End-to-end Injection Safety at Scale

Mike Samuel, Google Security Engineering

March 2019

About me

Security engineer @ Google

Hacks libraries, tools, languages

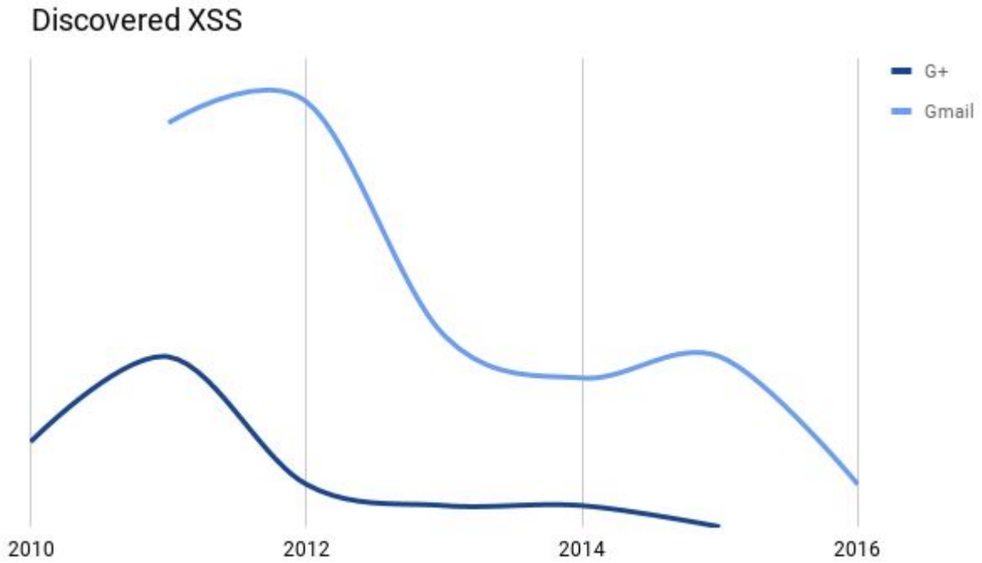
TC39 member (JavaScript language committee)

Editor of "A Roadmap for Node.js Security"

Mission: make the easiest way to express an idea in code, a secure way

@mvsamuel

Trusted Types



- W3 Proposal
- Builds on 6+ years of experience within Google
- Protects Gmail, many other complex apps
- Evidence of efficacy

Trusted Types

1. Problem statement
2. Small change ⇨ big organizational effect
3. Adoption in practice
4. Early adopters welcome

"Today, a novice programmer cannot write a complex but secure application."

Breaking XSS mitigations via Script Gadgets - blackhat 2017

What are Script Gadgets?

A **Script Gadget** is an *existing* JS code on the page that may be used to bypass mitigations:

```
<div data-role="button" data-text="I am a button"></div>

[...]

<script>
  var buttons = $("[data-role=button]");
  buttons.html(buttons.attr("data-text"));
</script>
```

```
<div data-role="button" ... >I am a button</div>
```

What are Script Gadgets?

A **Script Gadget** is an *existing* JS code on the page that may be used to bypass mitigations:

```
XSS <div data-role="button"
data-text="&lt;script&gt;alert(1)&lt;/script&gt;"></div>XSS


<script>
  var buttons = $("[data-role=button]");
  buttons.html(buttons.attr("data-text"));
</script>
```

Script Gadget

```
<div data-role="button" ... ><script>alert(1)</script></div>
```

So what? Why should I care?

- Gadgets are present in **all but one** of the tested popular web frameworks.
- Gadgets can be used to bypass **most** mitigations in modern web applications.
- We automatically created exploits for **20% of web applications** from Alexa top 5,000.

"Today, a novice programmer cannot write a complex but secure application."

Breaking XSS mitigations via Script Gadgets - blackhat 2017

Reduce developers' security burden.

Move it to security professionals.

Client-side JavaScript

```
<div id=foo></div>
<script>
var foo = document.querySelector('#foo');
foo.innerHTML = 'raw-string';
</script>
```

Client-side JavaScript

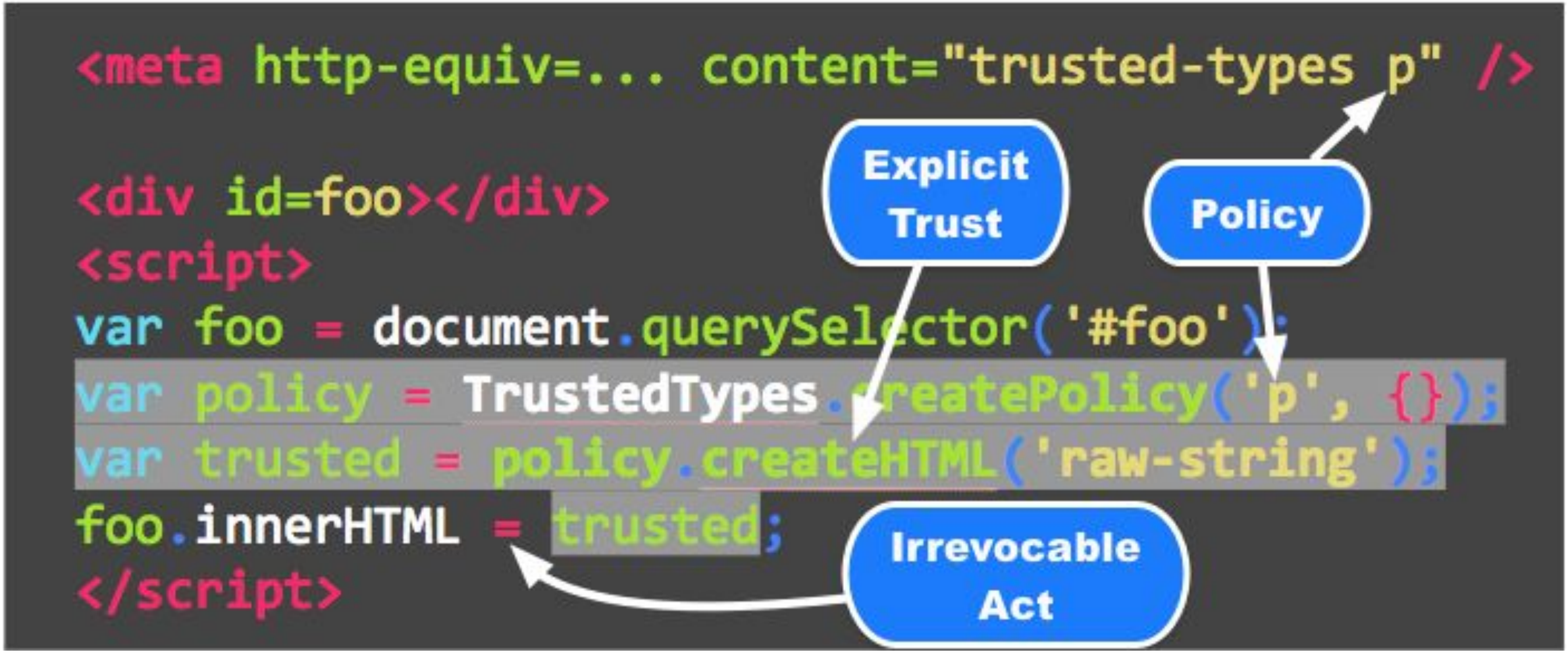
```
<meta http-equiv=... content="trusted-types p" />

<div id=foo></div>
<script>
var foo = document.querySelector('#foo');
foo.innerHTML = 'raw-string';
</script>
```

```
▶Uncaught TypeError: Failed to set the 'innerHTML' property on 'Element': This document requires  demo2.html:9
`TrustedHTML` assignment.
    at demo2.html:9
```

Trusted Types make Trust decisions explicit

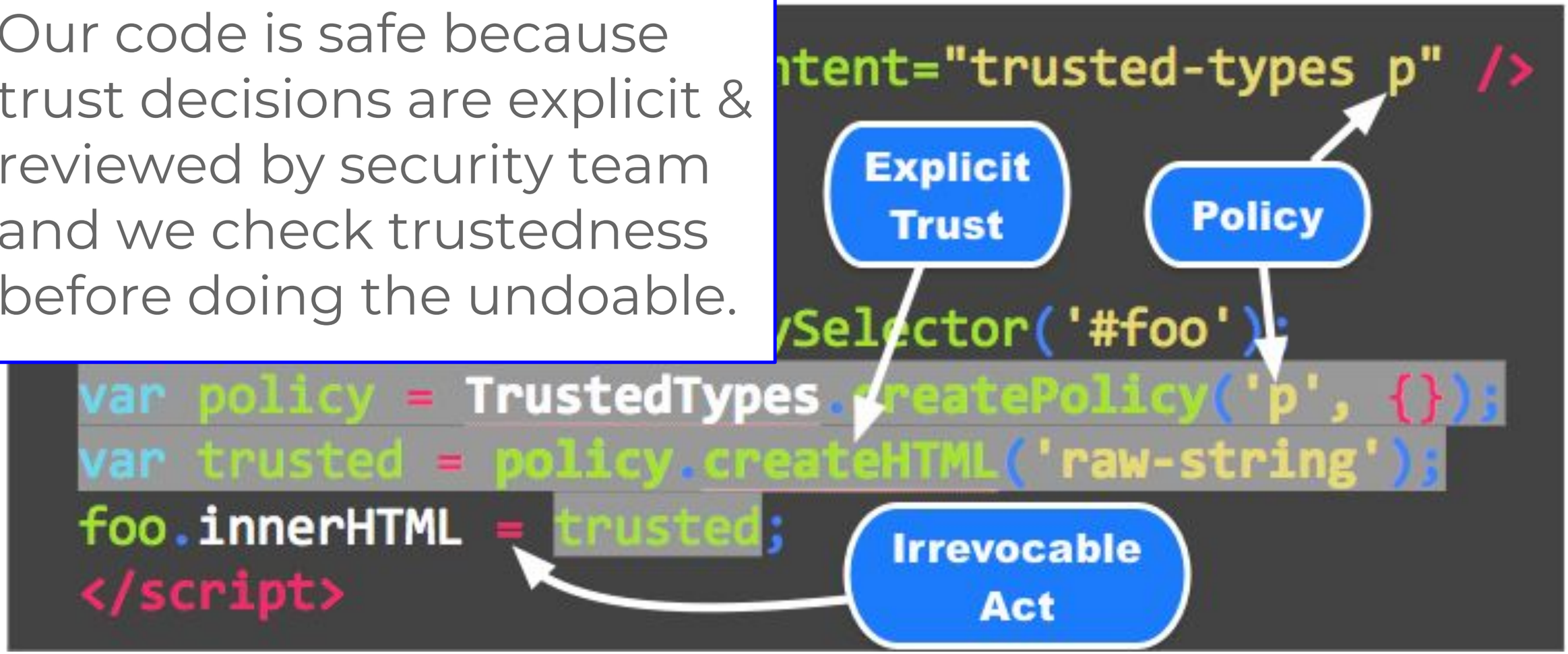
```
<meta http-equiv=... content="trusted-types p" />

<div id=foo></div>
<script>
var foo = document.querySelector('#foo');
var policy = TrustedTypes.createPolicy('p', {});
var trusted = policy.createHTML('raw-string');
foo.innerHTML = trusted;
</script>
```

Trusted Types make Trust decisions explicit



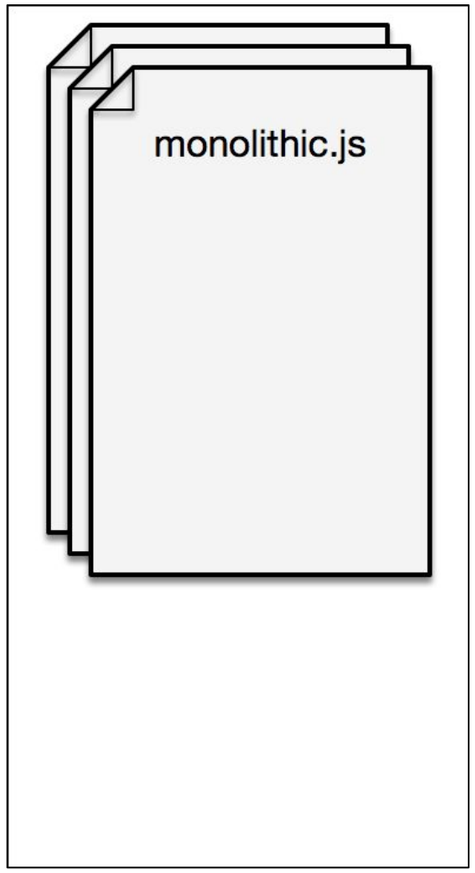
Trusted Types make Trust decisions explicit

Our code is safe because trust decisions are explicit & reviewed by security team and we check trustedness before doing the undoable.

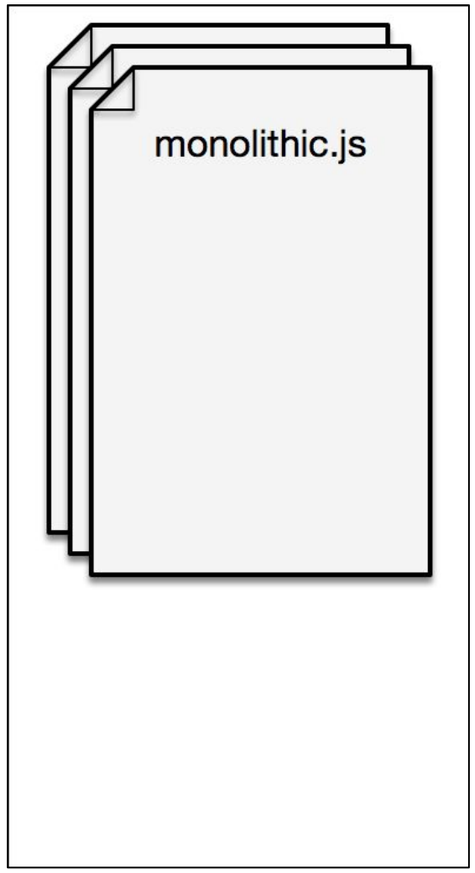


Step 1. Flip the switch

- Trust decisions Implicit
- No separation btw. sensitive and other code
- Un-undoable acts not checked
- Bugs non-obvious
- Fails unsafe

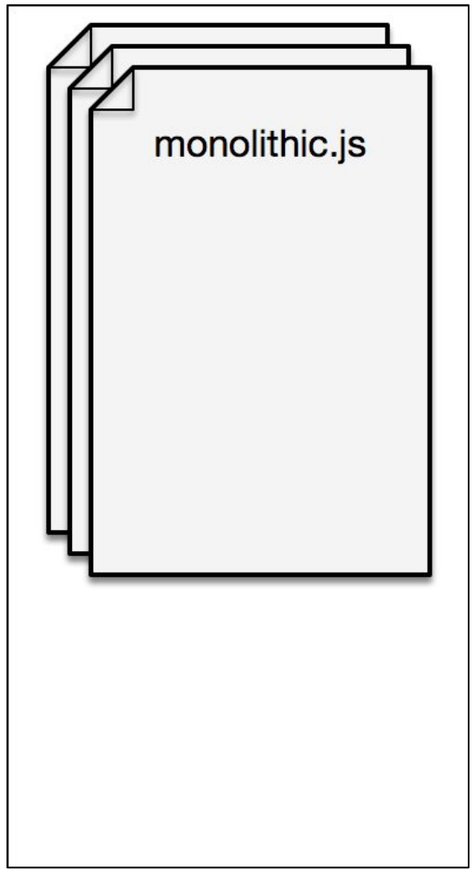


- Trust decisions Implicit
- No separation btw. sensitive and other code
- Un-undoable acts **checked**
- Bugs **obvious**
- Fails **safe**

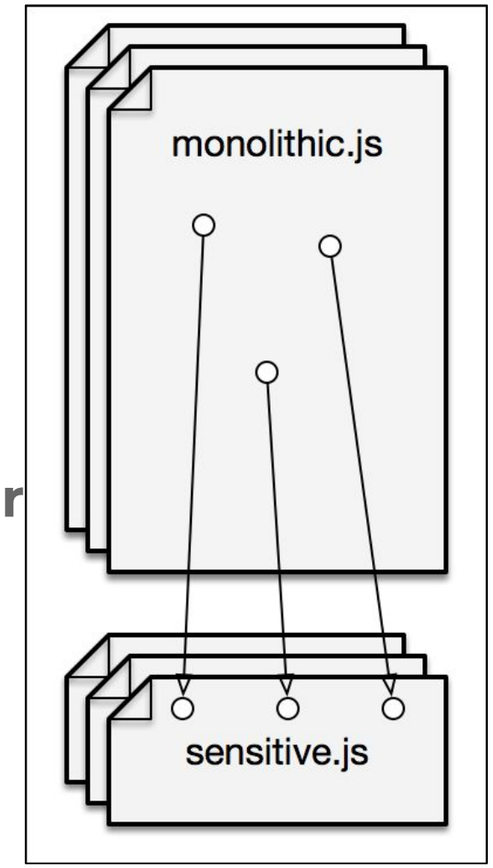


Step 2. Consolidate tricky code

- Trust decisions Implicit
- No separation btw. sensitive and other code
- Un-undoable acts checked
- Bugs obvious
- Fails safe

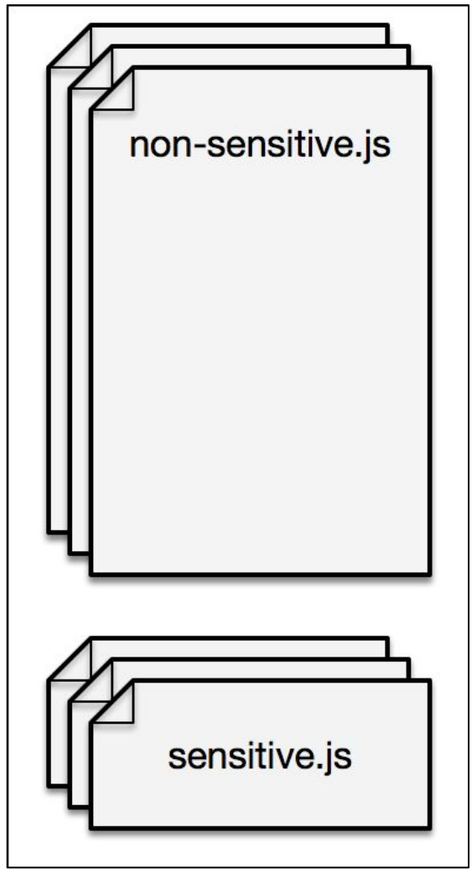


- Trust decisions **explicit**
- **Separation** btw. sensitive and other code
- Un-undoable acts checked
- Bugs obvious, **fewer**
- Fails safe
- **Sensitive code not scrutinized**

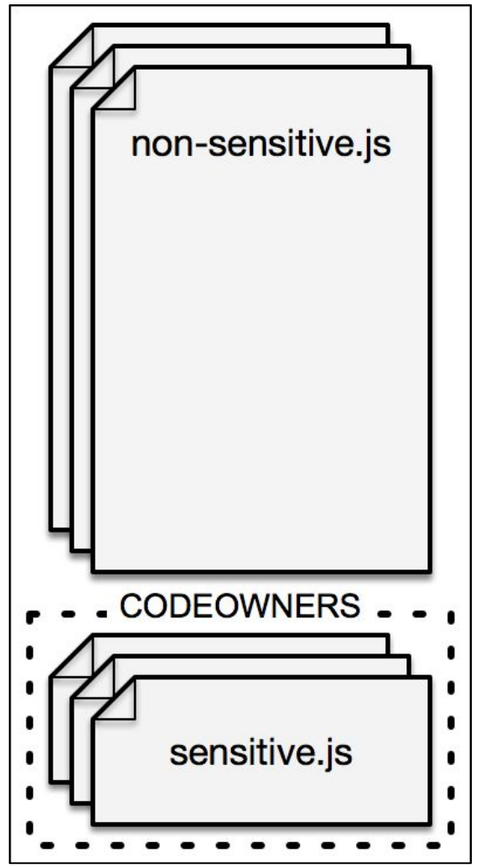


Step 3. Automatically loop in reviewers

- Trust decisions explicit
- Separation btw. sensitive and other code
- Un-undoable acts checked
- Bugs obvious, fewer
- Fails safe
- Sensitive code not scrutinized



- Trust decisions explicit
- Separation btw. sensitive and other code
- Un-undoable acts checked
- Bugs obvious, fewer
- Fails safe
- Sensitive code **scrutinized**



Lightweight process

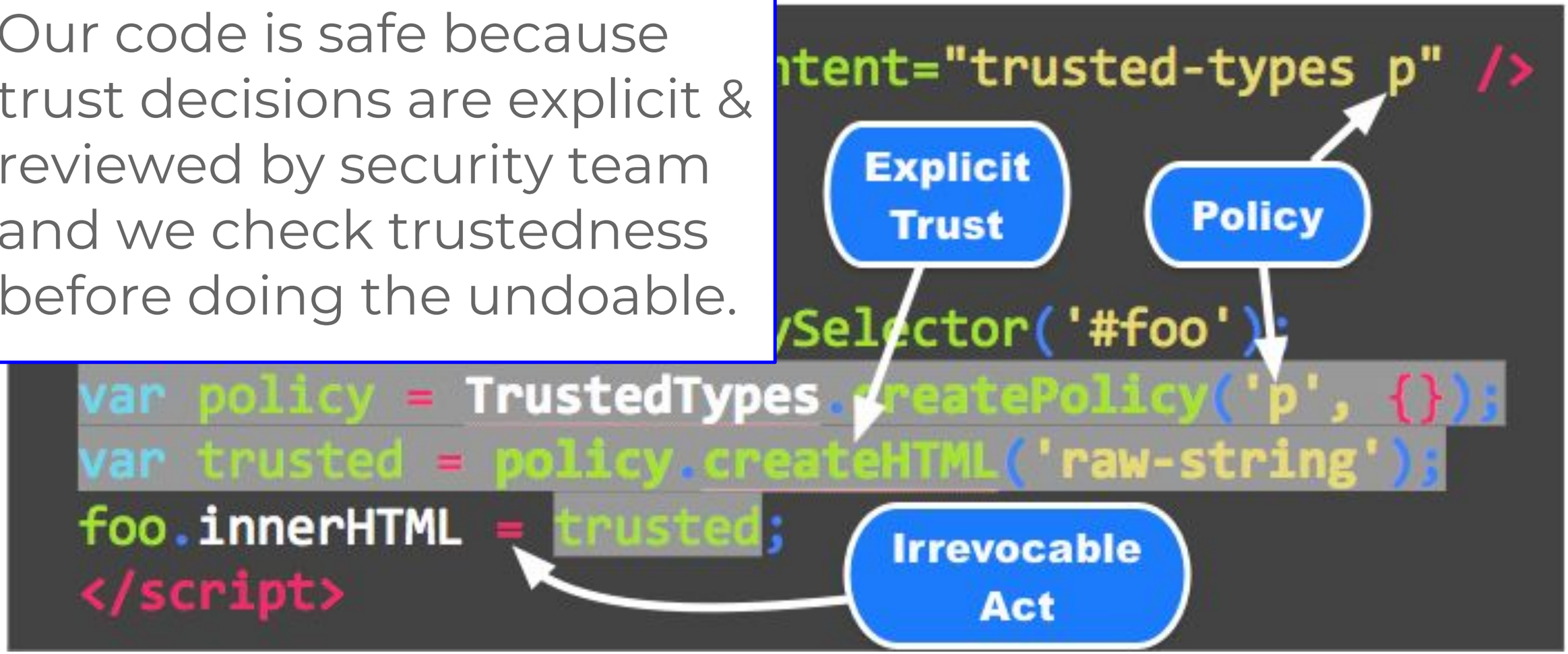
help.github.com/en/articles/about-code-owners

"Code owners are automatically requested for review when someone opens a pull request that modifies code that they own."

```
CODEOWNERS
# Files under sensitive/ need extra sign-off
sensitive/ @securityperson
```

Trusted Types make Trust decisions explicit

Our code is safe because trust decisions are explicit & reviewed by security team and we check trustedness before doing the undoable.



github.com/gothinkster/realworld

RealWorld example apps

Stay on the bleeding edge — join our Gitter room! 🎉

build passing | chat on gitter | Follow 11k

React ⇄ django

See how *the exact same* Medium.com clone (called Conduit) is built using any of our supported frontends and backends. Yes, you can mix and match them, because they all adhere to the same API spec 😮😎

RealWorld React App

public/index.html

```
7       <link rel="stylesheet" href="//demo.productionready.io/main.css">
8       <link href="//code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css"
rel="stylesheet" type="text/css">
9       <link href="//fonts.googleapis.com/css?
family=Titillium+Web:700|Source+Serif+Pro:400,700|Merriweather+Sans:400,700|Sou
rce+Sans+Pro:400,300,600,700,300italic,400italic,600italic,700italic"
rel="stylesheet" type="text/css">
10  +    <meta http-equiv="Content-Security-Policy" content="trusted-types default
react-article-markup">
11  +    <script src="https://wicg.github.io/trusted-
types/dist/es6/trustedtypes.build.js"></script>
12      <!--
13        Notice the use of %PUBLIC_URL% in the tag above.
14        It will be replaced with the URL of the `public` folder during the
```

RealWorld React App

src/components/Article/index.js

```
5   import { connect } from 'react-redux';
6   import marked from 'marked';
7   import { ARTICLE_PAGE_LOADED, ARTICLE_PAGE_UNLOADED } from
    '../../constants/actionTypes';
8   +import { createReactPolicy } from '../../trustedtypes';
9
10  const mapStateToProps = state => ({
11    ...state.article,


19      dispatch({ type: ARTICLE_PAGE_UNLOADED })
20  });
21
22  +const markedSanitizedPolicy = createReactPolicy('article-markup', {
23  +       createHTML: (s) => marked(s, { sanitize: true }),
24  +});
25  +
26  class Article extends React.Component {
```

RealWorld React App

src/components/Article/index.js

```
35          return null;
36        }
37
38  -     const markup = { __html: marked(this.props.article.body, { sanitize: true
    }) };
39        const canModify = this.props.currentUser &&
40          return null;
41        }
42
43  +     const markup = { __html:
    markedSanitizedPolicy.createHTML(this.props.article.body) };
44        const canModify = this.props.currentUser &&
45          this.props.currentUser.username === this.props.article.author.username;
46        return (
```

RealWorld React App

src/index.js

```
+// Allow http: and https: URLs
+createDefaultPolicy({
+    createURL: function(s) {
+      const u = new URL(s, document.baseURI);
+      if (['http:', 'https:'].includes(u.protocol)) {
+        return s;
+      }
+      throw new TypeError('Invalid URL!');
+    },
+});
+
```

RealWorld React App

src/trustedtypes.js

```
+// Dummy policies
+let createReactPolicy = (name, rules) => rules;
+let createDefaultPolicy = (rules) => {};
+
+if (window.TrustedTypes) {
+  createDefaultPolicy = (rules) => window.TrustedTypes.createPolicy('default', rules, true);
+  createReactPolicy = (name, rules) => window.TrustedTypes.createPolicy('react-' + name, rules);
+}
+
+export { createReactPolicy, createDefaultPolicy };⊘↵
```

Tools Integration

Most trust decisions in common infrastructure that is safe-by-construction:

- Template Systems (strict, contextually autoescaped)
- Sanitizers
- Programmatic Builder APIs
- Protobufs bridge server- and client-side trusted values.

Trusted Types

Gets security eyes on decisions to trust

Long experience migrating projects
Reduces XSS payments to vulnerability hunters

Proposed as web standard
Available in Chrome with origin trial token
Contributors in Munich, NY, Prague, Seattle, Zürich

Resources

- Early adopters: trusted-types@googlegroups.com
- Specification: github.com/WICG/trusted-types
- "Securing the Tangled Web" by C. Kern, ACM Queue 2014

@mvsamuel (Ask me about tools integration)