Google Hacking para Ataques SQL Injection

Conference Paper · November 2014

CITATIONS
0

READS
6,059

2 authors, including:

Amaury Carvalho
Instituto Federal Goiano
2 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Google *Hacking* para Ataques SQL *Injection*

Amaury Walbert de Carvalho, Antonio Pires de Castro Júnior

Faculdade de Tecnologia SENAI de Desenvolvimento Gerencial (FATESG)
Rua 227-A, no 95, St. Leste Universitário – CEP 74610-155 – Goiânia – GO – Brazil

{amaurywalbert,apcastrojr}@gmail.com

Abstract. *The objective of this paper is to present a study on the Google Hacking and SQL Injection, alerting administrators of web systems and information security professionals about the risks of combining these techniques for detecting faults and vulnerability exploits. A case study using the tools Google Dorks and SqlMap is described to show the ease with which an attacker can gain access to sensitive information of an organization by detecting and exploiting faults in web systems. Some measures are introduced to minimize the problems caused by this type of attack..*

Resumo. *O objetivo deste trabalho é apresentar um estudo sobre o Google Hacking e SQL Injection, alertando os administradores de sistemas web e profissionais de segurança da informação sobre os riscos da combinação dessas técnicas de detecção de falhas e exploração de vulnerabilidades. Um estudo de caso usando as ferramentas Google Dorks e SqlMap é descrito para mostrar a facilidade com que um atacante consegue ter acesso a informações sensíveis de uma organização através da detecção e exploração de falhas em sistemas web. Algumas medidas são apresentadas para minimizar os problemas causados por esse tipo de ataque.*

1. Introdução

A Internet permite que diversos tipos de negócios aconteçam através de sistemas web, e as ferramentas de busca são aliadas no processo de divulgação dos produtos e serviços de uma organização. Essas mesmas ferramentas podem ser usadas por atacantes para descobrir falhas de segurança nos sistemas. Existem diversas técnicas usadas para esse propósito, uma delas é o Google *Hacking* (GHDB, 2014).

Exploradas de maneira correta, as falhas de segurança permitem que o atacante tenha acesso irrestrito à informações sensíveis de uma organização. De acordo com um levantamento feito pela OWASP (2013), o maior número de incidentes de segurança em sistemas web no ano de 2013 foi através de injeção de códigos SQL em sistemas vulneráveis. Este método de exploração é conhecido como SQL *Injection*.

O objetivo deste trabalho é chamar a atenção dos profissionais de segurança da informação e dos administradores de sistemas para as ameaças existentes. Atacantes estão combinando ferramentas do mecanismo de buscas do Google para identificar alvos, e explorando essas vulnerabilidades com ferramentas que fazem injeção de códigos SQL de forma automatizada para terem acesso a informações sensíveis.

Na próxima seção, o leitor encontra um levantamento sobre a necessidade de proteção de sistemas informacionais. As Seções 3 e 4 trazem um estudo sobre as técnicas Google *Hacking* e SQL *Injection*, respectivamente. A Seção 5 descreve um

estudo de caso usando as ferramentas Google *Dorks* e *SqlMap* para detecção de um alvo vulnerável, exploração e acesso a informações sensíveis de uma organização.

2. Segurança da Informação

Já há alguns anos a informação movimenta negócios pelo mundo todo e muitas organizações dependem exclusivamente da manipulação de informações para a geração de receita, e por consequência, manter sua existência e lucratividade. Para PMG Solutions (2011) a informação se tornou uma mercadoria valiosa e é difícil observar qualquer processo de negócio que não trabalhe com informações.

Como uma boa parte das informações sensíveis de uma organização estão sendo manipuladas por sistemas computacionais, indivíduos mal-intencionados podem se aproveitar de falhas nos sistemas para ter acesso a essas informações. Empresas são testadas a todo momento sobre aspectos relacionados à segurança da informação e os noticiários constantemente mostram casos de incidentes que vão desde acessos indevidos a sistemas web até alterações em banco de dados. O fato é que as brechas de segurança existem e devem ser consideradas e tratadas.

Para Ferreira (2003) existem três principais atributos de qualidade que a informação deve ser submetida: integridade, confidencialidade e disponibilidade. Esses atributos formam os pilares da segurança da informação. Malandrin (2013) defende a ideia de que uma organização deve adotar um Sistema de Gestão da Segurança da Informação (SGSI) e relaciona o Modelo CIA como propriedades essenciais sobre a informação da organização. Para este autor, essas propriedades são assim definidas: (1) Confidencialidade: a propriedade que a informação tem de que não será divulgada a entidades não autorizadas. Essa propriedade pode se referir ao conteúdo da informação ou a própria existência da informação. Controles de acesso ajudam a manutenção de confidencialidade da informação. (2) Integridade: a propriedade que a informação tem de não ser alterada por entidades não autorizadas. Essa propriedade também se refere tanto quanto ao conteúdo da informação como a sua origem. Controles existem para permitir a prevenção contra ataques a integridade ou a detecção de ataques que já aconteceram. (3) Disponibilidade: a propriedade que a informação tem de que estará disponível para as entidades autorizadas.

Com o aumento considerável da preocupação com a segurança da informação, normas e padrões foram estabelecidos para que os interessados possam realizar procedimentos que diminuam os riscos de ataques contra seus sistemas, entre eles os sistemas web, um dos principais.

No Brasil, existe a NBR ISO/IEC 27002, um código de prática para a segurança da informação baseado em normas internacionais e publicado pela ABNT (Associação Brasileira de Normas Técnicas). Essa norma é baseada nas melhores práticas de segurança da informação reconhecidas mundialmente e por sua constante revisão e atualização, deveria ser utilizada por todas as empresas. Para Franciscatto (2013) a adoção da ISO/IEC 27002 garante, além da segurança da informação, uma correta comunicação entre fabricante e cliente, a proteção do consumidor e a eliminação de barreiras técnicas e comerciais.

3. Google *Hacking*

Toledo (2012) mostra que existe uma quantidade expressiva de informações sensíveis que podem ser acessadas através de uma simples pesquisa no Google. O ato de explorar

falhas em sistemas web ou localizar informações confidenciais usando a ferramenta de pesquisa do Google é chamado de Google *Hacking*.

McGuffe (2013) define Google *Hacking* como um conjunto de ferramentas e técnicas que podem ser combinadas ou usadas de forma isoladas para descobrir vulnerabilidades e problemas de segurança na Internet. Assim como os operadores avançados do Google permitem que usuários possam melhorar suas pesquisas e conseguir localizar um determinado conteúdo de forma rápida e precisa, usuários mal-intencionados podem usar os mesmos artifícios para identificar possíveis alvos para ataques.

Profissionais de segurança da informação e usuários avançados dos sistemas web tentam identificar os tipos de consultas que podem resultar em busca por informações sensíveis. Exitem diversos catálogos disponíveis na internet que descrevem as principais técnicas usadas pelos praticantes do Google *Hacking*. A intenção é divulgar o máximo possível essas técnicas para que todos os responsáveis por administrar sistemas web possam conhecê-las e tentar evitar que os seus sistemas sejam explorados. A coletânea mais conhecida talvez seja o GHDB (2014). Criado originalmente por Long (2008), autor do livro *Google Hacking for Penetration Testers,* o GHDB foi incorporado ao *Exploit Database*, e as técnicas de consulta usadas no Google *Hacking* passaram a ser conhecidas também como Google *Dorks*. A partir de então os diversos softwares que fazem testes de intrusão podem usar o GHDB para aprimorar os testes e verificar se os sistemas estão vulneráveis a estes tipos de consultas (GHDB, 2014).

De acordo com o site Acunetix (2014) os *dorks* disponíveis no GHDB podem identificar as seguintes tipos de dados: (1) Avisos e vulnerabilidades do servidor; (2) Mensagens de erro que contêm muita informação; (3) Arquivos contendo senhas; (4) Diretórios sensíveis; (5) Páginas que contêm portais de logon; (7) Páginas que permitem acesso à dispositivos de rede ou dados como *logs* de *firewall*.

3.1. Evitando o Google *Hacking*

Diariamente surgem novos *dorks* que exploram falhas de segurança e podem ser usados para levantar informações confidenciais ou dados sensíveis na internet. Billing (2008) faz uma síntese sobre as técnicas que podem minimizar os efeitos desse tipo de ataque e ressalta a importância em estabelecer uma política de segurança sólida, no que diz respeito aos tipos de informações que podem ser disponibilizadas na web.

Outro ponto interessante é usar as mesmas técnicas do Google *Hacking* para explorar os seus próprios sistemas. Estar atentos aos novos *dorks* e testá-los contra os seus sites é uma maneira de verificar quais são as falhas que podem ser exploradas e com isso conseguir corrigir antes de um possível ataque efetivo. Caso as informações sensíveis sejam disponibilizadas por funcionários das empresas, cursos e treinamento podem ser aplicados para que esses funcionários conheçam os riscos de uma informação veiculada na internet de maneira imprópria.

A partir do conhecimento das técnicas de Google *Hacking*, os administradores podem criar ferramentas automatizadas para testar continuamente seus sistemas. Existem também soluções prontas que fazem esse trabalho. Os *Scanners* de Vulnerabilidades Web já estão fazendo uso do GHDB e também de outras bibliotecas, para tentar identificar sistemas que podem estar divulgando informações sensíveis e passíveis de serem exploradas por técnicas de Google *Hacking*.

4. SQL Injection

De acordo com um levantamento feito pela OWASP (2013), o *Structured Query Language Injection*, ou SQL *Injection*, foi a técnica mais utilizada para ataques a aplicações web em 2013. Tentando explorar variáveis usadas pelas aplicações para acessar o banco de dados, o atacante pode conseguir inserir comandos SQL maliciosos e receber como retorno informações sensíveis diretamente do banco de dados, ou, em alguns casos, inserir comados do próprio sistema operacional.

Um exemplo simples pode ser descrito a partir de uma página de login, onde um usuário deve passar valores referentes ao login e senha para a aplicação web. A partir de comandos previamente definidos, a aplicação se comunica com o banco de dados para consultar os valores passados pelo usuário. Caso o ambiente descrito não tenha nenhum mecanismo para verificar os tipos de dados inseridos, outros comandos SQL podem ser inseridos nos campos de login e senha e assim modificar a consulta ao banco de dados. Tudo o que o atacante precisa saber é um pouco dos comandos SQL e intuição para tentar adivinhar nomes usados pelos administradores do banco de dados para as tabelas e campos (ACUNETIX, 2014).

Na prática, um ataque de SQL *Injection* funcionaria da seguinte maneira:

```
SELECT produto FROM tabela WHERE nomeprod = 'usuário insere nome do produto';
```

O comando acima irá retornar o campo produto de uma tabela onde os valores inseridos pelo usuário no campo correspondente ao produto forem iguais aos valores mantidos no banco de dados. Os valores inseridos pelo usuário na aplicação web será usado para completar o comando SQL. Se um atacante insere os seguintes valores:

```
teste' OR 'x' = 'x
```

a sintaxe da consulta passar a ser a seguinte:

```
SELECT produto FROM tabela WHERE nomeprod = 'teste' OR 'x' = 'x'
```

Com a injeção de caracteres que alteram o código SQL, a consulta irá retornar valores do campo produto onde o nome do produto for teste, ou onde x for igual a x. Como x é igual a x em qualquer situação, o teste lógico realizado pelo operador OR irá retornar um resultado verdadeiro e todos os valores do campo produto serão listados.

Caso um comando *SELECT* seja executado em uma loja online vulnerável a ataques de SQL *Injection*, é possível que um atacante consiga visualizar informações de clientes, como endereço, CPF, e até mesmo o número do cartão de crédito.

Para Sadeghian (2013), a flexibilidade da linguagem SQL permite que essa técnica de ataque possa ser usada com muita frequência nos dias atuais e por isso lojas online devem tomar cuidado com seus sistemas web. Já para OWASP (2014), esse tipo de ataque ocorre porque em caso de sucesso, o atacante pode obter acesso completo ao banco de dados de um sistema, tornado-o um alvo atrativo.

4.1. Evitando Ataques

Propor uma solução única para os ataques de SQL *Injection* é muito difícil por causa da flexibilidade da linguagem SQL. Sadeghian (2013) propõem dois modelos que devem ser usados em paralelo: Consultas Parametrizadas e Configuração Inteligente do Esquema de Privilégios do Banco de Dados.

As Consultas Parametrizadas são usadas pelo desenvolvedor no momento da codificação. Espaços devem ser reservados nas consultas SQL para receberem os valores das variáveis. As instruções SQL são executadas sem as variáveis e depois com elas. Assim, mesmo que um atacante passe comandos SQL, estes serão tratados apenas como strings normais.

A Configuração Inteligente do Esquema de Privilégios do Banco de Dados, sugere que sejam criados usuários diferentes e com privilégios restritos para cada um. O administrador pode conceder privilégios para certas páginas de apenas consulta. Nesse caso um atacante apenas conseguirá usar o comando *SELECT*, excluindo a possibilidade de alteração do banco de dados, o que não isenta a possibilidade de visualização do conteúdo armazenado.

Outro método empregado para diminuir os problemas com SQL *Injection* é tratar os valores de entrada para que as variáveis do sistema web não considere valores que representem um código SQL malicioso. Controlar o tamanho de uma determinada variável e eliminar caracteres especiais usados pelo banco de dados são exemplos de como tentar tratar esses valores.

Ataque de SQL *Injection* têm impacto direto na integridade e confidencialidade da informação, causando enormes prejuízos para o negócio de qualquer instituição. O ideal é que os administradores de banco de dados e os desenvolvedores dos sistemas web consigam combinar os métodos e técnicas de prevenção para que os riscos de um ataque bem sucedido sejam minimizados.

5. Estudo de Caso

Este estudo de caso foi desenvolvido com o propósito de mostrar ao leitor como é simples o uso das técnicas de Google *Hacking* para detectar um alvo de ataque e conseguir acesso a banco de dados através de falhas de segurança em sistemas web com variáveis dinâmicas que permitem o uso de SQL *Injection*.

Primeiramente definimos um escopo para o teste de detecção de sistemas web vulneráveis a SQL *Injection*. Restringimos a consulta utilizando os operadores avançados do Google para formar um *dork* específico para este estudo de caso. Escolhemos pesquisar apenas sistemas web com variáveis dinâmicas que estão no domínio edu.br. O Goole *Dork* ficou assim:

```
inurl:"php?id=" site:edu.br
```

O operador inrul irá restringir os resultados à apenas sites que contenham em sua url o valor "php?id=", nesse caso, sistemas que usam variáveis dinâmicas passadas como parâmetro para formar a consulta que será executa no banco de dados. O operador site irá retornar resultados apenas no domínio "edu.br".

O próximo passo é identificar uma página que seja vulnerável a ataques de SQL *Injection*. Para isso, digitamos um apóstrofo ao final do endereço da página, que é onde a variável do sistema recebe os parâmetros necessários para a consulta ao banco de dados. Caso o sistema web interprete o apóstrofo, uma mensagem de erro de sintaxe SQL é mostrada, apontando que o sistema web não faz o tratamento adequado do conteúdo das variáveis que são passadas ao SGBD. Caso o sistema não seja vulnerável, o apóstrofo será ignorado e a página será exibida sem nenhum problema.

Após identificar o alvo, o atacante pode usar a ferramenta *SqlMap* (http://sqlmap.sourceforge.net/) para automatizar o processo de injeção de código SQL

e ter acesso ao banco de dados. A ferramenta utiliza vários métodos de SQL *Injection* para explorar a vulnerabilidade existente. O comando usado para iniciar o processo de exploração da falha é mostrado a seguir:

```
sqlmap -u http://siteexemplo.edu.br/popUpVisualizar.php?id=49557
--dbs --random-agent –ignore-proxy
```

As opções do comando servem para indicar a url a ser explorada (-u), identificar o SGBD e os bancos de dados existentes (--dbs), usar cabeçalhos de requisições HTTP de diferentes navegadores de forma aleatória (--random-agent) e ignorar conexões com proxy (--ignore-proxy).

Neste ponto o atacante já tem acesso ao SGBD do alvo e é possível identificar softwares que ali estão instalados (Servidor Web e SGBD), quantidade e os nomes do bancos de dados. Para descobrir a estrutura de um dos bancos o atacante pode usar mais opções da ferramenta *SqlMap*. O código a seguir explora um banco de dados a fim de mostrar a estrutura de tabelas:

```
sqlmap -u http://siteexemplo.edu.br/popUpVisualizar.php?id=49557
-D database_exemplo --tables
```

A opção -D do comando, indica o banco de dados a ser explorado e a opção --tables indica que devem ser mostradas as tabelas existentes nesse banco.

O procedimento de análise da estrutura de um banco de dados pode prosseguir para as colunas de uma tabela e até mesmo para os valores armazenados nessas colunas. A ferramenta *SqlMap* é completa nesse sentido. Além de permitir que seja feito uma cópia da estrutura do banco de dados e do próprio conteúdo do banco, a ferramenta ainda consegue identificar colunas que armazenam *hashes* de senhas, possuindo a opção de tentar descobrir a senha por ataque de força bruta a partir de um dicionário de senhas da própria ferramenta ou por valores e dicionários passados pelo atacante. O comando usado para esse tipo de procedimento é mostrado a seguir:

```
sqlmap -u http://siteexemplo.edu.br/popUpVisualizar.php?id=49557
-D database_exemplo -T usuario --columns --dump
```

As opções do comando indicam uma tabela a ser explorada (-T), a estrutura de colunas (--column) e o despejo (--dump), ou cópia, da estrutura e do conteúdo da tabela para o dispositivo do atacante. A Figura 1 mostra o resultado da sequência dos comandos anteriores e a Figura 2 mostra a ferramenta *SqlMap* descobrindo as senhas a partir do *hash*.

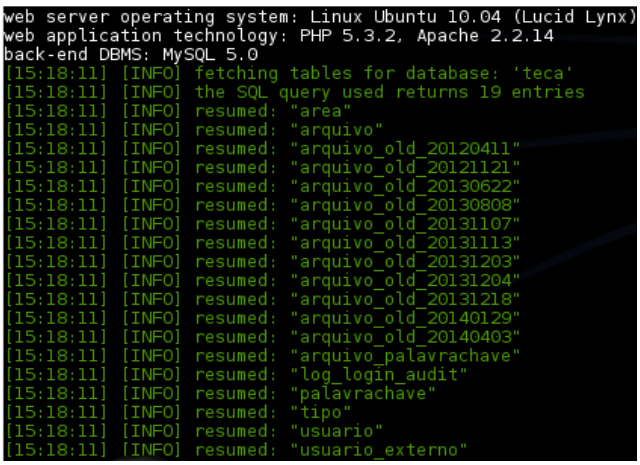

Figura 1. Tabelas Detectadas

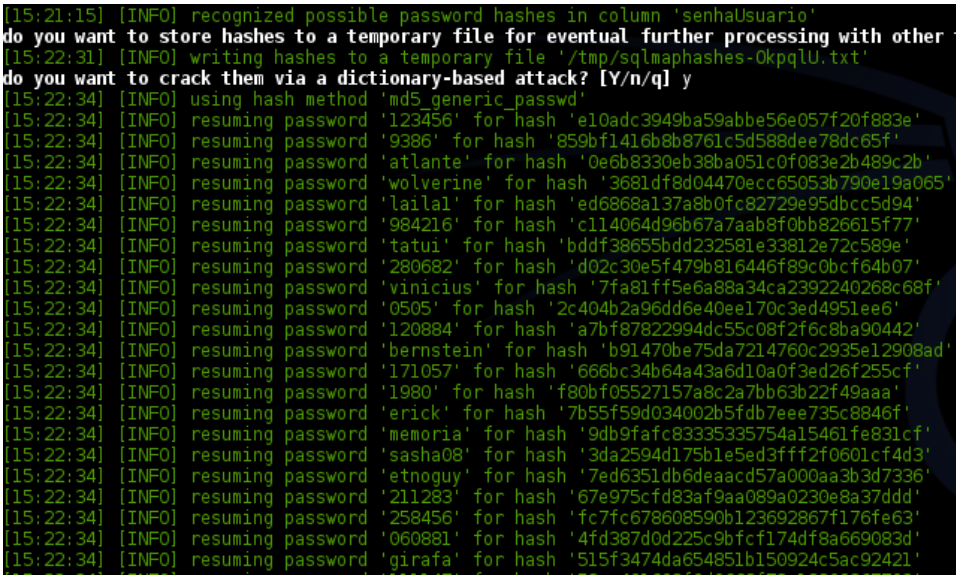

Figura 2. Descoberta de Senhas a partir do hash

6. Considerações Finais

Normas específicas, como a ISO/IEC 27002, estão disponíveis para auxiliar no processo de elaboração e manutenção de uma política que mantenha os pilares da segurança da informação. Avaliar o tipo de informação que pode ser veiculada em sistemas web também faz parte do processo de segurança, porém, no caso de ataques utilizando as técnicas descritas nesse trabalho, outros mecanismos de defesa devem ser adotados e colocados em prática.

Ataques de SQL *Injection* acontecem por negligência dos profissionais responsáveis pela implantação e manutenção do sistema ou por falhas nos softwares de terceiros adotados pela organização. De qualquer modo, a passagem de parâmetros em um sistema web deve ser testada a fim de detectar códigos inseridos pelo usuário que podem comprometer o banco de dados. Da mesma forma, a saída gerada pelo banco de dados pode conter informações sensíveis e também deve ser verificada. A construção de um sistema web pautado por tipos de consultas parametrizadas e a configuração inteligente de um SGBD são recomendadas.

No caso do Google *Hacking*, a recomendação é que os responsáveis pelo sistema web analise periodicamente os *dorks* existentes e aplique a técnica em seu próprio sistema para verificar que tipos de informações podem ser extraídas. Vários softwares de testes de vulnerabilidade já estão adotando os bancos de dados de *dorks* disponíveis na Internet para realizar uma verificação completa e identificar quais sistemas estão divulgando informações sensíveis e quais deles podem ser explorados pelas técnicas de SQL *Injection*.

Referências

Billig, J., Danilchenko, Y., and Frank, C. E. Evaluation of Google Hacking. In Proceedings of the 5th Annual Conference on Information Security Curriculum Development, InfoSecCD'08, pages 27–32, New York, NY, USA. ACM, 2008.

Ferreira, Fernando Nicolau Freitas. Segurança da Informação. 1a. ed. 176p. Editora Ciência Moderna, 2003.

Franciscatto, Roberto. Notas de Aula: Aula 2 - ISO 27002. Especialização em Gestão de TI. Universidade Federal de Santa Maria, 2013. Disponível em: <http://www.cafw.ufsm.br/~roberto/trabalhos/aulas/Aula%202%20-%20ISO-IEC %2027002.pdf> Acesso em: 28 Abr. 2014.

Google Hacking Database, GHDB, Google Dorks. Exploit Database. Disponível em: <http://www.exploit-db.com/google-dorks/> Acesso em: 29 Abr. 2014.

ISO/IEC 27002: Fundamentos. PMG Solutions. Manual de Treinamento. 2011. Disponível em: <http://www.pmgeducation.com.br/demo/e-book_ISO_IEC_27002_Online_DEMO. pdf> Acesso em: 20 Mai. 2014.

Long, J. Google Hacking for Penetration Testers, Volume 2. Syngress Publishing - Elsevier, 2008.

McGuffee, J. W. and Hanebutte, N. Google Hacking as a General Education Tool. J. Comput. Sci. Coll., 28(4):81–85, 2013.

Malandrin, L. J. A. A. Modelo de Suporte a Políticas e Gestão de Riscos de Segurança Voltado à Terceirização de TIC, Computação em Nuvem e Mobilidade. Dissertação (Mestrado) - Departamento de Engenharia de Computação e Sistemas Digitais. Escola Politécnica da Universidade de São Paulo. São Paulo, 2013.

Sadeghian, a., Zamani, M., and Ibrahim, S. Sql Injection is Still Alive: A Study on Sql Injection Signature Evasion Techniques. In Informatics and Creative Multimedia (ICICM), 2013 International Conference on, pages 265–268, 2013a.

SQL Injection - OWASP. Open Web Application Security Project. Disponível em: <https://www.owasp.org/index.php/SQL_Injection> Acesso em: 10 Mai. 2014.

Toledo, A. S. d. O. and Moraes, S. H. d. Google Hacking. Pós em Revista, (6):358–367, 2012.

Top 10 2013 - OWASP. Open Web Application Security Project. Disponível em: <https://www.owasp.org/index.php/Top_10_2013-Top_10> Acesso em: 10 Mai. 2014.

What is Google Hacking? Acunetix. Disponível em: <https://www.acunetix.com/websitesecurity/google-hacking/> Acesso em: 29 Abr. 2014.