Preventing Common Web Application Vulnerabilities

Mosh Hamedani

programmingwithmosh.com @moshhamedani

SQL Injection

Allows an attacker to execute malicious SQL statements in your application.

```
var sql = “select * from users where userId = “ + userId;

“1234 or 1 = 1”
```

SQL Injection

```
var sql = “select * from users where userId = @userId“
```

SQL Injection

```
_context.Gigs.Add(gig);
```

SQL Injection in GigHub

```
_context.Gigs.Add(gig);

“insert into Gigs (…) values (…)”
```

SQL Injection in GigHub

```
_context.Gigs.SqlQuery("select * …" + input);
```

SQL Injection in GigHub

Cross-site Scripting (XSS)

Enables an attacker to execute malicious script on the victim's computer.

Attacker

First Name

```
<script>…</script>
```

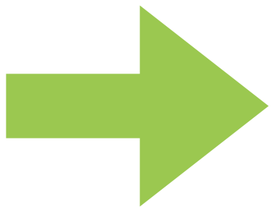
Victim

Execute Script

Escaping Content

Raw

`<script>`



Escaped

`&lt;script&gt;`

XSS in GigHub

ASP.NET MVC rejects Javascript in inputs

Razor views escape content

- Exception: Html.Raw()

Cross-site Request Forgery (CSRF)

Allows an attacker to perform actions on behalf a user without their knowledge.

How it Works

PayPal

paypal.com/transfer-money

```
POST paypal.com/transfer-money

targetAccount: 1234
amount: 100
```

How it Works

Malicious Page

```
POST paypal.com/transfer-money

targetAccount: 1234
amount: 100
```

How it Works

Malicious Page

```
POST paypal.com/transfer-money

targetAccount: 666
amount: 100
```

Cross-site Request Forgery (CSRF)

Allows an attacker to perform actions on behalf a user without their knowledge.

```
using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
}
```

```
[ValidateAntiForgeryToken]
public ActionResult Create()
```

Preventing CSRF Attacks

Summary

SQL Injection

Cross-site Scripting (XSS)

Cross-site Request Forgery (CSRF)

- View: @Html.AntiForgeryToken()
- Controller: [ValidateAntiForgeryToken]