Principais vulnerabilidades de segurança Web

João Mauro C. Lopes[1], Millys Fabrielle Araujo Carvalhaes[2]

[1,2]Bacharelado em Engenharia de Computação – Centro Universitário de Anápolis (UniEVANGÉLICA) – Anápolis - GO

[1]jjoaommauro@gmail.com, [2]millys.carvalhaes@docente.unievangelica.edu.br

Resumo. *Este artigo descreve uma das principais vulnerabilidades de segurança Web, de acordo com o OWASP (Open Web Application Security Project) que é uma comunidade online que cria e disponibiliza de forma gratuita artigos, metodologias, documentação, ferramentas e tecnologias no campo da segurança de aplicações web.*

1. Introdução

No início, a Web foi criada sem grandes preocupações com segurança, seu objetivo principal era disponibilizar conteúdos adequados aos recursos disponíveis na época. Mas a cada dia a Web vem se consolidando como um dos principais meios de comunicação, de relacionamentos e de negócios. De uma maneira crescente empresas disponibilizam informações, produtos, serviços e, ainda, realizam negócios cada vez mais importantes. Diante desses fatos tornou-se de extrema necessidade garantir a segurança nas aplicações Web. Por se tratar de serviços remotos, executado muitas vezes distribuidamente, conceitos fundamentais de segurança como confidencialidade, integridade, disponibilidade e autenticidade devem estar presentes nos sistemas para diminuir os riscos de ataques devido à exploração de vulnerabilidades [MELLO *et al.* 2006].

As vulnerabilidades são condições que quando exploradas por pessoas mal-intencionadas podem resultar em falhas de segurança. As vulnerabilidades Web são o resultado de um conjunto de fatores que, em sua maioria, envolve todo processo de desenvolvimento, de manutenção e de uso das aplicações. Prazos curtos de entrega das aplicações aliados a processos falhos de desenvolvimento resultam em maior taxa de falhas de segurança nas aplicações. O estudo e a análise das principais vulnerabilidades Web possibilita aos desenvolvedores uma compreensão dos principais erros cometidos no ciclo de desenvolvimento de software, evitando assim, custos futuros com manutenção relacionadas a falhas de segurança [MONSTEVERDE 2014].

Vulnerabilidades em softwares têm sido amplamente utilizadas por atacantes, para roubo de informações confidenciais e invasões de redes corporativas. Prover a segurança de um software, porém, não é um objetivo fácil de ser alcançado, dada a complexidade dos sistemas nos dias de hoje. Facilmente, eles atingem dezenas de milhares de linhas de código, que contêm, invariavelmente, uma quantidade de defeitos significativa. Alguns destes têm impacto direto em segurança, podendo acarretar desde a indisponibilidade do sistema, até o controle total do computador por um atacante. Para piorar ainda mais este cenário, considere-se que, normalmente, um ciclo de desenvolvimento de software seguro não é adotado, o que resulta, no mínimo, em especificações inseguras e configuração vulnerável das plataformas subjacentes.

2. OWASP (*Open Web Application Security*)

A OWASP é uma fundação aberta sem fins lucrativos dedicada a capacitar empresas e organizações a desenvolver aplicações confiáveis, seu foco principal são aplicações Web. Ela reúne informações que permitem avaliar os riscos de segurança das aplicações Web e combater assim formas de ataques a segurança através da Internet. Os documentos produzidos pela OWASP são disponibilizados a toda comunidade internacional gratuitamente e são utilizados como ponto de referência por diversas entidades e organizações na área de tecnologia e segurança como *U.S. Defense Information Systems Agency* (DISA), *U.S. Federal Trade Commission* e *PCI Council* [MONSTEVERDE 2014].

O trabalho mais divulgado da OWASP é o "*The Top 10 Most Critical Web Application Security Risks*" OWASP Top 10, que agrupa os maiores os riscos de ataques críticos a partir de vulnerabilidades em aplicações Web, com atualizações periódicas, com última publicação em 2013 [MONSTEVERDE 2014].

2.1. Cross Site Scripting (XSS)

Os furos XSS ocorrem sempre que uma aplicação obtém as informações fornecidas pelo usuário e as envia de volta ao navegador sem realizar validação ou codificação daquele conteúdo. O XSS permite aos atacantes executarem scripts no navegador da vítima, o qual pode roubar sessões de usuário, pichar sites Web, introduzir worms, etc [OWASP 2007].

Os ataques são frequentemente implementados em Java Script, que é uma ferramenta poderosa de *scripting*. O uso do Java Script habilita atacante a manipular qualquer aspecto da página a ser renderizada, incluindo a adição de novos elementos (como um espaço para login que encaminha credenciais para um site hostil), a manipulação de qualquer aspecto interno do DOM e a remoção ou modificação de forma de apresentação da página etc [OWASP 2007].

2.2. Falhas de Injeção

As falhas de Injeção, particularmente injeção SQL, são comuns em aplicações web. Existem muitos tipos de injeção: SQL, LDAP, XPath, XSLT, HTML, XML, comando de sistema operacional e muitas outras. Falhas de Injeção acontecem quando os dados que o usuário dá de entrada são enviados como parte de um comando ou consulta. Os atacantes confundem o interpretador para o mesmo executar comandos manipulados enviando dados modificados. As falhas de Injeção habilitam o atacante a criar, ler, atualizar ou apagar arbitrariamente qualquer dado disponível para a aplicação. No pior cenário, estes furos permitem ao atacante comprometer completamente a aplicação e os sistemas relacionados, até pelo contorno de ambientes controlados por *firewall* etc [OWASP 2007].

2.3. *Execução Maliciosa de Arquivo*

As vulnerabilidades de execução de arquivos são encontradas em muitas aplicações. Os desenvolvedores têm por hábito usar diretamente ou concatenar entradas potencialmente hostis com funções de arquivo ou *stream*, ou confiar de maneira imprópria em arquivos de entrada. Em muitas plataformas, frameworks permitem o uso de referências a objetos externos, como referências a URLs ou a arquivos de sistema. Quando o dado é insuficientemente verificado, isto pode levar a uma inclusão arbitrária remota que será processado ou invocado um conteúdo hostil pelo servidor web [OWASP 2007].

2.4. Referência insegura direta a objeto

Uma referência direta a um objeto acontece quando um desenvolvedor expõe uma referência a um objeto de implementação interna, como por exemplo, um arquivo, diretório, registro na base de dados ou chave, uma URL ou um parâmetro de um formulário. Um atacante pode manipular diretamente referências a objetos para acessar outros objetos sem autorização, a não ser que exista um mecanismo de controle de acesso [OWASP 2007].

2.5. Cross Site Request Forgery (CSRF)

É simples e devastador. Um ataque CSRF força o navegador logado da vítima a enviar uma requisição para uma aplicação web vulnerável, que realiza a ação desejada em nome da vítima [OWASP 2007].

2.6. Vazamento de Informações e Tratamento de Erros Inapropriados

Diversas aplicações podem sem intenção vazar informações sobre suas configurações, funcionamento interno, ou violar privacidade através de diversos problemas. Aplicações podem vazar o funcionamento interno via tempo de resposta para executar determinados processos ou respostas diferentes para entradas diversas, como exibindo mesma mensagem de erro, mas com código de erros diferentes. Aplicações Web frequentemente vazarão informações sobre seu funcionamento interno através de mensagens de erros detalhadas ou debug. Frequentemente, essa informação pode ser o caminho para lançar ataques ou ferramentas automáticas mais poderosas [OWASP 2007].

2.7. Furo de Autenticação e Gerência de Sessão

Autenticação e gerência de sessão apropriadas são críticas para a segurança na web. Falhas nesta área geralmente envolvem a falha na proteção de credenciais e nos tokens da sessão durante seu tempo de vida. Estas falhas podem estar ligadas à roubo de contas de usuários ou administradores, contornando controles de autorização e de responsabilização, causando violações de privacidade [OWASP 2007].

2.8. *Armazenamento Criptográfico Inseguro*

Proteger dados sensíveis com criptografia tem sido parte chave da maioria das aplicações Web. Simplesmente não criptografar dados sensíveis é muito comum. Ainda, aplicações que adotam criptografia frequentemente possuem algoritmos mal concebidos, usam mecanismos de cifragem inapropriados ou cometem sérios erros usando cifragem fortes [OWASP 2007].

2.9. Comunicações Inseguras

Aplicações geralmente falham na hora de encriptar tráfego de rede quando é necessário proteger comunicações sensíveis. A encriptação (geralmente SSL) deve ser usada em todas as conexões autenticadas, especialmente páginas web com acesso via internet, mas também conexões com o back-end. Senão, o aplicativo irá expor uma autenticação ou o token de sessão. Adicionalmente, a autenticação deve ser usada sempre que dados sensíveis, assim como cartões de crédito ou informações de saúde são transmitidos. Aplicações cujo modo de encriptação possa ser subvertido são alvos de ataques [OWASP 2007].

2.10. Falha ao Restringir Acesso à Urls

Comumente, a única proteção para uma URL é não mostrar o link para usuários não autorizados. No entanto, um motivado, hábil ou apenas um sortudo atacante pode ser capaz

de achar e acessar estas páginas, executar funções e visualizar dados. Segurança por obscuridade não é suficiente para proteger dados e funções sensíveis em uma aplicação. Verificações de controles de acesso devem ser executadas antes de permitir uma solicitação a uma função sensível, na qual garante que somente o usuário autorizado acesse a respectiva função [OWASP 2007].

3. Ciclo de Desenvolvimento de Software Seguro

Um software seguro é aquele que satisfaz os requisitos implícitos e explícitos de segurança em condições normais de operação e em situações decorrentes de atividade maliciosa de usuários. Para isso, deve resultar de um ciclo de desenvolvimento que considere aspectos de segurança em todas as suas fases, da especificação à implantação em produção. Todavia, muito desenvolvedor começa a se preocupar com segurança, somente quando o software está quase finalizado, pois acreditam, erroneamente, ser possível trabalhar dessa maneira. Tal abordagem só contribui para um maior custo, pois as correções de vulnerabilidades tornam-se mais caras à medida que se avança pelas fases de desenvolvimento [UTO, MELO 2009].

Cada fase de um ciclo de desenvolvimento de software seguro, portanto, tem sua parcela de contribuição para a qualidade do resultado final e não pode ser omitida durante o processo. Na etapa de especificação, requisitos explícitos de segurança devem ser numerados e requisitos funcionais devem ser avaliados, para verificar se não introduzem uma vulnerabilidade no sistema [UTO, MELO 2009].

Atividades comumente realizadas na fase de projeto, incluem o mapeamento do modelo de dados para estruturas lógicas e físicas, a definição de padrões de interface a serem utilizados, a escolha de elementos de hardware e software que farão parte da solução e o desenho da topologia a ser adotada. Nesse contexto, um problema muito comum de segurança que surge é o posicionamento incorreto do banco de dados na topologia de rede [UTO, MELO 2009].

Todo software sempre apresenta uma ou mais falhas de segurança, ao longo de sua existência. Assim, é razoável concluir que é utópico construir um sistema completamente invulnerável. Porém, com um ciclo de desenvolvimento seguro, é possível, no mínimo, produzir consistentemente sistemas com um número reduzido de brechas e que possuam mecanismos de proteção contra os diversos ataques conhecidos [UTO, MELO 2009].

4. Conclusão

Tendo como base os dados apresentados foi possível identificar que muitas vulnerabilidades de segurança são resultado de práticas de programação ruins. Nesse sentido a utilização de verificação, validação e teste de software é essencial para a qualidade de um software.

Outro ponto identificado que compromete a segurança do software é a entrada não validada de dados. Esta por sua vez é de difícil detecção, pois a falha ocorrerá para um conjunto específico de dados. Dessa forma, além a elaboração dos testes a escolha dos dados de entrada para execução destes é fundamental para descoberta de tais falhas.

Por fim, este trabalho apresentou a grande influência que decisões de projeto e conscientização dos programadores quanto a utilização de técnicas mais seguras de programação influenciam a qualidade de um software.

Referências

Mello, E. R., Wangham, M. S., Fraga, J. da S. Camargo, E. (2006). Segurança em Serviços Web. Santa Catarina: Universidade Federal de Santa Catarina. Departamento de Automação e Sistemas.

QWASP (2007). As 10 vulnerabilidades de Segurança mais críticas em aplicações web.

Monteverde, W.A. (2014). Estudo e Análise de Vulnerabilidades Web. Universidade Tecnológica Federal do Paraná

Uto, N., Melo, S. (2009). Vulnerabilidades em Aplicações Web e Mecanismos de Proteção. Minicursos SBSeg.