Congresso de Segurança da Informação das Fatec

VULNERABILIDADES EM APLICAÇÕES WEB E SUA RELAÇÃO COM O FATOR HUMANO

VULNERABILITIES ON WEB APPLICATIONS AND ITS RELATIONSHIP WITH THE HUMAN FACTOR

Luiggi Torricelli, FATEC Americana, luiggi.torricelli@fatec.sp.gov.br
Ricardo Hiro Nishizaki, FATEC Americana, ricardo.nishizaki@fatec.sp.gov.br
Orientador: Maxwel Vitorino da Silva, FATEC Americana, maxwel.silva5@fatec.sp.gov.br

Resumo

Nesse trabalho apresenta-se uma análise sobre as dez maiores vulnerabilidades *web*, conforme a OWASP (*Open Web Application Security Project*), visando inter-relacionar a segurança da informação, fator humano e vulnerabilidades a sistemas web. Para isso, foram utilizados artigos científicos e dados provenientes do próprio site da OWASP e do CERT (Centro de Estudos, Respostas e Tratamentos de incidentes de Segurança do Brasil). Foi constatado que os tópicos enfatizados pelo trabalho têm grande relação, uma vez que a quantidade de vulnerabilidades é inversamente proporcional ao nível de segurança da informação em um sistema, ou seja, menor são as possíveis vulnerabilidades quando o sistema é mais seguro. O fator humano continua sendo um vetor crítico para exploração de vulnerabilidades.

Palavras-chave: Segurança da informação, vulnerabilidades, fator humano.

Abstract

This paper presents an analysis around the ten biggest web vulnerabilities, based on the OWASP (Open Web Application Security Project), with the objective of interrelating information security, human factor and web systems' vulnerabilities. In order to do that, scientific articles and data coming from both the OWASP own website and from CERT (Centro de Estudos, Respostas e Tratamentos de incidentes de Segurança do Brasil) were used. It was found that the topics which were emphasized by the paper have a considerable connection, since the number of vulnerabilities is inversely proportional to the level of information security on a system, meaning that less possible vulnerabilities exist when the system is safer. The human factor is still a critical vector for vulnerability exploration.

Keywords*: Information security, vulnerabilities, human factor.*

1. Introdução

Esse artigo baseia-se nos conceitos da segurança da informação, que requer um entendimento aprofundado devido à sua complexidade (STALLINGS, 2015), junto ao fator humano, uma vez que, segundo Glaspie e Karwowski (2017), os humanos têm participação fundamental no sucesso da segurança de uma organização, e dessa forma realizou-se uma análise referente as vulnerabilidades contidas no *OWASP Top Ten*. A OWASP é uma entidade sem fins lucrativos e reconhecida internacionalmente que atua no campo da segurança de aplicações *web*, e seu *Top Ten* é um *ranking* das dez vulnerabilidades *web* mais exploradas (OWASP, 2021).

Sendo assim, com base no método analítico de pesquisa, no qual foram lidos diversos materiais, como artigos, publicações, livros etc., coletando informações oriundas deles, serão investigadas cada uma dessas dez vulnerabilidades, considerando como podem ser exploradas e de qual maneira é feita a prevenção, utilizando os pilares da segurança da informação para ilustrar como elas afetam as aplicações *web*, e a presença do fator humano como um ponto em comum em todas elas.

2. Referencial Teórico

2.1. Segurança da Informação

É fato que a tecnologia evoluiu de uma forma na qual não é possível ter o controle absoluto de tudo que é produzido. Por um lado, isso é algo positivo, pois dá ao usuário a liberdade de navegar, porém, também é necessário se atentar aos perigos atuais. Para Moraes (2015), segurança é uma das necessidades básicas da nossa vida, assim como educação, saúde etc., porém, no nosso cotidiano, não é algo que se é pensado a todo o momento. Na *internet*, isso funciona da mesma maneira e é necessária uma total atenção, pois o que está em risco nesse caso são suas informações.

Devido a isso, foi desenvolvido a área de Segurança da Informação, para que a informação seja protegida de agentes externos. O *National Institute of Standards and Technology* (NIST) (1995) se refere ao termo *Segurança de computadores* como:

> "[...] Segurança de computadores: A proteção oferecida para um Sistema de informação automatizado a fim de alcançar os objetivos de preservar a integridade, a disponibilidade e a confidencialidade dos recursos do Sistema de informação (incluindo *hardware*, *software*, *firmware*, informações/dados e telecomunicações".

Estes conceitos foram chamados de *tríade CIA* (do acrônimo em inglês *confidentiality*, *integrity*, *availability*), porém, com o tempo, foi necessário a adição de novos conceitos como a

autenticidade, não-repúdio e a responsabilização. De acordo com Stallings (2015, p.7), definem-se os conceitos: *confidencialidade*, que se trata da proteção da privacidade de dados do usuário fornecendo a eles meios de proteger suas informações; *integridade*, que se estabelece contra a modificação ou exclusão de informações de um determinado arquivo; *disponibilidade*, que assegura que as informações e os serviços estejam sempre disponíveis para usuário em seu momento de uso; *autenticidade*, que visa a confiança na validação de uma transmissão de dados; *responsabilização*, que se trata de atribuir a responsabilidade para uma pessoa pelas ações de entidade no caso de uma violação de segurança.

De acordo com Moraes (2015, p. 22), é descrita a existência de um pilar chamado de *não-repúdio*, exercendo um papel importante utilizando métodos de identificação para que, posteriormente, o usuário não possa negar a realização de qualquer ação executada.

2.2. Vulnerabilidades Web – OWASP Top Ten

Dentre as existentes vulnerabilidades atualmente, observam-se determinadas áreas de efeito: redes, aplicações, *web* etc. De acordo com o CERT (Centro de Estudos, Respostas e Tratamentos de incidentes de Segurança do Brasil), cerca de 665.079 incidentes foram relatados no ano de 2020 e, dentre eles, 3,99% são direcionados para a *web*, como visto nas Figuras 1 e 2:

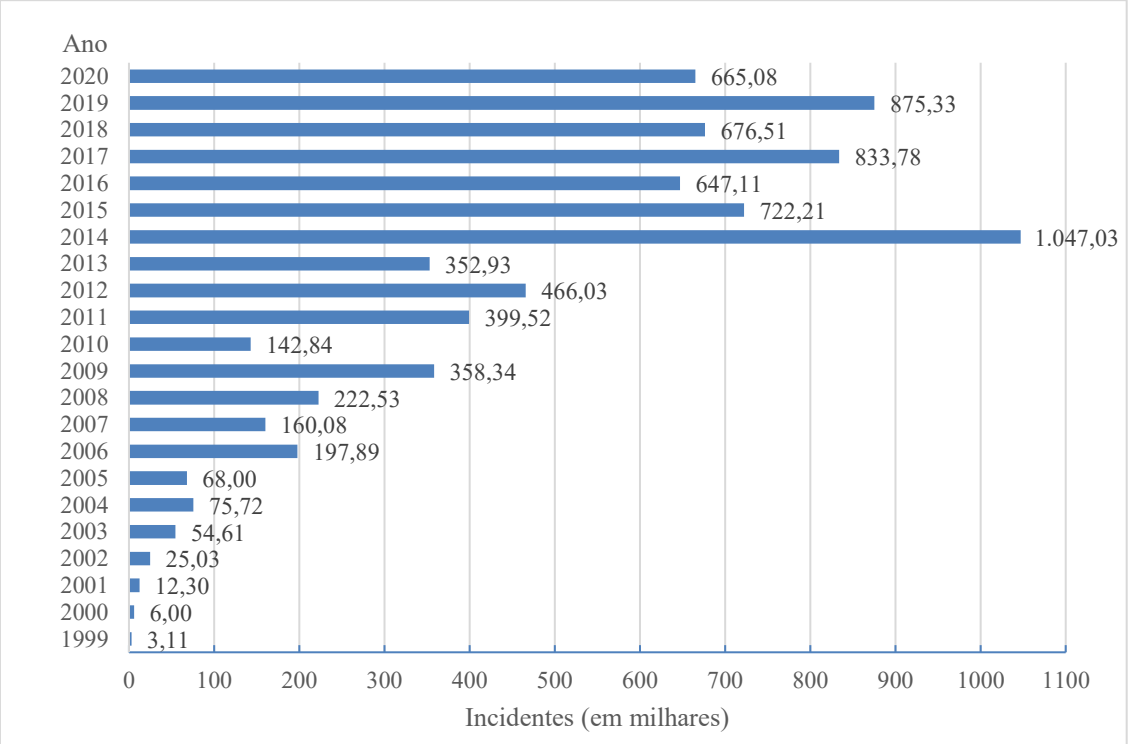

Figura 1 – Total de incidentes reportados ao Cert.Br por ano (CERT.br 2020).

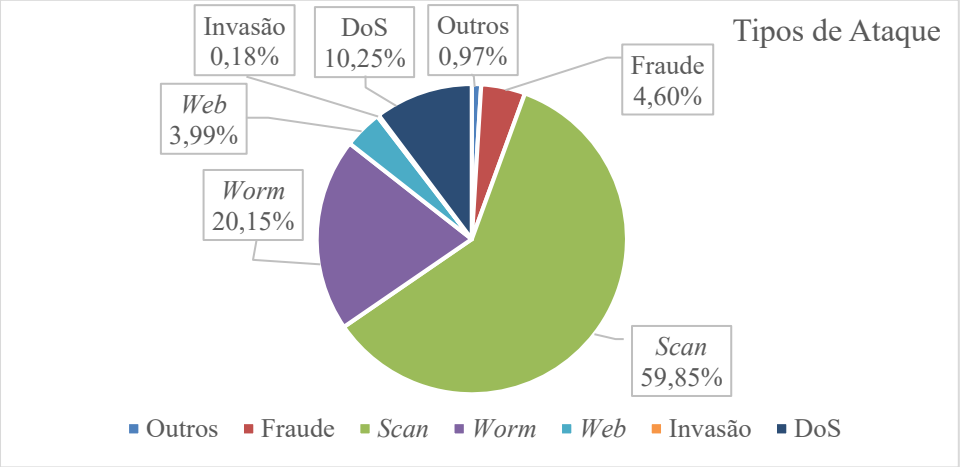

Figura 2 – Incidentes reportados ao Cert.Br – janeiro a dezembro 2020 (CERT.br 2020).

Há uma organização que atua no fortalecimento da segurança das aplicações *web*, sem fins lucrativos e com reconhecimento internacional chamada OWASP.

Formada por especialistas em segurança, ela atua realizando o levantamento das 10 principais vulnerabilidades mais exploradas a cada 4 anos, as quais serão discorridas mitigando possíveis vulnerabilidades, como ilustra o Quadro 1.

Quadro 1 – OWASP - 10 maiores riscos de segurança em aplicações web

Identificador	Nome
A01:2021	Controle de Acesso Falho
A02:2021	Falhas Criptográficas
A03:2021	Injeção
A04:2021	Design Inseguro
A05:2021	Configuração Incorreta de Segurança
A06:2021	Componentes Desatualizados e Vulneráveis
A07:2021	Falhas de Identificação e Autenticação
A08:2021	Falhas de Software e Integridade de Dados
A09:2021	Falhas de Monitoração e Registro de Segurança
A10:2021	Falsificação de Solicitação do lado do Servidor (SSRF)

Fonte: OWASP (2021)

2.2.1. Controle de acesso falho (A01:2021)

A quebra de controle de acesso acontece quando controles de acesso são implementados de forma inadequada, e isso pode acontecer frequentemente em APIs, podendo causar vazamento de informações, impactando a confidencialidade; modificação ou exclusão dos dados, impactando a integridade; e escalamento de privilégios, o qual impactaria todos os pilares da segurança da

informação.

Muitas vezes, o programador pode não haver implementado decoradores no código que apontem a necessidade do *login*. Segundo Harrison (2013), um decorador é um padrão no qual permite a inclusão de um comportamento a um objeto. De acordo com Freeman (2018), outro ponto importante é a implementação de guardas para criar rotas seguras para os usuários, controlando o acesso a essas rotas e respondendo com redirecionamentos ou cancelamento da navegação.

São maneiras de prevenção: implementar à aplicação o princípio de privilégio mínimo ou negação por padrão, verificar a alteração de URL forçada, verificar as permissões de visualização ou alteração de contas de outros usuários, não utilizar referências diretas não seguras a objetos, aplicar controle em APIs para métodos *POST*, *PUT*, *DELETE*, dificultar a adulteração de *tokens*, verificar configurações CORS.

2.2.2. Falhas de criptografia (A02:2021)

As falhas criptográficas acontecem quando os dados são violados, estando eles no servidor do cliente (trânsito) ou no banco de dados ou sistema de arquivos (repouso). O impacto da quebra da criptografia afeta diretamente a confidencialidade dos dados, pois, desta forma, o atacante terá acesso a arquivos encriptados.

O tráfego e armazenamento de dados em texto plano é uma das possíveis causas de falhas na criptografia, assim como o uso de protocolos inseguros como HTTP, SMTP, FTP etc., visto que desta forma torna-se possível farejar pacotes de rede e coletar o conteúdo deles. O uso de chaves criptográficas fracas, *hashes* obsoletos, tecnologias fracas, quando se trata de criptografia, como o DES que se torna vulnerável por conta do tamanho da sua chave e a natureza do algoritmo como cita Stallings (2015), são outros pontos de atenção para essa vulnerabilidade.

Quando se fala em implementação para solução em falhas criptográficas, existem diversas formas de prevenção como: não armazenar dados sensíveis sem necessidade, utilizar criptografia nos dados em repouso e em trânsito, utilizar *hashes* fortes para o armazenamento de senhas, unido a *salts*, que, de acordo com Rosulek (2021), é a adição da aleatoriedade dentro do *hash*, assim evitando ataques de *rainbow table*, que são tabelas pré-computadas com o intuito de reverter funções *hash* criptográficas.

2.2.3. Injeção (A03:2021)

A injeção acontece quando a aplicação é enganada, injetando-se comandos ao interpretador

ao invés de dados, visando buscar informações dentro da aplicação ou até mesmo conquistar acessos privilegiados (OWASP). Além do vazamento de dados, que impacta a confidencialidade, e a modificação, que impacta a integridade, caso o atacante consiga escalar privilégio, pode impactar em todos os outros pilares da segurança da informação (OWASP).

A injeção pode acontecer de diversas maneiras, através de c*ross site scripting*, *SQL injection*, *shell injection*, entre outros... Todas essas opções seguem o conceito de injetar comandos ao interpretador ao invés de dados, como demonstrado por Fogie *et al*. (2007).

É necessário validar, filtrar ou sanitizar a aplicação, parametrizar as consultas, aplicar medidas para impedir a concatenação nas *strings* de consulta, validar a entrada de usuários a partir de uma *whitelist* ou expressões regulares, utilizar *LIMIT* e outros controles de SQL para que, caso a injeção de SQL seja bem-sucedida, seja evitado a divulgação em massa, codificando as saídas que dependem da entrada de usuário (OWASP).

2.2.4. Design inseguro (A04:2021)

Inteiramente focado nas falhas de *design* e arquitetura do *software*, o *design* inseguro é diferente das falhas de implementação, podendo ter problemas de implementação levando a vulnerabilidades, mas não podendo ser corrigido nem mesmo com uma implementação correta (OWASP, 2021). Falhas desse tipo são capazes de impactar todas os pilares da segurança da informação, devido as diferentes aplicabilidades.

Ausência de modelagem de ameaça, falta de comunicação entre a equipe de desenvolvimento e segurança. Nota-se uma ausência de uma referência no padrão do *design* (OWASP, 2021), e de acordo com Black *et al*. (2021), todo o *design* deve ser pensado antes de implementado a aplicação.

Utilizar bibliotecas seguras para operações com usuários, estabelecer um ciclo de vida para desenvolvimento seguro, utilizar modelagem de ameaça, realizar testes no *software*, adotar conceitos como *Secure by design* e *Privacy by design* (OWASP, 2021).

2.2.5. Configuração incorreta de segurança (A05:2021)

São configurações incorretas de segurança que acontecem tanto do lado do servidor como do lado do cliente. Sendo a coleta de informações sobre a aplicação uma das etapas mais importantes durante um ataque, como descreve Mukhopadhyay e Nath (2014), a exploração dessa vulnerabilidade pode ocasionar diversos impactos em todos os pilares da segurança da informação, pois se trata de uma gama muito extensa de erros.

Ocorrem em grande parte das vezes por conta da falta de *hardening* (OWASP, 2021), falha de configurações em diversas partes do aplicativo, configurações incorretas de permissões, recursos desnecessários ativados ou instalados, contas padrão com usuários comuns, excesso de informações nos erros apresentados aos usuários, muitas vezes expondo *stack traces*, falta de configurações de segurança nos servidores atualizados, *software* desatualizado e vulnerável, exposição de informações nos cabeçalhos HTTP são algumas das possíveis causas, de acordo com a OWASP (2021).

Para que configurações faltantes ou incorretas sejam prevenidas, a utilização de um processo automatizado de verificação das configurações em todos os ambientes utilizados pela aplicação é uma recomendação. Desenvolver a aplicação com uma arquitetura segmentada permite uma separação segura entre componentes, garantindo que mudanças específicas realizadas tenham menor chance de interferir no funcionamento do outro. A remoção de *softwares* e componentes não utilizados facilitará no gerenciamento de toda a configuração no ambiente, visto que serão menos arquivos que o administrador deverá ter atenção.

2.2.6. Componentes desatualizados e vulneráveis (A06:2021)

Com o crescimento do desenvolvimento de novas tecnologias, bibliotecas, pacotes de *software* e a quantidade de conhecimento cada vez mais acessível na internet, uma aplicação *web* se torna composta de diversos componentes que, com intuito ou não, podem apresentar vulnerabilidades e serem alvos de ataques.

Devido ao uso de diversos componentes numa aplicação, a falta de controle do versionamento dos mesmos abre brechas para utilização de *softwares* vulneráveis, desatualizados, tanto no lado do cliente como no do servidor. Isso inclui o sistema operacional, *middlewares* como gerenciadores de bancos de dados e servidores de transação, bibliotecas, APIs etc.

A falta de escaneamento de vulnerabilidades no ambiente também pode ser uma das principais causas, assim como deixar de aplicar correções de segurança nos componentes utilizados pela aplicação quando elas surgem, e não garantir a segurança na configuração dos componentes.

Embora seja recomendado aplicar correções e atualizar os componentes, não testar a compatibilidade da aplicação com o novo nível desses componentes pode ocasionar o aparecimento de outros problemas, sejam eles novas vulnerabilidades ou o mau funcionamento da aplicação como um todo.

Para garantir o uso somente dos componentes que a aplicação necessita, recomenda-se

remover todas as dependências que não são mais utilizadas, como arquivos e documentações antigas, recursos inutilizados, entre outros. Essa ação por si só facilita o controle de inventário e versionamento dos componentes, podendo ser realizado por ferramentas automatizadas que, uma vez executadas, são capazes de verificar fontes de dados como, por exemplo, o *Common Vulnerability and Exposures* (CVE) e *National Vulnerability Database* (NVD), identificando vulnerabilidades já documentadas.

Também é uma boa prática obter componentes somente das fontes oficiais através de *links* seguros, assim diminuindo a chance de encontrar código malicioso ou modificado dentro do componente.

Um plano de monitoração, triagem e instalação de correções é algo que toda empresa deve ter para reduzir o máximo possível os riscos de ataques relacionados a componentes vulneráveis e desatualizados.

2.2.7. Falhas de identificação e autenticação (A07:2021)

Se soluções de autenticação, gerenciamento de sessão, e identificação do usuário não foram implementadas ou foram, porém de forma incorreta, possíveis vulnerabilidades passam a existir possibilitando os agentes ameaçadores a explorá-las, comprometendo senhas e outras chaves de autenticação e sendo capazes de assumir a identidade de diferentes usuários.

Os atacantes podem utilizar técnicas como *credential stuffing*, que consiste em tentativas de acesso com uma lista de possíveis usuários e senhas com base em vazamentos de dados prévios. Em união a isso, também são possíveis os ataques de força bruta, onde dicionários de possíveis palavras e exaustivas combinações de caracteres são utilizados na tentativa de sucesso na autenticação. Ambos os ataques citados, assim como outros, podem ser feitos de maneira automatizada.

Devido ao uso de dicionários para estes ataques, permitir com que os usuários utilizem senhas de baixa complexidade e simples, como palavras conhecidas ou padrões, pode facilitar o invasor a conseguir o acesso com mais facilidade.

A falta de autenticação com múltiplos fatores, o gerenciamento incorreto dos identificadores de sessão, e formas inefetivas e fracas de procedimento de recuperação de senha também são causas para que falhas de autenticação de identificação ocorram.

Ataques automatizados de força bruta e *credential stuffing*, assim como uso de credenciais roubadas, podem ser prevenidos com a utilização de um esquema de autenticação com múltiplos

fatores, que garante que o segundo ou mais fatores de autenticação sejam necessários e não só a senha do usuário.

Isso ainda não impede que o seu sistema seja alvo dos ataques citados, então em situações em que haja diversas falhas de autenticação, sugestiona-se fazer o *logging* dessas tentativas e notificar os administradores de que algum tipo de invasão pode estar ocorrendo. Estabelecer um limite de tentativas dentro de determinado tempo também é uma opção, embora precise ser feito cautelosamente para não tornar o sistema inacessível caso o usuário não consiga se autenticar devido a outros motivos.

A implementação de um sistema que verifica a complexidade da senha utilizada é uma recomendação, visto que isso impediria o uso de senhas fracas que seriam facilmente quebradas com o uso de técnicas de ataque pelos invasores.

O NIST possui diferentes documentos com diretrizes e orientações em diversas áreas da tecnologia, incluindo o *Digital Identity Guidelines* (SP 800-63b) cujo assunto é autenticação e gerenciamento do ciclo de vida dela. Sugere-se o alinhamento dos padrões utilizados na aplicação com as indicações de tamanho, complexidade e política de rotações de senha estabelecidos nesse documento.

O gerenciador de sessões da aplicação do lado do servidor deve gerar sempre um novo identificador após o acesso do usuário, que deve ser armazenado de forma segura, não deve ser utilizado em qualquer URL, e deve ser invalidado com a saída do usuário ou após um tempo pré-determinado.

2.2.8. Falhas de software e integridade de dados (A08:2021)

Em um código ou infraestrutura na qual não há verificação de integridade de qualquer dado recebido e armazenado, brechas são abertas para introdução de código malicioso, acesso não-autorizado, entre outras violações, causando o comprometimento do sistema.

Em suma, assim como demonstra Adachi e Omote (2015), esse tipo de vulnerabilidade pode surgir quando é feito o *download* de dados de fontes não-confiáveis ou em situações em que a adulteração dos dados ocorre na própria fonte, durante a transferência, no cache, ou durante a desserialização de dados, que consiste na extração de dados, como entradas onde o usuário insere informações, e transformação dos mesmos em diferentes tipos de objetos.

Para confirmar que o *software* ou dado recebido não foi alterado ou que ele pertence a fonte

correta, sugere-se o uso de mecanismos de verificação de integridade, como assinaturas digitais, ferramentas de verificação de dependências de software para encontrar vulnerabilidades conhecidas nos componentes utilizados pela aplicação e garantir que essas dependências estão consumindo repositórios confiáveis. No recebimento de qualquer dado serializado, sanitize-o antes de desserializá-lo, assim garantindo que código malicioso não seja considerado.

2.2.9. Falhas de monitoração e registro de segurança (A09:2021)

A monitoração e registro de segurança são ações extremamente necessárias em qualquer aplicação para que seja possível a detecção mais rápida possível de brechas capazes de serem exploradas por agentes ameaçadores, tornando factível a escalabilidade e a resposta à ameaça ativa.

De forma geral, falhas nesta categoria ocorrem quando se deixa de registrar e monitorar atividades dentro do ambiente em registros de segurança, o que dificulta a detecção de brechas, como no exemplo de eventos auditáveis, como acessos permitidos, tentativas de acesso e transações críticas. Caso esse registro seja armazenado somente localmente e haja um ataque utilizando *ransomware* (que bloqueia o uso e/ou encripta os arquivos até que um resgate seja pago ao atacante), esse arquivo fica inacessível e impossibilita o acesso a mais detalhes do que ocorreu durante e antes do momento do ataque.

A monitoração de controle de acesso e erros durante a validação de dados de entrada de usuários, assim como em transações ou tabelas com dados críticos, deve ser realizada e registrada nos registros de segurança de forma clara, contextual e com a codificação correta, facilitando a identificação de atividade suspeita e possibilitando análises forenses futuras, e impedindo injeção de dados que comprometam a monitoração durante o uso de gerenciadores de registros.

Um plano de resposta a incidentes e um plano de recuperação deve ser adotado para minimizar o impacto de possíveis explorações de brechas. Como guia, o NIST possui um documento com recomendações referentes ao tratamento de incidentes em segurança da informação: *Computer Security Incident Handling Guide* (SP 800-61r2).

2.2.10. Falsificação de solicitação do lado do servidor (A10:2021)

Solicitações do lado do servidor acontecem quando o servidor faz requisições à recursos remotos e, devido a crescente modernidade das aplicações *web*, vêm se tornando cada vez mais comum, o que abre espaço para falsificação destas solicitações quando as vulnerabilidades existem, de acordo com Jabiyev *et al*. (2021).

Fazer a segmentação da funcionalidade de acesso a recursos remotos em redes diferentes reduz é uma das maneiras na camada de rede de reduzir o impacto. Outra defesa seria utilizar políticas de *firewall* que, por padrão, bloqueie toda a entrada e saída de pacotes, a não ser que o tráfego seja dentro da rede interna. Para cada comunicação aceita ou negada, registrar nos registros de segurança para futuros controles. Recomenda-se, numa escolha dessa, também estabelecer um período de verificação das regras de firewall para determinar se ainda são válidas, assim como o responsável por cada uma.

Na camada da aplicação, garantir que toda e qualquer entrada de dados vindo do cliente seja validada e sanitizada, assim como impor que o destino, porta e *schema* da URL informada esteja na lista de permissão. Como resposta aos clientes, envie somente respostas que não estejam em formato *raw*. Pode-se também desabilitar os redirecionamentos HTTP na aplicação para prevenir-se do SSRF.

2.3. Fator humano

Considerando as vulnerabilidades descritas acima e suas origens e recomendações, pode-se observar um ponto de convergência: ambas se originam de implementações falhas ou inexistentes de determinadas soluções, não detectando previamente os riscos que possíveis vulnerabilidades trariam.

De acordo com Glaspie e Karwowski (2017), funcionários que não são estimulados a entenderem os conceitos de segurança da informação podem expor a companhia a riscos de segurança. Intencionalmente ou não, desenvolvedores que carecem de conhecimento e consciência de segurança podem cometer erros capazes de afetar a segurança da aplicação e da companhia.

Em diversas situações, a negligência do fator humano em riscos à segurança da informação causa maior susceptibilidade à existência de vulnerabilidades em sistemas de informação, que podem ser exploradas tanto de forma exclusivamente tecnológica como através da engenharia social, técnica que, segundo Peltier (2006), consiste na manipulação de outras pessoas com o intuito de obter informações ou acesso a um ou mais sistemas ou dispositivos. Nas palavras de Luo *et al.* (2011, p. 1), “a gentileza natural de usuários humanos, suas fraquezas psicológicas, e suas tendências de estarem inconscientes do valor da informação que possuem” é o que causa o aumento de sucesso de ataques onde a engenharia social está presente.

3. Considerações Finais

Partindo do pressuposto de trazer informações a respeito das categorias das vulnerabilidades da OWASP *Top Ten*, o estudo realizou uma metodologia exploratória buscando conteúdos em livros e estudos do tema já realizado. Dessa forma, utilizando as considerações finais para elucidar a questão: de qual premissa partem as vulnerabilidades descritas na OWASP?

O estudo realizado demonstra que, indiferentemente da categoria que se aborda dentro da OWASP, há a necessidade de implementações de padrões já estipulados para a segurança, além da utilização de ferramentas e estratégias para a criação de uma aplicação *web* segura. Não basta apenas que o sistema cumpra a sua função. É necessário que os pilares da segurança da informação sejam resguardados e, para isso, sugestiona-se o uso de ferramentas tais como o CORS para verificação de configurações de segurança, *salt* para segurança dentro dos *hashes*, política de rotação de senha, assinaturas digitais, dentre outras citadas previamente.

É necessário o incentivo e uso das tecnologias de segurança, pois mesmo as vulnerabilidades sendo sanadas e não mais apresentando ameaças, pode-se observar, com base nas listas feitas pela OWASP, que novas vulnerabilidades surgirão e, sucessivamente, essa listagem será atualizada.

Pode-se concluir que o motivo principal deste fenômeno é que tanto o desenvolvimento de novas tecnologias quanto a atualização de antigas são realizadas por humanos, e esse componente traz a compreensão que, mesmo não havendo vulnerabilidades detectadas no momento, é possível que elas somente não foram descobertas ainda.

Referências

ADACHI, T.; OMOTE, K. An Approach to Predict Drive-by-Download Attacks by Vulnerability Evaluation and Opcode. 2015 10th Asia Joint Conference on Information Security. DOI: 10.1109/asiajcis.2015.17. Citado na página 9.

BLACK, P. E.; OKUN, V.; GUTTMAN, B. Guidelines on Minimum Standards for Developer Verification of Software. NIST, 2021. DOI: 10.6028/NIST.IR.8397. Citado na página 6.

FOGIE, G.; HANSEN, R.; PETKOV, S.; JEREMIAH, R.; ANTON, P. XSS Attacks: Cross Site Scripting Exploits and Defense. Syngress, 2007. Citado na página 6.

FREEMAN, A. Pro Vue.js 2, 2018. DOI:10.1007/978-1-4842-3805-9. Citado na página 5.

GLASPIE, H. W.; KARWOWSKI, W. Human Factors in Information Security Culture: A Literature Review. Advances in Human Factors in Cybersecurity, 269–280, 2017. DOI: 10.1007/978-3-319-60585-2_25. Citado na página 2 e 11.

HARRISON, M. Guide to: Learning Python Decorators. Hairysun.com, 2013. Citado na página 5.

JABIYEV, B.; MIRZAEI, O.; KHARRAZ, A.; KIRDA, E. Preventing server-side request forgery attacks. 36th Annual ACM Symposium on Applied Computing, 2021. Citado na página 10.

LUO, X.; BRODY, R.; SEAZZU, A.; BURD, S. Social Engineering. Information Resources Management Journal, 24(3), 1–8, 2011. DOI: 10.4018/irmj.2011070101. Citado na página 11.

MORAES, A. d. Firewalls - Segurança no controle de acesso, Addison Wesley, 1ª edição, 2015. Citado na página 2 e 3.

MUKHOPADHYAY, R.; NATH, A. Ethical Hacking: Scope and challenges in 21st century. IJIRAE, 2014. Citado na página 6.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. An Introduction to Computer Security: The NIST Handbook. Special Publication 800-12. out 1995. Citado na página 2.

OPEN WEB APPLICATION SECURITY PROJECT. OWASP Top Ten, 2021. Disponível em: <https://owasp.org/Top10/>. Acesso em: 21 de out. de 2022. Citado na página 2, 6 e 7.

PELTIER, T. R. Social Engineering: Concepts and Solutions, EDPACS: The EDP Audit, Control, and Security Newsletter, 33:8, 1-13, 2006. DOI: 10.1201/1079.07366981/45802.33.8.20060201/91956.1. Citado na página 11.

ROSULEK, M. The Joy of Cryptography, 2021. pp. 4. Citado na página 5.

STALLINGS, W. Criptografia e segurança de redes: Princípios e práticas, 6ª edição, 2015. Citado na página 2, 3 e 5.