DOMXSS IS NOT DEAD

Olle Segerdahl – Principal Security Consultant
@nxsolle @sakerhetssnack

Home About

Search Twitter

Stefano Di Paola

@WisecWisec

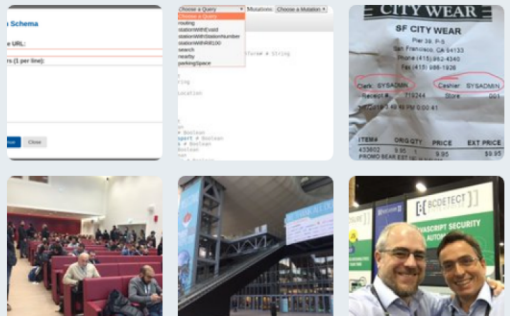
CTO & Chief Scientist of MindedSecurity. (Web) Application Security consultant, researcher and enthusiast. I love lateral thinking.

Florence

blog.mindedsecurity.com

Joined January 2010

61 Photos and videos



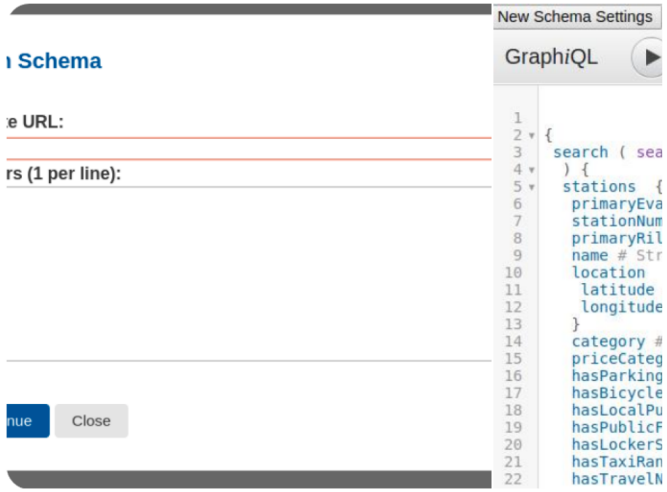
Tweets 3,795 | Following 282 | Followers 5,694 | Likes 1,519

Tweets | Tweets & replies | Media

Pinned Tweet

Stefano Di Paola @WisecWisec · Apr 4

If you're dealing with a GraphQL sec testing and you ca might want to give a try to graphqlschema2payload wh the schema: github.com/mindedsecurity...



43 92

MINDED SECURITY
The Software Security Company

Contact us | Subscribe to our newsletter

HOME PRODUCTS SERVICES RESEARCH NEWS MINDED SECURITY BLOG ABOUT US

WEDNESDAY, SEPTEMBER 22, 2010

A Twitter DomXss, a wrong fix and something more

A Twitter DOM Xss

It seems that twitter new site, introduced some issue resulting in a worm exploiting a stored Xss.
They also added some new JavaScript in their pages which I casually saw while searching in the html for the worm payload.

The code was the following :

```
//<![CDATA[
(function(g){var a=location.href.split("#!")[1];if(a)
{g.location=g.HBR=a;}})(window);
//]]>
```

Do you spot the issue?
It search for "#!" in the Url and assign the content after that to the window.location object. And it is present in (almost?) every page on twitter.com main site.

According to DOM Xss Wiki the location object is one of the first objects identified for being dangerous as it is both a source and a sink.

In fact the DOM Based Xss will be triggered by simply going to:

> http://twitter.com/#!javascript:alert(document.domain);

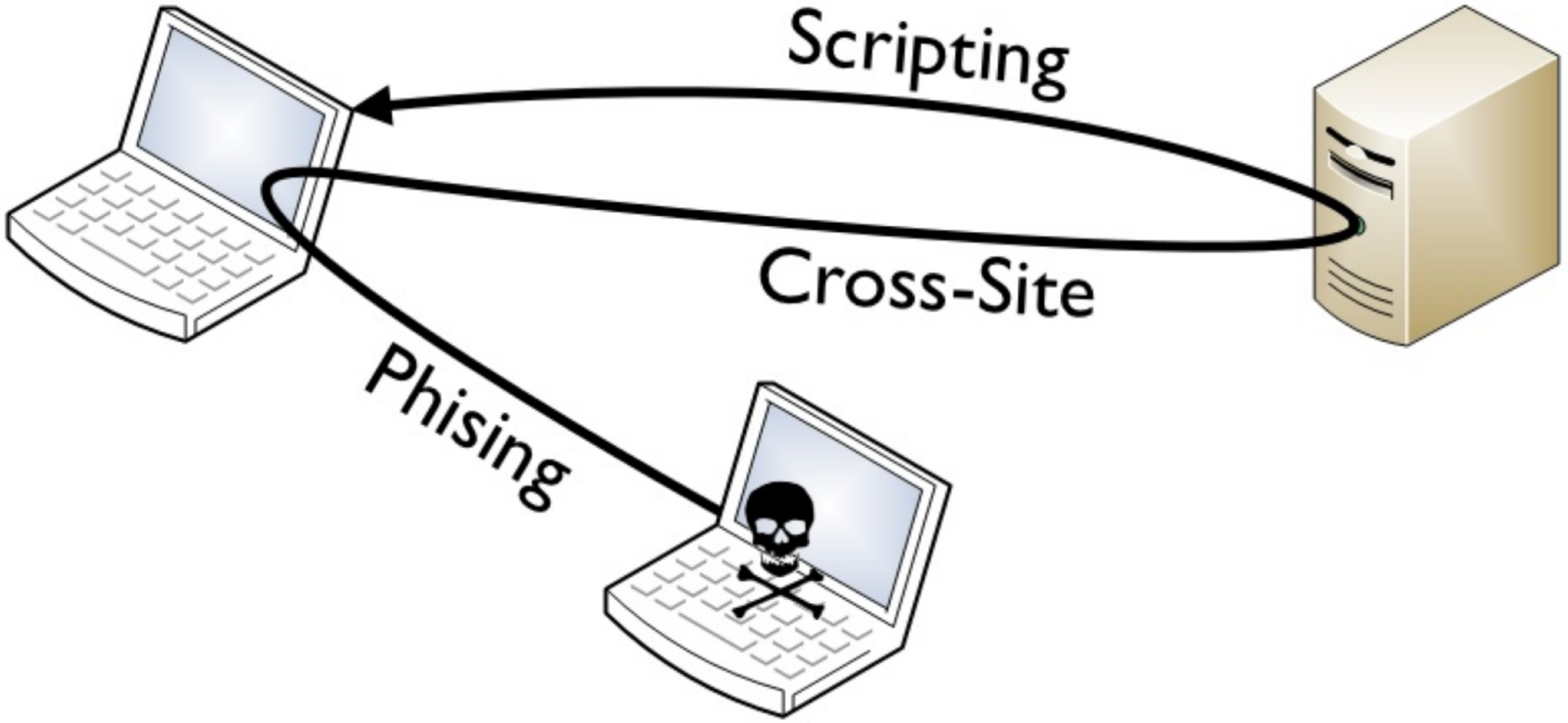
SEARCH

Search

LABELS

3rd party javascript (2)
absolute path check (1)
Adobe (3)
Advisory (5)
adware (1)
AFNetworking (1)
agile (1)
AMT (1)
Android Security (2)
Anti-Tampering (2)
Antitamper (2)
Applet Security (6)
Application Security (22)
Arbitrary Code Execution (4)
architecture (1)
asp.net (2)
ast (1)
attacks (1)
Autoloaded File Inclusion (1)
Banking (4)
Banking Malware (1)
blueclosure (1)
canonicalization (1)

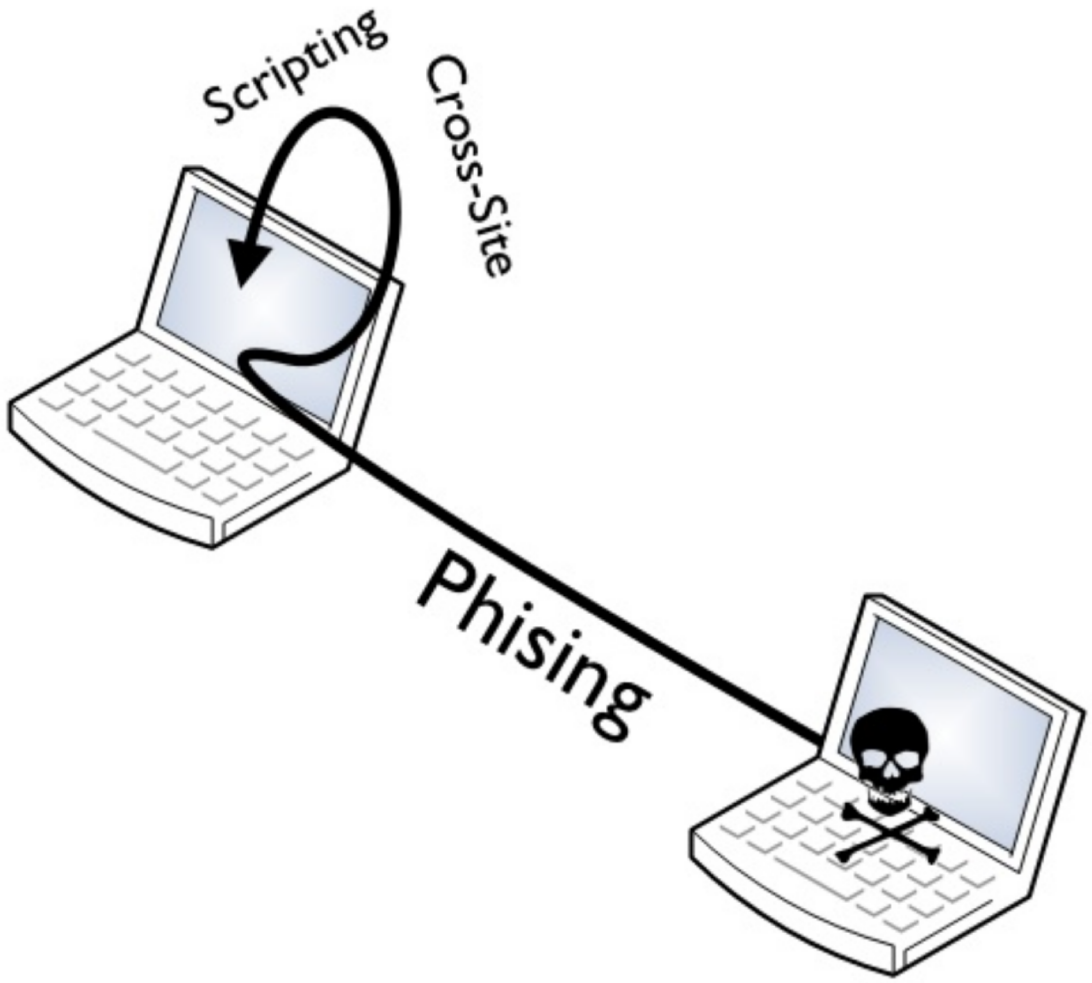
WHAT IS DOMXSS?



Visio skills: John Wilander - https://johnwilander.se/

F-Secure

WHAT IS DOMXSS ?



Visio skills: John Wilander - https://johnwilander.se/

F-Secure

WHAT IS DOMXSS?

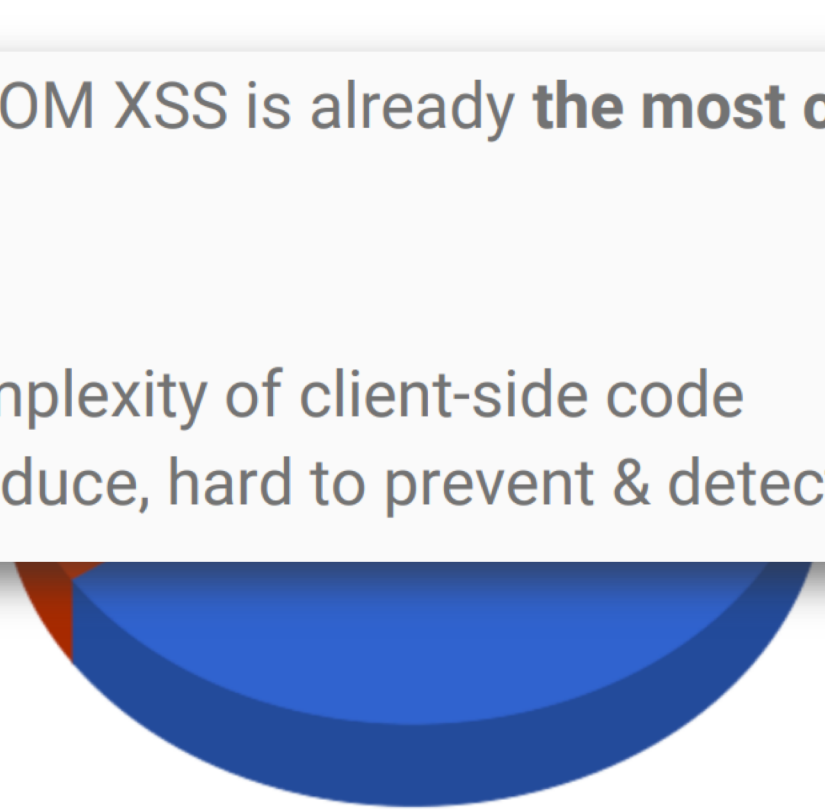
```
<script>
  var page = location.hash.substr(1);
  var div = document.getElementById('banner’);
  div.innerHtml = '<h1>' + page + '</h1>’;
</script>
```

F-Secure

DOMXSS IS PAINFUL

Google Web :: results :: bugs

What types of bugs do they find?



- At Google, DOM XSS is already **the most common XSS variant**

Reasons:

- Growing complexity of client-side code
- Easy to introduce, hard to prevent & detect

Source: Krzysztof Kotowicz / Google

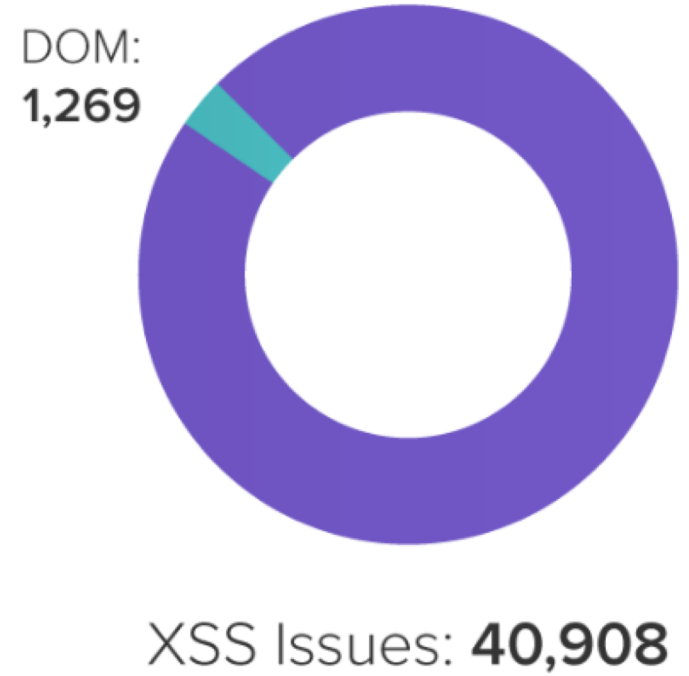
F-Secure

DOMXSS IS HARD TO DETECT

netsparker

Cross-site Scripting

Cross-Site Scripting (XSS) has been around for a very long time and is known by almost all developers. So it's surprising that it accounts for around one quarter of all detected vulnerabilities, a total of 40,908 issues. 1,269 of the detected XSS were DOM XSS. Cross-site scripting vulnerabilities are very difficult to get rid of, though they are very easy to detect automatically with the Netsparker web security solution.



F-Secure

Source: https://www.netsparker.com/blog/web-security/netsparker-web-security-scan-statistics-2018/

FINDING DOMXSS

```
<script>
  var page = location.hash.substr(1);
  var div = document.getElementById('banner');
  div.innerHtml = '<h1>' + page + '</h1>';
</script>
```

Source of 'attacker controlled' data

Sink, data inserted in unsafe context

`http://user@host/path?query#fragment`

F-Secure

Home

Stefano Di Paola edited this page on Jul 14, 2016 · 3 revisions

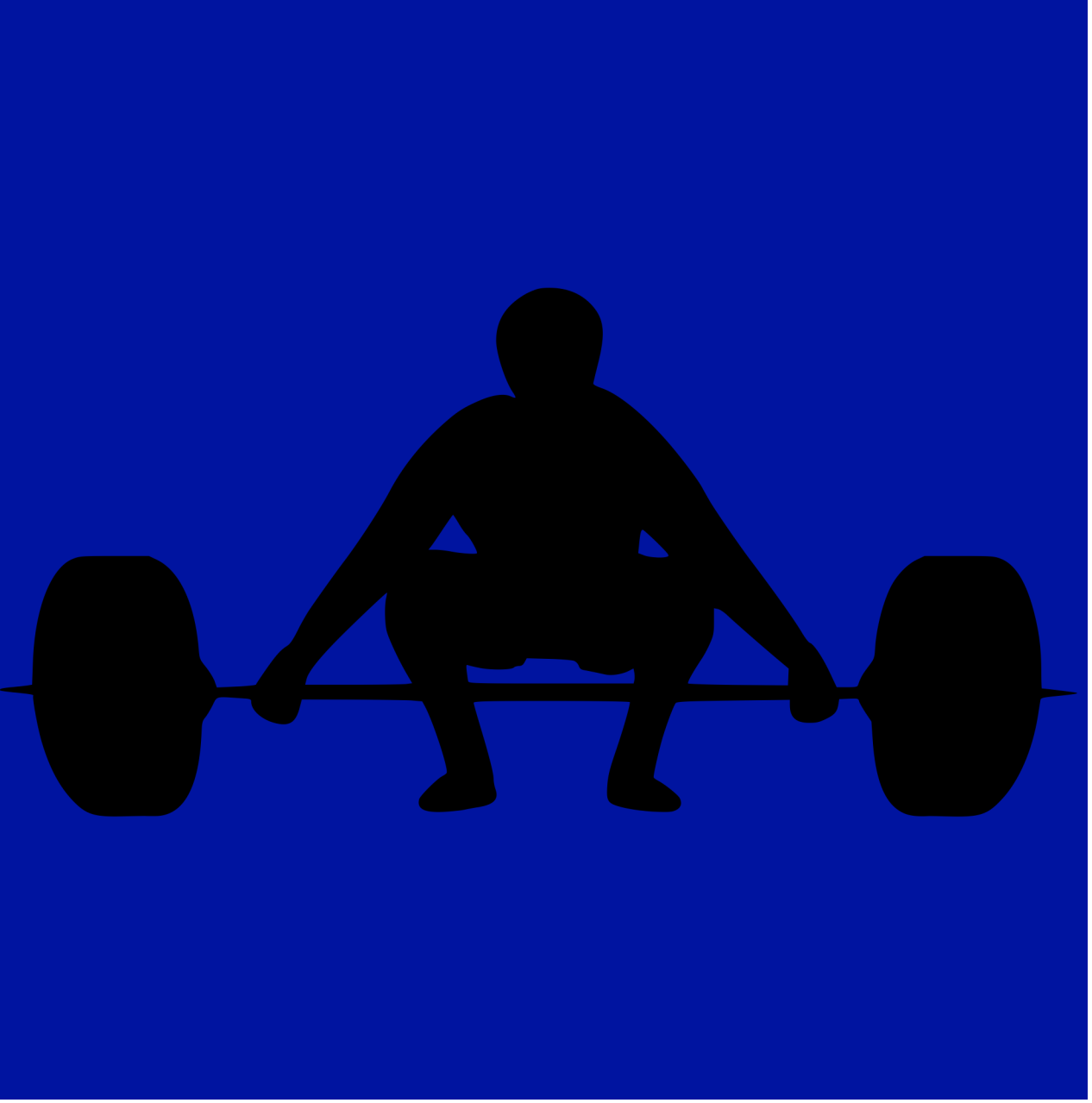
DOMXSS Wiki

The **DOMXSS Wiki** is a Knowledge Base for defining sources of attacker controlled inputs and sinks which potentially could introduce DOM Based XSS issues. DOMXSS first being thoroughly documented in a paper by Amit Klein in 2005 has risen in relevance over the last years - nevertheless still lacking a central place for collecting information and knowledge about it.

The project aims top be this very place and to identify sources and sinks methods exposed by public, widely used javascript frameworks. The project is a work in progress and will be extended over time. Contributions are welcome.

Please use the sidebar menu to navigate contents.

This project is mainly maintained by Stefano Di Paola.



EXAMPLES & TUTORIALS

www.domxss.com

www.daspatnaik.com/test/demo

xss-game.appspot.com

domgo.at

F-Secure

Client XSS Exercise-1

Previous | Next

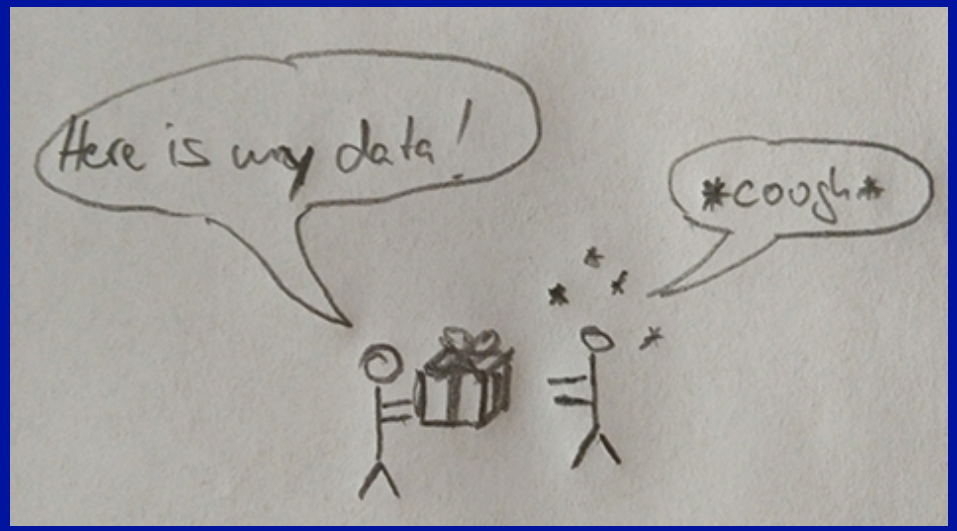
Welcome **12345**!!

Data Flow

Source of Data		Data from Source		Data to Sink		Sink causing Execution
location.hash	→	#12345	→	Welcome 12345!!	→	HTMLElement.innerHTML

Vulnerable Code

```
let hash = location.hash;
if (hash.length > 1) {
    let hashValueToUse = unescape(hash.substr(1));
    let msg = "Welcome <b>" + hashValueToUse + "</b>!!";
    document.getElementById("msgboard").innerHTML = msg;
}
```

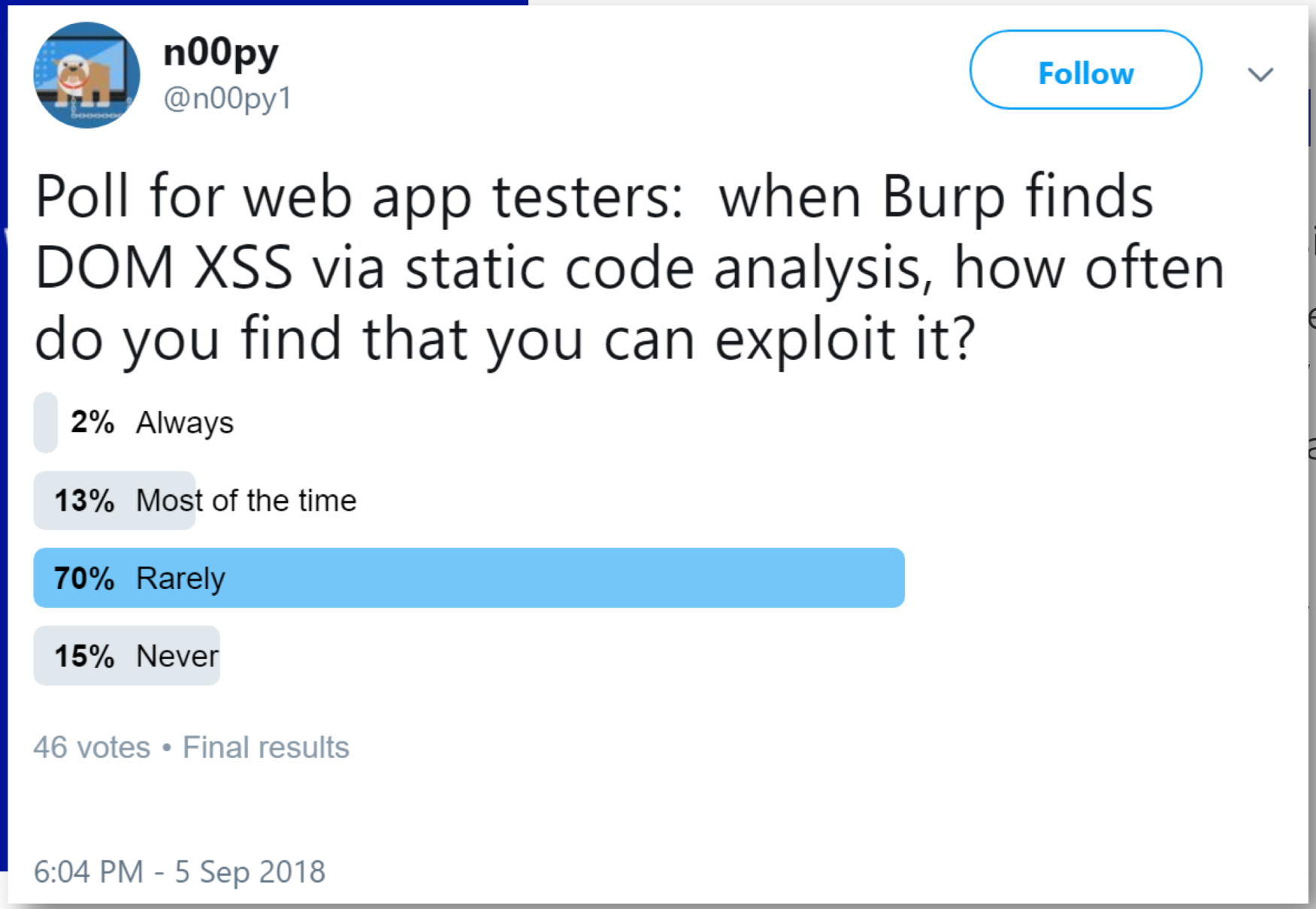


FINDING DOMXSS: TAINT ANALYSIS

- Mark data as 'tainted' at the source
- Follow the data as it is passed around
- Alert if 'tainted' data is passed to a sink!
- Two different ways of implementing: the static or dynamic approach

Drawing stolen from: http://www.thewhitespace.de/blog/2014/11/17/flowtwist-efficient-context-sensitive-inside-out-taint-analysis/

F-Secure



inks

e

at it”

.

F-Secure

BLUECLOSURE
JAVASCRIPT SECURITY

FINDING DOMXSS: DYNAMIC APPROACH

- Pioneered by DOMinator
- Instrument a ”real” browser JS engine
- Run the code to see what it does!
- Implemented in Burp since late 2018
- (DOMinator replaced by BlueClosure)

- Major pain to keep modifications to browser codebase up-to-date…
- Limited to commercial tools?

F-Secure

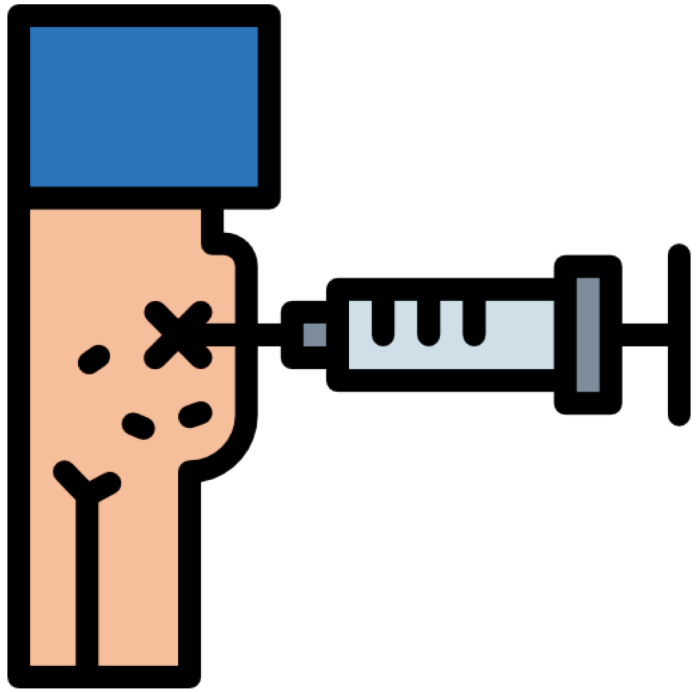
FINDING DOMXSS: HILLBILLY APPROACH

- We already have a browser
- JS allows overriding almost anything
- What if we 'taint' **inline** with the data
- 'input' -> 'input.t4int3d' at the source
- Now we match on 't4int3d' at all sinks

- 100% doable using in-browser JS!

F-Secure

INTRODUCING: TAINT TESTING TOOL



https://github.com/ollseg/ttt-ext

F-Secure

Awesome icon made by smalllikeart from www.flaticon.com

– Målet med vägledningen är ökad förutsägbarhet och rättssäkerhet. Skapa en någorlunda enhetlig bild i landet över vad det är som kan förbjudas, säger Magnus Olofsson, MSB.

Eldningsförbud avsett för skog

MEST LÄST

- Tuggar slang hela dan
- MSBs förstärkningsresurser imponerade
- Texell slutar i Attunda
- Vägledning om att släcka skogsbränder
- Jobbar för nästa generation

SKOGSBRÄNDER

Not Secure | www.tjugofyra7.se/?t4inT3d.param=t4inT3d.value"%27<h1>lol#!//t4inT3d.hash"'<h1>lol

Förnya prenumeration | Annonsera | Kontakta oss

TJUGOFYRA7

BLOGGARE

Stefan Svensson
docent, teknisk doktor

LADDA HEM

tidigare nummer av tidningen

START | BLOGGAR | ARTIKLAR | SKRIV EN SÖKFRÅGA HÄR

MEST LÄST

Tuggar slang hela dan

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Audits

7 8

top | Filter | All levels

'ZGAMClyHrYMAAAASKsy7Aw%3D%3D', v1: 't4int3d.example.com', v3: 'http%3A%2F%2Fwww.tjugofyra7.se%2F', v4: 'http%3A%2F%2Fwww.tjugofyra7.se%2F%3Ft4inT3d.param%3Dt4inT3d.value%2522%2527%253ch1%253Elol%23!%2F%2Ft4inT3d.hash%2522'%253ch1%253Elol' };</script><script id="pxscrpt" async="" defer="" src="//t.sharethis.com/1/d/t.dhj?rnd=1555926187772&cid=c010&dmn=www.tjugofyra7.se"></script></body></html>

2 GOOD VM31:40

▶DOM child node added to 'html' VM31:55
: <body onload="var pxscrpt = document.createElement('script'); pxscrpt.id = 'pxscrpt'; pxscrpt.async = true; pxscrpt.defer = true; pxscrpt.src = '//t.sharethis.com/1/d/t.dhj?rnd=1555926187772&cid=c010&dmn=www.tjugofyra7.se';document.body.appendChild(pxscrpt);"><script> var pxcelData = { v0: 'ZGAMClyHrYMAAAASKsy7Aw%3D%3D', v1: 't4int3d.example.com', v3: 'http%3A%2F%2Fwww.tjugofyra7.se%2F', v4: 'http%3A%2F%2Fwww.tjugofyra7.se%2F%3Ft4inT3d.param%3Dt4inT3d.value%2522%2527%253ch1%253Elol%23!%2F%2Ft4inT3d.hash%2522'%253ch1%253Elol' };</script>
<script id="pxscrpt" async="" defer="" src="//t.sharethis.com/1/d/t.dhj?rnd=1555926187772&cid=c010&dmn=www.tjugofyra7.se"></script></body>

2 GOOD VM31:40

▶DOM child node added to 'body' VM31:55
: <script> var pxcelData = { v0: 'ZGAMClyHrYMAAAASKsy7Aw%3D%3D', v1: 't4int3d.example.com', v3: 'http%3A%2F%2Fwww.tjugofyra7.se%2F', v4: 'http%3A%2F%2Fwww.tjugofyra7.se%2F%3Ft4inT3d.param%3Dt4inT3d.value%2522%2527%253ch1%253Elol%23!%2F%2Ft4inT3d.hash%2522'%253ch1%253Elol' };</script>

▶FP? DOM setAttribute: iframe#pxcelframe.src, //t.sharethis.com/a/t_.htm?ver=0.243.10378&cid=c010#rnd=1555926187772&cid=c010&dmn=www.tjugofyra7.se&tt=t.dhj&dhjLcy=29&lbl=pxcel&flbl=pxcel&ll=d&ver=0.243.10…&qs=test%3Dt4inT3d.referrer%2522%2527%253ch1%253Elol&cc=FI&cont=EU&ipaddr= VM31:50

▶FP? DOM child node added to 'body': <iframe id="pxcelframe" src="//t.sharethis.com/a/t_.htm?ver=0.243.10378&cid=c010#rnd=1555926187772&cid=c010&dmn=www.tjugofyra7.se&tt=t.dhj&dhjLcy=29&lbl=pxcel&flbl=pxcel&a…4inT3d.referrer%2522%2527%253ch1%253Elol&cc=FI&cont=EU&ipaddr=" style="display: none;"></iframe> VM31:50

>

Console | What's New × | Search

"button/buttons.js" Search

Need more results? Try internal pages search. query syntax

140330 web pages in 0.00 s.

URLs CSV CSV+snippets

Rank	Url	Snippets
1 817	https://itslearning.com/uk/	://ws.sharethis.com/**button/buttons.js**?ver=5.1.1'></script
1 917	http://fanfox.net/	://ws.sharethis.com/**button/buttons.js**"; var s = document.
2 204	https://www.diariolibre.com/	"//ws.sharethis.com/**button/buttons.js**?publisher=0dd6783e-
2 337	http://www.mangahere.cc/	://ws.sharethis.com/**button/buttons.js**"; var s = document.
3 958	http://www.ammonnews.net/	://ws.sharethis.com/**button/buttons.js**"></script> <script
4 511	http://www.mangatown.com/	'//ws.sharethis.com/**button/buttons.js**').queueWait(functio
4 632	https://www.alaraby.co.uk/portal	://ws.sharethis.com/**button/buttons.js**"></script> <script
4 708	http://iloveoldschoolmusic.com/	p://w.sharethis.com/**button/buttons.js**" async></script> <s
5 417	http://ancensored.com/	p://w.sharethis.com/**button/buttons.js**"></script> <script
6 325	https://www.hds.to/home.php	p://w.sharethis.com/**button/buttons.js**"></script> <script
6 835	https://www.unicef.org/	://ws.sharethis.com/**button/buttons.js**"></script> <script

media.ford.com

Go Further

COMMERCIAL VEHI
THE BACKBON
THE ECONOMY

DOMXSS on media.ford.com

Close

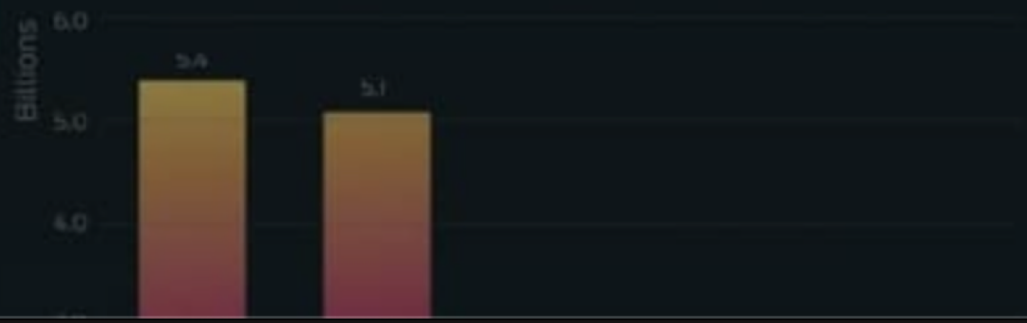
Go Further

THE VAN ECONOMY

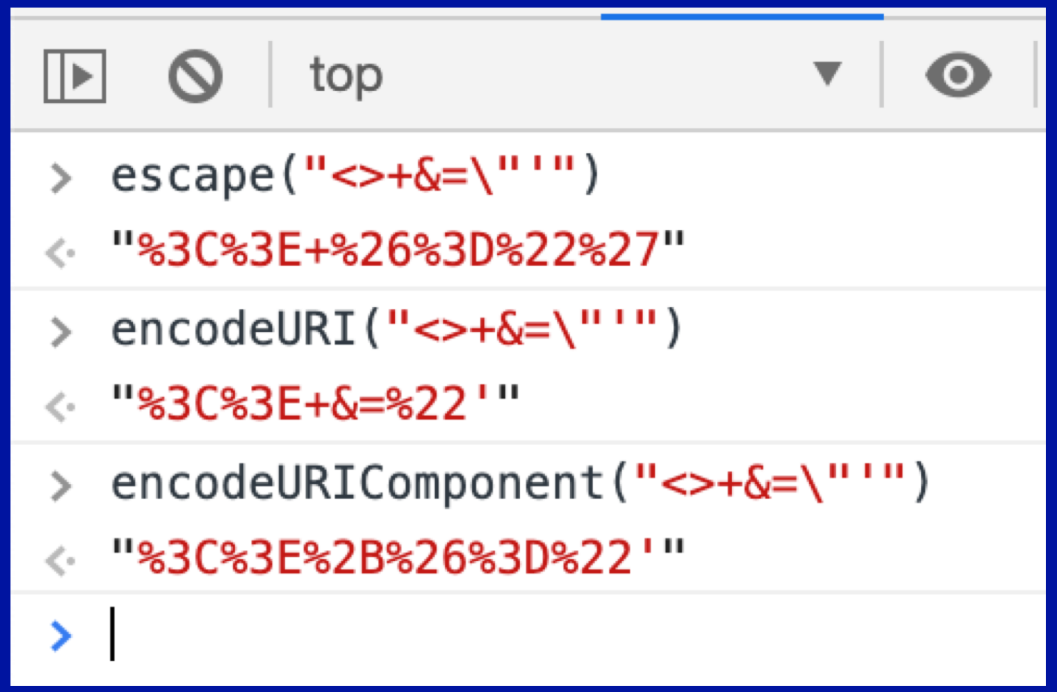
The financial contribution of industries that rely heavily on commercial vehicles (CVs) is growing and as of 2017, **stands at €675bn**.

Financial contribution to the

Fuel duty receipts from van use, €bns, 2018.



FIXING DOMXSS



DON'T use `encodeURI()`

It doesn't escape singe-quotes and is meant for use with HTTP.

What about `escape()`?

Well, it's deprecated by W3C and is meant for use with JS strings.

Use *someting* context-aware!

F-Secure

FIXING DOMXSS

Sign Up Log In

How to display escaped HTML special characters the right way?

Get Help

MarcusW 1 May '18

Hi,

I'm using a HTML-escape library on my server-side to escape any user-input before delivering it to my Vue.js frontend to mitigate XSS attacks.
Now I've the issue that this of course also escapes characters like & and german umlauts like ö,ä,ü so that german sentences like `Die süße Hündin & die Bären laufen in die Höhle.` are displayed everywhere in the frontend as `Die s&#xFC;&#xDF;e H&#xFC;ndin &amp; die B&#xE4;ren laufen in die H&#xF6;hle`.

This is of course not acceptable so I'm wondering what's the best-pracice to make the rendering in Vue.js work right.

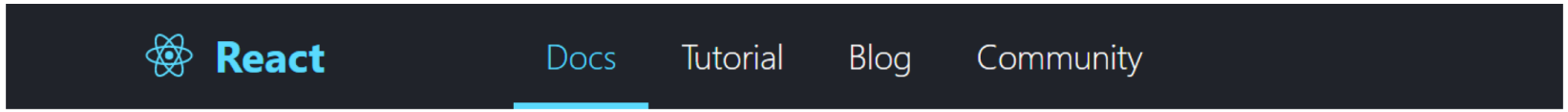
I have a quiet large codebase that's using {{…}} everywhere and replacing several hundrets of them with v-html is not an option.

What's the right way to handle this? Can I disable the escaping for the {{…}}-syntax completely?

Thank you in advance!

F-Secure

FIXING DOMXSS



dangerouslySetInnerHTML

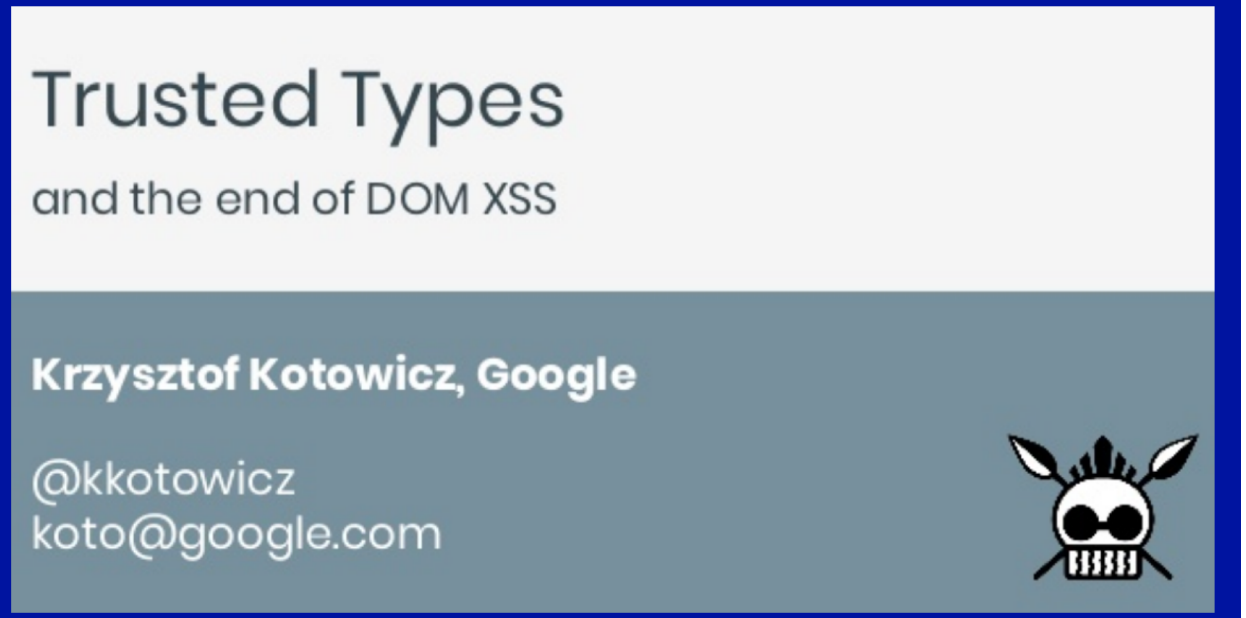
`dangerouslySetInnerHTML` is React's replacement for using `innerHTML` in the browser DOM. In general, setting HTML from code is risky because it's easy to inadvertently expose your users to a cross-site scripting (XSS) attack. So, you can set HTML directly from React, but you have to type out `dangerouslySetInnerHTML` and pass an object with a `__html` key, to remind yourself that it's dangerous. For example:

```
function createMarkup() {
  return {__html: 'First &middot; Second'};
}

function MyComponent() {
  return <div dangerouslySetInnerHTML={createMarkup()} />;
}
```

F-Secure

KILLING DOMXSS !



https://www.slideshare.net/kkotowicz/trusted-types-and-the-end-of-dom-xss

Follow @kkotowicz on Twitter!

https://wicg.github.io/trusted-types/

https://developers.google.com/web/updates/2019/02/trusted-types

https://jsbin.com/miwanobeva/

F-Secure

DOMXSS IS NOT DEAD (YET)

@nxsolle @sakerhetssnack

F-Secure