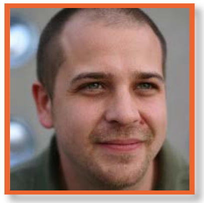
JavaScript/Browser-based Clients

Dominick Baier
http://leastprivilege.com
@leastprivilege



pluralsight
hardcore dev and IT training

Agenda

- Same Origin Policy
- Implicit Browser Authentication
- Cross Site Request Forgery (CSRF)
- Cross Origin Resource Sharing (CORS)

Same Origin Policy

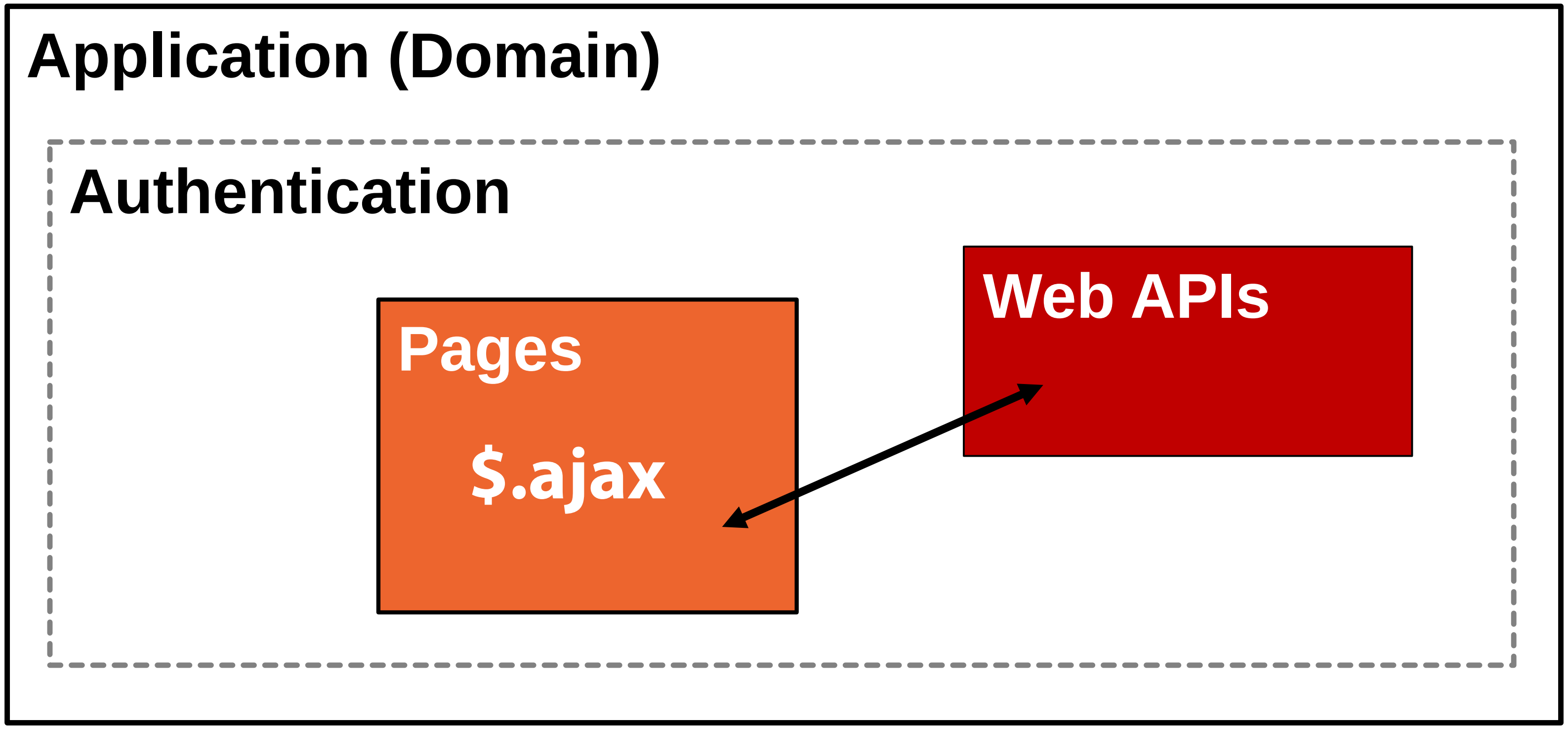
- **Sandbox mechanism**
  - affects scripts, communication, implicit browser authentication

e.g. https://www.example.com/customers/add

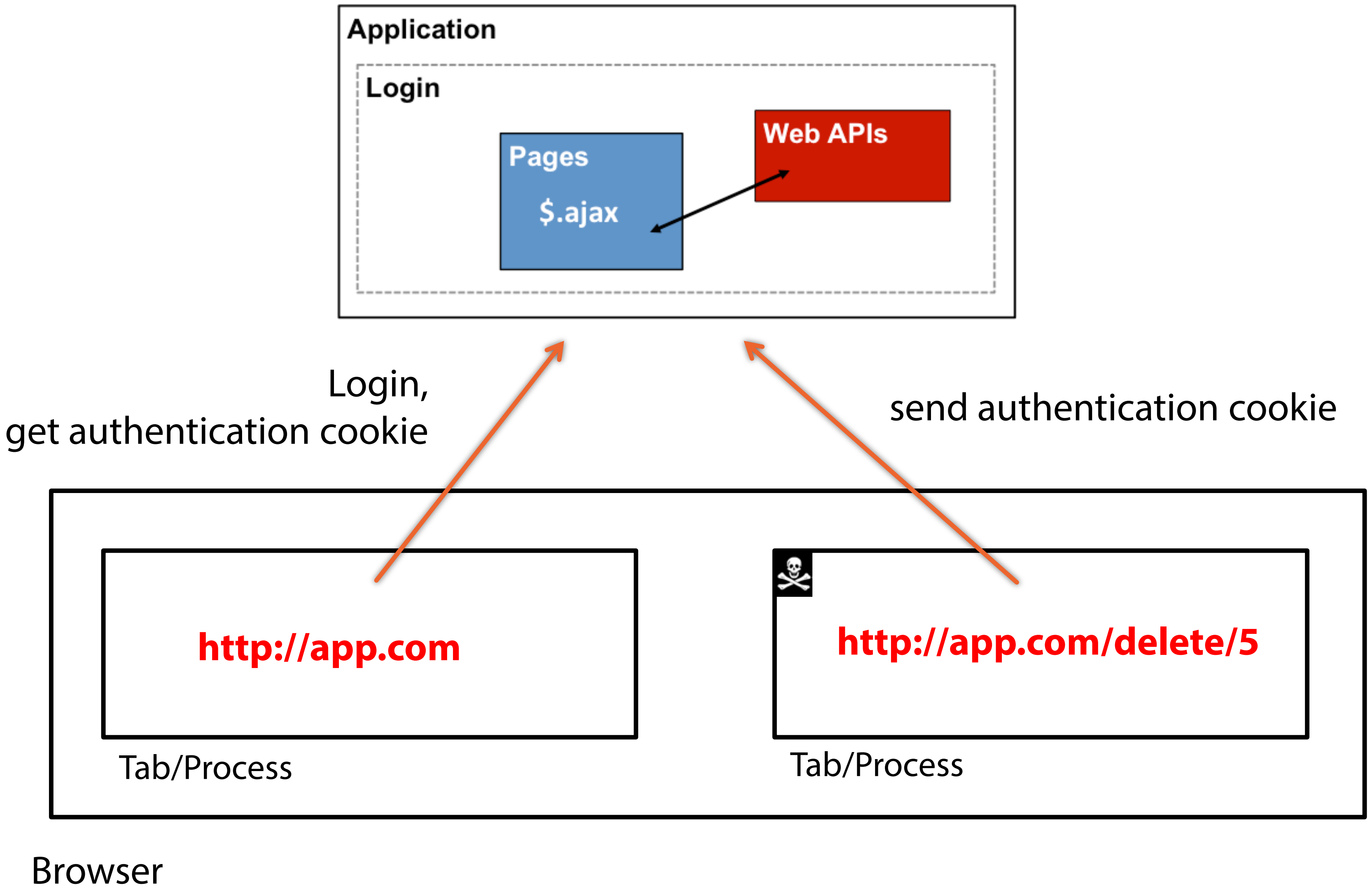
Compared URL	Outcome	Reason
https://www.example.com/api/customers	**Success**	Same protocol and host
https://www.example.com:444/api/customers	Failure	Different port
http://www.example.com/api/customers	Failure	Different protocol
https://example.com/api/customers	Failure	Different host
https://v2.www.example.com/api/customers	Failure	Different host

Using Same-Domain for Authentication

- **Web APIs inherit security settings of web host**
  - e.g. cookies, Windows/Basic authentication, client certs...

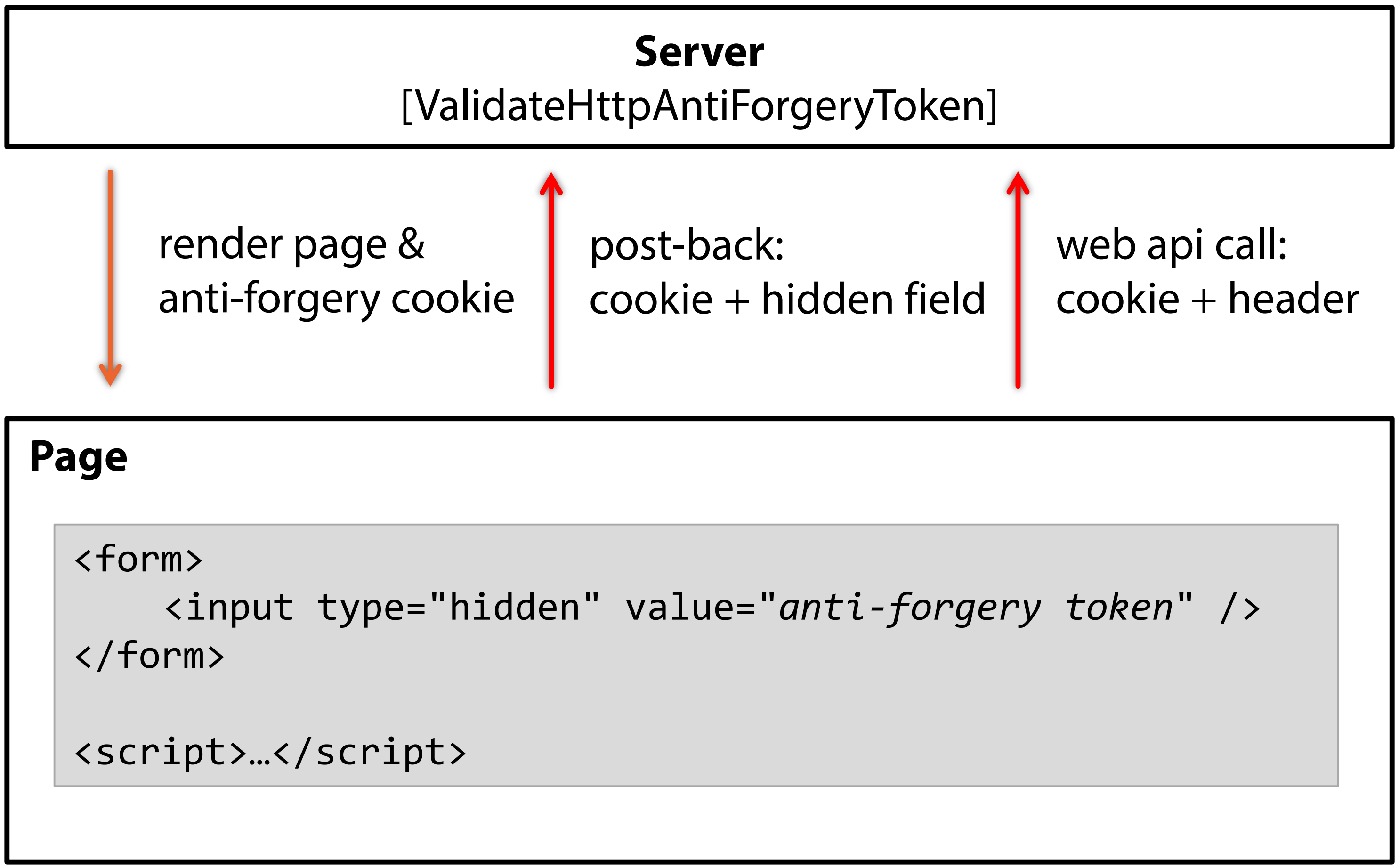


CSRF – The Problem



CSRF Protection – Web API v1 Approach

- Part of the SPA template in MVC 4 (Update 2)



Server
[ValidateHttpAntiForgeryToken]
render page & anti-forgery cookie
post-back: cookie + hidden field
web api call: cookie + header
Page

```
<form>
    <input type="hidden" value="anti-forgery token" />
</form>

<script>…</script>
```

CSRF Protection – Web API v2 Approach

```
// Configure Web API to use only bearer token authentication
config.SuppressDefaultHostAuthentication();

config.Filters.Add(new HostAuthenticationFilter(
  OAuthDefaults.AuthenticationType));
```

WebApiConfig.cs

```
protected override async Task<HttpResponseMessage> SendAsync(
  HttpRequestMessage request, CancellationToken cancellationToken)
{
  SetCurrentPrincipalToAnonymous(request);
  return await base.SendAsync(request, cancellationToken);
}
```

PassiveAuthenticationMessageHandler.cs

Cross Origin Resource Sharing

- **Same origin policy also used to restrict AJAX communication**
- **CORS is a W3C standard that allows relaxing those restrictions**
  - http://www.w3.org/TR/cors/
- **Web API has to opt-in to cross domain requests**
- **Not fully supported by all browsers**
  - http://caniuse.com/#search=cors

http://msdn.microsoft.com/en-us/magazine/dn532203.aspx

CORS Example I

- **Simple* CORS**

→

```
POST https://apiserver/resources/ HTTP/1.1

Host: apiserver
Accept: */*
Origin: https://appserver
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
…
```

←

```
HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8
Access-Control-Allow-Origin: https://appserver
…
```

***** GET or POST
application/x-www-form-urlencoded, multipart/form-data, text/plain
no additional request headers

CORS Example II

- **CORS with pre-flight request**

```
OPTIONS https://apiserver/resources/1 HTTP/1.1

Host: apiserver
Access-Control-Request-Method: PUT
Origin: https://appserver
Access-Control-Request-Headers: content-type
Accept: */*
```

```
HTTP/1.1 200 OK

Access-Control-Allow-Origin: https://appserver
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Headers: content-type
Access-Control-Max-Age: 600
```

Enabling CORS Support

Install-Package Microsoft.AspNet.WebApi.Cors

```
[EnableCors("origin", "headers", "verbs")]
public class CustomersController : ApiController
{
    // actions...
}
```

```
config.EnableCors();
```

WebApiConfig.cs

Summary

- **Browser based clients adhere to same origin policy**
- **“Classic“ AJAX/SPA type applications make use of implicit browser authentication**
  - this might lead to CSRF issues
- **Web API v1 used the anti-forgery token approach to mitigate CSRF**
- **Web API v2 tries discourages use of cookies altogether**
  - In favor of (explicit) token based authentication
- **CORS allows to relax cross domain communication restrictions**