Vulnerabilidades em Sistemas de Software Web baseados em Plug-ins? Um Estudo Exploratório do WordPress

Oslien Mesa[1], Marx Viana[1], Elder Cirilo[2], Carlos Lucena[1]

[1]Laboratório de Engenharia de Software
Pontifícia Universidade Católica (PUC-Rio) – Rio de Janeiro, RJ – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de São João del-rei (UFSJ) – São João del Rei, MG – Brasil

{orodriguez, mleles, lucena}@inf.puc-rio.br, elder@ufsj.edu.br

Abstract. *Plugin-based web software systems support simply and quickly the introduction and configuration of new functionalities, and for this reason, are increasingly popular. However, vulnerabilities related to the inclusion of plug-ins are recurrent and often lead to consequences that were not yet well investigated. To better comprehend the main challenges associated with vulnerabilities introduced by plugins in web applications, we mined and analyzed vulnerabilities reports from the CVE-D and changesets from the WordPress plug-ins repository. Our study was able to observe the main vulnerabilities introduced by plugins and their impact, associated maintenance effort and the developers' sentiment.*

Resumo. *Sistemas de software web baseados em plug-ins suportam de modo simples e rápido à introdução e configuração de novos comportamentos, e por esta razão, são cada vez mais populares. Porém, vulnerabilidades causadas por plug-ins são recorrentes e acarretam em consequências ainda pouco investigadas. Para compreender melhor os principais desafios associados às vulnerabilidades causadas por plug-ins em aplicações web, nos mineramos e analisamos relatos do CVE-D e conjuntos de mudanças do repositório de plug-ins do WordPress. Como resultado, conseguimos mapear as principais vulnerabilidades observadas e os impactos causados por elas, e compreender o esforço necessário de manutenção e o sentimento dos desenvolvedores.*

1. Introdução

Sistemas de software baseados em plug-ins oferecem uma série de benefícios para os desenvolvedores, incluindo a rápida configuração de novos comportamentos. Por isto, o mercado de aplicações web vem se estabelecendo em cima dos principais sistemas de software web baseados em plug-ins disponíveis, como *Wordpress*, *Drupal* e *Joomla*. O *Wordpress*, com mais de 50.000 plug-ins, é amplamente utilizado como base para a construção de mais de 60 milhões de aplicações web.

Enquanto a capacidade de rápida extensão de sistemas de software é atraente, ela também, por outro lado, induz uma série de desafios no desenvolvimento que podem comprometer a qualidade das aplicações produzidas. Falhas ou vulnerabilidades introduzidas por extensões realizadas por meio de plug-ins são reconhecidamente difíceis de serem descobertas e resolvidas antecipadamente [Sampaio and Garcia 2016]

[Thome et al. 2017]. A Figura 1 apresenta um segmento de código da resolução de uma vulnerabilidade causada pelo plug-in Easy SMTP. Podemos observar no seguimento de código o conjunto de modificações necessário para abordar a correção de uma possível vulnerabilidade XSS (*cross-site scripting*) causada pela não sanitização dos dados referentes ao remetente e corpo do e-mail. Um simples descuido, como o ilustrado neste exemplo (a não sanitização dos dados), pode causar prejuízos incalculáveis.

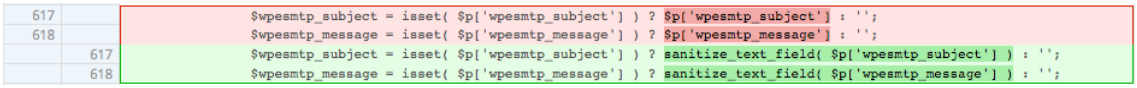

Figura 1 – Correção da vulnerabilidade causada pelo plug-in Easy SMTP.

É evidente então que vulnerabilidades em aplicações web ocorrem na prática e originam consequências graves [Bozorgi et al. 2010] [Smith and Williams 2011]. Estima-se que no geral, 15% das vulnerabilidades observadas em aplicações web possuem nível de gravidade máximo (classificação CVE), e que, em média, 35% delas são trivialmente exploráveis. Porém, pouco se sabe sobre à ocorrência, gravidade, custos de manutenção e o sentimento da comunidade de desenvolvedores em relação às vulnerabilidades causadas por plug-ins em aplicações web como o caso ilustrado na Figura 1. Neste sentido, nos buscamos analisar neste trabalho um conjunto de dados complementares com objetivo de construir um melhor entendimento dos principais desafios associados às vulnerabilidades causas por plug-ins em sistemas de software web. Especificamente, buscamos responder as seguintes questões de pesquisa:

> ***QP1***. *Quais são os principais tipos de vulnerabilidades causados por plug-ins no WordPress?*
> ***QP2***. *Qual a gravidade das vulnerabilidades causadas por plug-ins no WordPress?*
> ***QP3***. *Qual é o sentimento da comunidade de desenvolvedores do Wordpress em relação às vulnerabilidades introduzidas por plug-ins?*
> ***QP4***. *Qual o esforço necessário para resolução das vulnerabilidades causadas por plug-ins no Wordpress?*

Para responder as questões de pesquisa, nos mineramos e analisamos 254 relatos de vulnerabilidades cadastradas no CVE-D (*Common Vulnerabilities and Exposures Details*) e 522 conjuntos de mudanças do repositório de plug-ins do *WordPress*. Como resultado, nos conseguimos mapear os principais tipos de vulnerabilidades, observar que em sua grande maioria, vulnerabilidades causadas por plug-ins não afetam a integridade das aplicações web e que no geral, os desenvolvedores assumem uma postura neutra ao relatar vulnerabilidades. Por fim, com base nos conjuntos de mudanças, constatamos que uma porcentagem das vulnerabilidades demanda um esforço considerável de manutenção. Esses resultados apontam direcionamentos de pesquisa relevantes, destacando uma demanda real por suporte aos desenvolvedores na detecção antecipada de vulnerabilidades.

O restante deste artigo está organizado da seguinte maneira. A Seção 2 apresenta uma discussão sobre a ocorrência de vulnerabilidades em sistemas de software web causadas por plug-ins. A Seção 3 descreve a metodologia do estudo exploratório conduzido e a Seção 4 detalha os resultados obtidos. Na Seção 5 os riscos à validade do

estudo são discutidos. Finalmente, as Seções 6 e 7 apresentam os trabalhos relacionados e as conclusões obtidas respectivamente.

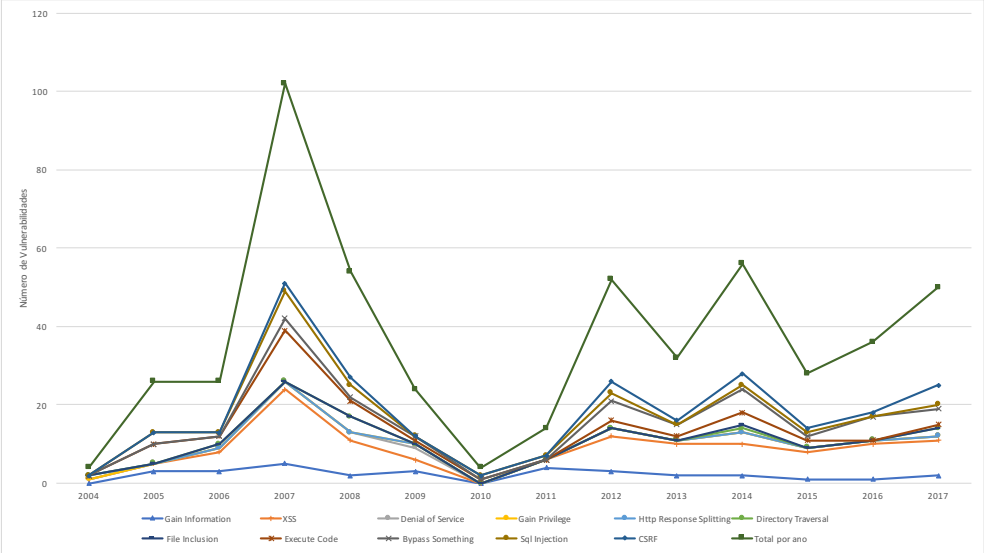

Figura 2 – Vulnerabilidades por tipo a cada ano.

2. Vulnerabilidades em Sistemas de Software Web baseados em Plug-ins

Os riscos eminentes causados por vulnerabilidades em aplicações web é uma realidade em mundo amplamente conectado por meio da internet. O *Wordpress*, por exemplo, tem sido um alvo natural para exploração de vulnerabilidades desde sua criação. Ele é hoje, um dos sistemas de software web baseados em plug-ins mais utilizado no mundo para produção de aplicações web. Para se ter uma noção, entre o período de 2004 a 2017, foram registradas no CVE-D, cerca de 254 vulnerabilidades relacionadas ao *WordPress*. A Figura 2 apresenta a variação no número de relatos de vulnerabilidades associadas ao *WordPress*. Podemos observar que os números seguem um padrão onde não há uma estabilidade ou decaimento na ocorrência de vulnerabilidades, ou seja, vulnerabilidades permanecem sendo introduzidas e corrigidas pelos desenvolvedores constantemente.

Recentemente, estudos conduzidos por consultorias internacionais revelaram um alto índice de vulnerabilidades de alto risco especificamente introduzidas no *WordPress* por plug-ins. Um exemplo famoso ocorreu em 2010. Na época, diversos plug-ins que utilizavam o *TimTumb*, popular utilitário para geração de *thumbnails*, suportaram a execução não autorizada de código PHP no lado do servidor. Infelizmente, porém, estas são vulnerabilidades muito mais complexas de serem observadas e resolvidas. Enquanto é possível controlar a qualidade do núcleo do *WordPress* por meio de ótimas políticas de revisão e processos rigorosos de verificação, não é tão simples verificar os milhares de plug-ins disponíveis no mercado e todas as suas possíveis configurações e interações.

3. Estudo Exploratório

O objetivo do estudo é investigar, quantitativamente, o impacto de vulnerabilidades causada por plug-ins em sistemas de software baseados em plug-ins. Em particular, queremos compreender melhor sobre a ocorrência, gravidade, o custo de manutenção e o sentimento da comunidade de desenvolvedores em ralação as vulnerabilidades causadas pelos mais de 50.000 plug-ins disponíveis para o *WordPress*.

Metodologia. Primeiramente, nos mineramos e manualmente analisamos 254 relatos de vulnerabilidades associados ao *WordPress* presentes no CVE-D. O CVE-D oferece uma interface web amigável para consulta e visualização dos dados relacionados as vulnerabilidades cadastradas no NVD (*National Vulnerability Database*). Para cada CVE é possível consultar: gravidade, complexidade, impacto, tipo, data de publicação, e etc. Nos também mineramos 522 conjuntos de mudanças (*changesets*) do repositório de plug-ins do *Wordpress*, que mantém a gerência de configuração dos mais de 50.000 plug-ins oficialmente cadastrados. Nos obtivemos os conjuntos de mudanças associados à correção de vulnerabilidades buscando pelos que possuíam o termo "*vulnerability*" na mensagem de descrição.

Métricas. Com o propósito de responder as questões de pesquisa, as seguintes métricas foram criadas com base nos dados disponíveis nos relatos e conjunto de mudanças coletados:

- **Tipo - M1**. Número de vulnerabilidades por tipo (XSS, Bypass, Sql Injection) causadas pela adição de plug-ins.
- **Gravidade - M2.** Nível de comprometimento causado pela vulnerabilidade na integridade, disponibilidade e complexidade da aplicação web.
    - **Integridade - M2.1: Nenhum** – não existe impacto na integridade; **Parcial** – é possível realizar modificações em arquivos ou obter acesso a informações do sistema, porém o atacante não possui controle sobre o que pode ser modificado ou o alcance é limitado; **Completo** – existe um total comprometimento da integridade do sistema.
    - **Disponibilidade - M2.2**: **Nenhum** – sem impacto na disponibilidade do sistema; **Parcial** – redução na performance ou interrupções parciais nos recursos disponíveis; **Completo** – paralização dos recursos afetados.
    - **Complexidade - M2.3**: **Baixa** – as condições de acesso não demandam expertise; **Média** – as condições de acesso demandam um pouco de expertise; e **Alta** – as condições de acesso demanda alta expertise.
- **Sentimento - M3**. Sentimento atrelado à descrição completa dos relatos de vulnerabilidades coletados do CVE-D. Consideramos a neutralidade, polaridade, positividade e negatividade dos termos presentes na descrição.
- **Esforço - M4**. Número de artefatos adicionados, modificados, copiados ou removidos do código para resolução das vulnerabilidades identificadas.

4. Resultados

QP1. Quais são os principais tipos de vulnerabilidades causados por plug-ins no WordPress?

Com base nos relatos minerados do repositório CVE-D, obtivemos os principais tipos de vulnerabilidades causados por plug-ins no *WordPress*. A Tabela 2 apresenta a listagem das vulnerabilidades e os respectivos tipos, atribuídos a cada uma delas. Nós observamos um volume extremamente diversificado de vulnerabilidades, com segue: (i) seis do tipo *XSS* — o código não neutraliza ou neutraliza incorretamente a entrada do usuário antes da mesma ser repassada como saída a outros usuários; (ii) cinco do tipo

SQL Injection — o código não neutraliza ou neutraliza indevidamente elementos especiais passados como entrada e que possuem a capacidade de modificar comandos SQLs internos; (iii) quatro do tipo *Direct Traversal* — o código não neutraliza adequadamente elementos especiais que podem fazer com que o caminho de acesso requisitado seja retornado para um outro diretório restrito; (iv) duas do tipo *Bypass* — permite a um invasor ludibriar usuários desavisados por meio de páginas com conteúdo malicioso; (v) duas do tipo *DoS* — permitem o uso excessivo de CPU, vazamentos de memória ou o uso excessivo de i/e de disco, pesquisas lentas e longas, chamadas de banco de dados ou até mesmo a execução de grandes operações de junção; (vi) uma do tipo *Gain Information* — permite o acesso não autorizado à informação sigilosas ou a interrupção de processamentos críticos; (vii) duas do tipo *Improper Access Control* — o código não restringe ou restringe incorretamente o acesso a recursos não autorizados; (viii) uma do tipo *Input Validation* — o código não valida ou valida incorretamente entradas que podem manipular o fluxo de controle ou o fluxo de dados da aplicação web; (ix) uma do tipo *Cross-Site Request Forgery* (*CSRF*) — o código não é capaz de verificar se requisições bem-formulada e validas foram intencionalmente ou não fornecidas pelo usuário que enviou a requisição.

Tabela 2 – Dados das vulnerabilidades encontradas.

CVE - Registro	M1	M2.1	M2.2	M2.3	M3			
					Subjetividade		Polaridade	
					Neutro	Polar	Positivo	Negativo
CVE-2013-2204	Input Valid.	Parcial	Nenhum	Média	0.8	0.2		
CVE-2008-0615	Direct. Trav.	Nenhum	Nenhum	Baixa	0.7	0.3		
CVE-2013-7240	Direct. Trav.	Nenhum	Nenhum	Baixa	0.7	0.3		
CVE-2016-6896	DoS; Direct. Trav.	Nenhum	Parcial	Baixa	0.8	0.2		
CVE-2008-0617	XSS	Parcial	Nenhum	Média	0.4	0.6	0.4	0.6
CVE-2008-0618	XSS	Parcial	Nenhum	Média	0.8	0.2		
CVE-2012-4271	XSS	Parcial	Nenhum	Média	0.6	0.4		
CVE-2013-2201	XSS	Parcial	Nenhum	Média	0.3	0.7	0.4	0.6
CVE-2010-5295	XSS	Parcial	Nenhum	Média	0.7	0.3		
CVE-2017-5488	XSS	Parcial	Nenhum	Média	0.4	0.6	0.4	0.6
CVE-2008-0491	Exec Code Sql	Parcial	Parcial	Baixa	0.6	0.4		
CVE-2008-0616	Exec Code Sql	Parcial	Parcial	Baixa	0.7	0.3		
CVE-2008-4625	Exec Code Sql	Parcial	Parcial	Baixa	0.6	0.4		
CVE-2011-5216	Exec Code Sql	Parcial	Parcial	Baixa	0.9	0.1		
CVE-2017-5611	Exec Code Sql	Parcial	Parcial	Baixa	0.5	0.5	0.4	0.6
CVE-2012-2402	Bypass	Parcial	Parcial	Baixa	0.9	0.1		
CVE-2012-4422	Improper Access Control	Parcial	Nenhum	Média	0.9	0.1		
CVE-2017-6816	Improper Access Control	Parcial	Nenhum	Baixa	0.8	0.2		
CVE-2016-10148	Bypass	Nenhum	Nenhum	Baixa	0.8	0.2		
CVE-2009-2334	DoS; XSS; GI	Parcial	Nenhum	Média	0.7	0.3		
CVE-2008-0198	CSRF	Parcial	Nenhum	Média	0.9	0.1		
CVE-2006-5705	Direct. Trav.	Parcial	Parcial	Média	0.8	0.2		

QP2. Qual a gravidade das vulnerabilidades causadas por plug-ins no WordPress?

Dos vinte e dois relatos do CVE-D analisados, nos observamos que em apenas quatro não se relatou um comprometimento significativo da integridade da aplicação web (M2.1 — Nenhum). Por outro lado, nos demais doze outros relatos os desenvolvedores relataram a possiblidade de acesso irrestrito a alguns arquivos ou de certas informações da aplicação web, porém nestes casos, o escopo do que os atacantes possuíam acesso era limitado (M2.1 — Parcial). Por fim, não observamos relatos em que a integridade da aplicação web estaria totalmente comprometida (M2.1 — Completo).

Analisando os dados relacionados a complexidade, observamos que existe uma correlação entre o impacto causado pelas vulnerabilidades na integridade da aplicação web e a expertise necessária para que se possa explorá-las. Por exemplo, nos onze dos doze relatos em que o nível de comprometimento da integridade foi considerado como parcial, a complexidade de exploração da vulnerabilidade foi classificada como média (M2.3 – Média). Já a complexidade de exploração das vulnerabilidades com nenhum impacto na integridade da aplicação web foi classificada como baixa (M2.3 – Baixa).

Por fim, em relação ao impacto das vulnerabilidades na disponibilidade das aplicações web, observamos que em somente oito, dos vinte e dois relatos, que os desenvolvedores consideraram a possiblidade de um comprometimento na performance ou interrupção parciais nos recursos disponíveis (M2.2 — Parcial). Os outros dezesseis relatos não consideraram qualquer comprometimento na disponibilidade das aplicações web (M2.2 – Nenhum).

QP3. Qual é o sentimento da comunidade de desenvolvedores do Wordpress em relação às vulnerabilidades introduzidas por plug-ins?

Com objetivo de compreender melhor a opinião da comunidade de desenvolvedores em relação as vulnerabilidades causadas por plug-ins em aplicações web, conduzimos uma análise de sentimento nos relatos de vulnerabilidade coletados do CVE-D. Em análise de sentimento, a opinião de um sujeito é expressa pelo grau de satisfação em relação a algum evento observado. Em nosso caso, a ocorrência das vulnerabilidades. Esse grau é representado em uma escala de intensidade que varia de negativa para positiva, de modo que a soma das intensidades (positiva e negativa) é um valor igual a um. Quando se observa um equilibro entre a negatividade e a positividade, o sentimento é classificado como neutro.

Aplicando o algoritmo de análise de sentimentos disponível na biblioteca NLTK[1] à cada relato de vulnerabilidade coletado do CVE-D, classificamos as opiniões de acordo com M3 em positivas, negativas, neutrais e polares e conforme apresentado na Tabela 2. Podemos observar que há uma postura de neutralidade dos desenvolvedores ao se relatar as vulnerabilidades. Em 81% dos relatos coletados, o valor da neutralidade é superior a 0.5. Porém, em quatro relatos observamos um sentimento negativo. Uma análise qualitativa conduzida nestes relatos apresentou que eles, em sua grande maioria, tratavam da ocorrência conjunta de múltiplas vulnerabilidades dispersas em diferentes pontos do código do plug-in.

[1] www.nltk.org

QP4. Qual o esforço necessário para resolução das vulnerabilidades causadas por plug-ins no Wordpress?

Analisamos o impacto de manutenção das vulnerabilidades causadas por plug-ins com base na métrica M4. Os resultados indicaram no geral um alto custo de manutenção associado as vulnerabilidades causadas por plug-ins. Considerando os 522 conjuntos de modificações analisados, observamos um número significativo e predominante de arquivos sendo adicionados a cada nova correção. No geral, ocorreram a adição de mais de 13.000 arquivos, ao contrário dos poucos menos de 3.200 arquivos modificados e 890 arquivos copiados. Em média, a correção de uma vulnerabilidade demandou a adição de 25 novos arquivos, a modificação de 6 arquivos existentes e a cópia de outros 2 arquivos.

Uma análise mais específica indicou, porém, uma alta concentração de adições de arquivos em poucos conjuntos de mudanças. Enquanto em 35% dos casos não houve quaisquer adições de novos arquivos, e sim, apenas modificações de no máximo dois arquivos existentes, em apenas 7% dos conjuntos de modificação contabilizamos mais de 11.033 arquivos adicionados. Também observamos uma baixa reincidência de vulnerabilidades por plug-in. Dos 332 plug-ins que causaram alguma vulnerabilidade, observamos em apenas 52 deles (15%) dois ou mais conjuntos de mudanças relacionados a vulnerabilidades.

5. Ameaças à Validade

Validade Externa. O estudo limitou-se à analise de 22 relatos de vulnerabilidade do CVE-D associados ao *WordPress*. Portanto, os resultados não podem ser generalizados com confiança para outros sistemas de software web baseados em plug-ins (e.g., *Drupal* e *Joomla*). Entretanto, o *WordPress* com seus 50.000 plug-ins é um sistema de software relevante, com alta popularidade, o que confere relevância para os resultados inicias observados.

Validade Interna. O modo de coleta dos dados é um dos aspectos que pode afetar os resultados esperados. Para mitigar essa ameaça, conduzimos uma verificação manual dupla tanto dos relatos de vulnerabilidade do CVE-D quanto dos conjuntos de mudanças coletados do repositório de plug-ins do *WordPress*.

6. Trabalhos Relacionados

Poucos trabalhos abordam a análise de vulnerabilidades em sistemas de software na presença de variabilidades. Em [Ferreira et. al. 2016] um estudo conduzido com base no kernel do Linux demostra que existe uma relação entre a complexidade introduzida por variabilidades e a ocorrência de vulnerabilidades. Os nossos resultados colaboram com as observações de Ferreira et. al., comprovando que a introdução de plug-ins também causa vulnerabilidades em sistemas de software.

Por outro lado, existem esforços no sentido de oferecer para os desenvolvedores suporte automatizado para a detecção antecipada de vulnerabilidades. Em [Sampaio and Garcia 2016] os autores propuseram uma abordagem baseada em analise de fluxo de dados para detecção online de vulnerabilidades. Existem dois desafios principais a serem superados no contexto de sistemas de software web baseados em plug-ins: (i) a explosão combinatória imposta pela introdução dos milhares plug-ins disponíveis; e (ii)

a natureza fracamente tipada e dinâmica da linguagem PHP, que vem governando o desenvolvimento dos principais sistemas de software web baseados em plug-ins (*WordPress, Drupal e Joomla)*.

7. Conclusão e Trabalhos Futuro

Este artigo apresentou um estudo exploratório conduzido em relatos de vulnerabilidades disponíveis no CVE-D e em conjuntos de mudanças coletados do repositório de plug-ins do *WordPress*. O estudo observou mais especificamente à ocorrência, gravidade, custos de manutenção e o sentimento da comunidade de desenvolvedores em relação às vulnerabilidades causadas pelo uso indiscriminado de plug-ins em aplicações web. Encontramos 254 vulnerabilidades catalogadas no CVE-D associadas ao *WordPress*, das quais analisamos manualmente 22 causadas por plug-ins. Como resultado, conseguimos mapear os principais tipos de vulnerabilidades, observar que em sua grande maioria, as vulnerabilidades causadas por plug-ins não afetam gravemente a integridade das aplicações e que no geral, os desenvolvedores assumem uma postura neutra ao reportar as vulnerabilidades. Por fim, com base nos conjuntos de mudanças, constatamos que apenas uma porcentagem das vulnerabilidades implica em um grande esforço de manutenção.

Como trabalho futuro e implicações dos resultados obtidos podemos visualizar algumas possibilidades. Primeiro, destacamos a possiblidade de mineração dos dados diretamente da *National Vulnerability Database* (NVD) que contempla relatos de vulnerabilidades mais completos do que os disponíveis na base do CVE-D. Segundo, entendemos que é relevante uma melhor compreensão da complexidade de configuração do *WordPress* e o seu impacto nas vulnerabilidades caudas por plug-ins. Por fim, o nosso estudo confirma que existe uma demanda real por uma detecção antecipada de vulnerabilidades.

References

Bozorgi, M., Saul, L. K., Savage, S., & Voelker, G. M. (2010). Beyond heuristics: learning to classify vulnerabilities and predict exploits. 16th International conference on Knowledge Discovery and Data Mining. ACM.

Smith, B., & Williams, L. (2011). Using SQL hotspots in a prioritization heuristic for detecting all types of web application vulnerabilities. 4th International Conference on Software Testing, Verification and Validation. IEEE.

Sampaio, L., & Garcia, A. (2016). Exploring context-sensitive data flow analysis for early vulnerability detection. Journal of Systems and Software, v. 113, pp. 337-361.

Thome, J., Shar, L. K., Bianculli, D., & Briand, L. (2017). JoanAudit: A Tool for Auditing Common Injection Vulnerabilities. 11th Symposium on the Foundations of Software Engineering. ACM.

Ferreira, G., Malik, M., Kästner, C., Pfeffer, J., & Apel, S. (2016). Do #ifdefs influence the occurrence of vulnerabilities? an empirical study of the linux kernel. 20th International Systems and Software Product Line Conference. ACM.