DOTNEXT

ASP .NET Core: Preventing attacks 2.0

Mikhail Shcherbakov
Independent Developer and Consultant

Who am I

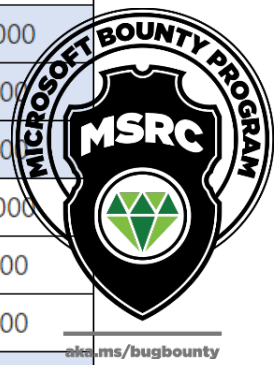
- Independent Developer and Consultant
- Co-organizer of .NET meetups http://dotnet.ru
- Public Speaker at DotNext, DotNetConf, ITGM, .NET meetups
- Former Product Manager at Cezurity, R&D Developer at Positive Technologies and Team Lead at Acronis, Luxoft, Boeing

What you can expect

- Preventing Open Redirect
- Data Protection
- Preventing XSS
- Setting up CSP
- Anti-Request Forgery
- Setting up CORS
- Using Cookies

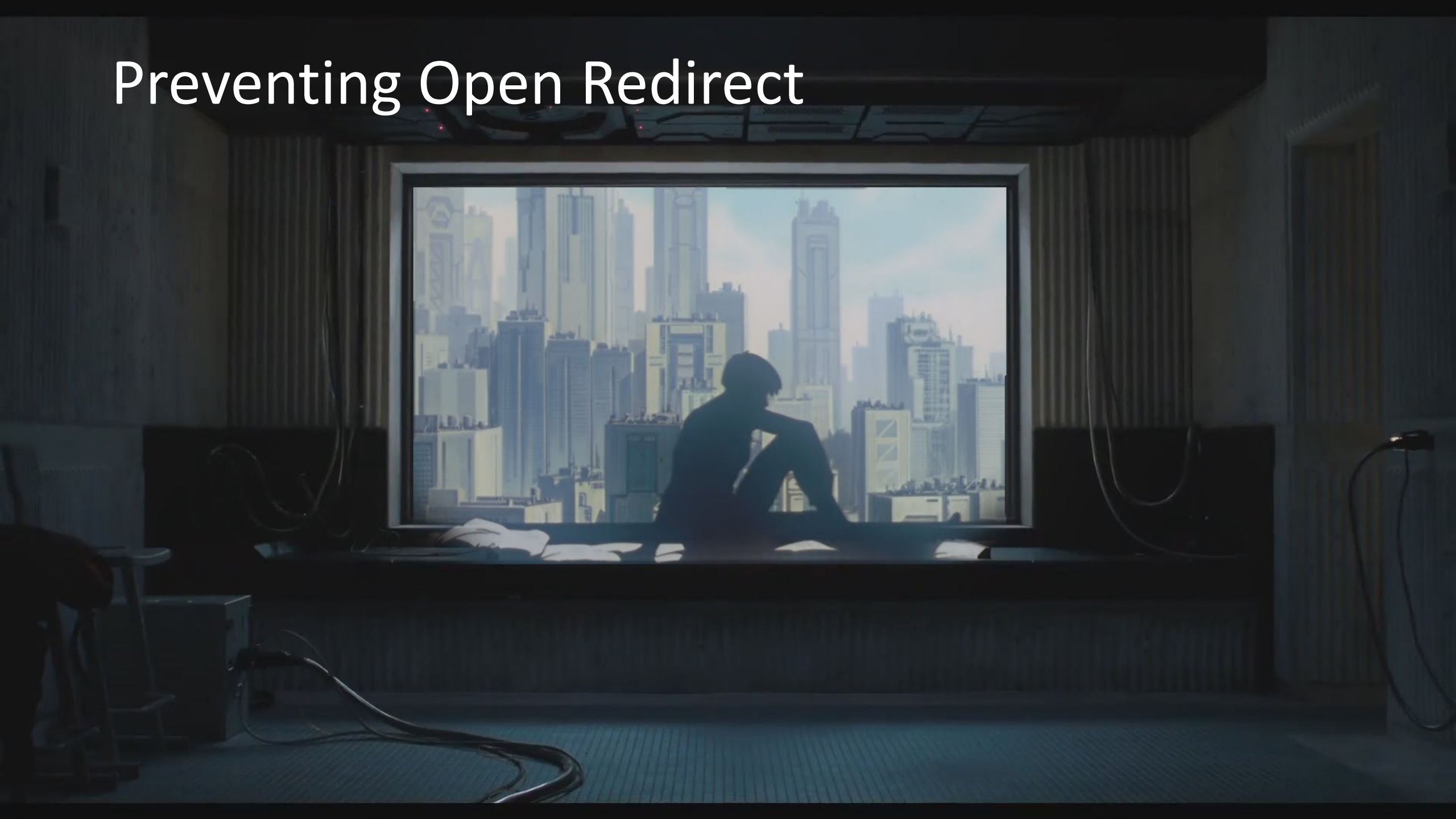
Microsoft Bug Bounty Program

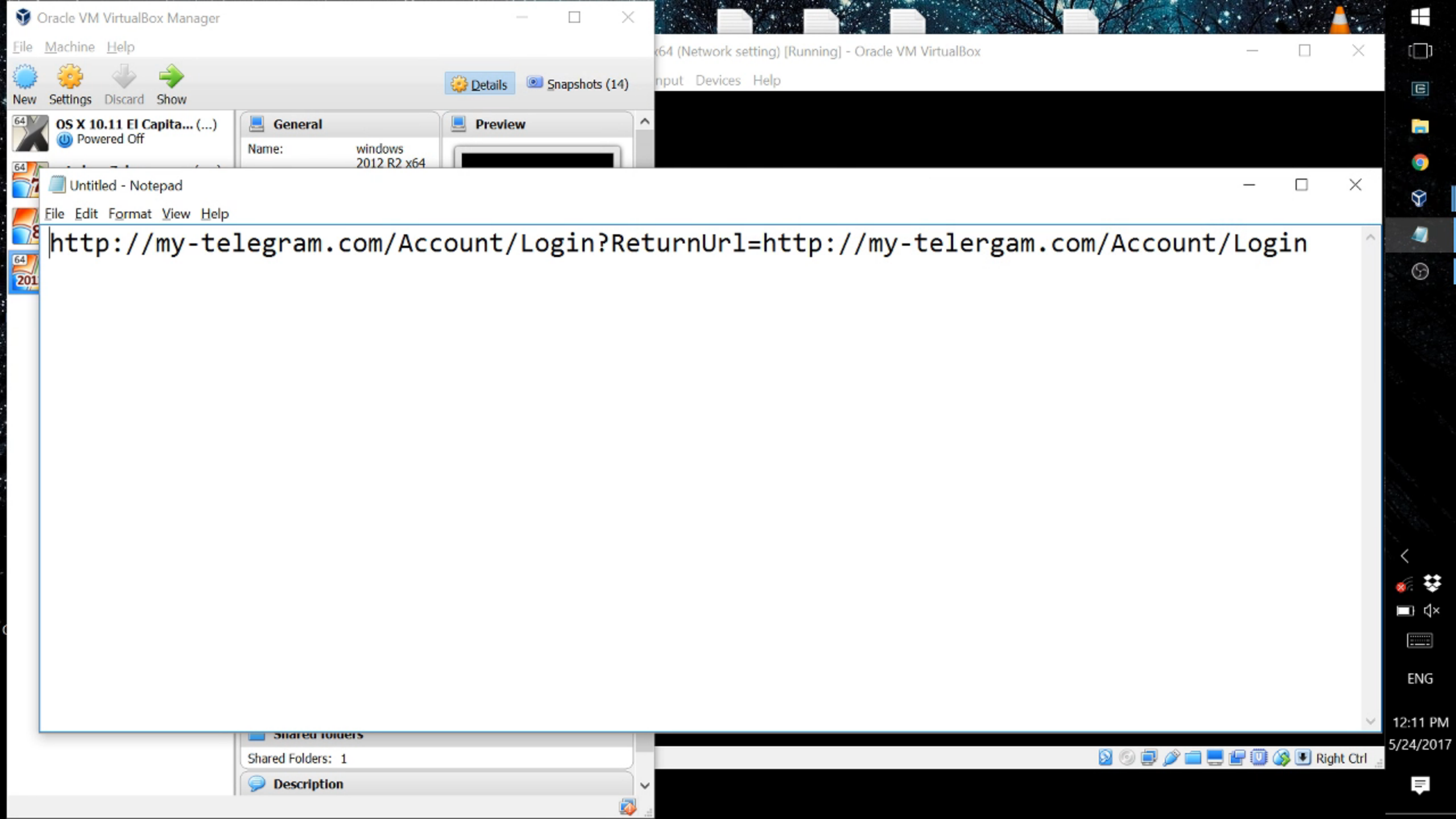
Vulnerability type	Proof of concept	Functioning Exploit	Whitepaper / Report Quality	Payout range (USD)
Remote Code Execution	Required	Required	High	Up to $15,000
	Required	No	High	Up to $6,000
	Required	No	Low	Up to $1,500
Security Design Flaw	Required	Required	High	Up to $10,000
	Required	Optional	High	Up to $5,000
	Required	No	Low	Up to $1,500
Elevation of Privilege	Required	Required	High	Up to $10,000
	Required	No	Low	Up to $5,000
Remote DoS	Required	Optional	High	Up to $5,000
	Required	No	Low	Up to $2,500
Tampering / Spoofing	Required	Optional	High	Up to $5,000
	Required	No	Low	Up to $2,500
Information Leaks	Required	Optional	High	Up to $2,500
	Required	No	Low	Up to $750
Template CSRF or XSS	Required	Optional	High	Up to $2,000
	Required	Optional	Low	Up to $500



https://aka.ms/corebounty

Preventing Open Redirect




Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Show
Details Snapshots (14)
OS X 10.11 El Capita... (...)
Powered Off
General
Name: windows 2012 R2 x64
Preview
Shared Folders: 1
Description
x64 (Network setting) [Running] - Oracle VM VirtualBox
nput Devices Help
Untitled - Notepad
File Edit Format View Help
http://my-telegram.com/Account/Login?ReturnUrl=http://my-telergam.com/Account/Login
Right Ctrl
ENG
12:11 PM
5/24/2017

How to prevent

```
public async Task<IActionResult> Login(LoginViewModel model,
    string returnUrl)
{
    // ...
    return LocalRedirect(returnUrl);
}
```

How to prevent

```
private IActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
        return Redirect(returnUrl);
    else
        return RedirectToAction(
            nameof(HomeController.Index), "Home");
}
```

Open Redirect Attack

14	What is your primary assessment methodology?	Totals	Ou
15	Number of Cross-Site Scripting (XSS) Vulnerabilities Found (CWE-79)?	1923423	
16	Number of SQL Injection Vulnerabilities Found (CWE-89)?	182810	
17	Number of Unchecked Redirect Vulnerabilities Found (CWE-601)?	57177	
18	Number of XML eXternal Entity Injection (XXE) Vulnerabilities Found (CWE-611)?	42337	
19	Number of Path Traversal Vulnerabilities Found (CWE-22)?	12382	
20	Number of Security Misconfiguration Vulnerabilities Found (CWE-2)?	11841	
21	Number of Cryptographic Vulnerabilities Found (CWEs-310/326/327/etc)?	9692	
22	Number of Command Injection Vulnerabilities Found (CWE-77)?	8021	
36	Number of Cross-Site Request Forgery (CSRF) Vulnerabilities Found (CWE-352)?	1310	
37	Number of Cleartext Storage of Sensitive Information Vulnerabilities Found (CWE-312)?	1293	
38	Number of Insufficient Security Logging Vulnerabilities Found (CWE-778)?	990	
39	Number of Unvalidated Forward Vulnerabilities Found (No CWE)?	831	
40	Number of Insufficient Anti-automation Vulnerabilities Found (CWE-799)?	732	

https://github.com/OWASP/Top10/tree/master/2017/datacall

Microsoft Security Advisory 4021279

Vulnerabilities in .NET Core, ASP.NET Core Could Allow Elevation of Privilege

Published: May 9, 2017 | Updated: May 10, 2017

Issue CVEs and Description

CVE	Description
CVE-2017-0247	Denial of Service
CVE-2017-0248	Security Feature Bypass
CVE-2017-0249	Elevation of Privilege
CVE-2017-0256	Spoofing

Acknowledgments

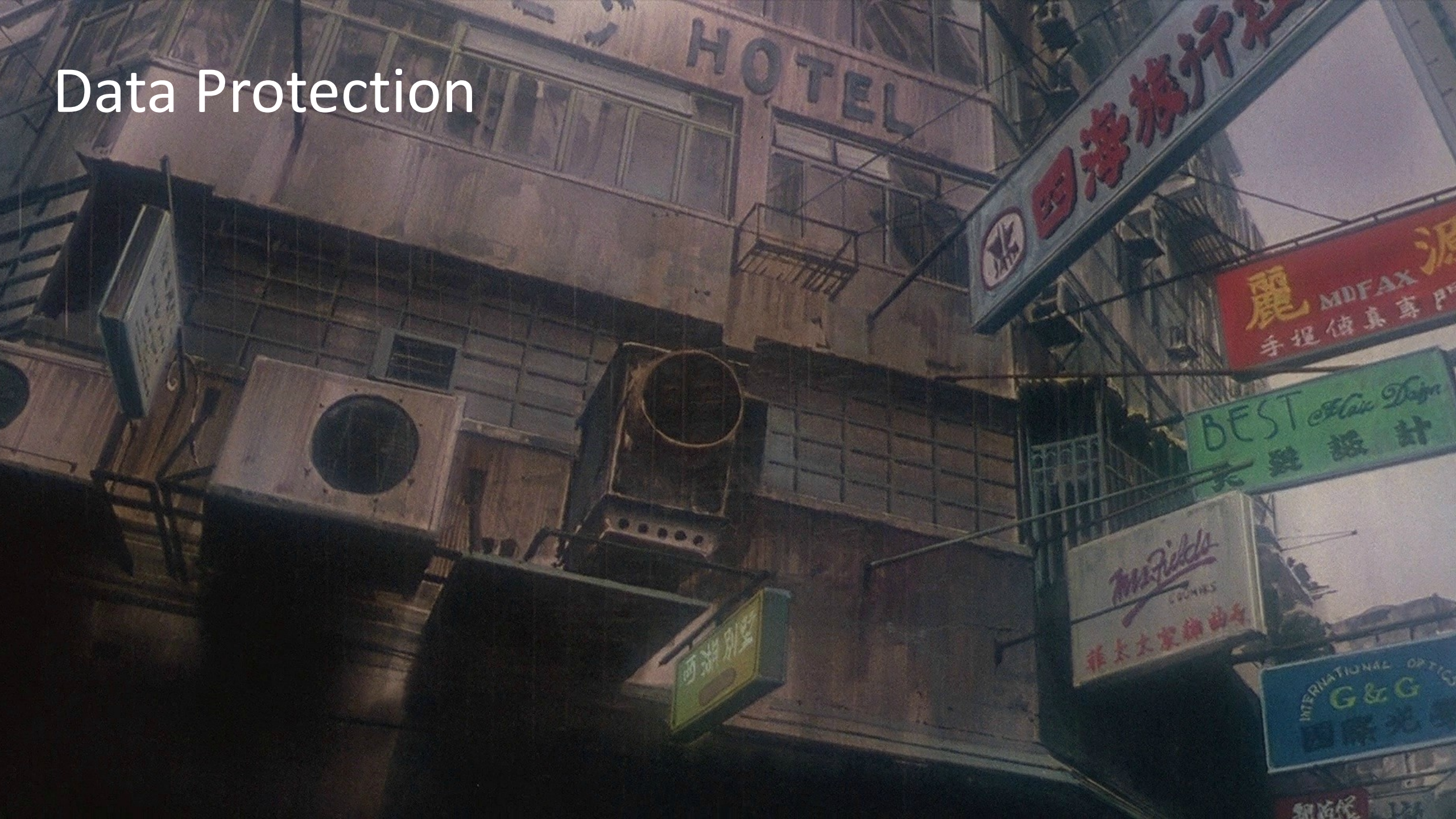
Microsoft thanks the following for working with us to help protect customers:

- David Fernandez of Sidertia Solutions for reporting the ASP.NET Core Denial of Service Vulnerability (CVE-2017-0247)
- Mikhail Shcherbakov for reporting the ASP.NET Core Spoofing Vulnerability (CVE-2017-0256)

Disclaimer

https://technet.microsoft.com/library/security/4021279

Data Protection



Overview

- No machine keys
- High level cryptography out-of-the-box
- Key stores out-of-the-box
- Ability to extend key stores
- Supports key rotation automatically
- Provides isolation automatically
- Custom algorithms supported

Protect / Unprotect

```
public class HomeController : Controller
{
    private readonly IDataProtector protector;

    public HomeController(IDataProtectionProvider provider)
    {
        protector = provider.CreateProtector("isolation-purpose");
    }

    public IActionResult Index(string input)
    {
        var protectedPayload = protector.Protect(input);
        var unprotectedPayload = protector.Unprotect(protectedPayload);
        return View();
    }
}
```

Protect / Unprotect

```
public IActionResult Index(string input)
{
    var timeLimitedProtector = protector.ToTimeLimitedDataProtector();

    var protectedData = timeLimitedProtector.Protect(input,
        lifetime: TimeSpan.FromMinutes(30));

    return View();
}
```

Password Hashing

```
public void StorePassword(SecureString password)
{
    var salt = new byte[128 / 8];
    using (var rng = RandomNumberGenerator.Create())
    {
        rng.GetBytes(salt);
    }

    var hash = Convert.ToBase64String(KeyDerivation.Pbkdf2(
        password: password,
        salt: salt,
        prf: KeyDerivationPrf.HMACSHA512,
        iterationCount: 10000,
        numBytesRequested: 256 / 8));

    // store salt and hash to DB...
}
```

Configuration

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDataProtection()
        .SetApplicationName("isolation-name")
        .SetDefaultKeyLifetime(TimeSpan.FromDays(14))
        .PersistKeysToFileSystem(new DirectoryInfo(@"\\server\share\"))
        .ProtectKeysWithDpapi()
        .UseCryptographicAlgorithms(new AuthenticatedEncryptionSettings
        {
            EncryptionAlgorithm = EncryptionAlgorithm.AES_256_CBC,
            ValidationAlgorithm = ValidationAlgorithm.HMACSHA256
        });
}
```

Configuration

- HKLM\SOFTWARE\Microsoft\DotNetPackages\Microsoft.AspNetCore.DataProtection
  - EncryptionType
  - DefaultKeyLifetime
  - KeyEscrowSinks

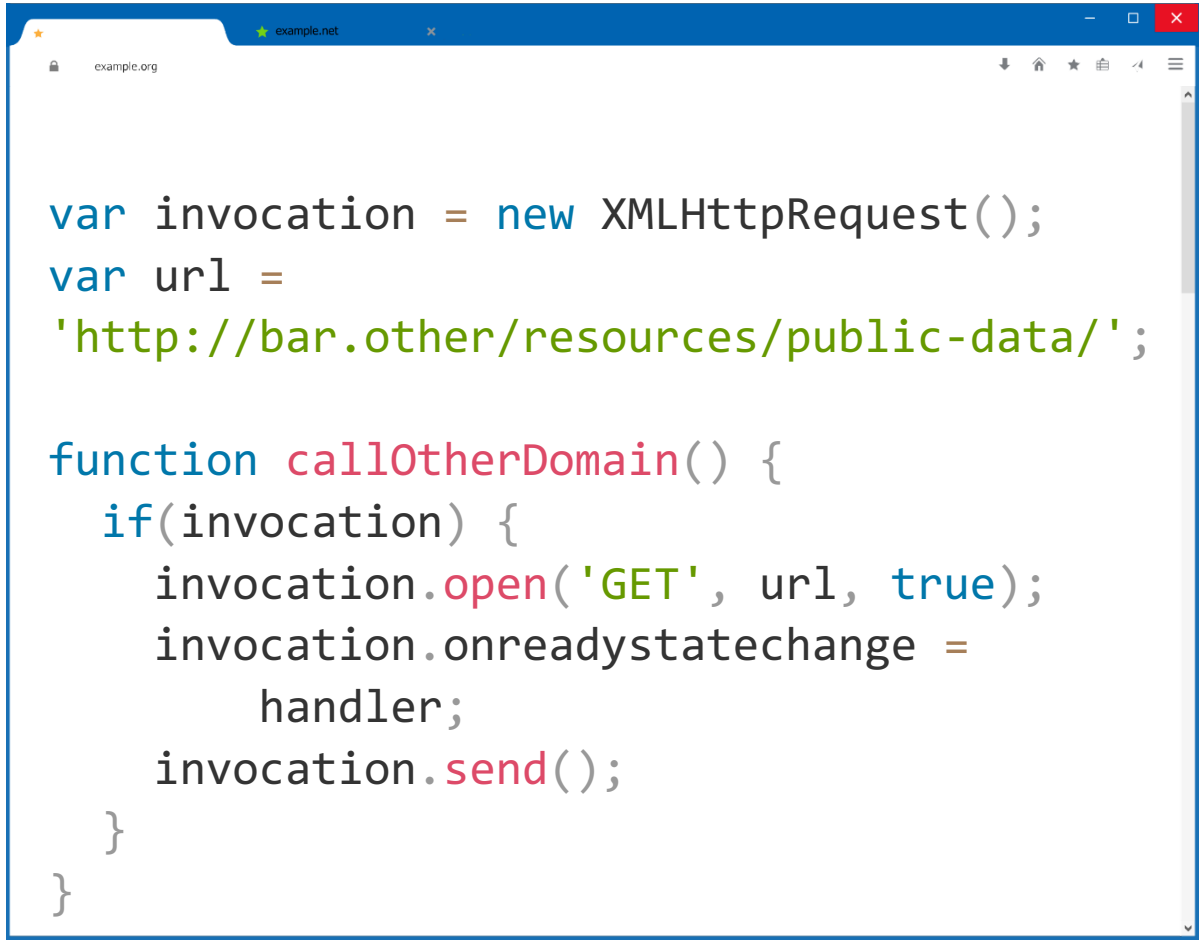
Details

- The default payload protection algorithm used is AES-256-CBC for confidentiality and HMACSHA256 for authenticity. A 512-bit master key, rolled every 90 days
- Protected payload format
  - 32-bit magic header
  - 128-bit key id
  - the part of specific to the encryptor

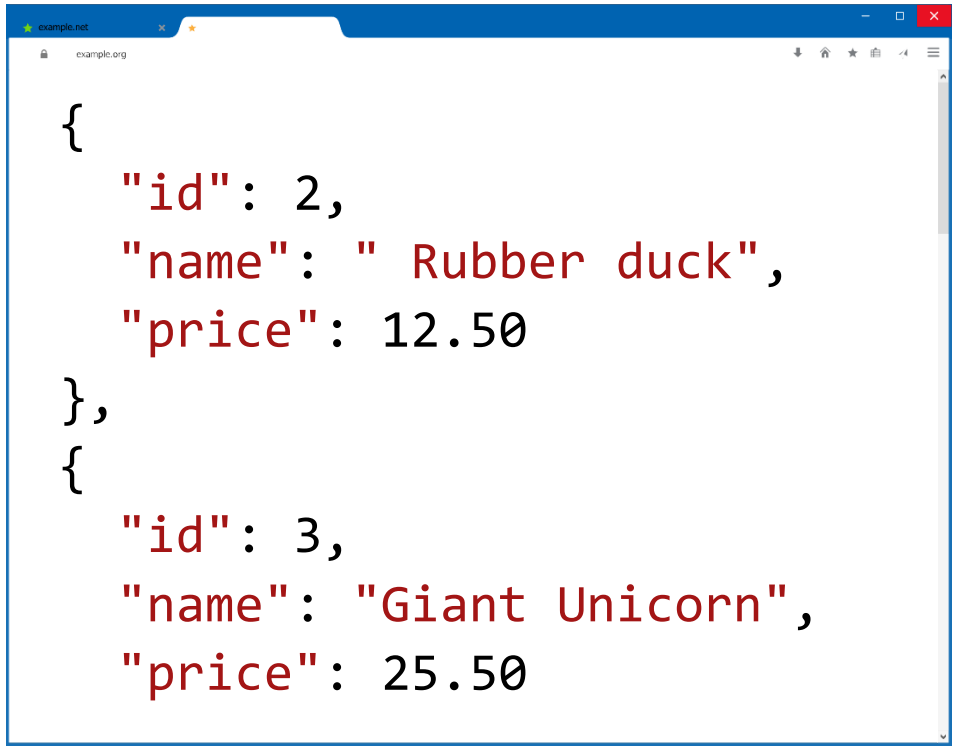
Preventing XSS

Same-origin policy (SOP)

https://foo.example



http://bar.other



Same-origin policy (SOP)

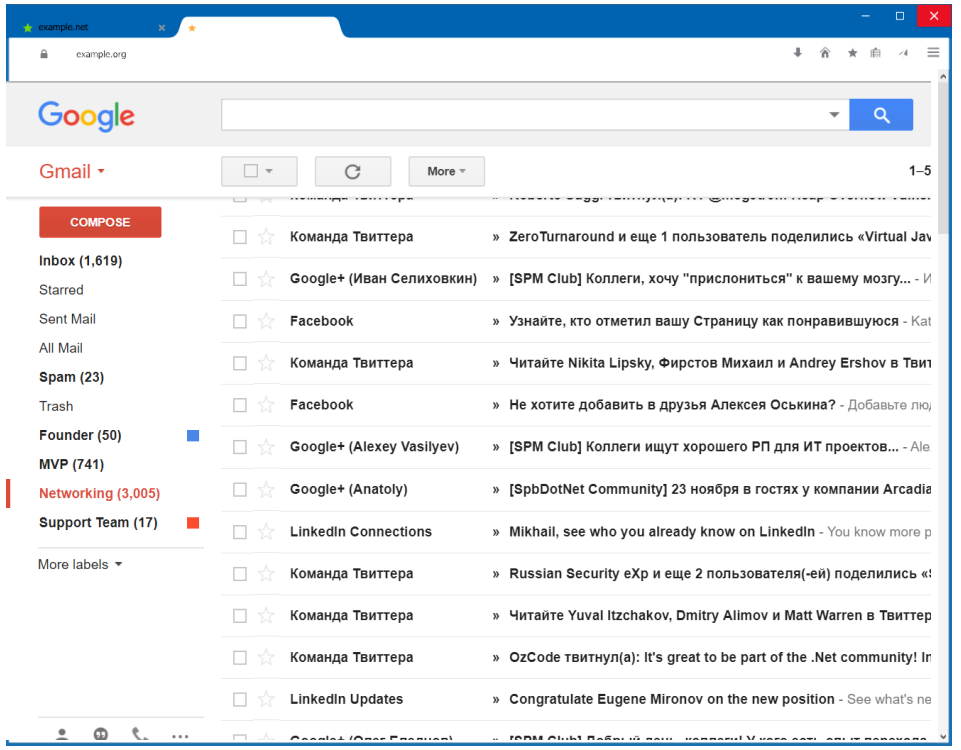
https://evil.com

example.net

example.org

```
var invocation = new XMLHttpRequest();
var url =
'https://mail.google.com/mail/u/0/';

function callOtherDomain() {
  if(invocation) {
    invocation.open('GET', url, true);
    invocation.onreadystatechange =
        handler;
    invocation.send();
  }
}
```

https://mail.google.com/



Same-origin policy (SOP)

The same-origin policy (SOP) **restricts** how a document or script loaded from **one origin** can **interact** with a resource from **another origin**. It is a critical security mechanism for isolating potentially malicious documents.

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

Definition of an origin

- URI scheme *(http)*
- Hostname *(my-domain.com)*
- Port number *(80)*

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) attacks are a **type of injection**, in which **malicious scripts** are injected into otherwise benign and **trusted web sites**.

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Cross-Site Scripting (XSS)

- The XSS is the most popular attack to web applications
  - https://github.com/OWASP/Top10/tree/master/2017/datacall
- This is a good entry point for other attacks with more impact
- Only some people know and correctly use built-in XSS prevention mechanisms

Potential XSS

```
<script src="@ViewData["UntrustedInput"]">
</script>
```

Potential XSS

```
<script>
    @ViewData["UntrustedInput"];
</script>

...

<img src="@ViewData["UntrustedInput"];">
```

How to prevent

- Input data validation and filtering
- Output sanitization for HTML, JavaScript, XML, SVG using sanitization libs

Encoding points

- HTML tags
- HTML attributes
- JavaScript
- URL
- XML
- SVG

Encoding points

- HtmlEncoder (*@<member>* in .cshtml)
- JavaScriptEncoder
- UrlEncoder
- Any sanitizer (e.g., https://github.com/mganss/HtmlSanitizer or https://github.com/LibProtection/libprotection-dotnet)

Potential XSS

```
<script src="@ViewData["UntrustedInput"]">
</script>

<script>
    @ViewData["UntrustedInput"];
</script>
```

Content Security Policy (CSP)

```
app.UseMvc();

app.Use(async (context, next) =>
{
    context.Response.Headers.Add("Content-Security-Policy",
        "default-src 'self'; report-uri /cspreport");
    await next();
});
```

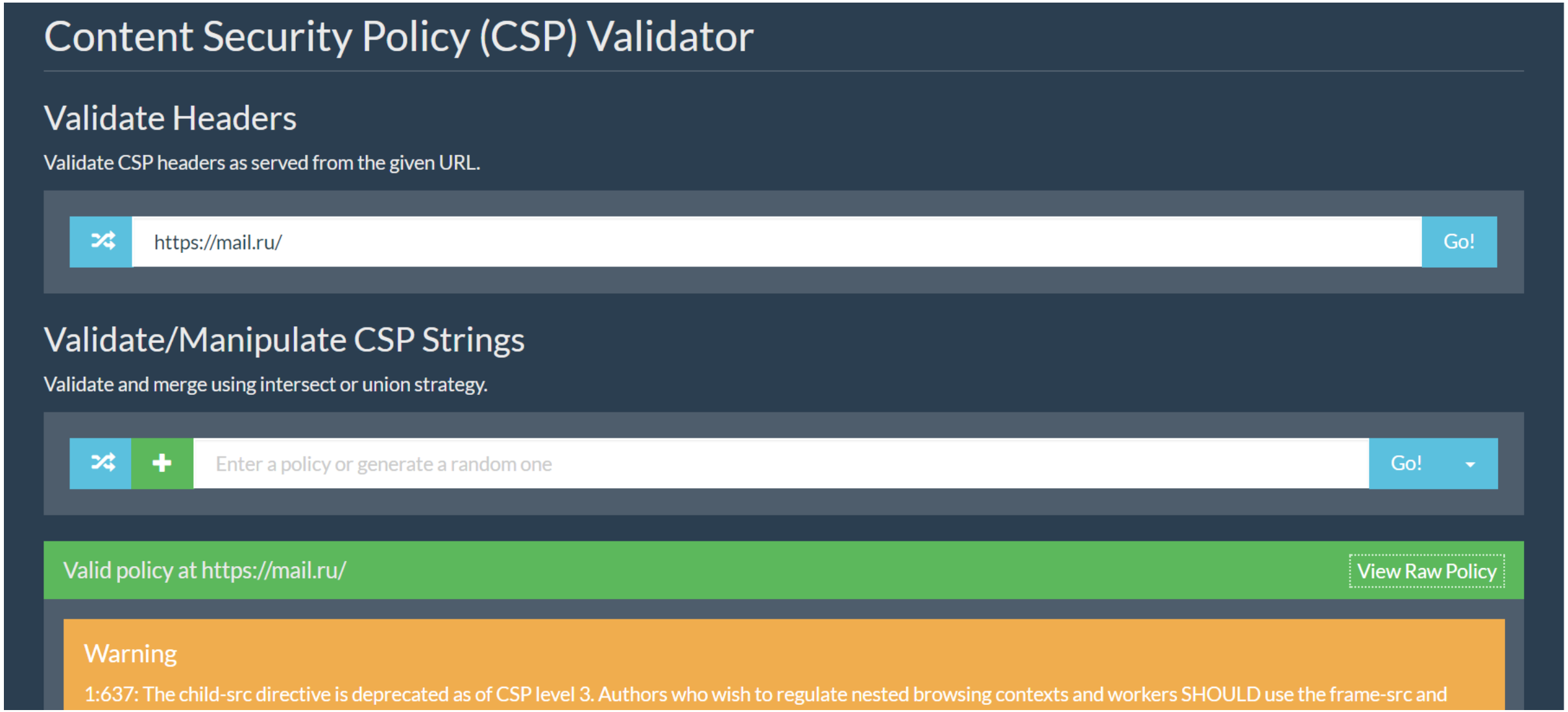
CSP Report Request

```
public class CspReportRequest
{
    [JsonProperty(PropertyName = "csp-report")]
    public CspReport CspReport { get; set; }
}

public class CspReport
{
    [JsonProperty(PropertyName = "document-uri")] public string DocumentUri { get; set; }
    [JsonProperty(PropertyName = "referrer")] public string Referrer { get; set; }
    [JsonProperty(PropertyName = "violated-directive")] public string ViolatedDirective { get; set; }
    [JsonProperty(PropertyName = "effective-directive")] public string EffectiveDirective {get; set;}
    [JsonProperty(PropertyName = "original-policy")] public string OriginalPolicy { get; set; }
    [JsonProperty(PropertyName = "blocked-uri")] public string BlockedUri { get; set; }
    [JsonProperty(PropertyName = "status-code")] public int StatusCode { get; set; }
}
```

CSP Report Endpoint

```
[HttpPost("~/cspreport")]
public IActionResult CspReport([FromBody] CspReportRequest request)
{
    // log and analyze the report...
    return Ok();
}
```

https://cspvalidator.org/



Bypass

```
<base href='http://evil.com/'>
```

```
<form method="post" class="form-horizontal"
    action="/Account/Login">
    <h4>Use a local account to log in.</h4>
    <input type="email" id="Email" name="Email" value="" />
    <input type="password" id="Password" name="Password" />
    <button type="submit">Log in</button>
    <input name="__RequestVerificationToken" type="hidden"
        value="CfDJ8MNYtGIQJ0NKvmDVDgK_YQ1W616zRzP4KVx5Pud1Z0kwNgsP4fkkzW3kapQF2i
</form>
```

x-xss-protection

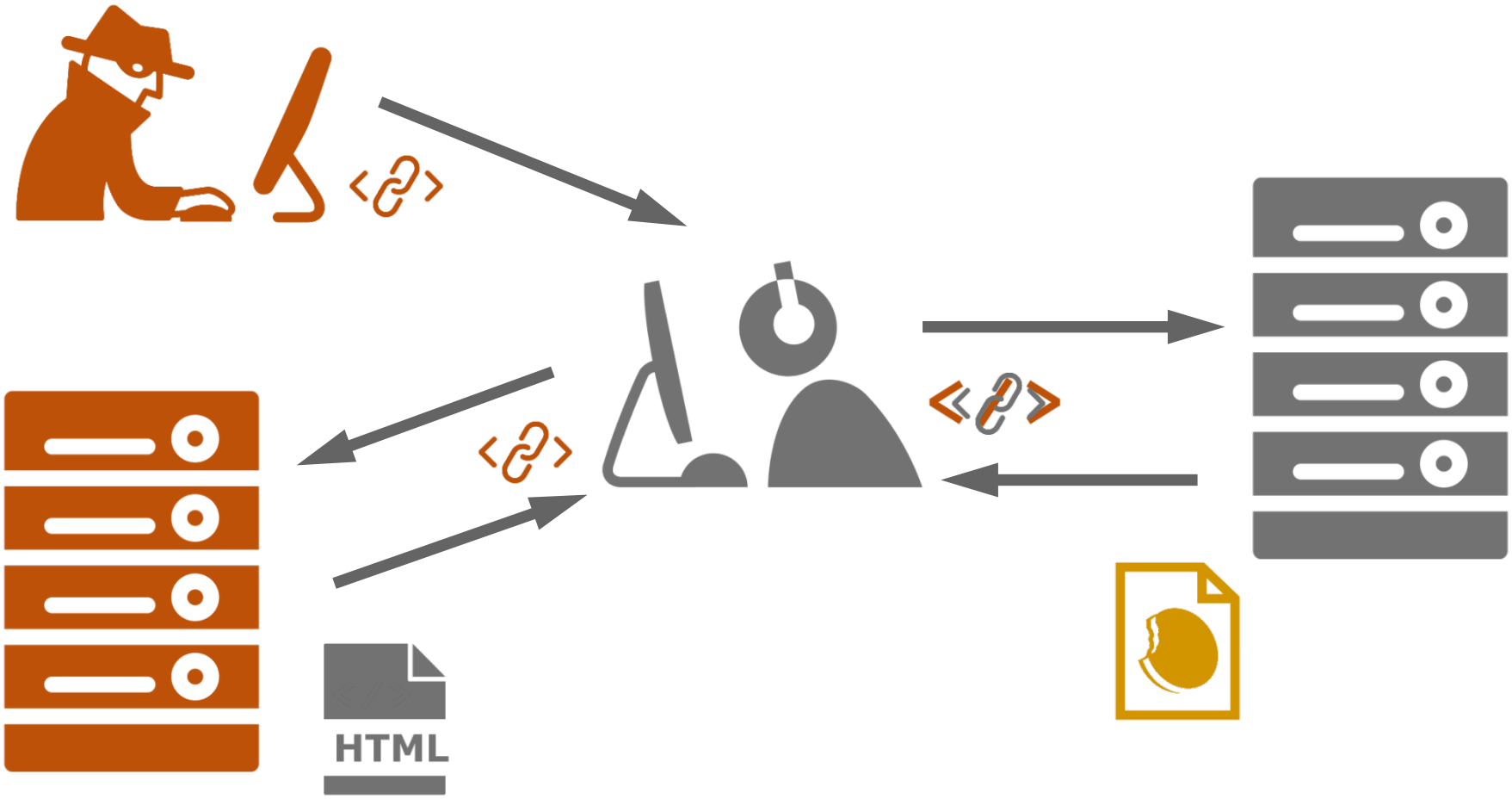
```
app.UseMvc();

app.Use(async (context, next) =>
{
    context.Response.Headers.Add("x-xss-protection", "1");
    await next();
});
```

How to prevent

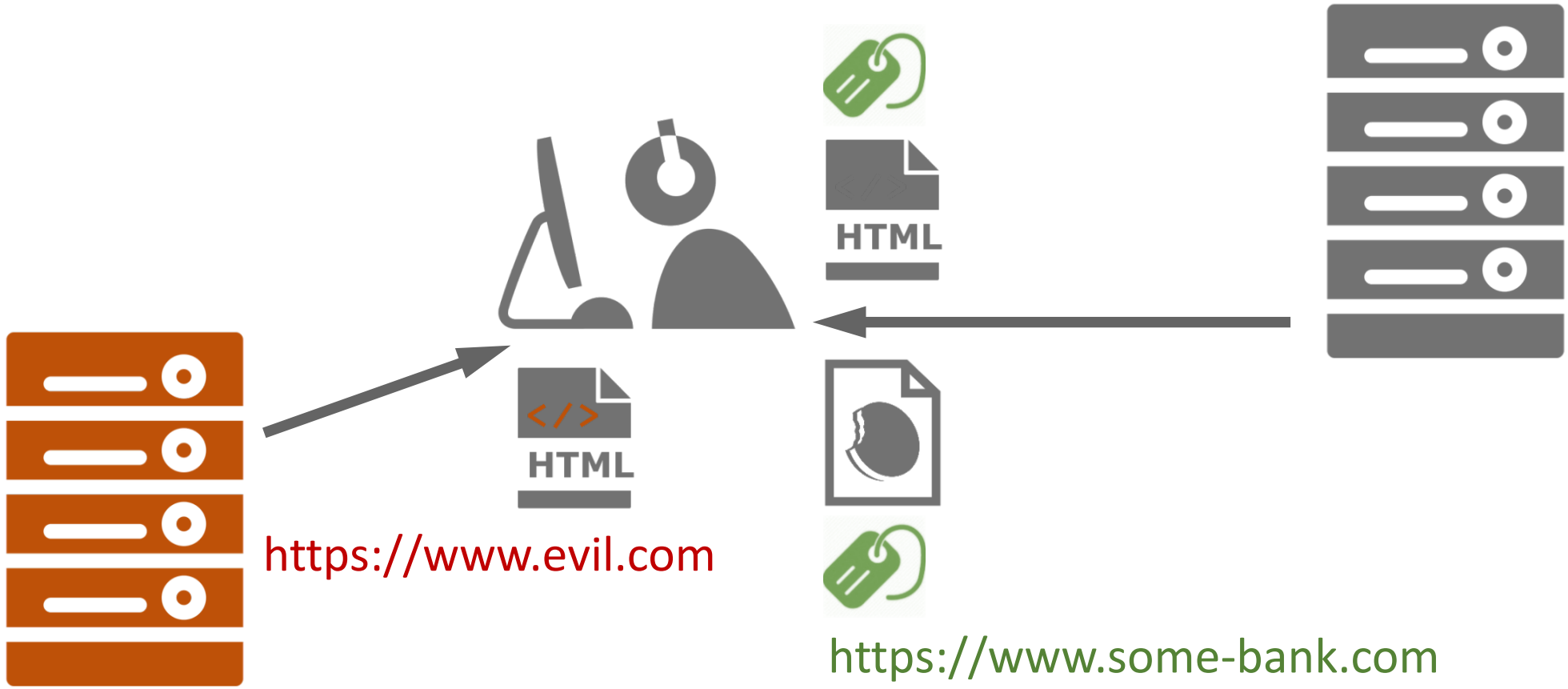
- Input data validation and filtering
- Output sanitization for HTML, JavaScript, XML, SVG
- *Good practice: using Content Security Policy (CSP)*
- *Good practice: using x-xss-protection*

Anti-Request Forgery

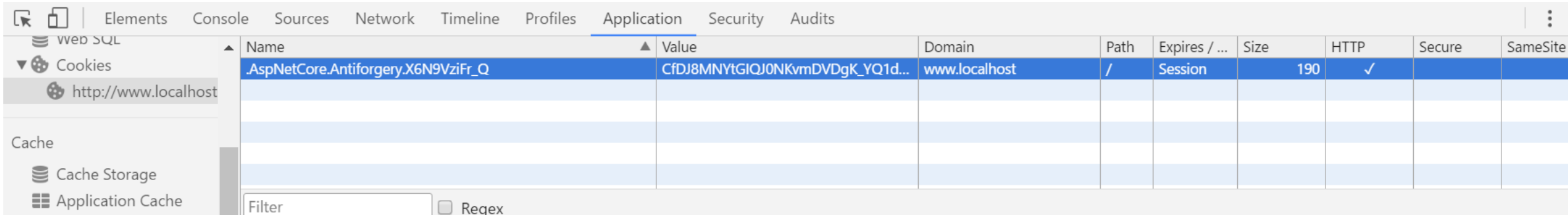
Cross-Site Request Forgery



Double-Submit Cookie Pattern



Double-Submit Cookie Pattern




```
<form method="post" action="/Home/MyAction">
    <div>...</div>
    <input name="__RequestVerificationToken" type="hidden"
           value="CfDJ8MNYtGIQJ0NKvmDVDgK_YQ2alNtW7VHnQAVGEoKzZhHQgrj0A0o8L8s904
</form>
```

https://github.com/aspnet/Antiforgery

New Token

```
private static readonly RandomNumberGenerator
    randomNumberGenerator = RandomNumberGenerator.Create();


private static byte[] GenerateNewToken(int bitLength)
{
    var data = new byte[bitLength / 8];
    randomNumberGenerator.GetBytes(data);
    return data;
}
```

New Token

```
cryptoSystem = provider.CreateProtector(
  "Microsoft.AspNetCore.Antiforgery.AntiforgeryToken.v1");

// ...

var bytes = cryptoSystem.Protect(stream.ToArray());
```

AntiForgery Token

```
<form asp-controller="Account" asp-action="Login“ method="post"
      asp-route-returnurl="@ViewData["ReturnUrl"]" class="form-horizontal">
    <h4>Use a local account to log in.</h4>
    <div asp-validation-summary="All" class="text-danger"></div>
    <div class="form-group">...</div>
</form>
```

```
<form method="post" class="form-horizontal" action="/Account/Login">
    <h4>Use a local account to log in.</h4>
    <div class="text-danger validation-summary-valid" data-valmsg-summary="true">
        <ul><li style="display: none"></li></ul>
    </div>
    <div class="form-group">...</div>
    <input name="__RequestVerificationToken" type="hidden"
           value="CfDJ8MNYtGIQJ0NKvmDVDgK_YQ2alNtW7VHnQAVGEoKzZhHQgrj0A0o8L8s9049_Z1ELltvpTkCt978aCpj1q9DQyb_XIsQ4
</form>
```

Enable Auto Validation

```
public class Startup
{
    public void ConfigureServices(
        IServiceCollection services)
    {
        services.AddMvc(options =>
          options.Filters.Add(
            new AutoValidateAntiforgeryTokenAttribute()));
    }
}
```

Attributes

```
[AutoValidateAntiForgeryToken]
public class CustomController : Controller
{
    // some methods…
}
```

An antiforgery token is required for HTTP methods other than:

- GET
- HEAD
- OPTIONS
- TRACE

Attributes

```
[ValidateAntiForgeryToken]
public class CustomController : Controller
{
    [HttpPost]
    [IgnoreAntiforgeryToken]
    public IActionResult DoSomething(SomeViewModel model)
    {
        // no antiforgery token required
    }
}
```

AJAX, WebAPI, SPAs…

- Do you use authentication cookies?
  - No cookies, no problems… except for stealing a token by XSS ☺ For example, you may use JSON Web Token (JWT)
  - Yes, I do… Go to next page

AJAX, WebAPI, SPAs...

```
@inject Microsoft.AspNetCore.Antiforgery.IAntiforgery Xsrf
@functions{
    public string GetAntiXsrfRequestToken()
    {
        return Xsrf.GetAndStoreTokens(Context).RequestToken;
    }
}

<input type="button" id="antiforgery" value="Antiforgery" />
<script>
    $("#antiforgery").click(function () {
        $.ajax({
            type: "post",
            dataType: "html",
            headers:
            {
                "RequestVerificationToken": '@GetAntiXsrfRequestToken()'
            },
            url: '@Url.Action("Antiforgery", "Home")',
        });
    });
```

AngularJS

```
HttpContext.Response.Cookies.Append("XSRF-TOKEN",
    antiforgery.GetAndStoreTokens(HttpContext).RequestToken,
    new Microsoft.AspNetCore.Http.CookieOptions
    {
        HttpOnly = false
    });
```

```
services.AddAntiforgery(options =>
    options.HeaderName = "X-XSRF-TOKEN");
```

Cross-Origin Requests

Cross-Origin Requests

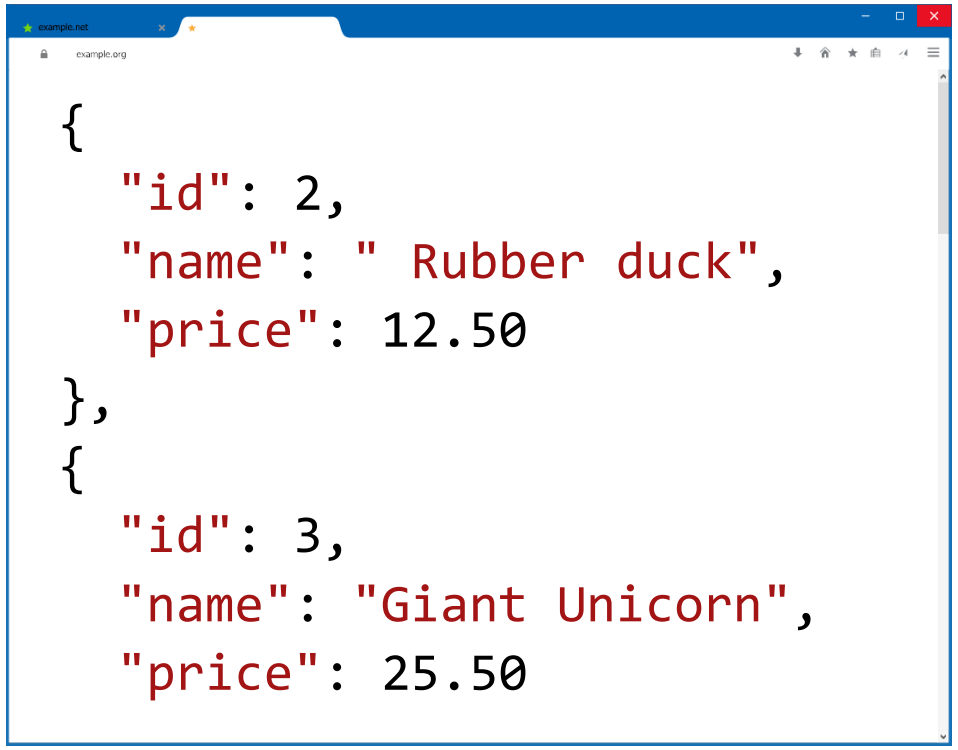
https://foo.example

example.net

example.org

```
var invocation = new XMLHttpRequest();
var url =
'https://bar.other/resources/public-data/';

function callOtherDomain() {
  if(invocation) {
    invocation.open('GET', url, true);
    invocation.onreadystatechange =
        handler;
    invocation.send();
  }
}
```

https://bar.other



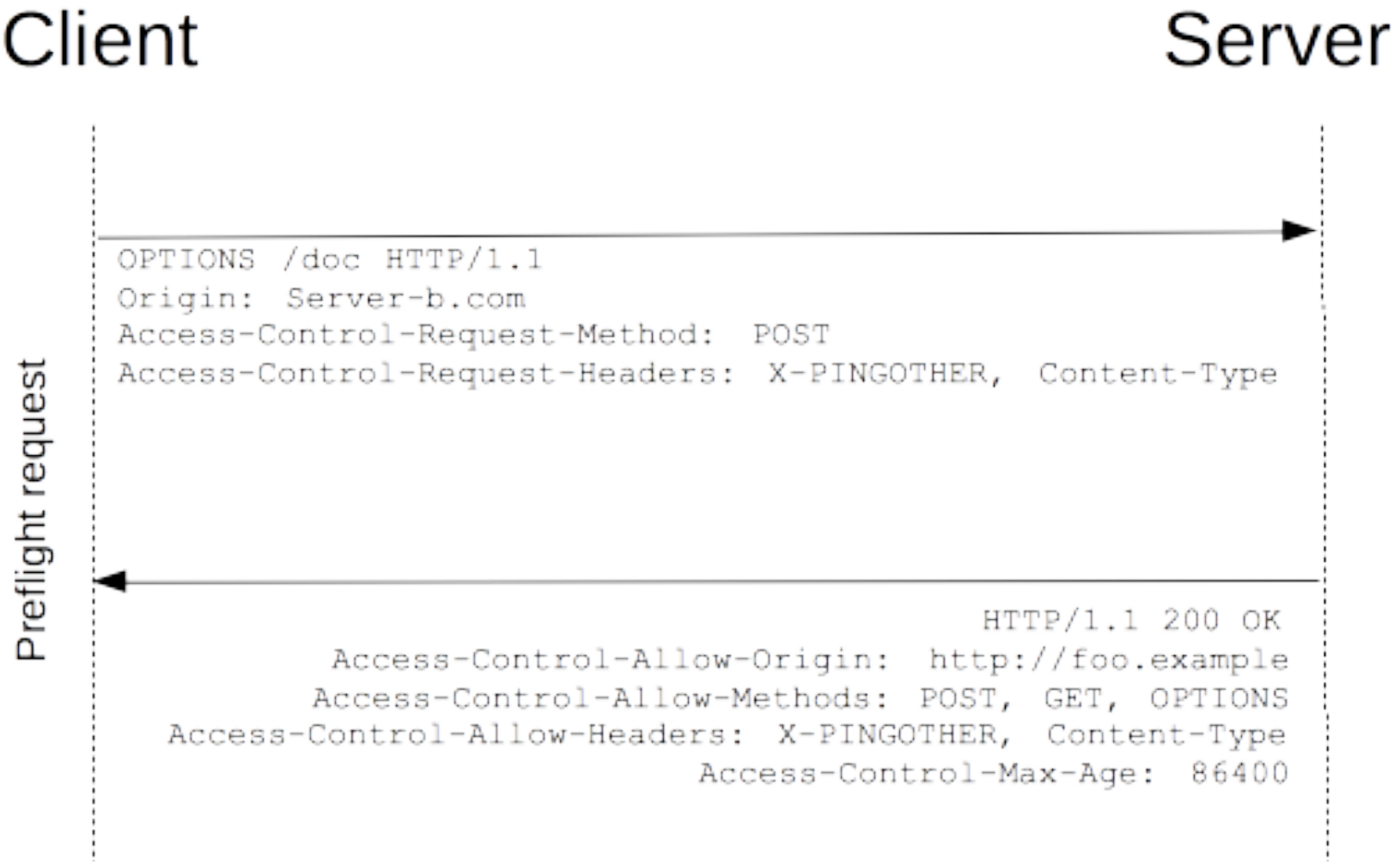
Cross-Origin Resource Sharing (CORS)

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Simple requests

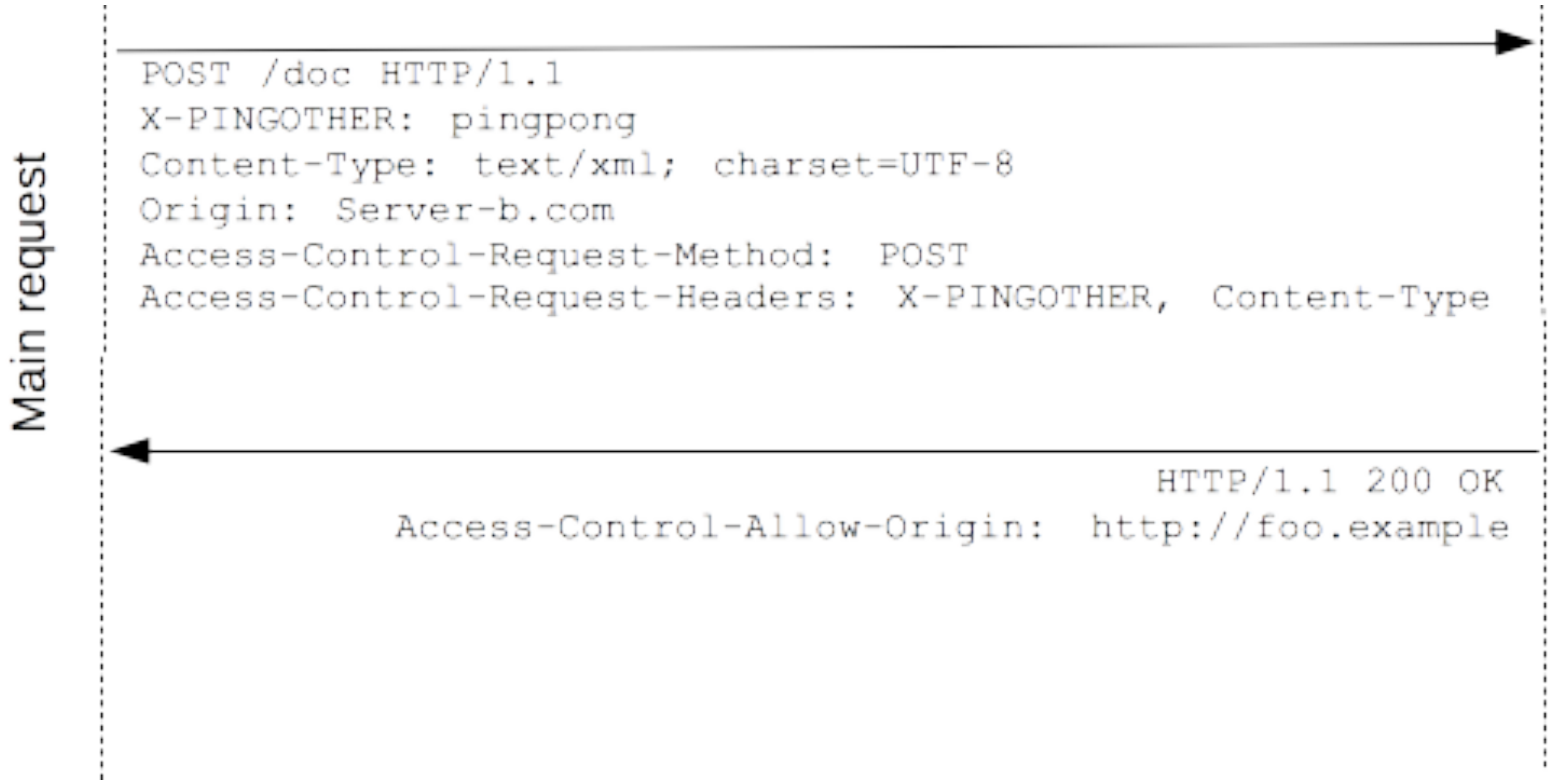
- The request has a method: GET, HEAD, POST
- The request contains only the headers set automatically by the user agent and CORS-safelisted request-headers.
- The Content-Type header values are: *application/x-www-form-urlencoded, multipart/form-data, text/plain*
- No event listeners are registered on any *XMLHttpRequestUpload* object
- No *ReadableStream* object is used in the request

Preflighted requests



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Preflighted requests



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Setting up CORS

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors();
}


public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseCors(builder => builder.AllowAnyOrigin());

    // ...
```

Setting up CORS

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("AllowSpecificOrigin",
            builder => builder.WithOrigins("http://example.com"));
    });
}
```

```
[Route("api/[controller]")]
[EnableCors("AllowSpecificOrigin")]
public class ValuesController : Controller
```

Requests with credentials

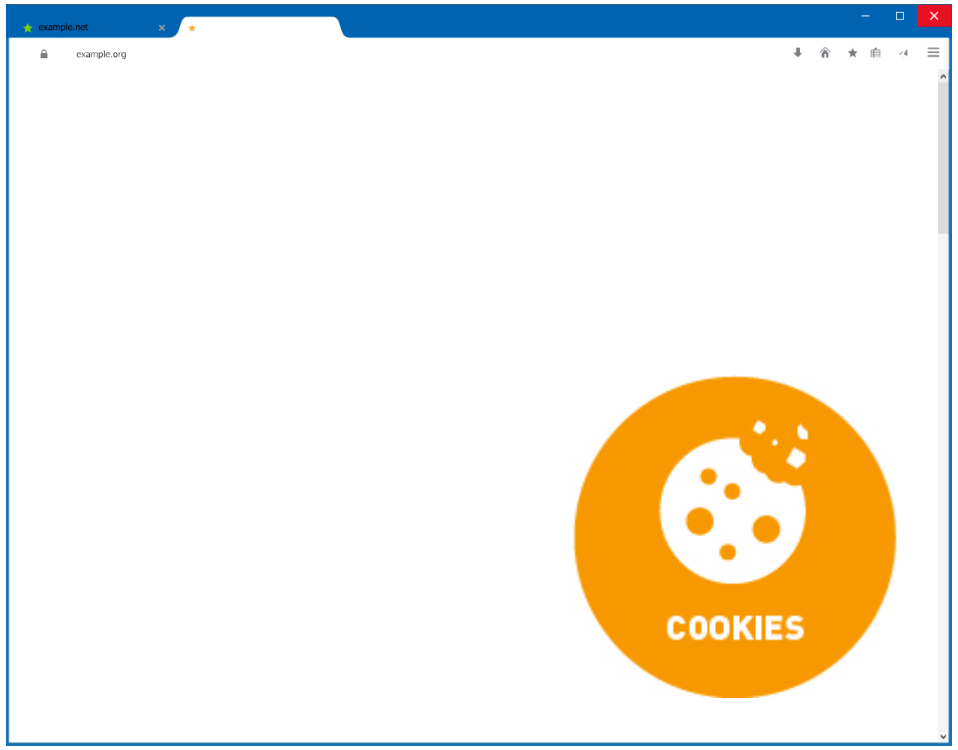
https://ex.com

example.net

example.org

```
var invocation = new XMLHttpRequest();
var url =
'https://example.com/resources/';

function callOtherDomain() {
  if(invocation) {
    invocation.open('GET', url, true);
    invocation.withCredentials = true;
    invocation.onreadystatechange =
        handler;
    invocation.send();
  }
}
```

https://example.com



Requests with credentials

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Requests with credentials

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseCors(builder => builder
        .WithOrigins("https://ex.com")
        .AllowCredentials());

    // ...
```

Other Cross-Origin Requests

use tokens

CORS

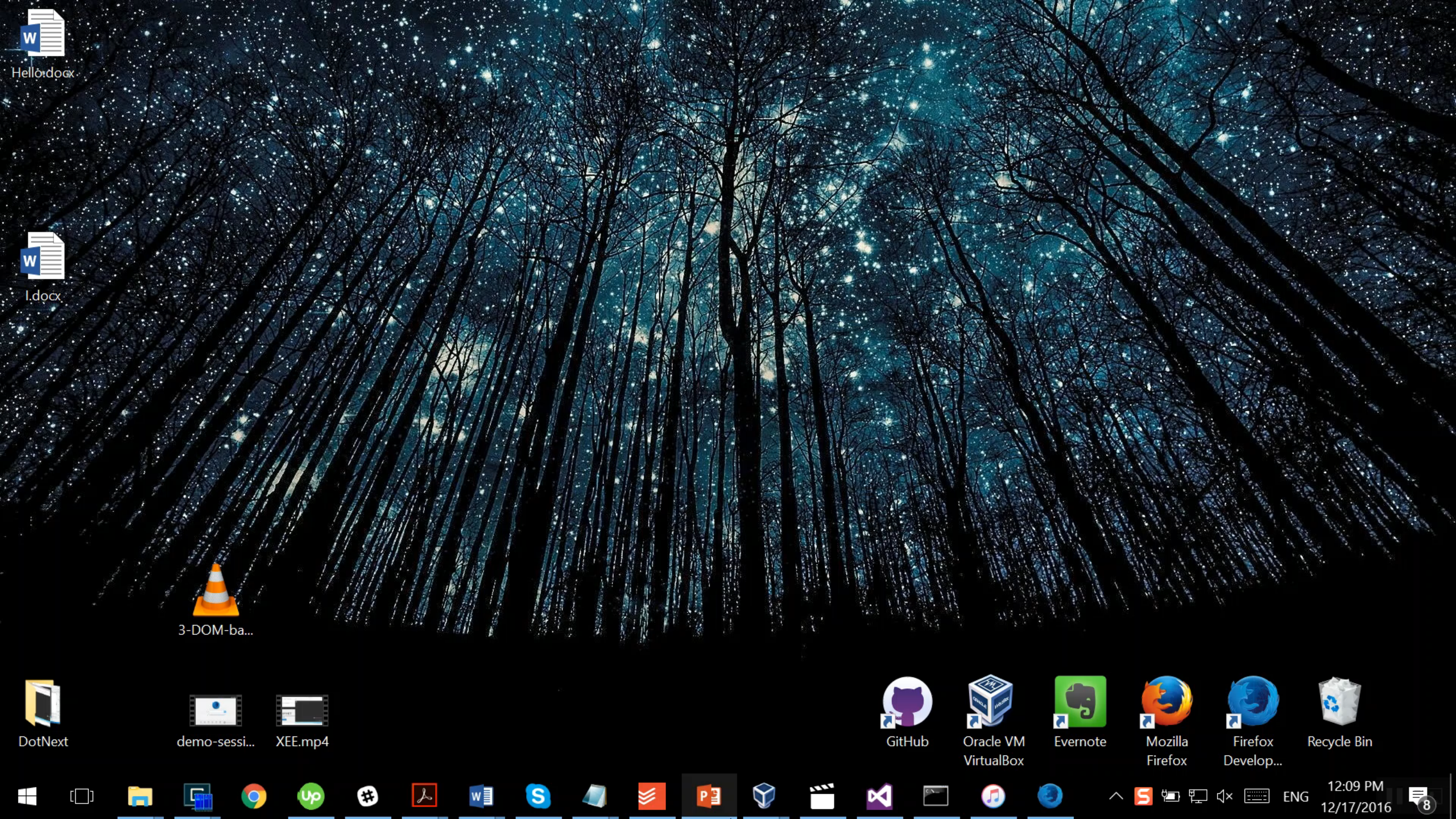
`postMessage()`

check the ORIGIN

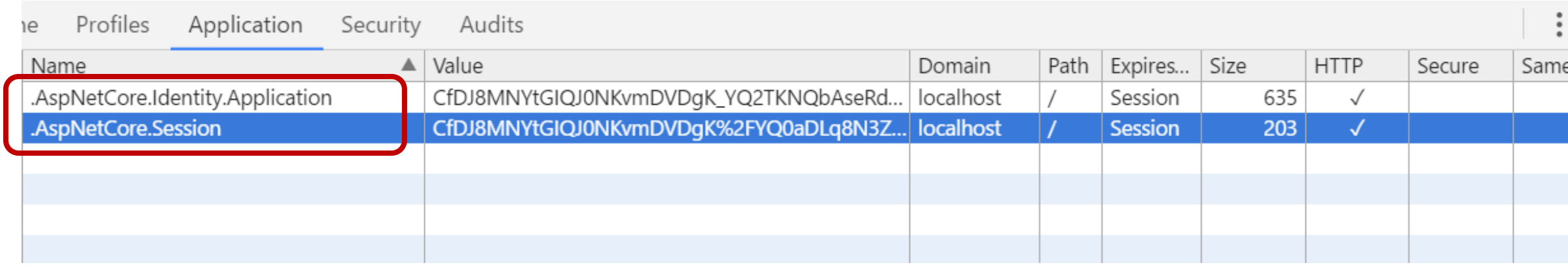
Session Management

Overview

```
private string GetMessageFromCacheOrDb()
{
    // check session cache
    byte[] value;
    if (HttpContext.Session.TryGetValue("msg", out value))
    {
        return System.Text.Encoding.UTF8.GetString(value);
    }

    var valueMessage = GetMessageFromDb(
        HttpContext.User.Identity.Name);
    HttpContext.Session.SetString("msg", valueMessage);
    return valueMessage;
}
```


Hello.docx
I.docx
3-DOM-ba...
DotNext
demo-sessi...
XEE.mp4
GitHub
Oracle VM VirtualBox
Evernote
Mozilla Firefox
Firefox Develop...
Recycle Bin
ENG
12:09 PM
12/17/2016

Overview


Profiles
Application
Security
Audits
Name
Value
Domain
Path
Expires...
Size
HTTP
Secure
.AspNetCore.Identity.Application
CfDJ8MNYtGIQJ0NKvmDVDgK_YQ2TKNQbAseRd...
localhost
/
Session
635
✓
.AspNetCore.Session
CfDJ8MNYtGIQJ0NKvmDVDgK%2FYQ0aDLq8N3Z...
localhost
/
Session
203
✓

ASP .NET Core Session Fixation

- Don’t store security sensitive data in a session!..
- Fix it in your project.

Fix Session Fixation

The **NWebsec.SessionSecurity** library improves ASP .NET session security by enforcing a strong binding between an *authenticated user's identity* and the *user's session identifier*.

https://docs.nwebsec.com/projects/SessionSecurity/en/latest/

https://github.com/NWebsec/NWebsec

Cookies

Cookies

An *HTTP cookie* (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with the next request to the same server.

Cookies are mainly used for:

- Client-side storage
- Session management
- Personalization
- Tracking

Cookies

```
Response.Cookies.Append(
    "cookie-name",
    "secret-value",
    new CookieOptions
    {
        Domain = ".my-domain.com",
        Path = "/docs",
        Secure = true,
        HttpOnly = true,
        Expires = DateTimeOffset.Now.AddHours(1),
        SameSite = SameSiteMode.Lax
     });

var value = Request.Cookies["cookie-name"];
```

Domain and Path

```
new CookieOptions
{
    Domain = "my-domain.com",
    Path = "/docs",
    Secure = true,
    HttpOnly = true,
    Expires = DateTimeOffset.Now.AddHours(1),
    SameSite = SameSiteMode.Lax
};
```

Domain and Path

```
public class ResponseCookies : IResponseCookies
{
    public void Append(
        string key, string value, CookieOptions options)
    {
        var cookieHeaderValue = new SetCookieHeaderValue(
            Uri.EscapeDataString(key),
            Uri.EscapeDataString(value));

        cookieHeaderValue.Domain = options.Domain;
        cookieHeaderValue.Path = options.Path;
```

HttpOnly and Secure

```
new CookieOptions
{
    Domain = ".my-domain.com",
    Path = "/docs",
    Secure = true,
    HttpOnly = true,
    Expires = DateTimeOffset.Now.AddHours(1),
    SameSite = SameSiteMode.Lax
};
```

Using HttpOnly is enough?

```
cryptoSystem = provider.CreateProtector(“MyCompany.MyService.v1");

// ...

var bytes = cryptoSystem.Protect(stream.ToArray());
```

Why isn't it enough again?

```
/* The serialized format of the anti-XSRF token is as follows:
 * Version: 1 byte integer
 * SecurityToken: 16 byte binary blob
 * IsCookieToken: 1 byte Boolean
 * [if IsCookieToken != true]
 *   +- IsClaimsBased: 1 byte Boolean
 *   |  [if IsClaimsBased = true]
 *   |    `- ClaimUid: 32 byte binary blob
 *   |  [if IsClaimsBased = false]
 *   |    `- Username: UTF-8 string with 7-bit integer length prefix
 *   `- AdditionalData: UTF-8 string with 7-bit integer length prefix
 */
```

Hosting

Hosting options

- Kestrel and IIS
- Kestrel and Nginx
- Kestrel only (not recommended way)

Kestrel options

- HTTPS support
- Maximum client connections
- Maximum request body size
- Minimum request body data rate
- Keep-alive timeouts
- Request/Response buffer sizes
- Request header limits

https://github.com/aspnet/KestrelHttpServer/blob/rel/2.0.0/src/Microsoft.AspNetCore.Server.Kestrel.Core/KestrelServerLimits.cs

Summary

- Michal Zalewski “Tangled Web. A Guide to Securing Modern Web Applications”
- Stan Drapkin “Security Driven .NET”
- OWASP Developer Guide http://bit.ly/1JcQLoh
- OWASP Testing Guide v4

Thanks! Questions?!

Mikhail Shcherbakov

Independent Developer and Consultant

 @yu5k3

 https://www.linkedin.com/in/mikhailshcherbakov

The presentation contains footage from “Ghost in the Shell” and Kowloon city photo