

DETECÇÃO DE VULNERABILIDADES EM AMBIENTES WEB CONTROLADOS

Isadora Garcia Ferrão¹

Guilherme Neri Bustamante Sá²

Diego Luis Kreutz³

Resumo:

Neste trabalho são implementadas, em um ambiente controlado, as dez principais vulnerabilidades recorrentemente detectadas em sistemas Web segundo a classificação tri-anual da OWASP 2017. O ambiente controlado, além de criar um cenário conhecido para a avaliação das ferramentas, torna possível investigar outras coisas, como falsos positivos. Neste contexto, o objetivo principal do trabalho é avaliar os principais scanners de vulnerabilidades gratuitos, escolhidos através de uma pesquisa em sites de classificação, em um cenário controlado, dando um passo além (em relação a trabalhos existentes na literatura) na investigação da eficácia dessas ferramentas. Para atingir este objetivo, o desenvolvimento do trabalho foi organizado em cinco etapas bem definidas, indo desde a pesquisa do estado da arte até a análise dos resultados dos scanners. Surpreendentemente, os scanners de vulnerabilidades detectaram apenas 50% das dez categorias de vulnerabilidades implementadas no cenário controlado. A ferramenta que se destacou foi o Skipfish, identificando 80% do total de vulnerabilidades detectadas pelos scanners. Os resultados da avaliação da eficácia das diferentes ferramentas de varredura de vulnerabilidades, em um ambiente controlado, reforça a necessidade de se utilizar múltiplas ferramentas para encontrar e corrigir defeitos em sistemas Web. Outro aspecto surpreendente, constatado na pesquisa, é o fato de alguns scanners não detectarem algumas das vulnerabilidades que prometem detectar. Em outras palavras, isto significa que os usuários não podem confiar cegamente na documentação das ferramentas. Como trabalhos futuros podem ser citados: a avaliação do impacto de frameworks e Web Application Firewalls na segurança dos sistemas Web; e a utilização prática dos scanners de vulnerabilidades para identificar problemas em potencial em sites de prefeituras, escolas e outros órgãos públicos da região de abrangência da UNIPAMPA, prestando um serviço prático e importante à comunidade de inserção da instituição.

Palavras-chave: Segurança de sistemas; Segurança na Web; Scanners de vulnerabilidades

Modalidade de Participação: Iniciação Científica

DETECÇÃO DE VULNERABILIDADES EM AMBIENTES WEB CONTROLADOS

¹ Aluno de graduação. isadora-gf@hotmail.com. Autor principal

² Participante. guinbsa@gmail.com. Co-autor

³ Docente. diegokreutz@unipampa.edu.br. Orientador

DETECÇÃO DE VULNERABILIDADES EM AMBIENTES WEB CONTROLADOS

1 INTRODUÇÃO

Apesar da evolução das linguagens de programação, frameworks e outras tecnologias utilizadas no desenvolvimento de sistemas *Web*, o número de incidentes de segurança e vulnerabilidades antigas (exemplos: *SQL injection* e *cross site scripting*) detectadas nesses sistemas ainda é alta [Assunção, F. 2015; Coelho, S. et al. 2015]. As ferramentas mais utilizadas para detectar vulnerabilidades em sistemas *Web* são os conhecidos *scanners* de vulnerabilidades [Bau, J. et al. 2010]. Tipicamente, estas ferramentas são projetadas para identificar, classificar e diagnosticar problemas de sistemas. Por exemplo, *scanners* ajudam a detectar erros de programação que permitem a um atacante inserir e executar remotamente arquivos maliciosos no sistema *Web*.

Em uma pesquisa recente, foram investigados os 10 principais *scanners* de vulnerabilidades gratuitos [Ferrão, I. G, Kreutz, D. L. 2017]. Como ambiente de avaliação, os pesquisadores utilizaram uma máquina virtual da OWASP (<https://www.owasp.org/>) conhecida como BWA. Os resultados da pesquisa destacaram a eficácia, em termos de número de vulnerabilidades detectadas, dos *scanners* Zed Attack Proxy e Paros Proxy. Além disso, constataram que são necessários pelo menos 4 *scanners* para cobrir todas as vulnerabilidades encontradas na BWA.

O maior desafio do trabalho [Ferrão, I. G, Kreutz, D. L. 2017] foi comparar os *scanners* de vulnerabilidades em pé de igualdade levando em conta um cenário com um número muito grande de vulnerabilidades desconhecidas. Isto por que o BWA é um ambiente complexo, composto por dezenas de aplicações vulneráveis, criado especificamente para a formação de especialistas em segurança de sistemas. Na prática, é inviável tentar enumerar as vulnerabilidades existentes nas centenas de milhares de linhas de código (em diferentes linguagens de programação) das aplicações *Web* da BWA. A enumeração das vulnerabilidades desta máquina virtual demandaria um tempo extremamente longo (vários meses de trabalho).

Neste trabalho é proposto um ambiente controlado, onde são implementadas as 10 principais vulnerabilidades recorrentemente detectadas em sistemas *Web* segundo a classificação tri-anual OWASP de 2017 [OWASP 2017]. O ambiente controlado, além de criar um cenário conhecido para a avaliação das ferramentas, algo utilizado na prática por trabalhos similares [Vieira, M. et al. 2009; Bau, J. et al. 2010; Doupé, A. et al. 2012], torna possível investigar outras coisas, como falsos positivos. Um falso positivo consiste na identificação de uma vulnerabilidade, que, na verdade, não representa nenhum perigo eminente.

Considerando o contexto exposto, o objetivo principal deste trabalho é avaliar os principais *scanners* de vulnerabilidades gratuitos em um cenário controlado, dando um passo além (em relação a trabalhos existentes na literatura) na investigação da eficácia dessas ferramentas. As principais contribuições do trabalho são:

- a criação do ambiente controlado (uma máquina virtual GNU/Linux contendo a implementação das 10 principais vulnerabilidades OWASP de 2017);
- a execução e avaliação dos 10 principais *scanners* de vulnerabilidades gratuitos sobre o cenário controlado;
- a identificação de falhas de cobertura (vulnerabilidades críticas não detectadas) nos resultados dos *scanners* para o cenário controlado;
- a identificação e apresentação de trabalhos futuros, que possam dar continuidade à pesquisa.

O restante deste texto está organizado como segue. A Seção 2 detalha as etapas de desenvolvimento do trabalho. Os resultados são discutidos na Seção 3. Por fim, as considerações finais e os trabalhos futuros são apresentados.

2 METODOLOGIA

O desenvolvimento da pesquisa foi estruturado em cinco etapas distintas, como resumido na subseção a seguir. Os resultados foram coletados a partir do ambiente controlado, cujos (alguns) detalhes técnicos são apresentados na Subseção 2.2.

2.1 Etapas da Pesquisa

A **primeira etapa** foi caracterizada pela escolha, instalação e estudo dos *scanners*. Os *scanners* escolhidos foram os mesmos utilizados no trabalho [Ferrão, I. G, Kreutz, D. L. 2017], ou seja, Uniscan, Paros, Zed Attack Proxy, Nessus, Ratproxy, Grabber, Wapiti, Andiparos, Skipfish e Vega. Os principais motivos da escolha foram o fato destas ferramentas serem gratuitas e de código aberto, não existirem estudos que comparem estas ferramentas, similaridade de objetivos entre elas e sites de classificação de ferramentas de segurança.

Na **segunda etapa** da pesquisa, foram escolhidas as vulnerabilidades do cenário controlado. O parâmetro de seleção das vulnerabilidades foi o índice de recorrência ao longo do tempo. Segundo a OWASP, organização internacional que monitora vulnerabilidades em sistemas *Web*, as mais recorrentes, conforme a classificação de 2017, são: injeção de código (V1), quebra de autenticação e gerenciamento de sessão (V2), *cross-site scripting* (XSS) (V3), quebra de controle de acesso (V4), má configuração de segurança (V5), exposição de dados sensíveis (V6), falta de proteção contra ataques (V7), *cross-site request forgery* (CSRF) (V8), uso de componentes com vulnerabilidades conhecidas (V9) e APIs desprotegidas (V10).

A **terceira etapa** foi constituída pelo projeto e implementação de um cenário controlado contendo as 10 vulnerabilidades selecionadas na etapa anterior. O cenário controlado foi criado a partir da distribuição Kali Linux (versão 4.9.0), servidor MySQL (versão 5.6) e servidor *Web* Apache (versão 2.4.27). O Kali Linux foi escolhido pelo fato de já possuir a maioria das ferramentas instaladas no sistema e ser voltado para testes de segurança. As vulnerabilidades foram implementadas na linguagem de programação PHP (versão 7.0).

Na sequência, a **quarta etapa** foi utilizada para a execução dos *scanners* de vulnerabilidades sobre o ambiente controlado. O tipo mais comum de testes para sistemas *Web* é o black-box, também utilizado neste trabalho. Vale ressaltar que este é o tipo de testes que mais se aproxima ao que é empregado na prática pela maioria dos atacantes. Os testes black-box são realizados de forma automatizada e sem acesso direto a detalhes de infraestrutura e codificação do sistema alvo.

Os testes práticos foram realizadas em uma máquina Dell Inspiron Special Edition, Intel core i7, sexta geração, memória de 16 GB e 1 TB de HD + 8 GB de SSD. Enquanto os *scanners* foram instalados no sistema hospedeiro Kali Linux, o cenário controlado foi instalado em uma máquina virtual utilizando o VirtualBox (versão 5.2.18).

Finalmente, na **quinta etapa** foram analisados os relatórios gerados pelos *scanners* na etapa anterior. Cada *scanner* gera um relatório de saída específico, num formato próprio, contendo as especificações das vulnerabilidades encontradas, como o nível de gravidade da vulnerabilidade, como explorar, como corrigir e onde encontrar. Portanto, na fase de análise foram inspecionados todos os detalhes técnicos dos relatórios de saída e, no final, foi gerada uma síntese por *scanner* (ver Seção 3).

2.2 O Cenário Controlado

No cenário controlado, para a implementação das 10 vulnerabilidades mais recorrentes em sistemas *Web*, foram criados dois formulários *Web* utilizando a linguagem de programação PHP. O primeiro para o acesso ao sistema (login) e o segundo para o cadastro de usuários. Estes formulários foram escolhidos por estarem presentes na maioria dos sistemas *Web*. Ambos os formulários possuem os campos *login/nome* e *senha*.

O trecho de código da **Figura 1** representa a consulta SQL utilizada no sistema para verificar o *nome* e a *senha* do usuário. No exemplo, o comando *SELECT* retorna todas as linhas da tabela *usuarios*, sendo que *nome* é igual a variável *\$nome* e *senha* é igual a variável *\$senha* cifrada. Aparentemente, parece ser um código trivial e correto. Contudo, na verdade, este código é vulnerável e permite ataques do tipo *SQL Injection* (injeção de código SQL).

Figura 1- Trecho de código suscetível a *SQL injection*

```
mysqli_query ($conexao, "SELECT * FROM usuarios WHERE nome = '$nome' AND  
senha = '$senhaencriptada' ");
```

As falhas de injeção de código ocorrem quando dados não confiáveis são enviados, por usuários maliciosos, para o sistema alvo. Este tipo de falha pode permitir o acesso a informações sensíveis do sistema, as quais o usuário não deveria ter acesso, isto é, não tem autorização explícita para acessar. No exemplo, as variáveis *nome* e *senha* representam um problema grave de segurança. Como os dados de entrada delas não são verificados, um atacante pode injetar código, modificando o comando SQL original.

Suponha que o atacante entre com o valor *"1' OR '1' = '1"* no campo *nome* do formulário de login. Consequentemente, a variável *nome* vai alterar o comando SQL, uma vez que o caractere aspas simples (') é especial e *OR* é uma operação SQL. Neste caso, o comando SQL resultante vai "deixar de verificar" a *senha* armazenada no banco de dados. Com a alteração do comando SQL, sempre que 1 for igual a 1, a sessão será liberada. Como 1 é sempre igual a 1, a sessão será sempre liberada.

3 RESULTADOS

A **Tabela 1** apresenta os resultados obtidos com os testes black-box no cenário controlado. As ferramentas são identificadas pelos seguintes índices: [1] Andiparos, [2] Nessus, [3] Ratproxy, [4] Uniscan, [5] Wapiti, [6] Grabber, [7] Paros Proxy, [8] Skipfish, [9] Vega e [10] Zed Attack proxy. Nas colunas da tabela aparecem os nomes das vulnerabilidades e as ferramentas que a detectaram, respectivamente.

Surpreendentemente, como pode ser observado na **Tabela 1**, os *scanners* de vulnerabilidades detectaram apenas 50% das 10 categorias de vulnerabilidades implementadas no cenário controlado (lembrando que são as 10 mais recorrentes, na Internet, segundo o ranking da OWASP 2017). Apenas as vulnerabilidades injeção de código, *cross site scripting*, exposição de dados sensíveis, má configuração de segurança e uso de componentes com vulnerabilidades conhecidas foram detectadas pelos *scanners*.

A ferramenta que se destacou em relação às demais foi o Skipfish, detectando 80% do total de vulnerabilidades identificadas pelo conjunto dos 10 *scanners*. Em segundo lugar, o Nessus detectou 60% das vulnerabilidades. Além disso, ele foi o único *scanner* a detectar a vulnerabilidade de usar componentes com vulnerabilidades conhecidas. Estes resultados apontam para um cenário que pode ser considerado preocupante para o ecossistema *Web*. Os *scanners* de vulnerabilidades, em especial os gratuitos, precisam claramente evoluir, isto é, melhorar a eficácia em termos de detecção de vulnerabilidades em sistemas *Web*. Isto fica ainda mais evidente na **Tabela 2**, onde são relacionadas as vulnerabilidades que os *scanners* prometem detectar, mas, de fato, não detectam ou detectam de forma muito limitada

(exemplo: em casos muito específicos de ocorrência da vulnerabilidade). Isto significa que não é possível confiar-se na documentação técnica dos *scanners*.

Tabela 1 -Detecção de vulnerabilidades no cenário controlado.

Vulnerabilidades	Scanners	Qntd. de vulnerabilidades
Injeção de Código	[4,8]	[7]
Quebra de Autenticação e Gerenciamento de Sessão	[]	[]
<i>Cross site scripting</i>	[1,2,6,8,9,10]	[8]
Quebra de controle de acesso	[]	[]
Má configuração de segurança	[1,4,7,8,9,10]	[6]
Exposição de dados sensíveis	[1,2,8,9,10]	[6]
Falta de proteção contra ataques	[]	[]
<i>Cross site requesty forgery</i> (CSRF)	[]	[]
Uso de componentes com vulnerabilidades	[2]	[1]
APIs desprotegidas	[]	[]

Fonte: do autor, 2018.

Tabela 2 -Falhas na detecção de vulnerabilidades.

Vulnerabilidades	<i>Scanners</i> que falharam em detectar
Injeção de Código	[1,3,5,6,7,9]
<i>Cross site scripting</i>	[3,4,5,7]
Quebra de controle de acesso	[8]
Má configuração de segurança	[2,3,5,6]

Fonte: do autor, 2018.

A taxa de falsos-positivos no cenário controlado, considerando o conjunto de *scanners* testados, ficou em 20%. A identificação dos falsos-positivos foi realizada através da verificação dos relatórios de saída de cada ferramenta. A ferramenta Skipfish foi responsável 85% do conjunto de todos os falsos-positivos encontrados. Como os falsos-positivos representam trabalho de verificação, geralmente manual, das supostas falhas dos sistemas, fica evidente que ainda há espaço para investigação e desenvolvimento.

4 CONSIDERAÇÕES FINAIS

A avaliação da eficácia de diferentes *scanners* de vulnerabilidades, em um ambiente controlado, reforça a necessidade de se utilizar múltiplas ferramentas para encontrar e corrigir defeitos em sistemas *Web*. Além disso, surpreendentemente, os *scanners* detectaram apenas 50% das 10 vulnerabilidades mais recorrentes em sistemas *Web*. A partir deste resultado pode-se concluir que os *scanners* investigados ainda carecem de pesquisa e desenvolvimento.

Outro aspecto surpreendente constatado na pesquisa é o fato de alguns *scanners* não detectarem algumas das vulnerabilidades que prometem detectar. A exemplo, *SQL injection* foi a vulnerabilidade onde os *scanners* mais falharam em detectar.

SQL injection e *cross site scripting* são as vulnerabilidades mais frequentemente detectadas pelos *scanners*. Apesar disto, e do nível de criticidade, estas vulnerabilidades continuam na lista das mais recorrentes em sistemas *Web*. Este cenário, aliado aos dados apresentados nesta pesquisa, permitem concluir que há uma falta de atenção por parte dos desenvolvedores de sistemas e falha na utilização de ferramentas como *scanners* de vulnerabilidades no ciclo de vida de sistemas online.

Finalmente, como trabalhos futuros podem ser citados: a avaliação do impacto de frameworks e *Web Application Firewalls* na segurança dos sistemas *Web*; e a utilização dos *scanners* de vulnerabilidades para identificar problemas em potencial em sites de prefeituras, escolas e outros órgãos públicos da região de abrangência da UNIPAMPA, prestando um serviço prático e importante à comunidade de inserção da instituição.

REFERÊNCIAS

ASSUNÇÃO, F. Análise da eficiência na detecção de vulnerabilidades em ambientes web com o uso de ferramentas de código aberto. In: Projetos e Dissertações em Sistemas de Informação e Gestão do Conhecimento 4.2. Vol. 4, n. 2, 2015.

BAU, J. et al. State of the art: Automated black-box web application vulnerability testing. In: IEEE Symposium on Security and Privacy. IEEE, 2010. p. 332-345

COELHO, S. et al. Identificando vulnerabilidades de segurança em uma aplicação web. In: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG). WTIC, 2015.

DOUPÉ, A. et al. Enemy of the state: A state-aware black-box web vulnerability scanner. In: Usenix Security Symposium. Vol. 14, 2012.

FERRÃO, I. G, KREUTZ, D. L. Segurança na Web: análise black-box de scanners de vulnerabilidades. In: Escola Regional de Engenharia de Software (ERES), 2017. p. 149-156.

OWASP. Owasp Top Ten, 2017. Disponível em <<https://www.owasp.org>>

VIEIRA, M. et al. Using web security scanners to detect vulnerabilities in web services. In: Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on. IEEE, 2009. p. 566-571.