Segurança de aplicações *web*: Experimentos e taxonomia de ataques.

Cícero Roberto Ferreira de Almeida

Resumo

A segurança nas interações eletrônicas via *web* é um tema relevante para usuários e organizações que mantém aplicações *web*. No entanto, as aplicações *web* que proporcionam serviços on-line podem conter vulnerabilidades que produzem riscos de acesso indevido a informações privativas de empresas e usuários de sistemas *web*. Este artigo apresenta um estudo com simulação de ataques comuns a aplicações *web* e também uma taxonomia de ataques sugestões de defesa no sentido de melhorar a segurança de aplicações *web*.

Palavras chave: Segurança da informação. Aplicação *web*. Ataque. SQL Injection. Banco de dados. Vulnerabilidade.

Abstract

Safety on the electronic interactions via world wide web is an important issue for users and organizations which maintains web applications. However, web applications that provide online services may contain vulnerabilities which produce risks of unauthorized access to private information of companies and users of web systems. This article presents a study with simulated attacks common to web applications and also a taxonomy of attacks and defense suggestions to improve the security of web applications.

Keyword: Information security. Web application. Attack. SQL Injection. Database. Vulnerability.

1 Introdução

A ampla interação entre pessoas e organizações via Internet é uma realidade da época atual. A internet é o meio facilitador destas interações. Por meio da rede mundial, os sistemas *web* possuem um papel importante nas mais distintas finalidades. Assim, aplicações *web* podem ser destinadas a gerenciar informações pessoais dispostas em um blog, transações bancárias, operações com cartões de crédito, compra e venda de produtos e serviços, informações judiciais, restituição do imposto de renda entre outros.

Paralelamente à crescente demanda de serviços, evolui a preocupação com o aumento dos incidentes de segurança na *web* que frequentemente ocupam o noticiário. Nakamura e Geus (2003) comentam que a necessidade de segurança vem transcendendo o limite da produtividade e da funcionalidade. A vantagem competitiva alcançada pelos processos eficientes e velozes das novas tecnologias se contrapõe à falta de segurança dos mesmos meios que proporcionam tal vantagem. Isto possivelmente resulta em prejuízos e limitam novas oportunidades de negócios. Tanenbaum (2003) alerta para este problema, afirmando: "a *Web* é o lugar em que encontramos a maioria dos intrusos, espionando e fazendo seu

trabalho sujo".

A construção de *websites* dinâmicos envolve um esforço de desenvolvimento e codificação que segue a metodologia adotada para o desenvolvimento de sistemas. Neste sentido, um sistema *web* de informação também está sujeito a erros de programação que podem torná-lo vulnerável à diversos tipos de ataques.

O conhecimento das vulnerabilidades em sistemas é importante na construção de soluções de segurança dos sistemas, assim a proposta deste artigo é demonstrar um experimento numa aplicação fictícia, descrevendo as vulnerabilidades relacionadas aos ataques de SQL Injection e vulnerabilidades em aplicativos que utilizam AJAX (*Assincronous Javascript And XML*), bem como propor uma profilaxia para a solução destes problemas.

2. Experimento realizado

O estudo realizado utilizou um experimento de ataque em uma aplicação fictícia desenvolvida em ASP. A aplicação contém vulnerabilidades para demonstrar as possibilidades de ataque. A aplicação simula uma loja de venda de veículos contendo uma área restrita controlada por usuário e senha.

2.1 Ambientes dos ataques desferidos:

A aplicação ASP foi submetida a um experimento de ataque realizado em um ambiente Windows XP Service Pack 2 com servidor HTTP IIS 5.1 com acesso ao SGBD em SQL Server 2000.

2.2 Primeiro ataque: Acesso à área restrita da aplicação

A primeira investida contra a aplicação ASP realiza uma cópia da página de login para a máquina do atacante. Isto permitiu alterar os campos da página, burlando um controle *client-side* desenvolvido em javascript. Ao executar este procedimento, foi possível obter um formulário ajustado para conseguir enviar informações manipuladas ao servidor. Após o ajuste do formulário de login, empregou-se uma injeção de SQL para conseguir o acesso à área controlada por usuário e senha sem o conhecimento destes parâmetros. Foi passado um parâmetro manipulado (um fragmento de código SQL) que se inseriu na *string* de consulta do banco de dados, permitindo o acesso indevido.

2.3 Segundo ataque: Revelação de estrutura e formação da *string* de consulta

Neste segundo ataque, a técnica de injeção de SQL foi empregada amplamente para provocar erros que revelaram informações sobre a estrutura da tabela. À medida que os erros ocorreram, a *string* injetada foi sendo complementada até a total descoberta dos campos da tabela de usuários, bem como seus tipos de dados. Após a descoberta do último campo da tabela de usuários, uma nova *string* foi injetada, desta vez, contendo comandos para a inclusão de um novo registro, confirmada posteriormente ao ser realizado novo acesso utilizando o usuário e a senha criados externamente pela injeção de SQL.

2.4 Terceiro Ataque: Descoberta de conteúdos de campos

De posse do acesso à área restrita obtido nos ataques anteriores, foi implementado de um terceiro ataque com objetivo de descobrir conteúdos armazenados. Primeiramente utilizou-se uma URL do sistema que utiliza um parâmetro numérico, este parâmetro foi manipulado e trocado por uma instrução SQL de consulta à tabela de usuários que retorna um campo texto, isto provoca erros de sintaxe que exibem o conteúdo do campo texto no lugar ao parâmetro numérico que a URL passaria para o servidor. Este método permitiu descobrir o primeiro usuário da tabela de usuários, que corrobora com a afirmação de Ulbrich e Della Valle (2007), na qual, em mais de 90% dos casos, o primeiro identificador de usuário em tabelas de usuários é justamente do desenvolvedor, com privilégios especiais. Foram também empregadas instruções em Transact SQL para criar uma nova tabela. Na mesma instrução, os dados da tabela de usuários foram transferidos para esta nova estrutura e posteriormente recuperados. O avanço do experimento permitiu executar comandos de DCL (Data Control Language) que levou à queda de serviço do SGBD SQL Server.

3 Vulnerabilidades

A vulnerabilidade, segundo Moreira (2001), está relacionada com a suscetibilidade a um determinado ataque e é composta de uma condição existente em recursos, processos, configurações etc. Esta condição é causada, em muitos casos, por insuficiência ou ausência de medidas defensivas.

Aplicativos *web* estão sujeitos a diversas ameaças que provém de vulnerabilidades. As vulnerabilidades *web* são classificadas em duas categorias: Vulnerabiliades na plataforma e vulnerabilidades de aplicação (SHEMA, 2003).

3.1 Vulnerabilidades na plataforma

São as vulnerabilidades relacionadas à infra-estrutura de hospedagem da aplicação, ou seja, o sistema operacional utilizado, SGBD, servidor HTTP entre outros.

Vulnerabilidades de plataforma identificadas nas aplicações estudadas:

- *Drivers* de conexão com SGBD que executam de múltiplas instruções SQL
- Permissão de leitura em diretórios não recomendados para usuários externos à organização.

3.2 Vulnerabilidades de aplicação

Consistem em implementações de código inadequadas ou mesmo em erros de programação que possibilitam a um invasor atuar de forma inesperada para explorar estas falhas.

Vulnerabilidades de aplicação identificada na aplicação estudada:

- Validação inexistente ou inadequada de parâmetros e informações provenientes de formulários HTML preenchidos por usuários externos;

- Utilização de senhas de superusuários de SGBD dentro de aplicações destinadas a usuários de internet.
- Ausência de tratamento de erro ou exceção com mensagem padronizada que impeça a manipulação de parâmetros na URL em aplicações *web*.

4 Taxonomia dos ataques

Atacar a uma aplicação consiste na explorar determinada vulnerabilidade. O foco deste artigo está nos ataques direcionados a aplicações, ou seja, exploração de vulnerabilidades de aplicação.

Esta seção apresenta uma proposta de taxonomia para enquadramento dos ataques e técnicas de invasão. Entretanto antes de estabelecer em uma descrição e classificação dos ataques, é importante observar a Figura 1 que ilustra um pouco do universo dos ataques a aplicativos *web*.

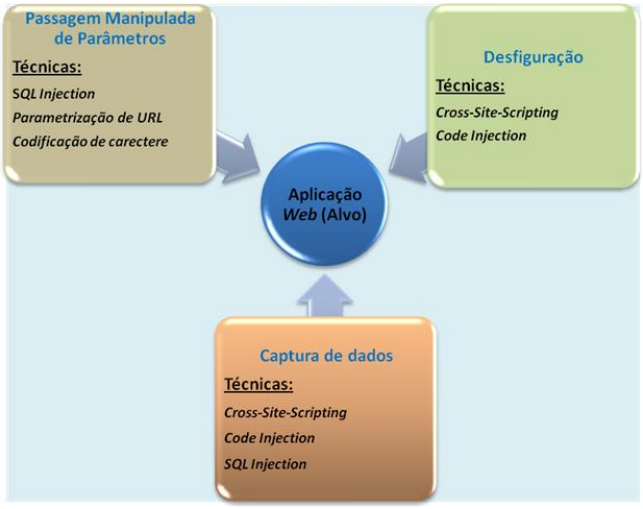

Figura 1 – Taxonomia de Ataques e Técnicas Exemplo de figura
Fonte: O autor.

Conforme mostra a Figura 1, um ataque pode empregar uma ou mais técnicas, ou seja, o objetivo de uma investida maliciosa em aplicações *web* pode ser alcançado por meio das técnicas de ataque.

Os mecanismos caracterizadores de ataques são configurados através destas técnicas e são os responsáveis por adaptarem as técnicas para a obtenção de um ataque. As técnicas não são exclusivas de um determinado ataque em razão de muitas investidas maliciosas combinarem diversas técnicas.

Os ataques são divididos em três categorias:

4.1 Passagem Manipulada de Parâmetros.

Neste tipo de ataque, os parâmetros de entrada de uma aplicação são alterados ou manipulados. Os parâmetros em aplicações *web* são em geral enviados para o servidor pelos métodos GET ou POST. O atacante pode se beneficiar e ter acesso a informações e modificar os parâmetros na URL ou dentro do formulário HTML da aplicação. Um bom exemplo seria

um sistema de reservas de livros, um usuário que tenha acesso à aplicação possui um identificador poderia modificar este identificador na URL ou dentro do formulário e ter acesso às reservas de um outro usuário. Entretanto os atacantes utilizam a passagem manipulada de parâmetros para provocar erros e descobrir mais sobre a aplicação e seu banco de dados. A passagem manipulada de parâmetros pode ser realizada combinando o emprego de técnicas de SQL Injetion, parametrização de URL e codificação de caracteres na URL.

4.2 Captura de Dados.

Ataques de captura de dados, como o próprio nome diz, visam capturar os dados armazenados em uma aplicação para fins diversos. Este tipo de ataque, geralmente combina técnicas Coss-Site-Scripting XSS (SHEMA, 2003) e Code Injection ou mesmo de SQL Injection.

4.3 Desfiguração.

O objetivo das investidas de desfiguração ou *defacement* contra aplicações *web* é causar alteração visual em uma ou mais páginas de um sistema ou *web*site. Geralmente altera-se a página principal. O dano visual produzido pelo ataque geralmente expõe uma imagem negativa para a organização mantenedora do *web*site. A desfiguração pode ser aplicada por meio da técnica de XSS (Cross Site Scripting).

5 Técnicas de ataque

Esta seção apresenta as principais técnicas utilizadas em ataques a aplicações *web*.

5.1 SQL Injection

A técnica de SQL Injection explora fragilidades em aplicações *web* que permitem receber *strings*, ou mesmo fragmentos de *strings* que contenham comandos da linguagem SQL. Esses códigos provindos do cliente incorporam-se nas *strings* de consultas a banco de dados formando instruções não esperadas pela programação. Em geral, é feito o uso de tautologias para validações, caracteres separadores de instruções, caracteres de comentários próprios da linguagem SQL. A técnica de SQL Injection possui algumas variações que podem ser utilizadas nos três tipos de ataques classificados de acordo com a taxonomia proposta.

5.2 Code Injection

Na técnica Code Injection, os códigos injetados são do tipo *server-side*. Trata-se de uma generalização de técnicas que utilizam códigos *server-side*. Se um atacante emprega um código PHP, utiliza-se a seguinte nomenclatura: PHP Injection (SHIFLETT, 2005). Essa denominação é definida com base na linguagem *server-side* na qual a aplicação alvo foi desenvolvida, deste modo podemos ter ASP Injection, PERL Injection entre outros. Este trabalho descreve um pouco da variante PHP Injection em razão de ser uma linguagem

muito popular e de uso amplo no desenvolvimento de aplicações *web*. Esta variante é direcionada para aplicações cuja linguagem *server-side* é PHP. Ataques que implementam estas técnicas podem trazer severos danos à aplicação, desde coleta de informações sigilosas e até negação de serviços com a inoperância da aplicação mediante a inserção de scripts maliciosos.

5.3 Cross-Site Scripting (XSS)

O XSS nada mais é do que a inserção de códigos HTML em um aplicativo *Web*. O que o difere do Code Injection é o tipo de código inserido. Dentro da técnica de Cross-Site Scripting existe uma especificação denominada Prototype Hijacking

5.4 Codificação de caractere

Codificação de caractere (SHEMA, 2003) é uma técnica empregada em ataques do tipo Passagem Manipulada de Parâmetro, onde o atacante utiliza códigos Unicode (URL-encode) nos caracteres enviados ao servidor. Por exemplo, o caractere aspa simples (') pode ser substituído pela codificação URL-encode (%27).

Exemplos de caracteres que podem ser manipulados com codificação URL-encode e passados para a aplicação:

Sequência de caracteres	URL-encode da sequência
../	%2e%2e%2f
<script>	%3cscript%3e
;	%3b

Tabela 1 – Exemplos de codificação unicode manipulada para sequência de caracteres
Fonte: O autor

5.5 Taxonomia das especializações das técnicas SQL Injection

Este artigo propõe também uma taxonomia para as especializações das técnicas de SQL Injection, de acordo com os objetivos a que se propõe cada tipo de investida contra aplicações *web* que faça uso de injeções de SQL.

5.5.1 Revelação de estrutura

Esta variação da técnica de SQL Injection se propõe a descobri a estrutura de tabelas no banco de dados. A técnica injeta caracteres e *string*s que de adaptam à *string* de consulta a banco interna da aplicação provocando erros que revelam a estrutura de tabelas, com a sequência de campos e seus tipos. É uma variação da técnica direcionada para ataques de validação de parâmetros.

5.5.2 Autenticação indevida de usuário

Aplicações dispostas na *web* geralmente possuem uma interface pública, e uma restrita a usuários que necessitam de autenticação. Na interface pública, os usuários externos da

aplicação têm acesso a serviços e informações ofertados pela organização que mantém a aplicação em funcionamento. Na interface restrita, a organização mantenedora da aplicação administra os conteúdos dispostos na interface pública; para isto, os administradores do conteúdo precisam estar autenticados. Utiliza-se em geral um módulo de login para autenticar estes usuários. Uma técnica de autenticação indevida de usuário pressupõe conseguir acesso não autorizado a esta interface restrita. Em muitas aplicações vulneráveis a ataques de validação de entrada de parâmetros, é possível conseguir acesso como um usuário válido por meio da técnica de SQL Injection ao inserir *strings* apropriadas para adaptarem-se às *string*s internas de consulta que autenticam usuários.

5.5.3 Manipulação de registros em tabelas

Aplicações possíveis de ataques de validação de entrada de parâmetros e especialmente sensíveis a técnicas de SQL Injection permitem a manipulação de registros, onde é possível incluir novos registros ou alterar campos dos registros existentes, no entanto esta é uma variação que em geral se segue após revelada a estrutura da tabela, pois com este conhecimento, o atacante poderá montar uma *string* a ser injetada no interior da *string* de consulta da aplicação para incluir, alterar ou mesmo excluir registros da tabela.

5.5.4 Recuperação de conteúdos.

A técnica de SQL Injection pode ser aplicada para recuperar conteúdos de tabelas em banco de dados. Para recuperar dados das organizações, os atacantes em geral exploram as vulnerabilidades dos drivers de conexão com de banco que expõem os conteúdos de determinados tipos de campos quando encontram uma condição de erro. Normalmente, os campos alfanuméricos armazenados nestes bancos são os mais vulneráveis a este tipo de problema, sendo revelados ao atacante.

5.5.5 Destruição de informações.

A destruição de informações vai muito além de meramente excluir um determinado registro em uma tabela. Uma simples exclusão fica melhor classificada dentro de manipulação de registro em tabelas. Ao invés disso, a vulnerabilidade a ataques de validação de entrada de parâmetros com o uso de técnica SQL Injection pode permitir a exclusão de uma tabela inteira ou mesmo do próprio banco de dados, caso na programação do sistema for utilizada senha de super-usuário nas conexões implementadas dentro da aplicação. Neste aspecto, uma injeção de SQL pode levada a termo com o uso de comandos DROP TABLE ou mesmo DROP DATABASE, promovendo a destruição em massa das informações, vindo a comprometer toda aplicação e suspender todos os serviços.

5.5.6 Interrupção de serviços.

A destruição de informações pode comprometer os serviços ofertados por uma aplicação, entretanto o uso de injeções de SQL permite uma suspensão ou interrupção de serviços, em função do emprego de comandos DCL (Data Control Language) utilizados normalmente pelos

administradores de banco de dados par suspender o serviço do banco de dados. Uma aplicação que utilize senha de super-usuário possivelmente é passível de sofrer as consequências de um ataque de passagem manipulada de parâmetros pela variante da técnica de SQL Injection para Interrupção de Serviços.

6 Profilaxia e defesas

A defesas e profilaxias descritas aqui não têm a pretensão de solucionar os problemas de vulnerabilidades, são apenas algumas sugestões que podem ser empregadas para deixar as aplicações um pouco mais seguras.

Ataques de passagem manipulada de parâmetros exploram vulnerabilidades nas entradas de dados provindos do usuário externo. Uma solução para minimizar o problema é buscar uma forma de filtrar os dados vindos do usuário mediante funções. Qualquer informação que venha do meio externo precisa ser tratada antes de ser armazenada ou exibida no navegador. A passagem manipulada de parâmetros é muitas vezes desferida com o objetivo de provocar erros que revelem detalhes sobre a aplicação ou o banco de dados, com isto uma forma de dificultar o ataque seria padronizar mensagens de erros ou mesmo a criação de um módulo de tratamento de erros e exceções para impedir que os erros provocados revelem estruturas ou conteúdos do banco de dados. Por exemplo, um determinado módulo da aplicação que espere receber um dado numérico não deve exibir um erro de execução se vier a receber um parâmetro manipulado como string, ao invés disso, deve tratar o erro dentro do código e emitir uma mensagem padronizada pela aplicação.

Para evitar o ataque de passagem manipulada de parâmetros em nível de aplicação, o arquivo de configuração do PHP deveria sofrer as seguintes alterações:

Diretiva	Medida profilática
allow_URL_fopen (disponível desde o PHP 4.0.4)	Defina como off ou 0 (zero) para evitar a leitura de arquivos externos.
allow_URL_include(disponível desde o PHP 5.0)	Defina como off ou 0 (zero) para evitar a inclusão de arquivos externos.

Tabela 2 – Diretivas do arquivo php.ini
Fonte: O autor

Desta forma, o atacante não conseguiria realizar a implantação do XSS Prototype Hijacking na aplicação.

7 Conclusão

No estudo realizado, foi possível entender que os atos imprevistos intencionais, ou ataques, desferidos no intuito de explorar as vulnerabilidades da aplicação. Um ataque pode ser realizado com o emprego de uma ou mais técnicas específicas, entretanto o mais comum é o emprego associado de várias técnicas num mesmo ataque. Este trabalho apresentou uma taxinomia destes ataques no sentido de classificá-los para melhor entender as técnicas envolvidas. Este artigo fornece um breve levantamento sobre vulnerabilidades e ataques, no entanto não esgota este tema, pois diariamente novas vulnerabilidades surgem e técnicas

para sua exploração evoluem paralelamente. Como trabalho futuro, a proposta é criar aplicações *web* com tecnologias mais modernas e submetê-las a experimentos de ataques em busca de vulnerabilidades com vista a propor novas profilaxias e defesas.

Referências

DELLA VALLE, James e ULBRICH, Henrique César – Universidade Hacker. ed Digerati Books, 5ª edição, 2007.

MOREIRA, Nilton Stringasci. Segurança Mínima. Axcel Books, 2001.

NAKAMURA, Emilio T. e GEUS, Paulo L. Segurança de Redes, ed Futura, 2003.

SHEMA, Mike. Hack Notes – Segurança na Web, ed Campus, 2003.

SHIFLETT, Chris. Essential PHP Security, First Edition, United State: O'Reilly, 2005.

TANENBAUM, Andrew S. Redes de computadores, ed Campus. 2003.