A8 – Cross-Site Request Forgery (CSRF)

Segurança de Aplicações

Leonardo Barbetta

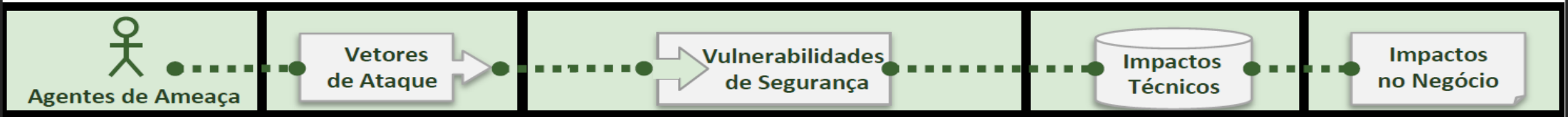
A8 – Cross-Site Request Forgery

O **CSRF** é um tipo de ataque malicioso a um site no qual comandos não autorizados são transmitidos através de um utilizador em quem o site confia., ou seja consiste na inserção de requisições em uma sessão já aberta pelo usuário.

Com esse tipo de ataque o atacante consegue inserir, alterar ou apagar dados de quem está sendo atacado.

A8 – Cross-Site Request Forgery - OWASP

A8 Cross-Site Request Forgery (CSRF)



Agentes de Ameaça	Vetores de Ataque	Vulnerabilidades de Segurança		Impactos Técnicos	Impactos no Negócio
Específico da Aplicação	**Exploração MÉDIO**	**Prevalência COMUM**	**Detecção FÁCIL**	**Impacto MODERADO**	**Específico do Negócio / Aplicação**
Considere qualquer pessoa que possa carregar conteúdo nos navegadores dos usuários, e assim forçá-los a fazer uma requisição para seu site. Qualquer site ou outro serviço html que usuários acessam pode fazer isso.	O atacante forja requisições HTTP falsas e engana uma vitima submetendo-a a um ataque através de tags de imagem, XSS, ou inúmeras outras técnicas. Se o usuário estiver autenticado, o ataque é bem sucedido.	O CSRF se aproveita do fato de que a maioria das aplicações web permitem que os atacantes prevejam todos os detalhes de uma ação particular da aplicação. Como os navegadores automaticamente trafegam credenciais como cookies de sessão, os atacantes podem criar páginas web maliciosas que geram requisições forjadas indistinguíveis das legítimas. Detecção de falhas de CSRF é bastante simples através de testes de penetração ou de análise de código.		Os atacantes podem enganar suas fazendo com que executem operações de mudança de estado que a vítima está autorizada a realizar, por ex., atualizando detalhes da sua conta, comprando, deslogando ou até mesmo efetuando login.	Considere o valor de negócio dos dados ou funções afetadas da aplicação. Imagine não ter a certeza se os usuários tem a intenção de realizar tais ações. Considere o impacto na sua reputação.

Vulnerabilidades

- Usuário que utiliza a mesma instancia do browser para navegar em sites malévolos;
- Manter-se logado em vários sistemas (não fechar as sessões);
- Não utilização de token de sessão para as requisições;
- Executar ações importantes via **GET**

Exemplo de ataque

Imagine que o usuário deseje cancelar uma compra, para isso ele cairá em uma tela de confirmação que irá solicitar o motivo para efetivação do cancelamento.

Código do formulário.

```
<form action="MeuSistema/Produto/CacelarCompra" method="post">

    <input type="hidden" name="compraId" value="15552">

    <textarea name="Motivo"></textarea>

    <input type="submit" value="Cancelar Compra">

</form>
```

Nesse formulário podemos notar que o id da compra é passado em um input hidden e caso um usuário mal intencionado altere o valor a compra errada será cancelada.

Exemplo de ataque

Vamos imaginar o seguinte cenário: Você está conversando em um chat com uma pessoa mal intencionada e você está com uma sessão aberta no site de um banco (esse site não possui proteção **CSRF**), a pessoa que você está conversando lhe envia um link que contem a tag de uma imagem, a página teria um código parecido com esse:

```
<img src="SiteDoBanco.com.br?ContaTransferencia=123456&ValorTranferencia=15000"/>
```

Ao abrir o link provavelmente você veria apenas uma página em branco, porém ao buscar o caminho da imagem foi feita uma requisição (via **GET**) de transferência para a conta 123456 no valor de R$15000;

Exemplo de defesa

Utilização de token de sessão por ação, ou seja em toda ação é criado um campo hidden que contém um hash que é valido apenas para aquela ação, daquele usuário, naquele momento, caso o usuário venha a fazer a ação uma segunda vez, outro hash será gerado, o que garante que cada requisição terá uma validação.

Exemplo de token:

```
<form action="MeuSistema/Produto/CacelarCompra" method="post">

    <input type="hidden" name="compraId" value="c91usclasGA0S49uarath0A1Iuzainguq7Tnuetser1ACGD7snA">
    <input type="hidden" name="CSRFToken" value="Eqfc91nA70ogh1ah0A0S4961Iuza6yeAIE4QRE871ACGD73OVxnUG0aOYhuq7Tnv09ktI73OVxuza6yedv8">

    <textarea name="Motivo"></textarea>

    <input type="submit" value="Cancelar Compra">

</form>
```

Outra forma de prevenir ataques **CSRF** é criptografando os dados passados via hidden e/ou os dados passados através de sessão.

Referencias

http://www.redesegura.com.br/2012/03/serie-ataques-os-ataques-cross-site-request-forgery-csrf/

https://seclab.stanford.edu/websec/csrf/

http://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html

http://www.veracode.com/security/csrf

http://blog.caelum.com.br/protegendo-sua-aplicacao-web-contra-cross-site-request-forgerycsrf/

https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)