Uma Avaliação da Proteção de Dados Sensíveis através do Navegador Web

Carlo M. R. da Silva[1], **Vinicius C. Garcia**[1]

[1]CIn – Universidade Federal de Pernambuco (UFPE)
Av. Jorn. A. Fernandes, s/n. – CDU, 50740-560 – Recife – PE – Brasil

cmrs, vcg@cin.ufpe.br

Abstract. *Web browsers are tools of utmost importance with regard to the use of data on the internet, because they allow interaction and consumption of information provided by various services available on the Web. However, it is clear the difficulty of these tools to prevent your users are victims vulnerabilities, are located in its own internal, coming from Web applications available or results the ingenuity of the user. Our study aims at two contributions: (i) the development of a controlled environment, which is defined as a Web application that simulates real vulnerabilities susceptible to 13 separate attacks, considered most emerging today. (ii) and an evaluation of the effectiveness of 25 tools to protect users.*

Resumo. *Navegadores Web são ferramentas de extrema importância no que diz respeito ao consumo de dados na internet, pois possibilitam a interação e consumo de informações providas por diversos serviços disponíveis na Web. Em contrapartida, é nítida a dificuldade destas ferramentas em evitar que seus usuários sejam vítimas de vulnerabilidades, sejam localizadas em seus próprios mecanismos internos, oriundas de aplicações Web disponíveis ou resultadas pela ingenuidade do próprio usuário.Nosso estudo tem como objetivo duas contribuições: (i) o desenvolvimento de um ambiente controlado, que se define como uma aplicação Web que simula vulnerabilidades reais suscetíveis a 13 ataques distintos, considerados mais explorados da atualidade. (ii) E uma avaliação da eficácia de 25 ferramentas que visam proteger os usuários de navegadores web contra estes ataques.*

1. Introdução

Não há dúvidas que a Web se consagrou como um dos mecanismos computacionais mais preciosos para a humanidade. E-mail, e-commerce, redes sociais e transações online são alguns dos recursos que são oferecidos, seja para fins de lazer ou negócios, beneficiando pessoas, empresas ou organizações. Consequentemente, um software se popularizou de tal forma que faz parte do dia a dia de diversos usuários, ele é conhecido como navegador Web. Contudo, o que podemos observar é uma grande dificuldade dos desenvolvedores de navegadores web em atender com eficiência os requisitos relacionados à Segurança da Informação (SI) diante a desenfreada evolução da web. Um bom exemplo são os casos cada vez mais frequentes de vulnerabilidades nos navegadores mais populares e seus recursos adicionais [Secunia], resultando no comprometimento de dados sensíveis.

Segundo Allen [Allen 2001], um dado sensível é todo e qualquer tipo de informação que ao ser visualizado, modificado ou indisponível sem a devida autorização,

poderá acarretar em algum transtorno ao proprietário da informação. Geralmente estes incidentes são catalogados em fontes literárias no intuito de inventariar as falhas, facilitando a identificação e prevenção dos ataques e vulnerabilidades decorrentes. Algumas organizações especializadas no assunto, a exemplo do MITRE[1] e OWASP[2], se esforçam em manter um catálogo destas ameaças.

Este estudo possui dois objetivos: (i) O desenvolvimento de um ambiente controlado, que se define como uma aplicação web com vulnerabilidades existentes de forma proposital. O intuito disso é provê um ambiente para estudo e análise fazendo uso de simulações do comportamento das vulnerabilidades existentes nos ambientes reais. Este tipo de aplicação não é algo novo, contudo, nossa contribuição tem como proposta uma reciclagem. Nossa análise exploratória de ambientes controlados disponíveis trouxe à tona a conclusão que nenhum dos ambientes disponíveis seria suficientemente apto a reproduzir os ataques e vulnerabilidades mais exploradas da atualidade. Nós descrevemos em detalhes o desenvolvimento do nosso ambiente controlado, onde revelamos desafios e limitações diante as opções disponíveis.

O segundo objetivo (ii) é apresentar um diagnóstico sobre a eficácia de mecanismos que visam minimizar os ataques mais explorados que atuam diretamente entre o usuário e o navegador Web. Para atingir nossos objetivos, executamos nossa avaliação fazendo uso de uma metodologia baseada em métricas, onde submetemos 25 ferramentas que atuam diretamente no navegador, com o intuito de minimizar a violação de dados sensíveis. As ferramentas selecionadas foram extraídas de repositórios públicos[3][4], gerenciados pelo próprio desenvolvedor do navegador submetido ao respectivo teste. Por fim, nós apresentamos os resultados das ferramentas submetidas e descrevemos nossas conclusões e trabalhos futuros a serem explorados.

2. Contextualização

Nesta seção serão apresentadas as principais referências literárias, além de certos termos e definições utilizados durante o desenvolvimento desta pesquisa. Como ponto de partida, foi necessário realizar um levantamento de ameaças que trazem como consequência a violação de dados sensíveis aos usuários de navegador web. Através do levantamento preliminar, foram selecionadas algumas referências literárias de alto respaldo sobre a problemática deste estudo. A primeira delas é a OWASP Top Ten 2013 [OWASP 2013], esta referência caracteriza-se como um catálogo dos 10 principais ataques mais explorados em aplicações web. Esta obra é mantida pela OWASP, uma entidade sem fins lucrativos e de respaldo internacional que visa melhorias e boas práticas de segurança.

Outra referência é a CWE/SANS 25 Most Dangerous Software Errors [MITRE 2011], que relaciona um grande conjunto de vulnerabilidades conhecidas em aplicações de diferentes ambientes. No caso de aplicações web, existe um catálogo de mais de 20 vulnerabilidades distintas. Os artefatos são mantidos pela SANS/MITRE, organizações que já publicaram inúmeras referências relevantes ao tema. E por fim, a HTML5Security [HTML5Security 2013], que é um grupo de pesquisa direcionado aos

[1]MITRE: http://www.mitre.org/
[2]OWASP: https://www.owasp.org/
[3]Chrome Web Store: https://chrome.google.com/webstore/
[4]Add-ons for Firefox: https://addons.mozilla.org/

aspectos de segurança do HTML5.

Para descrever uma visão geral da problemática, a Figura 1 apresenta um fluxograma que ilustra o cenário comum de interação na Web. O fluxo do usuário geralmente se inicia através de um navegador, este software é capaz de oferecer praticamente tudo que o usuário precisa para acessar o conteúdo disponível na web [DeRyck et al. 2013]. O intuito do navegador é provê a interação entre um usuário e uma aplicação Web, esta que se define como o serviço que administra um ativo pertencente ao usuário. Neste contexto, o ativo representa todo e qualquer conjunto de dados sensíveis que tem seu acesso restrito [Allen 2001].

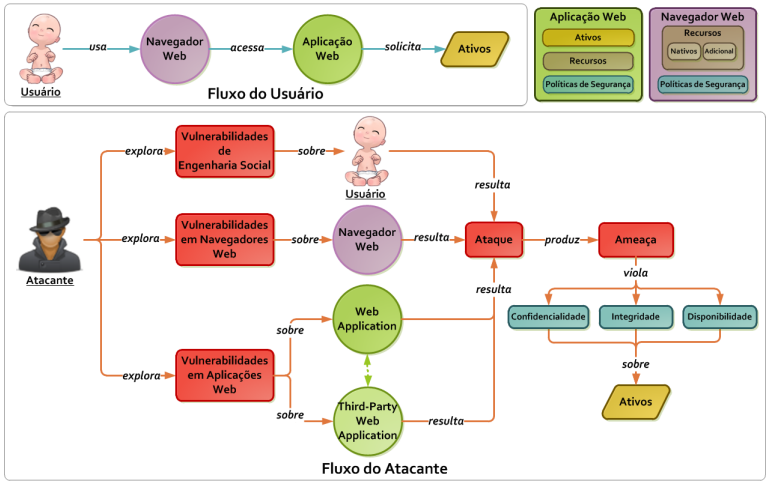

Figura 1. Fluxograma da problemática do estudo

O fluxo do atacante é mais complexo, pois a problemática se endereça em buscar comportamentos não esperados pelos administradores do ativo do usuário. Seu primeiro passo é buscar meios de realizar suas ações ilícitas, que basicamente é condicionado a diferentes métodos de exploração de vulnerabilidades. Nosso estudo tem o propósito de estabelecer uma tríade desta problemática, representada em três métodos, a saber:

Vulnerabilidades em Aplicações: Esta ameaça tange as falhas oriundas do desenvolvimento da aplicação web, seja pela ausência ou ineficiência de um solido processo de desenvolvimento seguro, ou brechas nas entradas da aplicação ou pela utilização de componentes com vulnerabilidades conhecidas. Apesar de existentes na aplicação, ainda sim, suas vulnerabilidades são exploradas quando a aplicação é renderizada no navegador e suas funcionalidades são disponibilizadas ao usuário.

Vulnerabilidades em Navegadores Web: Esta ameaça se caracteriza em componentes ou complementos do navegador com vulnerabilidades conhecidas. Uma vez que o navegador trafega dados sensíveis constantemente, esta ameaça proporciona um impacto de negócio equivalente à propagação da popularidade destas ferramentas [Gaurav Aggrawal and Boneh 2010]. Este estudo cobre as vulnerabilidades dos componentes nativos, bem como complementos de terceiros que são incorporados no navegador web [Silic et al. 2010].

Vulnerabilidades de Engenharia Social: A Engenharia Social possui um conceito muito amplo, portanto é importante salientar que no escopo desta pesquisa está

intrinsecamente relacionada às ações ilícitas de usuários mal-intencionados que vão bem além dos aspectos tecnológicos, como por exemplo, persuadir um usuário a uma determinada ação que resulte na violação de seus dados sensíveis. Este aspecto está restritamente relacionado às intervenções maliciosas que induzem o usuário a expor seus dados a riscos acessíveis pelo navegador e propagados por mal-intencionados.

Definida a tríade da problemática, foi possível identificar nas referências literárias [OWASP 2013] [MITRE 2011] [HTML5Security 2013] os ataques considerados mais explorados da atualidade que estão intrinsecamente relacionados ao uso de navegador web, conforme ilustrado na Tabela 1.

Tabela 1. Listagem dos ataques mais explorados

Ataques	Sigla	Categoria
Clickjacking	A1	Vulnerabilidades de Engenharia Social
Cross-Site Request Forgery (CSRF)	A2	Vulnerabilidades em Aplicações Web
Cross-Site Scripting (XSS)	A3	Vulnerabilidades em Aplicações Web
Exposição de Dados Sensíveis da Aplicação	A4	Vulnerabilidades em Aplicações Web
Exposição de Dados Sensíveis do Navegador	A5	Vulnerabilidades em Navegadores Web
Falhas na Política de Mesma Origem (PMO)	A6	Vulnerabilidades em Aplicações Web
Injeção SQL (SQLi)	A7	Vulnerabilidades em Aplicações Web
Malwares	A8	Vulnerabilidades de Engenharia Social
Phishing	A9	Vulnerabilidades de Engenharia Social
Redirecionamentos e Encaminhamentos Inválidos	A10	Vulnerabilidades em Aplicações Web
Utilização de Complementos Vulneráveis Conhecidos no Navegador	A11	Vulnerabilidades em Navegadores Web
Utilização de Componentes Vulneráveis Conhecidos na Aplicação	A12	Vulnerabilidades em Aplicações Web
Utilização de Componentes Vulneráveis Conhecidos no Navegador	A13	Vulnerabilidades em Navegadores Web

3. Avaliação

Nesta seção será descrito um diagnóstico sobre a eficiência das principais ferramentas de proteção disponíveis para usuários de navegador web. A justificativa de optarmos por extensões como mecanismos de segurança é que mesmo existindo outras ferramentas, tais como antivírus, existem diversos aspectos entre a aplicação e o navegador que precisam ser considerados, como por exemplo, a execução de scripts. Certos comportamentos precisam ser interceptados antes da interação entre o navegador do usuário ser concluída. Ou seja, as requisições precisam ser analisadas antes que sua resposta seja processada, e este tratamento só pode ser realizado por uma ferramenta que possua um alto nível de interação e privilégio sobre as ações do navegador.

Um fato que justifica este argumento é o número expressivo de extensões com este propósito que se encontram disponíveis em repositórios de aplicativos. Estes repositórios se tornaram uma prática comum entre os proprietários de navegadores, onde analisam e gerenciam tais ferramentas, muitas desenvolvidas por terceiros, para em seguida serem distribuídas aos seus usuários como um complemento de navegação.

3.1. Metodologia

A metodologia da avaliação baseia-se na abordagem GQM (Goal, Question, Metric) proposta por [V. Basili and Rombach 1994], com o intuito de proporcionar um formalismo e planejamento quanto à medição dos resultados a serem extraídos nas ferramentas submetidas aos testes. Essa abordagem divide-se em quatro fases: (i) Planejamento; (ii) Definição; (iii) Coleta de Dados; e (iv) Interpretação e Análise dos Resultados.

3.2. Planejamento

Segundo [V. Basili and Rombach 1994], o objetivo desta fase é definir o plano de projeto, no qual se descreve: O ambiente, participantes e o que será avaliado. O primeiro passo foi realizar uma busca por ambientes controlados disponíveis na web, estes têm como objetivo simular um cenário real, a fim de testar uma determinada causa e efeito. Para a realização dessa avaliação, foi utilizado o projeto "OWASP Broken Web Application"[5], OBWP, que se define como um conjunto de aplicações que auxiliam no estudo de vulnerabilidades. Algumas aplicações são fictícias com vulnerabilidades intencionais, outras são reais em versões antigas com vulnerabilidades conhecidas.

Nossa pesquisa também detectou a existência de outras aplicações que não fazem parte do projeto OBWP, mas que se encontram no contexto deste estudo, totalizando em 29 ambientes, conforme ilustrado na Tabela 2. Como triagem inicial, os únicos critérios estabelecidos foram que a ferramenta cobrisse ao menos um dos 13 ataques emergentes e seu conteúdo fosse disponível gratuitamente.

Tabela 2. Ambientes controlados disponíveis e a cobertura aos ataques

Ambientes	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
Acunetix VulnWeb			X				X						
AltoroMutual		X	X				X						
Bad Store			X				X						
Cenzic (CrackMe Bank)			X				X						
Cyclone Transfers			X	X			X						
Damn Vulnerable Web Application			X				X						
EeSafe.org			X				X						
EnigmaGroup			X				X						
Exploit- DB							X						
Exploit KB Vulnerable Web App							X						
Google Gruyere		X	X	X			X						
Hackxor		X	X				X						
Hacme Apps			X	X			X						
Moth			X				X						
Mutillidae			X	X			X						
OWASP Hackademic			X				X						
OWASP SiteGenerator			X				X						
OWASP Vicnum			X	X			X						
OWASP Web Goat		X	X	X		X	X						
PCTechtips Challenge							X						
Peruggia		X	X	X			X						
Stanford SecuriBench			X	X			X						
The Bodgeit Store		X	X	X			X						
The Butterfly Security Project			X				X						
WackoPicko			X				X						
Watchfire			X				X						
Web Security Dojo			X				X						
WebMaven/Buggy Bank			X				X						
XSS Encoding Skills x5s			X										

De posse dos ambientes controlados, o próximo passo foi realizar uma triagem mais criteriosa com o objetivo de selecionar as aplicações mais aderentes ao tema, selecionando assim aquelas que serão utilizadas na avaliação. Essa seleção foi baseada em

[5]OWASP Broken Web: https://goo.gl/NScjJ

Tabela 3. Resultado da triagem dos Ambientes Controlados

Ambiente Controlado	Ataques Relacionados	Última Atualização
OWASP WebGoat 6.4	5	2012
The BodgeIt Store 1.4	4	2012
Google Gruyere	4	2010
Peruggia 1.2	4	2010

dois critérios, a saber: (i) A aplicação com maior número de ataques relacionados; E (ii) a aplicação com atualizações mais recentes, considerando o semestre e ano. Como critério de desempate, foi considerada a ferramenta com maior período de atividade e documentação disponível. A Tabela 3 ilustra o resultado dessa triagem, exibindo as aplicações consideradas mais aderentes ao tema. Conforme os critérios mencionados, a aplicação selecionada foi a OWASP WebGoat[6], pois além de estar disponível há mais de 10 anos de atividade, também é rica em documentação.

Contudo, analisando a listagem da Tabela 2, é possível observar que nenhum dos ambientes controlados disponíveis cobre os ataques A1, A5, A8, A9, A10, A11, A12 e A13. Além disso, também não há avaliação para alguns comportamentos do HTML5 que estejam relacionados os ataques A3, A4 e A7, bem como outros aspectos específicos abordados sobre os ataques A3, A4 e A6. Baseado nessas conclusões, este estudo precisou desenvolver um novo ambiente controlado, intitulado "Aegis Web Threats" (AegisWT), uma aplicação J2EE que tem por objetivo apresentar intencionalmente vulnerabilidades que até o momento são impossíveis de serem reproduzidas nas opções disponíveis de ambientes controlados, conforme evidenciado na Tabela 2. De posse da AegisWT e WebGoat, se faz possível a construção de um cenário que contemple todas as vulnerabilidades.

3.3. Definição

As abordagens de [V. Basili and Rombach 1994] relatam que é na fase da definição onde devem ser descritos os objetivos, questões e métricas, conforme apresentados nas Tabelas 4, 5 e 6. Os ambientes controlados irão disponibilizar páginas vulneráveis nas quais uma aplicação teste, denominada CheckVulnTest, irá aguardar um determinado comportamento resultante de um ataque bem-sucedido. Nesse momento, as Extensões deverão tomar alguma ação que impeça o ataque de ser realizado, seja bloqueando recursos do navegador, cancelando a requisição ou resposta ou mesmo solicitando uma confirmação ao usuário para continuar o processo suspeito. O objetivo esperado é que a extensão modifique o comportamento final, resultando em algo que não seja esperado pela CheckVulnTest, ou seja, impedindo assim a reprodução do ataque. As métricas são baseadas nas principais técnicas de ataques descritas neste estudo.

3.4. Coleta de Dados

A coleta de dados foi realizada com base nas questões e métricas definidas na seção Descrição. Essas métricas foram obtidas por meio da execução de Testes Funcionais que, na Engenharia de Software, são responsáveis por realizar uma inspeção dos componentes de um determinado sistema com o objetivo de analisar resultados esperados [Leitner et al. 2007]. A técnica valida se o comportamento e funcionalidades do sistema estão de acordo com os objetivos definidos. Como o objetivo da extensão é a proteção do

[6]WebGoat: https://goo.gl/9RN63

lado cliente, os comportamentos internos dos sites visitados pelo usuário não são visíveis a estes mecanismos, portanto o método de avaliação será baseado em testes funcionais, também conhecidos como caixa preta [Leitner et al. 2007].

Conforme ilustrado na Figura 2, a aplicação CheckVulnTest ficou responsável em executar as métricas para avaliar o comportamento de uma extensão instalada no navegador. Cada extensão foi avaliada individualmente, ou seja, quando uma extensão era avaliada, as demais permaneciam desabilitadas no navegador, a fim de eliminar a possibilidade de eventuais conflitos. A automação em testes é a utilização de um software para controlar a execução dos testes da funcionalidade de outro software [Leitner et al. 2007]. Quanto à adoção do processo automatizado, se faz partindo do princípio que uma vez permitida à utilização de simulação automatizada, espera-se um resultado que proporcione um experimento mais quantitativo. Diante disto, fica evidente a possibilidade de alternar variáveis e comparar comportamentos distintos, o que proporciona um considerável número de extensões submetidas aos testes.

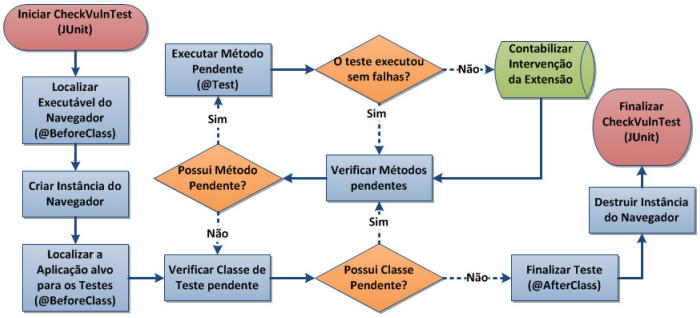

Figura 2. Fluxo de funcionamento da aplicação CheckVulnTest

A ferramenta para a automação dos testes foi a Selenium Server, a qual oferece automação diretamente no navegador. A aplicação CheckVulnTest foi desenvolvida em J2SE e faz uso do Framework JUnit como apoio na execução dos Testes Unitários. Além do JUnit, também utiliza as bibliotecas do Selenium WebDriver, que realiza os testes de forma automática em uma instância do navegador, dando suporte a diversos navegadores disponíveis. Cada classe de teste possui uma série de métodos, baseados nas métricas que aguardam um determinado comportamento na página.

O primeiro passo é a inicialização da CheckVulnTest, que irá localizar no computador o caminho absoluto do executável do navegador. Uma vez localizado, o Selenium WebServer irá criar uma instância do navegador e executará os métodos dos testes. Entretanto, uma vez o método não recuperando um comportamento esperado, causará um erro ao processo de teste, e é nesse momento que a CheckVulnTest irá contabilizar o evento, significando assim a intervenção defensiva da extensão.

Para garantir maior precisão nos resultados, após a execução dos testes, foi realizada uma triagem no log resultante da conclusão da CheckVulnTest, verificando assim o motivo das falhas nos métodos, para confirmar a intervenção das Extensões quanto ao comportamento do resultado destes métodos. Outro ponto que vale ressalva é que em nenhum dos casos a proteção nativa do navegador, a exemplo da Política de Mesma Origem, interveio quanto à reprodução dos testes. Na Tabela 7 são listadas as Extensões com suas respectivas versões utilizadas nessa avaliação. Estes recursos compõem o ambiente

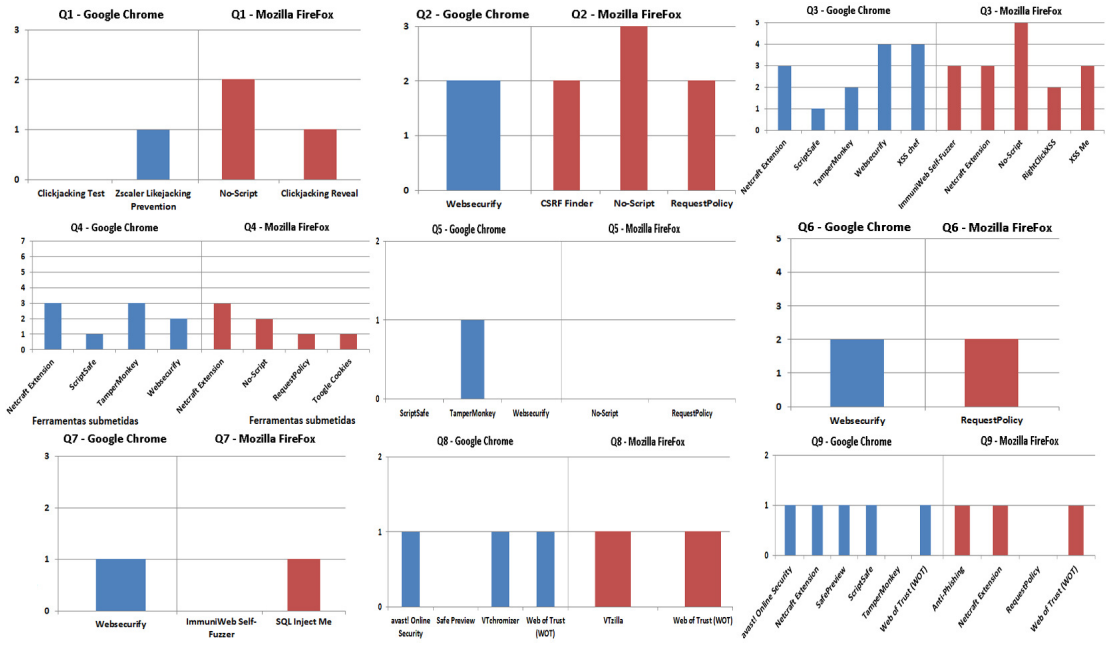

Figura 3. Resultados das avaliações

controlado para a execução das Questões, Q, com suas respectivas Métricas, M, com base nas definições descritas nas Tabelas 4, 5 e 6.

3.5. Análise dos Resultados

Com base na Tabela 7, foram submetidas 25 ferramentas aos testes desta experimentação, e seus respectivos resultados seguem conforme ilustrado na Figura 3.

O cenário proposto abrange os comportamentos mais elementares considerados de risco por esta pesquisa, portanto para uma ferramenta ser considerada satisfatória, ela deve atender todas as métricas estabelecidas em uma respectiva questão. Outro fato que vale ser ressaltado é que os falsos positivos não foram considerados nestes testes, uma vez que o estudo atribui o fator humano responsável pelas correções dos erros de filtragem. Sendo assim os cenários apresentados foram definidos como ameaças que simulam riscos reais ao usuário. Nas Tabelas 8 e 9 são descritos os resultados e considerações do estudo com base das métricas descritas nas Tabelas 4, 5 e 6.

3.6. Limitações do Presente Estudo

Em relação às ameaças da nossa avaliação, como em qualquer tipo de experimento, a automação pode ocasionar em riscos na confiabilidade dos resultados obtidos. Contudo, nós consideramos que a nossa metodologia adquiriu ganhos em produtividade e precisão quando decidimos eliminar o caráter subjetivo da avaliação, no caso, a intervenção manual. E diante do contexto desse estudo não há nada que um ser humano possa fazer que a automação não seja capaz de reproduzir. Quanto à limitação de escopo, no estado atual da nossa avaliação só pudemos submeter dois navegadores, o Google Chrome Desktop 32.0.1700.76 for Windows e o Mozilla Firefox Desktop v27.0 for Windows, em nosso ambiente controlado. Além disso, em relação ao critério de seleção das ferramentas de proteção, apenas consideramos opções disponíveis nos repositórios mantidos pelos desenvolvedores dos respectivos navegadores.

Tabela 4. Questões e Métricas para os ataques A1 até A5.

Questões	Objetivos (Goals)	Métricas	Extensões
[Q1] Quão eficaz é a proteção das ferramentas diante dos ataques A1?	(1).[AegisWT] A CheckVulnTest acessará uma página a qual possui um formulário com campos sobrepostos. Ela irá submeter a página e aguardar o envio concluído com sucesso, indicando que o ataque foi realizado. (2).[AegisWT] A CheckVulnTest acessará uma página que possui links sobrepostos com elementos que ao serem clicados, redirecionam para um Phishing. A aplicação irá aguardar o carregamento da página, indicando que o possível sequestro do click. (3).[AegisWT] A CheckVulnTest acessará uma página com frames cruzados e o cabeçalho com ausência do X-Frame-Options. A permissão de acesso à página já representa uma ameaça, portanto a CheckVulnTest aguardará apenas o carregamento da mesma.	(M1).Roubo de informações em Formulário (M2).Roubo de cliques (M3).Análise de X-Frame-Options	(1).Clickjacking Reveal. (2).Clickjacking Test. (3).No-Script. (4).Zscaler Likejacking Prevention.
[Q2] Quão eficaz é a proteção das ferramentas diante dos ataques A2?	(1).[WebGoat] A CheckVulnTest acessará uma página que possui um formulário de envio de email no qual é possível injetar um elemento <*img*> que realiza uma requisição através do atributo SRC com parâmetros modificados. A CheckVulnTest interpretará o ataque como concluído caso seja possível submeter o formulário.linebreak (2).[WebGoat] A CheckVulnTest acessará uma página a qual possui um formulário de envio de e-mail que possui controle de submissão dos dados através de um Token. O Token em questão está armazenado de forma insegura, através de um campo hidden com seu valor exposto de forma legível no código fonte da página. A CheckVulnTest considerará o ataque caso seja possível submeter o formulário.linebreak (3).[AegisWT] A CheckVulnTest acessará uma página que possui um método que será disparado em seu carregamento, e do qual renderizará, através de DOM, um elemento <*img*> com um atribuído SRC apontando para uma requisição forjada. A CheckVulnTest interpretará o ataque concluído caso a imagem seja carregada.	(M1).CSRF através de <*img*> (M2).ByPass em Token (M3).Forjando requisição através de DOM	(1).CSRF Finder. (2).No-Script. (3).RequestPolicy. (4).Websecurify.
[Q3] Quão eficaz é a proteção das ferramentas diante dos ataques A3?	(1).[WebGoat] A CheckVulnTest acessará uma página na qual injetará o comando <*script*>alert('XSS');</*script*> no campo do formulário e irá submetê-lo. Posteriormente a CheckVulnTest irá visualizar o resultado, esperando uma janela de alerta, indicando que o ataque foi realizado com sucesso. (2).[WebGoat] A CheckVulnTest acessará uma página na qual injetará o comando: <*script*>alert('XSS');</*script*> no campo do formulário, que ao ser submetido fará uma busca por palavras-chave na base de dados. A CheckVulnTest irá esperar a janela de alerta na resposta, indicando sucesso no ataque. (3).[AegisWT] A CheckVulnTest acessará uma página que possui um método JS que insere uma string com valor "Olá Mundo" na página. Esta string possui o comando replace combinada a uma expressão regular maliciosa, conforme o seguinte código: "Hello World".replace(/Olá Mundo/g,alert). A CheckVulnTest irá esperar a janela de alerta quando a página for carregada. (4).[AegisWT] A CheckVulnTest acessará uma página que irá carregar um conteúdo malicioso que será interpretado apenas no carregamento da página, por meio de DOM, o qual irá reproduzir mediante document.write o seguinte código: <*script*>alert('XSS');</*script*>. A CheckVulnTest irá esperar a execução do DOM. (5).[AegisWT] A CheckVulnTest acessará uma página que irá carregar um arquivo .js através de WebWorker. O conteúdo do arquivo possui o comando document.write("<*script*>alert('XSS');</*script*>");. A CheckVulnTest irá esperar a janela de alerta quando o carregamento da página for concluído.	(M1).XSS Persistente Padrão (sem escapes) (M2).XSS Refletido Padrão (sem escapes) (M3).XSS através de escapes com expressão regular. (M4).XSS através de DOM (M5).XSS baseado em DOM através de Web Workers	(1).ImmuniWeb Self-Fuzzer (2).Netcraft Extension. (3).No-Script. (4).RightClickXSS. (5).ScriptSafe. (6).TamperMonkey. (7).Websecurify. (8).XSS chef. (9).XSS Me.
[Q4] Quão eficaz é a proteção das ferramentas diante dos ataques A4?	(1).[WebGoat] A CheckVulnTest acessará uma página na qual possui exposição dos dados do identificador da sessão. Esse identificador não possui uma data definida de expiração, indicando que a mesma estará sempre armazenada no navegador. A CheckVulnTest irá obter o identificador deste cookie, comprovando o acesso à página. (2).[WebGoat] A CheckVulnTest irá acessar uma página que contém um formulário com dados sensíveis sem criptografia, utilizando codificações fracas como Base64. A CheckVulnTest irá aguardar a requisição da página. (3).[AegisWT] A CheckVulnTest irá acessar uma página na qual realiza autenticação por parâmetro Authorization do tipo Basic. O objetivo da CheckVulnTest será conseguir logon mesmo que o protocolo não seja HTTPS. (4).[AegisWT] A CheckVulnTest irá acessar uma página na qual realizará um logon em um formulário. Em seguida a CheckVulnTest irá fechar o navegador com a sessão ativa. Ao final do teste, se a instância do navegador estiver ativa, significa que o processo foi realizado de forma insegura. (5).[AegisWT] A CheckVulnTest irá acessar uma página na qual possui um formulário para autenticação. Os dados permitem armazenar as informações imputadas, além de por meio de padrão traz marcada a opção para lembrar a autenticação. Uma vez submetido o formulário de autenticação com os comportamentos inseguros, a CheckVulnTest irá considerar a operação insegura.	(M1).Utilização de Cookies Persistentes (M2).Armazenamento indevido baseado codificação Base64 (M3).Autenticação Basic sem HTTPS (M4).Fechamento do navegador com sessão aberta (M5).Campos com autocompletar habilitados e lembretes de Logon	(1).Netcraft Extension. (2).No-Script. (3).RequestPolicy. (4).ScriptSafe. (5).TamperMonkey. (6).Toogle Cookies. (7).Websecurify.

Tabela 5. Questões e Métricas para os ataques A5 até A9.

Questões	Objetivos (Goals)	Métricas	Extensões
[Q5] Quão eficaz é a proteção das ferramentas diante dos ataques A5?	(1).[AegisWT] A CheckVulnTest irá acessar uma página que armazena dados sensíveis através do LocalStorage. O objetivo da CheckVulnTest será recuperar um parâmetro que armazena o ID da sessão do usuário, indicando que o recurso, apesar de inseguro, ainda se encontra disponível. (2).[AegisWT A CheckVulnTest irá acessar uma página que armazena dados sensíveis através de IndexDB. O objetivo da CheckVulnTest será recuperar um parâmetro que armazena o ID da sessão do usuário, indicando que o recurso, apesar do comportamento inseguro, ainda se encontra disponível	(1).Armazenamento indevido através do LocalStorage (2).Armazenamento indevido através do IndexDB	(1).No-Script. (2).RequestPolicy. (3).ScriptSafe. (4).TamperMonkey. (5).Websecurify.
[Q6] Quão eficaz é a proteção das ferramentas diante dos ataques A6?	(1).[WebGoat] A CheckVulnTest acessará uma página que possui dois links, um que aponta para o mesmo domínio e outro que fará uma requisição em um domínio externo da aplicação e que o mesmo não consta nas permissões de domínios da PMO da aplicação. A CheckVulnTest irá clicar no primeiro e no segundo link, aguardando logo em seguida a presença de um elemento DIV com ID="message". Indicando que a falha foi explorada com sucesso. (2).[AegisWT] A CheckVulnTest acessará uma página que possui um método que será disparado um consumo via AJAX em um serviço externo. A resposta deste serviço contém um ataque de XSS que executa um alert através de codificação URIComponent. A CheckVulnTest irá aguardar como resposta a mensagem de alerta, indicando que o ataque foi bem-sucedido. (3).[AegisWT] A CheckVulnTest acessará uma página que possui um arquivo SWF. O arquivo crossdomain.xml está definido para permitir acesso de qualquer domínio externo. A CheckVulnTest irá aguardar o carregamento da página, indicando que a falha de segurança foi desprezada pela extensão. (4).[AegisWT] A CheckVulnTest acessará uma página que possui uma conexão via protocolo WebSocket. O "handshake" entre cliente e servidor possui falhas de PMO. A CheckVulnTest interpretará o ataque como bem-sucedido caso consiga recuperar a informação no retorno do WebSocket. (5).[AegisWT] A CheckVulnTest acessará uma página que possui uma troca de mensagens via Web Messaging. O controle de segurança para restringir domínios possui falhas. A CheckVulnTest interpretará o ataque como bem-sucedido caso consiga recuperar a mensagem.	(M1).Proteção de PMO (M2).Aplicando XSSI via AJAX (M3).Checagem do arquivo crossdomain.xml (M4).Falhas de PMO através de WebSocket (M5).XSS através de Web Messaging	(1).RequestPolicy. (2).Websecurify.
[Q7] Quão eficaz é a proteção das ferramentas diante dos ataques A7?	(1).[WebGoat] A CheckVulnTest acessará uma página que possui vulnerabilidade SQLi através de concatenação de parâmetros numéricos. Ela irá manipular os parâmetros do POST, aplicando uma verdade absoluta na sintaxe do SQL. A aplicação irá aguardar o carregamento da página, indicando que o acesso à página foi permitido. (2).[WebGoat] A CheckVulnTest acessará uma página na qual possui vulnerabilidade SQLi através de concatenação de parâmetros texto. Ela irá manipular os parâmetros do POST, aplicando uma verdade absoluta na sintaxe do SQL. A Aplicação irá aguardar o carregamento da página, indicando que o acesso à mesma foi permitido. (3).[AegisWT] A CheckVulnTest acessará uma página que possui parâmetros inseguros na utilização do WebSQL. Ela irá aguardar o carregamento da página, indicando que o acesso a ela foi permitido.	(M1).Vulnerabilidade via GET (M2).Vulnerabilidade via POST (M3).Vulnerabilidade em Web SQL	(1).ImmuniWeb Self-Fuzzer (2).SQL Inject Me (3).Websecurify.
[Q8] Quão eficaz é a proteção das ferramentas diante dos ataques A8?	(1).[AegisWT] A CheckVulnTest acessará uma página na qual disponibiliza um link para download, o arquivo em questão trata-se de um Malware e a CheckVulnTest aguardará a sua conclusão. (2).[AegisWT] A CheckVulnTest acessará uma página que está programada para receber a adição de um elemento $<a>$ que direciona para um Phishing. A injeção deste elemento será realizada por uma extensão instalada no navegador. A avaliação do resultado é baseada no estado do navegador, na qual será verificado se a extensão maliciosa está ativa.	(M1).Download de Malware (Plug-in) (M2).Detecção de Phishing em Complemento (Extensão)	(1).avast! Online Security (2).Safe Preview (3).VTchromizer (4).VTzilla (5).Web of Trust (WOT)
[Q9] Quão eficaz é a proteção das ferramentas diante dos ataques A9?	(1).[AegisWT] A CheckVulnTest acessará uma página que possui um elemento $<a>$ que direciona para um Phishing. Após o carregamento da página, a CheckVulnTest irá acionar o link e espera direcionar a página para a URL, indicando que o elemento não foi bloqueado pela extensão. (2).[AegisWT] A CheckVulnTest acessará uma página na qual possui um link com uma URL curta que redireciona para um Phishing. Após 10 segundos do carregamento da página, o próximo passo da CheckVulnTest será conseguir clicar no link mencionado, encontrando o link ainda funcional.	(M1).Redirecionamento para Phishing (M2).Checagem de URL curtas	(1).avast! Online Security (2).Anti-Phishing (3).Netcraft Extension (4).RequestPolicy (5).Safe Preview (6).ScriptSafe (7).TamperMonkey (8).Web of Trust (WOT)

Tabela 6. Questões e Métricas para os ataques A10 até A13.

Questões	Objetivos (Goals)	Métricas	Extensões
[Q10] Quão eficaz é a proteção das ferramentas diante dos ataques A10?	(1).[AegisWT] A CheckVulnTest acessará uma página na qual após 5 segundos irá atribuir o location.href com uma URL que possui um parâmetro que contem outra URL que pertence a outro domínio. O resultado final será observar se o redirecionamento foi executado com sucesso.	(M1).Redirecionamentos inválidos em URL	N/D
[Q11] Quão eficaz é a proteção das ferramentas diante dos ataques A11?	(1).[AegisWT] A CheckVulnTest irá checar a versão do serviço acessado e analisará se consta em blacklist de versões com vulnerabilidades conhecidas.	(M1).Analise de Zero-Day em serviços ou aplicações disponíveis na web.	N/D
[Q12] Quão eficaz é a proteção das ferramentas diante dos ataques A12?	(1).[AegisWT] A CheckVulnTest irá checar a versão do navegador e analisará se consta em uma blacklist de versões com vulnerabilidades conhecidas.	(M1).Versão do Navegador com Vulnerabilidades conhecidas.	N/D
[Q13] Quão eficaz é a proteção das ferramentas diante dos ataques A13?	(1).[AegisWT] A CheckVulnTest irá acessar uma página a qual possui um link para instalar um plug-in com vulnerabilidade conhecida. O resultado final será conseguir instalar o referido plug-in no navegador. (2).[AegisWT] Primeiramente será instalada no navegador uma extensão com uma versão com vulnerabilidade. Posteriormente a CheckVulnTest acessará uma página. O resultado final será observar se a extensão foi removida.	(M1).Plug-in instalado com vulnerabilidade conhecida (M2).Extensão instalada com vulnerabilidade conhecida	N/D

Tabela 7. Listagem das 25 Extensões submetidas à avaliação

Extensão	Navegador	Versão	Extensão	Navegador	Versão
avast! Online Security	Chrome	10.0.2208	Anti-Phishing	Firefox	1.0.1
Clickjacking Reveal	Firefox	1.1.1	Clickjacking Test	Chrome	1.0
CSRF Finder	Firefox	1.2.1	ImmuniWeb Self-Fuzzer	Firefox	0.9.3.1
Netcraft Extension	Chrome	1.6.1	Netcraft Extension	Firefox	1.10.1.1
No-Script Security Suite	Firefox	2.6.9.10.1	RequestPolicy	Firefox	0.5.28.1
RightClickXSS	Firefox	0.2.1.1	Safe Preview	Chrome	2.6.0
SQL Inject Me	Firefox	0.4.7.1	TamperMonkey	Chrome	3.9.202
Web of Trust (WOT)	Firefox	20131118	Websecurify	Chrome	5.0.1
XSS chef	Chrome	1.0	XSS Me	Firefox	0.4.6
Zscaler Likejacking Prevention	Chrome	1.1.9	VTchromizer	Chrome	1.2
VTzilla	Firefox	1.2			

4. Trabalhos Relacionados

Nesta seção serão descritos alguns trabalhos disponíveis na literatura que apresentam objetivos correlatos ao nosso estudo.

Silic et al [Silic et al. 2010] trazem uma abordagem bastante correlata ao nosso estudo, onde destacam problemas acerca dos usuários da internet que tem o navegador Web como ponto de partida para consumir informações. Eles apresentam vulnerabilidades especificas neste cenário, entretanto, após quatro anos muito do comportamento mudou, algumas dessas vulnerabilidades foram minimizadas, já outras permanecem persistentes e com novas técnicas de execução. Nosso estudo visa apresentar uma ótica atual sobre a problemática e realizar uma avaliação do estado de arte atual considerando as principais soluções disponíveis aos usuários de navegadores.

Akhawe & Felt [Akhawe and Felt 2013] apresentam uma avaliação empírica sobre a ineficiência dos alertas de seguranças que os navegadores web notificam aos seus usuários. A pesquisa é direcionada aos avisos de Malwares e Phishing que são ou que deveriam ser exibidos durante a navegação, trazendo como conclusão a importância de

Tabela 8. Resultados e Considerações das Questões Q1 até Q10.

Questões	Resultados e considerações
Q1	Nos dados extraídos quanto ao A1, conforme ilustrado na Figura 3, a maioria das soluções apresentaram um comportamento abaixo das expectativas, dentre elas, uma apresentou um comportamento nulo. Em contrapartida, a ferramenta No-Script além de detectar sobreposição para furto de cliques, também obteve controle dos cabeçalhos HTTP referente a permissões de frames. Contudo, nenhuma das ferramentas cobriu a sobreposição de campos.
Q2	Conforme descrito na Figura 3, a ferramenta No-Script apresentara um comportamento satisfatório nos testes que envolvem ataques de A2. As demais ferramentas tiveram ineficiência quanto a detecção da métrica 2, a qual é baseada na forja de token. Para os usuários do Chrome não há muitas opções, além do fato da única não atender com plenitude.
Q3	Conforme observado na Figura 3, os resultados obtidos de A3 demonstram que apenas a ferramenta No-Script teve um resultado satisfatório quanto aos ataques de XSS, porém essa solução encontra-se disponível apenas no navegador Firefox. Quanto ao Chrome, nenhuma das ferramentas disponíveis apresentou êxito nos testes, já que os melhores resultados foram obtidos através da Websecurify e a XSS Chef, e ambas falharam em uma das métricas. Outra questão importante é que no geral, apenas uma ferramenta teve sua funcionalidade classificada como bem abaixo das expectativas. Contudo, nenhuma delas apresentou um comportamento nulo em relação à proteção de XSS. Estes índices, além de consolidar-se como a avaliação com maior número de ferramentas disponíveis, demonstram que as opções de proteção contra XSS apresentam um maior amadurecimento em comparação com os demais ataques.
Q4	Conforme descrito na Figura 3, todas as ferramentas submetidas tiveram alguma ação preventiva ao ataque A4. O aspecto mais explorado foi a prevenção da quebra da confidencialidade de Cookies com informações sensíveis. Contudo, conforme descrito na Figura 3, no geral estas ferramentas obtiveram uma baixa ação preventiva nos demais aspectos. Apenas uma ferramenta observou o armazenamento inseguro da sessão da aplicação, porém, foi ineficiente em demais ameaças como a ausência de HTTP. Esta questão analisa um considerável número de métricas quanto às observações descritas neste estudo, além de ser a terceira avaliação com maior número de extensões.
Q5	Em relação ao A5, conforme ilustrado na Figura 3, a TamperMonkey se destacou por ser a única ferramenta a observar o armazenamento inseguro de LocalStorage do HTML5. Contudo, ela não pode ser considerada satisfatória por ter falhado na outra métrica. O caso mais grave é que as demais ferramentas tiveram um comportamento nulo.
Q6	Com base nos resultados da avaliação de A6, conforme ilustrado na Figura 3, identificamos apenas 2 opções. Além disso, nenhuma considerou certos critérios abordados neste estudo, a saber: a segurança em arquivos SWF e as falhas ocasionais em recursos do HTML5 como o WebSocket e o WebMessaging.
Q7	Seguindo para a análise dos resultados de A7, conforme ilustrado na Figura 3, os dados obtidos vão em contradição aos de XSS, tanto em respeito à quantidade de ferramentas disponíveis, quanto à eficácia. O fato é que as técnicas de detecção, ao menos no contexto de extensões, ainda apresentam muitas limitações. Primeiramente pela heurística aplicada por todas as ferramentas, as quais realizam uma inspeção baseada em um nível mais superficial, observando comportamentos como erro da aplicação ou mensagens ao usuário quando os parâmetros são manipulados. A primeira métrica possui um comportamento de simples detecção, ainda sim, uma ferramenta teve seu comportamento nulo. Outra conclusão importante é que nenhuma das ferramentas considerou possíveis vulnerabilidades de WebSQL.
Q8	Em relação aos resultados obtidos nos testes do ataque A8, com exceção a SafePreview, que apresentou seu comportamento nulo, as demais ferramentas conseguiram detectar o download de um Malware, conforme descrito na Figura 3. Porém, nenhuma pode ser considerada satisfatória devido à incapacidade de realizar a prevenção contra uma extensão maliciosa já instalada ou prestes a ser instalada no navegador. Alguns serviços como e-Banking, estabelecem módulos de proteção que visam minimizar ataques por malwares, tais módulos nada mais são que extensões para o navegador que tem o intuito de monitorar tráfego ou armazenamento no front-end.
Q9	Considerado o segundo com maior número de ferramentas submetidas, a avaliação de A9 identificou que nenhuma ferramenta fora capaz de detectar um Phishing em uma URL curta, definido como a segunda métrica desta avaliação, conforme ilustrado na Figura 3. Tal fato possibilita que o usuário clique em um link, e com isso acaba jogando a responsabilidade de prevenção do Phishing para o navegador. Dessas ferramentas, a única prevenção apresentada foi referente ao ataque de forma mais básica, descrito na primeira métrica. Outra observação é o fato de duas ferramentas terem apresentado um comportamento nulo por serem baseadas em blackist com considerável tempo de depreciação. A depreciação foi confirmada através de testes que utilizavam links catalogados no PhishTank ou Reasonable Anti-Phishing, que apesar de já confirmados[a], os links maliciosos só foram detectados por algumas extensões após um período maior do que 20 horas. O mecanismo nativo do Chrome e Firefox, o SafeBrowsing[b], apresentou um tempo de resposta ainda maior em comparação ao resultado das Extensões. Podemos considerar 20 horas como um tempo mais que suficiente para um mal-intencionado coletar informações sensíveis de diversos usuários. Outro comportamento que comprova este fato é a extensão Netcraft, que faz uso da base de links do PhishTank, ter detectado com com êxito todos os links fraudulentos. Contudo, nenhuma ferramenta foi satisfatória já que todas falharam na segunda métrica.
Q10	Também não foi possível detectar ferramentas com o propósito de solucionar a questão relacionada a A10. Constatamos que uma prática simples seria analisar os redirecionamentos comparando o domínio da aplicação com o domínio a ser direcionado para o usuário. Contudo, é importante reconhecer que o contexto de gerenciar os redirecionamentos do usuário é de responsabilidade da aplicação. Casos desta natureza para outros domínios não é algo incomum, mas precisam ser previsíveis e gerenciados, logo a utilização de Whitelist merece ser considerada. Um fato curioso observado durante os testes realizados neste estudo[c] é que em serviços como goo.gl, bitly.com, e o mcaf.ee foram bloqueadas as tentativas de criar uma URL curta que redirecionasse para uma URL catalogada no PhishTank. Em contrapartida, serviços como Ow.ly, Tiny.cc e o Bit.do, não realizaram esta checagem, possibilitando a geração de uma URL curtas maliciosas. Outra questão foi que as URL encurtadas geradas nos serviços mencionados, uma vez inseridas no PhishTank, o serviço anti-phishing não faz uma análise considerando o destino da URL original, portanto, erroneamente não detecta que a URL curta em questão é maliciosa.

[a] Links registrados na API com status "VALID PHISH": http://www.phishtank.com/developer_info.php

[b] SafeBrowsing API: https://developers.google.com/safe-browsing/

[c] Avaliação iniciada em Fevereiro de 2014 e concluída em Abril de 2014

Tabela 9. Resultados e Considerações das Questões Q11 até Q13.

Questões	Resultados e considerações
Q11	Seguindo a mesma linha do A10, para o A11 também não foram encontradas ferramentas disponíveis. Um fato curioso é que identificamos um meio relativamente simples de controlar de forma automatizada um plug-in instalado no navegador. Um exemplo deste meio é através do Chrome, que disponibiliza uma listagem de seus plug-ins instalados por meio do endereço chrome://plug-ins. Constatamos que se faz possível desenvolver uma extensão que poderia desabilitar um determinado plug-in da listagem, fazendo uso de uma blacklist, de forma automática. Em contrapartida, o mesmo comportamento não poderia ser reproduzido quanto a listagem about:plugins do Firefox.
Q12	Em relação aos ataques A12, nenhuma ferramenta foi encontrada para uma possível avaliação. Isso também reflete na pequena quantidade de publicações disponíveis na literatura sobre esse ataque. Com esse resultado identificamos uma lacuna muito preocupante em meio às ferramentas disponíveis na atualidade. Nós observamos que uma forma de prevenção desta lacuna seria um controle via blacklist de serviços com versões vulneráveis no intuito de alertar o usuário de um ambiente temerário.
Q13	Em relação aos ataques A13, não foi encontrada nenhuma ferramenta para uma possível avaliação. Isso também reflete na pequena quantidade de publicações disponíveis na literatura sobre esse ataque. Com esse resultado identificamos uma lacuna muito preocupante em meio às ferramentas disponíveis na atualidade. Uma alternativa bastante simples seria a extensão observar a versão do navegador, e alertar ao usuário para a devida atualização. É importante salientar que no caso do Chrome a atualização é automática, contudo o Firefox apenas notifica ao usuário sobre uma nova atualização, sem apresentar nenhum mecanismo que impeça a continuação da navegação em uma versão desatualizada com falhas.

uma eficiente política de alerta destes incidentes, além de evidenciar um significativo impacto destes alertas ao comportamento do usuário final. Nossa avaliação é direcionada ao mesmo ambiente e público alvo, porém, também é focada nas ameaças oriundas de vulnerabilidades não intencionais, a exemplo das falhas de desenvolvimento que possam refletir e comprometer os dados sensíveis do usuário de navegador web.

Monteverde & Campiolo [Monteverde and Campiolo 2014] apresentam um estudo e análise de vulnerabilidades em diversos tipos de aplicações Web no cenário brasileiro. Os autores fazem uso de Web Scanners, que se definem como mecanismos que automatizam o processo de busca por vulnerabilidades em aplicações. Aliado ao processo, também foram utilizadas ferramentas para interceptar as requisições solicitadas nas aplicações avaliadas. A proposta evidencia questões de segurança que são centradas especificamente as aplicações Web, e não faz menção aos aspectos do navegador Web ou Engenharia Social. Nossa avaliação enfatiza o uso do navegador web sob a ótica dos três aspectos da tríada, considerados como os principais vetores para propagação de violações de dados sensíveis nestes cenários.

5. Conclusão e Trabalhos Futuros

Este estudo teve o objetivo de apresentar uma avaliação de ameaças ao usuário de navegador web. Nesta avaliação foram consideradas 25 Extensões, dentre as ferramentas selecionadas, a No-Script, para Firefox, e Websecurify, para Chrome, tiverem bons resultados em alguns aspectos. Em contrapartida, considerando todos os testes submetidos e as prevenções pretendidas no contexto geral, foi observado que nenhuma das ferramentas avaliadas pode ser considerada como satisfatória.

Outro ponto importante foi evidenciarmos que existe uma nítida carência de cobertura para os ataques relacionados aos aspectos de HTML5. Concluindo assim que no estado atual das Extensões disponíveis, não existe uma proteção totalmente satisfatória, quer seja isoladamente ou em conjunto com outras extensões. Contudo, alguns ataques apresentam certo nível de amadurecimento, com um expressivo número de Extensões relacionadas, conforme pode ser observado nos ataques A3, A4 e A9.

Em contrapartida, existem grandes lacunas em diversos cenários. Os diagnósticos preocupantes desta avaliação cabem aos resultados dos ataques A1, A5, A6 e A7, visto

que nestes casos quando alguma ferramenta era identificada, seu propósito de proteção apresentava um comportamento quase nulo. E casos ainda mais alarmantes, como A10, A11, A12 e A13, houve ausência de opções como proposta de proteção. Apesar da prática de extensões como proteção ao usuário ser um tanto questionável, o estudo identificou essa causa bem apoiada pelos mantenedores dos navegadores.

E por fim, o estudo evidenciou que são poucas as opções de ambientes controlados disponíveis para pesquisadores. E as existentes não receberam atualizações frequentes nos últimos anos. Diante disso, surgiu a necessidade de desenvolver um novo ambiente que proporcionou a reprodução de todos os ataques emergentes identificados pelo nosso estudo, intitulado AegisWT. Adicionalmente, desejamos submeter este ambiente ao projeto OBWP, para possibilitar que outros pesquisadores possam realizar estudos e avaliações similares. Como trabalho futuro, temos o intuito de realizar avaliações automatizadas de scanners de vulnerabilidades no AegisWT, possibilitando analisar comportamentos em profundidade das ferramentas disponíveis.

Referências

Akhawe, D. and Felt, A. P. (2013). Alice in warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 257–272, Berkeley, CA, USA. USENIX Association.

Allen, J. H. (2001). *The CERT guide to system and network security practices Series in software engineering*. Addison-Wesley Professional; 1 edition.

DeRyck, P., Desmet, L., Joosen, W., Muhlberg, J., Piessens, F., Johns, M., Lekies, S., Davies, E., Farrell, S., Bos, B., and Roessler, T. (2013). Web-platform security guide: Security assessment of the web ecosystem. Technical report.

Gaurav Aggrawal, Elie Bursztein, C. J. and Boneh, D. (2010). An analysis of private browsing modes in modern browsers. *In Proc. of Usenix Security*.

HTML5Security (2013). Html5security cheatsheet.

Leitner, A., Ciupa, I., Oriol, M., Meyer, B., and Fiva, A. (2007). Contract driven development = tdd - writing test cases. In *ESEC/SIGSOFT FSE*, pages 425–434. ACM.

MITRE (2011). Cwe/sans top 25 most dangerous software errors. *Disponível em: http://cwe.mitre.org/top25/.*

Monteverde and Campiolo (2014). Estudo e análise de vulnerabilidades web. *VIII Workshop de Trabalhos de Iniciação Científica e de Graduação, 2014, Belo Horizonte. XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

OWASP (2013). Top ten 2013. *Disponível em: https://goo.gl/VKz94B*.

Secunia. Secunia vulnerability review 2015. *Disponível em: http://goo.gl/ZrdrRh*.

Silic, M., Krolo, J., and Delac, G. (2010). Security vulnerabilities in modern web browser architecture. *MIPRO, 2010 Proceedings of the 33rd International Convention*, pages 1240–1245.

V. Basili, C. C. and Rombach, H. D. (1994). Goal question metric paradigm. *Encyclopedia of Software Engineering, pp. 528-532.*