Server-Side Request Forgery

Nas aplicações modernas é normal os servidores realizarem diversas requisições para outros serviços com o objetivo de realizar diversas ações, como salvar e recuperar dados. No entanto, essas requisições podem ser falsificadas, quando isso acontece o servidor sofreu um ataque de Server-Side Request Forgery, ou, Falsificação de Requisições do Servidor.

Nesse tipo de ataque o atacante pode se aproveitar dessa oportunidade para usar de funcionalidades que ele normalmente não teria acesso, como a leitura e manipulação de dados confidenciais, acesso às configurações do servidor, como usuários e admins, API's de uso interno do servidor que, normalmente, não ficam disponíveis ao público, entre diversas outros cenários que podem gerar prejuízos à empresa.

Como ele é realizado?

O ataque SSRF é realizado aproveitando-se de vulnerabilidades em protocolos de requisição. O protocolo mais comum é o HTTP, mas, internamente, a aplicação pode usar outros protocolos que também abrem espaço para outras vulnerabilidades, exemplos de outros protocolos são FTP, SMB, SMTP, entre outros.

Veja abaixo um exemplo de como este tipo de ataque pode ser realizado:

Imagine um cenário em que um sistema busca a informação de estoque de uma loja para exibir ao cliente, para isso ele utiliza a seguinte URL: `stockApi=http://stock.weliketoshop.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1`

Se não houver nenhuma forma de proteção um hacker pode fazer a seguinte modificação da URL: `stockApi=http://localhost/admin`

Isso faz com que os dados retornados sejam do /admin, que são informações que não devem ser tornadas públicas.

Como se proteger desses ataques?

- Evitar uso de entrada controlado pelo usuário diretamente em funções que podem fazer solicitações em nome do servidor;
- Garantir tratamento de resposta adequado;
- Desativar esquemas de URL não utilizados, pré definindo https por exemplo;
- Bloquear o envio de URL's completas pelo usuário;
- Aplicar whitelist de permissão contendo nomes de DNS ou IP's que sua aplicação precisará acessar, negando solicitações para qualquer outro destino;
- Verificar se o formato dos nomes de DNS ou IP's acessados são válidos;
- Exigir autenticação durante comunicação com serviços como Redis, Memcached, etc… para reduzir a superfície de ataque;
- Sanitizar a entrada do usuário, usando, por exemplo, expressões regulares para limitar o formato que a entrada aceita para um formato já esperado.