal intencionados identificam falhas, eles podem roubar informações, interceptar comunicações entre cliente e servidor, parar ou até mesmo destruir o serviço *web*, deixando-o sem acesso ou inutilizado, gerando diversos transtornos aos usuários de um serviço.

As principais falhas em aplicações *web* ocorrem durante o desenvolvimento do projeto, onde a fase de codificação do sistema muitas vezes é feita às pressas e sempre extrapolando o prazo, fato comprovado pela teoria da Crise do Software [Dijkstra 1972], que afirma que cada vez os sistemas são cada vez mais complexos e com menores prazos de desenvolvimento. Com isso muitos gestores acabam optando pela diminuição de custos e deixando a segurança como um acessório para o cliente final, costumando dizer que se a aplicação está funcionando, portanto ela está segura [Luz 2011]. A segurança na realidade é um investimento, para tentar evitar custos e indisponibilidade do serviço *web*.

Toda a equipe responsável por um projeto deve receber treinamento acerca da

segurança de aplicações *web*, a fim de entender como medir riscos nessas aplicações e como as falhas podem gerar vulnerabilidades. A segurança de aplicações *web* deve ser abordada desde o princípio do desenvolvimento, pois geralmente é mais barato construir um sistema seguro, do que corrigir os problemas após a entrega do produto ao cliente como versão final. Algumas empresas possuem pessoas ou equipes especializadas em segurança, para auxiliar as outras equipes em seus projetos, porém essa não é uma realidade na maioria das empresas de desenvolvimento [OWASP 2013].

Com a crescente onda de desenvolvimento de *softwares* para a Internet, o objetivo desta pesquisa é expor para alunos de cursos de tecnologia, que estão aprendendo a desenvolver *softwares* para *web*, e a desenvolvedores de *software* que estão entrando no mundo *web*, a respeito dos riscos de aplicações mal desenvolvidas e apresentar um roteiro de como pensar na segurança em cada uma das partes do ciclo de desenvolvimento de um projeto para a Internet. O intuito do trabalho é de que os relatos aqui apresentados sirvam para chamar a atenção de desenvolvedores a respeito da importância da segurança nos sistemas *web* por eles desenvolvidos.

2. Trabalhos relacionados

A organização *Open Web Application Security Project* (OWASP) publica a cada triênio uma pesquisa conhecida como *Top Ten*, que lista e classifica as dez principais vulnerabilidades em aplicações *web*. A classificação é feita de acordo com a frequência e o risco de cada vulnerabilidade, fornecendo informações sobre a probabilidade da ocorrência e impactos técnicos que cada uma das vulnerabilidades podem causar [OWASP 2013].

No trabalho de Souza (2012) foi avaliado a metodologia proposta pela OWASP, no desenvolvimento seguro de aplicações *web*. Foram realizadas pesquisas qualitativas e o desenvolvimento de uma aplicação *web* para aplicar e avaliar as recomendações propostas pela OWASP. O trabalho concluiu que o uso de guias e metodologias auxiliam os envolvidos no projeto a construírem aplicações seguras, e que a conscientização dos envolvidos no projeto, a respeito de práticas de segurança da informação, é de extrema importância para que um projeto tenha a garantia de integridade, confidencialidade, disponibilidade e autenticidade.

Braga (2007) abordou em seu trabalho aspectos de segurança na construção de *softwares* em geral, desde a definição dos requisitos, casos de estudos, modelos de ameaças, até a escolha dos mecanismos de implementações para atender esses requisitos. O trabalho apresentou várias boas práticas e métodos de segurança que podem ser usados no desenvolvimento seguro de *software*, chegando a conclusão de que a estratégia reativa de correção de falhas após ataques não é mais suficiente, sendo necessária uma abordagem proativa, durante todo o ciclo de vida do *software*.

Desenvolvedores buscam por métricas para medir a qualidade do *software* desenvolvido a todo instante, no trabalho de Batista (2007) procurou-se avaliar a criação e utilização de métricas de segurança em *softwares* e chegou-se a conclusão de que ainda não se conseguiu obter métricas confiáveis para dizer que um *software* é seguro, apenas pode-se provar quando não é seguro. O trabalho ainda expõe que a necessidade de se construir *software* menos frágeis e de se procurar maneiras eficientes para detectar e resolver vulnerabilidades.

3. A segurança deve ser pensada no início do projeto

No início dos cursos de computação aprende-se a analisar e levantar requisitos, a partir desse momento deve-se ter em mente que sistemas mal desenvolvidos podem ser invadidos a todo o momento. Através disso é necessário arquitetar muito bem quais informações de um sistema serão expostas para o usuário e definir os níveis de permissão de acesso aos recursos que serão criados.

Falhas como as que geram a vulnerabilidade de Quebra de Autenticação e Gerenciamento de Sessão podem ser evitadas se pensadas nas fases inicias do projeto. Essas falhas permitem que atacantes roubem dados de usuários da aplicação, seja por mostrar detalhes das sessões de usuários em URLs, sessões que não expiram ou *tokens* de acesso que não são invalidados ao se fazer *logout*. Ao se arquitetar e modelar um sistema que não exponha dados desnecessários aos usuários, para que se possa evitar esse tipo de falha [Montanheiro et al. 2017].

Uma das partes importantes a serem feitas no início do projeto, no que tange a segurança, é mapear o cenário de utilização do *software* e os tipos de ataques que esse sistema pode vir a receber. Por exemplo, se estiver sendo desenvolvendo um *software* para usos militares, os atacantes podem usar de técnicas bem mais elaboradas para conseguirem invadir o sistema do que se comparado a um aplicativo de notas [SegInfoCast 2016].

Deve-se manter em mente que todo *software* está sujeito a falhas e que a cada dia é travada uma guerra entre profissionais de segurança e usuários maliciosos no desenvolvimento de ferramentas para conter ataques e para realizar ataques, que são espalhadas pela *Internet* a todo momento. É de extrema importância que o desenvolvedor sempre mantenha atualizado dos riscos que sua aplicação está exposta.

4. Cenário de um ataque real: Como uma falha de gerenciamento de sessão afetou um site de busca de empregos

Em maio de 2016 o pesquisador de segurança Rafael Fidelis postou em seu *blog* como conseguiu utilizar o cartão de crédito do CEO de um site de busca de empregos para pagar a sua assinatura no próprio *site*. Em seu *post* ele detalha como que uma falta na validação de sessão poderia ter gerado uma grande dor de cabeça para o *site* de empregos [Fidelis 2016].

Segundo o pesquisador, quando ele navegava pelo *site* acabou encontrando as falhas acidentalmente. A plataforma que possui um plano de assinaturas recorrentes mensais por meio de cartão de crédito, usava um *id* sequencial na página de confirmação de compra. O pesquisador precisou apenas alterar a URL no próprio navegador para ter acesso a dados de outros usuários, podendo inclusive confirmar compras no cartão de crédito de todos os usuários do serviço.

O pesquisador comunicou a empresa sobre a falha em seu sistema, a qual rapidamente fez uma correção e mitigou um possível vetor de ataque. Porém as falhas no *site* não paravam por ali, o pesquisador continuou a investigar o *site* e descobriu que a empresa salvava os dados de cartão de créditos dos usuários para que quando eles resolvessem fazer uma nova compra no *site*, não precisasse digitar novamente os dados do cartão.

O pesquisador resolveu analisar a requisição que era enviada ao servidor e

identificou que um dos parâmetros daquela requisição era mais um *id* sequencial, dessa vez o *id* do cartão salvo. De posse dessas informações o pesquisador resolveu testar se a aplicação estava validando da maneira correta o acesso aos cartões de créditos, fazendo uma nova requisição utilizando o *id* 1, que em teoria seria o primeiro cartão de crédito cadastrado no sistema.

Figura 1. Reprodução da aprovação da compra [Fidelis 2016]

Para a sua surpresa a compra foi aprovada em nome do CEO do *site* de empregos, como pode-se ver na Figura 1. Após confirmar mais essa falha o pesquisador entrou em contato com a empresa mais uma vez e notificou do problema. Com isso é possível entender como uma simples falha de autenticação pode ter um prejuízo enorme para a empresa e para sua imagem.

5. Falhas durante o desenvolvimento do projeto

A má codificação de um projeto pode ser resolvida com o uso de padrões e convenções de codificação, propostos pela comunidade da linguagem utilizada, a fim de diminuir falhas. Uma vez que ao se programar seguindo os conceitos da literatura, espera-se que não tenham falhas que possam gerar vulnerabilidades e que podem expor detalhes da implementação da aplicação para o usuário durante sua utilização, que pode ser utilizada de vetor de entrada para um atacante.

Os desenvolvedores também devem levar em consideração o documento que precisa ser feito pelos analistas no momento do levantamento dos requisitos e da modelagem do sistema. Esse documento deve mostrar quais os tipos de ataques cada um dos recursos a serem desenvolvidos estão propensos, para auxiliar o desenvolvedor a tomar decisões sobre o nível de implementação básica de segurança que cada recurso do sistema deve conter. Caso este documento não tenha sido confeccionado pelos analistas, é recomendado que toda a equipe do projeto se junte antes do desenvolvimento,

inclusive os *stakeholders*, para que fique claro em qual cenário o sistema será executado e os tipos de ataques que ele pode receber.

As falhas mais comuns em aplicações se dão em entrada de dados, essas quando não tratadas de maneira correta podem permitir injeção de códigos maliciosos, ocasionando em severos riscos para a aplicação. Recomenda-se que toda entrada de dados seja tratada e que se utilize bibliotecas de auto sanitização de páginas, que são focadas em segurança para garantir que entradas não sejam mal interpretadas pelos navegadores, possibilitando que os desenvolvedores neutralizem as tentativas de ataques à aplicação.

A falha citada no cenário de um ataque, seção 3, representa bem o que a falta de validação de dados durante o desenvolvimento do projeto pode ocasionar. Uma vez que os desenvolvedores não validaram se o usuário tinha permissões para fazer pagamentos com o cartão salvo, eles abriram uma brecha que poderia ter levado ao fim da empresa, caso algum atacante tivesse explorado.

6. Testes de segurança nas aplicações são essenciais

As práticas de testes automatizados são voltadas para melhorar a qualidade do *software* desenvolvido e são grandes aliadas na segurança de um sistema, uma vez que um código bem escrito é menos propenso a falhas, que podem gerar vulnerabilidades.

Algumas metodologias de desenvolvimento de *software* incorporam o desenvolvimento orientado a testes, ou TDD do inglês *Test Driven Development*, como parte da rotina de desenvolvimento, porém poucas equipes cumprem essas rotinas. Desenvolvedores costumam testar as funcionalidades de sua aplicação, todavia, durante a rotina de testes automatizados do sistema, normalmente em sua fase de homologação ou durante o desenvolvimento, também é necessário submeter a aplicação a testes de seguranças.

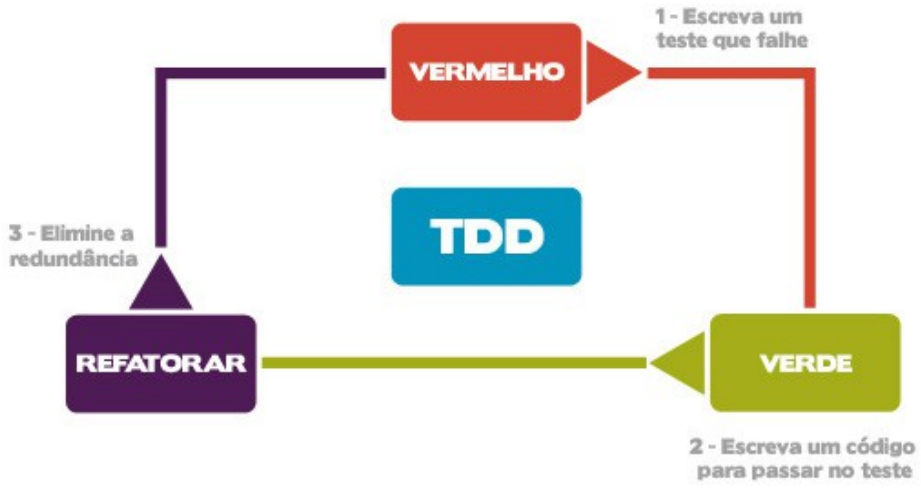

Figura 2. Ciclo de desenvolvimento orientado a testes [DevMedia 2018]

Como pode-se observar na Figura 2 a rotina de desenvolvimento com TDD se baseia em primeiro escrever um teste com os requisitos da função a ser desenvolvida. Após escrito, esse teste ele irá falhar, pois ainda não há nenhuma função desenvolvida

para atender as condições daquele teste, e então é feita a codificação da função, com o objetivo de cumprir todos os requisitos pré estabelecidos no teste. Com a função completa é feita a refatoração do código, a fim de eliminar todas as redundâncias e funções que podem ser abusadas por atacantes.

A revisão de todo código fonte escrito, a procura de falhas, é a mais aconselhada para minimizar a ocorrência de ataques nas aplicações. Essa parte costuma ser ignorada, porém é de extrema importância que seja feita para garantir o mínimo de qualidade no *software* desenvolvido [Chess B. and McGraw 2011].

Outra maneira de testar um *software* desenvolvido é com a utilização de *scanners* automatizados para encontrar falhas de segurança nas aplicações. A utilização desses *scanners* deve ser feita com muita atenção, pois este tipo de teste pode gerar falsos positivos e falsos negativos, exibindo falhas onde não existem e deixando passar falhas existentes [Basso 2010]. Se faz necessário a utilização de vários *scanners* e também a conferência manual dos resultados exibidos por eles, para confirmar se condizem com a realidade.

Grande parte dos ataques na *Internet* são provenientes de indivíduos conhecidos como "*Script Kiddie*", que normalmente possuem rasos conhecimentos e assistem vídeos e tutoriais na *Internet* aprendendo a utilizar ferramentas prontas para realizar ataques. É recomendado aos desenvolvedores que pesquisem as principais ferramentas automatizadas de invasão de sistemas, pois ao testar sua aplicação com ferramentas conhecidas no mercado, poderá tomar ações proativas de segurança e obter uma drástica diminuição nas possibilidades de sua aplicação ser invadida por esses tipos de indivíduos.

7. Cuidados a serem tomados na implantação

Durante a implementação do *software* é aconselhado que seja feito um plano de ação, descrevendo como a equipe tratará uma eventual descoberta de vulnerabilidades ou a ocorrência de um ataque, para que se este vier a ocorrer, a equipe não demore muito para responder e aplicar as correções necessárias.

Outro ponto importante na implementação é o servidor que irá rodar a aplicação desenvolvida. Neste servidor deve ser instalado apenas o necessário para que a aplicação execute e cada servidor deve ter suas responsabilidades bem definidas. Recomenda-se colocar um servidor para o banco de dados e outro para executar a aplicação, nesses servidores os serviços que são executados devem estar sempre atualizados, evitando assim que falhas descobertas em *softwares* utilizados possam ser exploradas.

Um cenário muito comum de ataque ocorre em sites que utilizam o gerenciador de conteúdo WordPress, que tem à sua disposição variadas extensões para diferentes tipos de *site* na *Internet* [Kyaw et al. 2015]. O WordPress foi feito para que pessoas leigas conseguissem criar e administrar seus *blogs*. Pela facilidade de sua utilização foram surgindo extensões que permitem até mesmo rodar uma loja virtual dentro do sistema de *blogs*. Muitas dessas extensões podem conter falhas na sua codificação e expor o sistema a vulnerabilidades. Por esse motivo, é recomendado sempre utilizar bibliotecas e extensões que são atualizadas constantemente e verificar se não há nenhuma vulnerabilidade em seus códigos antes de colocar em produção.

Senhas padrões devem ser evitadas em qualquer ambiente, principalmente quando lidamos com servidores. Uma vez colocados em rede, vários *scripts* automáticos ficam testando as senhas para conseguir acesso ao servidor, que podem ser invadidos e usados para minerar *bitcoins*, disseminar fraudes pela *Internet* e até mesmo para atacar outros servidores através de negação de serviço ou formar uma *botnet* de amplificação de ataques.

Em 2016 a *botnet* Mirai tomou o controle de vários dispositivos de Internet das Coisas que possuíam senhas padrões, atingindo principalmente câmeras IPs e roteadores domésticos, transformando os equipamentos em zumbis e usando seus recursos de rede para ataques de negação de serviço distribuídos. Variantes da Mirai são encontrados constantemente. Em 26 de janeiro de 2018 foram encontradas variantes que eram capazes de infectar roteadores da marca D-Link, burlando o sistema de autenticação e permitindo a execução remota de códigos arbitrários [Kolias et al. 2017].

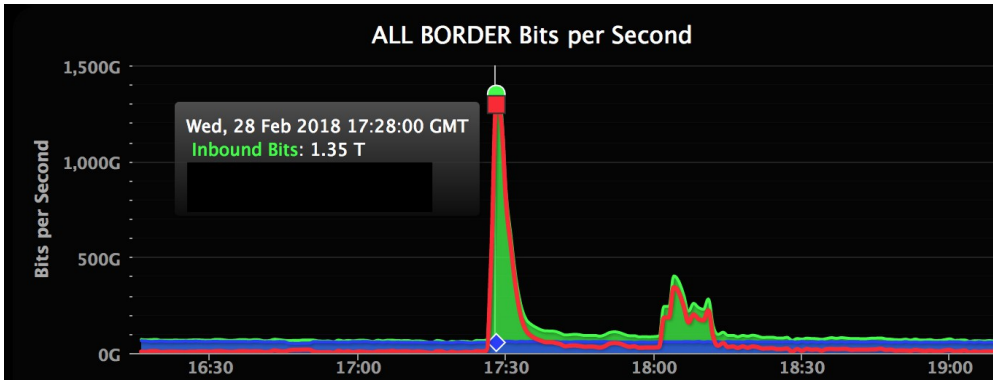

Figura 3. Gráfico de tráfego de rede do Github [GitHub 2018]

Recentemente, em fevereiro de 2018, foi descoberta uma vulnerabilidade de configuração incorreta na aplicação Memcached, que é um sistema distribuído de cache usado por servidores para acelerar redes e sites. Cerca de 51 mil servidores estavam expostos na *Internet* com essa falha. Através dos servidores com a falha atacantes conseguiram fazer um dos maiores ataques de negação de serviço já registrado, usando os servidores como fator de amplificação conseguiram chegar a 1,35 Tbps de tráfego contra os servidores do GitHub.

A Figura 3 mostra o tráfego de rede nos servidores do GitHub no dia do ataque. É possível perceber um pico de entrada 1,35 Terabits por segundo e outro pico de 400 Gigabits por segundo um pouco mais tarde, com esses ataques o GitHub ficou fora do ar por cerca de 10 minutos e só conseguiu estabilizar sua conexão utilizando os servidores de borda da Akamai, empresa que presta o serviço de mitigação de ataques.

A configuração correta do servidor é algo essencial para que sejam evitados esses tipos de vetores de ataque, além disso, devem estar sempre atualizados. Ainda é recomendado para servidores Linux não utilizar senha para fazer *login* no SSH e sim a utilização de um par de chaves, pública e privada, conhecido como *SSH key*.

A utilização e boa configuração de *firewalls* a nível de aplicação e de rede em ambientes de produção são de extrema importância, para negar várias solicitações, feitas muitas das vezes por *scanners*, a procura de vulnerabilidades e portas abertas que possam ser pontos de ameaça. Existem até mesmo sistemas automatizados que detectam e bloqueiam ataques em serviços *web* com base em anomalias do tráfego de rede dos

servidores [Pereira 2011].

Invasores estão a todo momento tentando roubar dados, dados esses que normalmente são armazenados em bancos de dados em que a preocupação com a segurança dos mesmos é bem básica. Uma das táticas que vem sendo utilizado para diminuir o roubo de informações nos bancos de dados é a criptografia dos dados armazenados, que causa perda da performance. Pesquisas como as de Jordão (2014) tentam encontrar um equilíbrio entre criptografia e performance em bancos de dados não relacionais, para termos uma maior confiabilidade nos dados armazenados.

8. Manutenção do projeto

Após a implantação do sistema, um projeto de *software* entra em fase de manutenção, seja para adicionar novas funcionalidades ou corrigir algum problema que venha a ocorrer. Durante essa fase é importante que os desenvolvedores fiquem atentos às atualizações de segurança de bibliotecas e *frameworks* utilizados durante o desenvolvimento da aplicação, pois uma falha nesses componentes de terceiros pode gerar uma falha na aplicação desenvolvida.

Todas as falhas que venham a ser encontradas na aplicação precisam ser investigadas detalhadamente, para se ter o conhecimento se foram utilizadas por atacantes para roubar, alterar ou forjar dados de alguma parte da aplicação.

9. Considerações finais

Vulnerabilidades são encontradas em aplicações *web* a todo momento, sejam elas grandes ou pequenas, para isso os desenvolvedores e toda equipe envolvida em um projeto de *software* precisam estar sempre informados sobre as formas e como funcionam os ataques, para assim então poderem se prevenir de ameaças.

A busca por estar sempre se atualizando já é comum para desenvolvedores quando considerado novas linguagens e tecnologias emergentes do mercado e precisa também ser levada para a área de segurança, uma vez que é ela que irá garantir a saúde e a qualidade de um sistema.

Há uma estimativa de déficit de 3,5 milhões de profissionais de segurança em 2021, isso demonstra que estamos perto de uma grande explosão do crime cibernético, através de ataques às aplicações. Portanto os desenvolvedores que optarem por se especializar na área de segurança da informação terão boas oportunidades de trabalho em um futuro próximo [IDGNow 2017].

Não existe *software* totalmente seguro, por isso a proteção de dados com o máximo de mecanismos de segurança que estiverem disponíveis é crucial para o sucesso de um projeto. Pois a partir do momento que uma aplicação é disponibilizada na *Internet* são grandes as chances de ser alvo de algum atacante, na maioria das vezes são até sistemas automatizados que varrem toda a rede a procura de vulnerabilidades para serem exploradas.

10. Trabalhos Futuros

Como trabalhos futuros pretende-se expor essa pesquisa para alunos, iniciantes em programação *web*, dos cursos de graduação em tecnologia do Instituto Federal Goiano e medir, através de pesquisa qualitativa, o nível de percepção de segurança durante todo o

ciclo de desenvolvimento de software para *web*, antes do conhecimento do guia e após seu conhecimento.

11. Agradecimentos

Agradecimentos ao Instituto Federal de Educação, Ciência e Tecnologia Goiano – Campus Morrinhos e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio e financiamento deste trabalho através do Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI) e do Programa Institucional de Incentivo à Participação em Eventos Científicos e Tecnológicos (PIPECT).

Referências bibliográficas

Basso, T. (2010) “Uma abordagem para avaliação da eficácia de scanners de vulnerabilidade em aplicações web”. Dissertação de Mestrado, Universidade Estadual de Campinas – Faculdade de Engenharia Elétrica e de Computação. Campinas, São Paulo.

Batista, C. F. A. (2007) “Métricas de Segurança de Software”. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

Braga, A. M. (2007) “Visão geral das boas práticas para construção de softwares seguros”. Revista Técnica IPEP, São Paulo, SP, 7(2), 65-78.

Chess B., McGraw G. (2004) “*Static analysis for security*”. IEEE Security & Privacy, v. 2, n. 6, p. 76-79.

DevMedia (2018) “TDD: fundamentos do desenvolvimento orientado a testes”. Disponível em: <https://www.devmedia.com.br/tdd-fundamentos-do-desenvolvimento-orientado-a-testes/28151> Acesso em: 20 Março 2018.

Dijkstra, E. W. (1972) “ *The humble programmer*”. Communications of the ACM, v. 15, n. 10, p. 859-866.

Fidelis, R. (2016) “Como Eu Usei o Cartão de Crédito do CEO do Trampos.co para Pagar Minha Assinatura Premium”. Disponível em: <http://www.fidelis.work/como-eu-usei-o-cartao-de-credito-do-ceo-do-trampos-co-para-pagar-minha-assinatura-premium/> Acesso em: 10 Março 2018.

GitHub (2018) “*February 28th DDoS Incident Report*”. Disponível em: <https://githubengineering.com/ddos-incident-report/>. Acesso em: 20 Março 2018.

IDGNow (2017) “Déficit de profissionais de segurança em TI deve bater 3,5 milhões em 2021”. Disponível em: <http://idgnow.com.br/internet/2017/06/08/deficit-de-profissionais-de-seguranca-em-ti-deve-bater-3-5-milhoes-em-2021/>. Acesso em: 10 Março 2018.

Jordão, R. E. M. (2014) “Armazenamento Distribuído de Dados Seguros com Esquema de Consultas Diretas”, Trabalho de Conclusão de Curso, Faculdade de Tecnologia, Universidade de Brasília - UnB. Brasília-DF.

Kolias, C., Kambourakis, G., Stavrou A. and Voas J. (2017) "DDoS in the IoT: Mirai and Other Botnets," in Computer, vol. 50, no. 7, p. 80-84.

Kyaw, A. K., Sioquim F., and Joseph J., (2015) "*Dictionary attack on Wordpress:*

Security and forensic analysis" *Second International Conference on Information Security and Cyber Forensics (InfoSec)*, Cape Town, 2015, p. 158-164.

Luz, H. J. F. (2011) “Análise de Vulnerabilidades em Java Web Applications”, Trabalho de Conclusão de Curso, Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha. Marília-SP.

Montanheiro, L. S., Carvalho, A. M. M., Rodrigues, J. A. (2017) “Utilização de JSON Web Token na Autenticação de Usuários em APIs REST”, XIII Encontro Anual de Computação. Anais do XIII Encontro Anual de Computação EnAComp 2017. Catalão: Universidade Federal de Goiás (UFG) - Regional Catalão, 2017. v. XIII. p. 186-193.

OWASP (2013) “OWASP Top 10 2013 – The Ten Most Critical Web Application Security Risk”. Disponível em: <https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf>. Acesso em: 10 Março 2018.

Pereira, H. (2011) “Sistema de Detecção de Intrusão para Serviços Web Baseado em Anomalias”, Dissertação (mestrado), Pontifícia Universidade Católica do Paraná - PUCPR. Curitiba-PR.

SegInfocast (2016) “Análise de Código e Segurança de Software”. Disponível em: <https://seginfo.com.br/2016/11/23/seginfocast-44-analise-de-codigos-e-seguranca-de-software/>. Acesso em: 10 Março 2018.

Souza, L. L. (2012) “Desenvolvimento seguro de aplicações web seguindo a metodologia OWASP”. Monografia, Universidade Federal de Lavras. Minas Gerais, Brasil.