



**black hat**<sup>®</sup>  
USA 2015

**DOM FLOW**  
**UNTANGLING THE DOM FOR EASY BUGS**

# #whoami

Ahamed Nafeez (@skeptic\_fx)

Security Engineer with interest in browsers

Speaker at BlackHat Asia, Hack In The Box, nullc0n,  
c0c0n.

# Overview

Modern web apps and their problems w.r.t pen tests

Hookish! tool and how it works

Dom Flow and its techniques

Few JavaScript / DOM nuances and how to catch them

# Today's web apps



*Knockout.*



# Today's state

Classic XSS is already fading away

Static analysis is becoming harder for client side JS code

Frameworks are getting more complex (JSX?)

# DOM XSS / Javascript injection

XSS triggered due to client side code (Mostly..)

Most generic class of vulnerability on browser.

**Sources** - Entry point for untrusted data

**Sinks** - Executes untrusted data

# The hello world of DOM XSS

[https://damnvulnerable.me/domxss/  
location\\_hash\\_to\\_window\\_eval#firstName](https://damnvulnerable.me/domxss/location_hash_to_window_eval#firstName)

```
var hash = document.location.hash //source  
  
firstName=hash.slice(1)  
  
document.write(firstName) //sink
```

# Common Sources / Sinks

## Sources

	URL	Cookie	referrer	name	postMessage	WebStorage	Total
HTML	1356796	1535299	240341	35466	35103	16387	3219392
JavaScript	22962	359962	511	617743	448311	279383	1728872
URL	3798228	2556709	313617	83218	18919	28052	6798743
Cookie	220300	10227050	25062	1328634	2554	5618	<b>11809218</b>
WebStorage	41739	65772	1586	434	194	105440	215165
postMessage	451170	77202	696	45220	11053	117575	702916
<b>Total</b>	<b>5891195</b>	<b>14821994</b>	<b>581813</b>	<b>2110715</b>	<b>516134</b>	<b>552455</b>	<b>24474306</b>

## Sinks

**25 Million Flows Later - Large-scale Detection of DOM-based XSS (2013)**

*Sebastian Lekies, Ben Stock, Martin Johns*



# String into Code

Everyone(Frameworks, Developers, . .) use  
**'strings'** in a way that directly or indirectly turns  
into code

The DOM specification is rich in doing that

# Direct

`eval()`

`setTimeout`

`Function(x)()`

`execScript(x)`

# Indirect

jQuery's -  $\$(x)$

document.write

Element.setAttribute(x)

Element.innerHTML=x

# jQuery - \$(x)

**\$('#id'), \$('.class'), \$('a')** - Acts as a query selector

**\$('#')** - Creates a new IMG element

**So why is it hard to pen test them?**

# Usually they look like this!

```
cc=function(a){if(a&&"number"==typeof a.length){if(!_bc(a))return function==typeof a.item||"string"==typeof a.item;if(!_fa(a))return function==typeof a.item}return!};dc=
(functionpadding:"cellPadding",cellspacing:"cellSpacing",colspan:"colSpan",frameborder:"frameBorder",height:"height",maxlength:"maxLength",role:"role",rowspan:"rowSpan",type:"type
emapi:"useMap",valign:"vAlign",width:"width");ec=constructor hasOwnProperty isPrototypeOf propertyIsEnumerable toLocaleString toString valueOf".split(" ");
_fc=function(a,c,d){for(var e in a)c.call(d,a[e],e,a)};_gc=function(a){var c=a.length;if(0<c){for(var d=Array(c),e=0;e<c;e++)d[e]=a[e];return d}return[]};_hc=function(a,c)
{return 0<=(0,_.qa)(a,c)};_ic=function(a){var c=_.da(a);return array=="object"==c&&"number"==typeof a.length};j=function(a,c,d){function e(d){d&&c.appendChild(_u(d)?
a.createTextNode(d):d)}for(var f=2;f<d.length;f++){var g=d[f];!_ic(g)||_bc(g)&&0<g.nodeType?e(g):(0,_.ra)(cc(g)?_gc(g):g,e)};
_kc=function(a,c){_fc(c,function(c,e){"style"==e?a.style.cssText=c:"class"==e?a.className=c:"for"==e?a.htmlFor=c:dc.hasOwnProperty(e)?
a.setAttribute(dc[e],c):0==e.lastIndexOf("aria-")||0==e.lastIndexOf("data-")?a.setAttribute(e,c):a[e]=c});_lc=function(a,c){for(var d,e,f=1;f<arguments.length;f++)
e=arguments[f];for(d in e)a[d]=e[d];for(var g=0;g<c.length;g++)d=ec[g],Object.prototype.hasOwnProperty.call(e,d)&&(a[d]=e[d]));_mc=function(a){return "number"==typeof a};
_nc=function(a,c){var d=c[0],e=c[1];if(!_.lb&&e&&(e.name||e.type)){d=["<",d];e.name&&d.push(" name="+_sc(e.name)+"");if(e.type){d.push(" type="+_sc(e.type)+"");var f=
{};_lc(f,e);delete f.type;e.f.push(">");d=d.join("")}d=a.createElement(d);e&&(!_u(e)?d.className=e:_.ea(e)?d.className=e.join(" ")!:_kc(d,e));2<c.length&&j(c,a,d,c);return
d};_oc=function(a){return"/[\s\x0d]*/.test(a)};pc=function(a){var c=[],d=0,e;for(e in a)c[d++]=e;return c};_qc=function(a){var c=[],d=0,e;for(e in a)c[d++]=a[e];return
c};_rc=function(a){return _pa.concat.apply(_pa,arguments)};_sc=function(a,c){var d=(0,_.qa)(a,c),e=(e=0<d)&&_pa.splice.call(a,d,1);return e};_tc=function();
_uc=function(a){_uc[" "]&&a);return a};_uc[" "]&= _tc;_vc=function(a,c){try{return _uc(a[c]),10}catch(d){}return 1};var
xc,yc;_wc=1;_L||9<= _xb;xc=1;_L||9<= _xb;yc= _L&&1;_N("9");1;_M||_N("528");_ub&&_N("1.9b")||_L&&_N("8")||_sb&&_N("9.5")||_M&&_N("528");_ub&&1;_N("8")||_L&&_N("9");
c=function(a,c){this.type=a;this.o=this.target<c;this.C=1;this.R=10};_zc.prototype.stopPropagation=function(){this.C=10};_zc.prototype.preventDefault=function()
{this.R=1;}_Ac= _L?focusin:"DOMFocusIn";_Bc= _M?webkitTransitionEnd:"_sb?otransitionend";_transitionend";_Cc=function(a,c){_zc.call(this,a?
a.type:"");this.A=this.o=this.target=null;this.D=this.keyCode=this.button=this.clientY=this.clientX=0;this.F=this.G=this.w=this.H=1;this.b=this.state=null;a&&this.init(a,c)};
_Cc,_zc);
_Cc.prototype.init=function(a,c){var d=this.type+a.type;this.target=a.target||a.srcElement;this.o=c;var e=a.relatedTarget;e?_ub&&(!_vc(e,"nodeName")||e=null);"mouseover"==
a.fromElement:"mouseout"==d&&(e=a.toElement);this.A=e;this.clientX=void 0!==a.clientX?a.clientX:a.pageX;this.clientY=void 0!==a.clientY?
a.clientY:a.pageY;this.button=a.button;this.keyCode=a.keyCode||0;this.D=a.charCodeAt(0|"keypress"==d?
a.keyCode:0);this.B=a.ctrlKey;this.w=a.altKey;this.G=a.shiftKey;this.F=a.metaKey;this.state=
a.state;this.b=a;a.defaultPrevented&&this.preventDefault();_Cc.prototype.stopPropagation=function(){_Cc.H.stopPropagation.call(this);this.b.stopPropagation?
this.b.stopPropagation():this.b.cancelBubble=10};_Cc.prototype.preventDefault=function(){_Cc.H.preventDefault.call(this);var a=this.b;if(a.preventDefault)a.preventDefault();
if(a.returnValue=11,yc)try{if(a.ctrlKey||112<=a.keyCode&&123>=a.keyCode--1)catch(c){};_Cc.prototype.U=function(){return this.b};
var Fc;_Dc="closure_listenable_"+(1E6*Math.random())|0};_Ec=function(a){return!(!a||!a[_Dc]);Fc=0;var Gc=function(a,c,d,e,f)
{this.listener=a;this.b=null;this.src=c;this.type=d;this.ic=1;e;this.Cc=f;this.key=++Fc;this.Hb=this.Hc=1},Hc=function(a)
{a.Hb=10;a.listener=null;a.b=null;a.src=null;a.Cc=null};var Ic=function(a){this.src=a;this.b={};this.o=0},Kc,Jc;Ic.prototype.add=function(a,c,d,e,f){var
g=a.toString();a=this.b[g];a||(a=this.b[g]={},this.o++);var h=Jc(a,c,e,f);1<h?h?c=a[h],d||{c.Hc=1}:c=new Gc(c,this.src,g,h,e,f),c.Hc=d,a.push(c);return
c};Ic.prototype.remove=function(a,c,d,e){a=a.toString();if(!a in this.b)return 1;var f=this.b[a];c=Jc(f,c,d,e);return 1<c?Hc(f[c]),_pa.splice.call(f,c,1),0==f.length&&delete
this.b[a],this.o--),10):1};
Kc=function(a,c){var d=c.type;if(!d in a.b)return 1;var e=_sc(a.b[d],c)&&Hc(c),0==a.b[d].length&&(delete a.b[d],a.o--);return e};_Lc=function(a,c,d,e,f)
{a=a.b[c.toString()];c--1;a&&(c=Jc(a,c,d,e,f));return 1<c?a[c]:null};Jc=function(a,c,d,e){for(var f=0;f<a.length;f++){var g=a[f];if(!g.Hb&&g.listener==c&&g.ic==1&d&&g.Cc==e)retur
f}return 1};
var Mc,Nc,Oc,Qc,Rc,Sr,Tc,Uc,Vc,Wc,Xc,Yc,Zc;Mc="closure_lm_"+(1E6*Math.random())|0;Nc={};Qc=0;_P=function(a,c,d,e,f){if(!_ea(c)){for(var g=0;g<c.length;g++)_P(a,c[g],d,e,f);return
null}d=_Rc(d);return _Ec(a)?a.$a(c,d,e,f):Sr(a,c,d,11,e,f)};
Sc=function(a,c,d,e,f,g){if(!c)throw Error("x");var h=1;if(!_Tc(a)|!|a[Mc]=1=new Ic(a));d=1.add(c,d,e,f,g);if(d.b)return
d;e=Uc();d.b=je.src=a;e.listener=d;if(a.addEventListener)a.addEventListener(c.toString(),e,h);else if(a.attachEvent)a.attachEvent(Vc(c.toString()),e);else throw
Error("y");Qc++;return d};Uc=function(){var a=Wc,c=xc?function(d){return a.call(c.src,c.listener,d)}:function(d){d=a.call(c.src,c.listener,d);if(!d)return d};return c};
_Xc=function(a,c,d,e,f){if(!_ea(c)){for(var g=0;g<c.length;g++)_Xc(a,c[g],d,e,f);return null}d=_Rc(d);return _Ec(a)?a.xc(c,d,e,f):Sc(a,c,d,10,e,f)};_Yc=function(a,c,d,e,f)
{if(!_ea(c)){for(var g=0;g<c.length;g++)_Yc(a,c[g],d,e,f);else d=_Rc(d),_Ec(a)?a.qd(c,d,e,f):a&&(a=_Tc(a))&&(c=_Lc(a,c,d,11,e,f))&&_Zc(c)};
_Zc=function(a){if(!_mc(a)||!a.Hb)return 1;var c=a.src;if(!_Ec(c))return c.fc(a);var d=a.type,e=a.b;c.removeEventListener?
c.removeEventListener(d,e,a.ic):c.detachEvent&&c.detachEvent(Vc(d),e);Qc--;(d=_Tc(c))?(Kc(d,a),0==d.o&&d.src==null,c[Mc]=null):Hc(a);return 10};Vc=function(a){return a in Nc?
Nc[a]:Nc[a]="on"+a};ad=function(a,c,d,e){var f=10;if(a=_Tc(a))if(c=a.b[c.toString()])for(c=c.concat(),a=0;a<c.length;a++){var g=c[a];g&&g.ic==d&&g.Hb&&
```

# Existing tools (DOM XSS)

Dominator Pro - Dynamic taint tracking using Firefox.

Plethora of static analysis tools - Regex pattern match, Parse JS code and analyse.

What can we look for?



# All cases of DOM Injection

DOM XSS / Javascript injection

DOM based open redirection

Second order DOM injection (XHR, WebSocket)

WebStorage manipulation

# Quirky DOM behaviour

Globally exposed variables in the DOM

DOM Clobbering

Usage of certain methods which could have unforeseen security implications

# [damnvulnerable.me](http://damnvulnerable.me)

**DamnVulnerable.me** is a webapp that is deliberately vulnerable to DOM based attacks.

Its goal is to provide a platform to learn, test and practice DOM based bugs and other exotic cases.

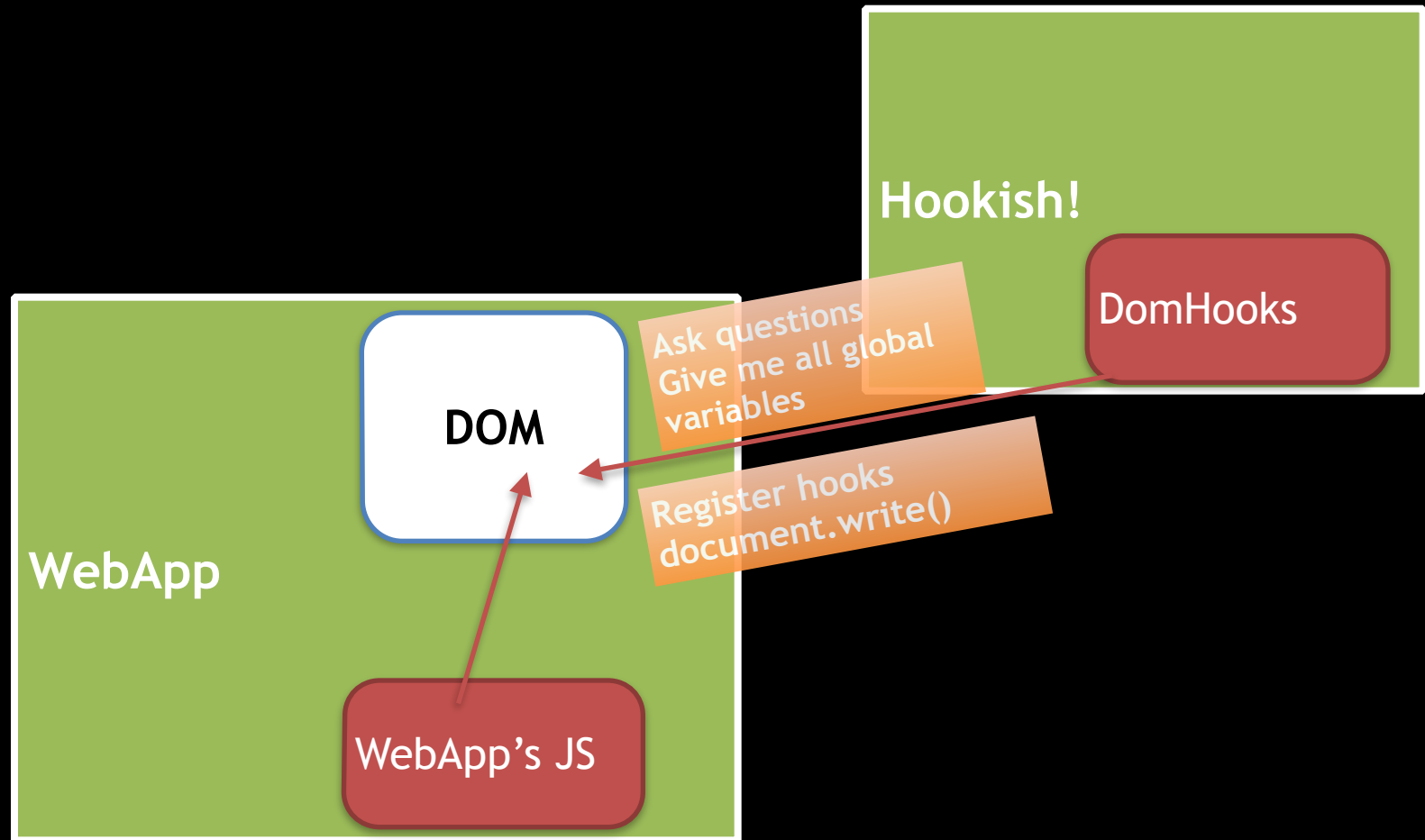
# How Hookish works

Inject DomHooks for sources and sinks

Wait for page to load

Track all sources and sinks

# Injecting DomHooks



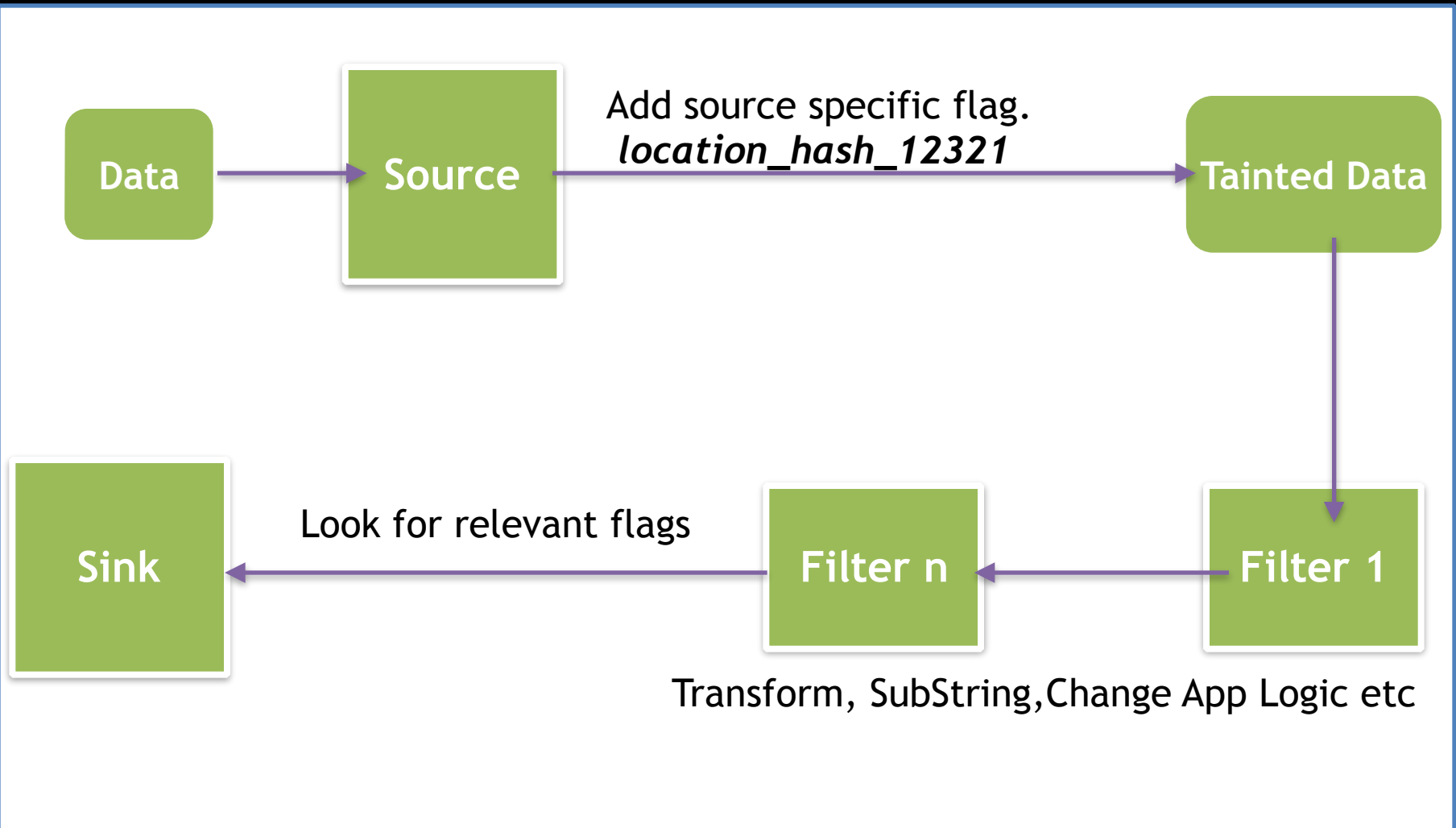
# domhooks.js

Standalone library which selectively registers required DOM properties & methods.

<https://github.com/skepticfx/hookish/blob/master/src/js/domHooks.js>

Can be used in other tools for performance analysis, hardening DOM, DOM based IDS etc.

# DomFlow



# DomFlow- cookie to innerHTML

Every time a cookie is accessed, the data is tagged with a unique flag - ***doc\_cookie\_12391***

This data may go through various transformations.

When a registered innerHTML receives data with this tag, it marks that as a possible DOM XSS.



# Overriding filters

Example: **XHR to innerHTML**

XHR responses are usually JSON content

```
JSON.parse({'data1': 'value1', 'data2': 'value2'})
```

```
Object.Stringify({'data1': 'value1Flag', 'data2':  
                  'value2Flag'})
```

# Boxing strings in JS

```
var str = "hello"
```

```
typeof str; // string
```

```
str.flag = true;
```

```
// JS propagates this string flag in most cases
```

# Navigating across the flows

window_eval	sink	console.log(1)	http://damnvulnerable.me
dom_nodes	sink	<ul style="list-style-type: none"><li>↳ xhr.onreadystatechange (http://damnvulnerable.me/js/domxss/xhr_to_eval.js:17:9)</li><li>↳ Object.c.(anonymous function) [as dispatchEvent] (&lt;anonymous&gt;:224:102)</li></ul>	http://damnvulnerable.me
document_cookie	sink	<ul style="list-style-type: none"><li>↳ n (&lt;anonymous&gt;:289:124) in AndCantPossiblyBeSet= ...</li><li>↳ d (&lt;anonymous&gt;:296:129)</li><li>↳ d (&lt;anonymous&gt;:296:74)</li><li>↳ C (&lt;anonymous&gt;:297:19)</li><li>↳ XMLHttpRequest.H.onreadystatechange (&lt;anonymous&gt;:302:45)</li></ul>	http://damnvulnerable.me

# Getting the stack trace in V8 Engine

Dynamically throw the error and filter to remove  
Hookish! specific stacks

Easily integrates with Chrome's dev tools and  
helps analyse vulnerable lines of code

# Tracking status of all hooks

[domstorm.skepticfx.com](http://domstorm.skepticfx.com)

Hooking Storage objects,

[http://domstorm.skepticfx.com/modules?  
id=529d4f84090faf000000000002](http://domstorm.skepticfx.com/modules?id=529d4f84090faf000000000002)

# Second order DOM injection

DOM injection where the sources doesn't flow directly.

Rather, they are fetched from a persistent storage at some point.

XHR/WS response flowing in to sinks

# Four Scenarios

The following 4 scenarios talks about bugs/special cases that are often missed while security testing a web app.

Hookish! tool is built to easily find / analyse such bugs

# 1. Do you check how XHR responses are handled in your application?

Most common issue which pen testers miss / scanners usually ignore.

The choke point is how you treat these data before populating into the DOM (regardless of how you store untrusted input)



# XHR response - innerHTML

```
var response = JSON.parse(xhr.responseText);  
  
var description = response.description;  
  
var div = document.getElementById('vulnerableDiv');  
  
div.innerHTML = description;
```

## 2. DOM Clobbering using Global Variables

Consider an IFrame sandbox which executes arbitrary code.

Exposed global variables can change logic in parent window.

# Classic Iframe sandboxing

Trusted Parent window

Untrusted but sandboxed IFrame child

```
<iframe sandbox="allow-scripts"></iframe>
```

Defaults to origin 'null'

# About this sandbox

IFrame sandboxes have 'null' origin.

The JS in sandboxed IFrame should not interact with the parent Window's DOM.

<http://www.html5rocks.com/en/tutorials/security/sandboxed-iframes/>

# Spot the bug and break out of this sandbox

[https://damnvulnerable.me/misc/  
insecure\\_global\\_variable](https://damnvulnerable.me/misc/insecure_global_variable)

# Setting global variables using window.name

DOM sets the name of iframe windows to the window object (DOM CLOBBERING)

Trusted Parent window

window name is SECURE\_FLAG    No window name

Untrusted but sandboxed IFrame child

```
<iframe sandbox="allow-scripts"></iframe>
```

```
<script>  
name= 'SECURE_FLAG'  
</script>
```

This sets the global variable `SECURE_FLAG` in the parent window's DOM and bypasses the security check

# 3. Redirect parent window while opening links in new tab

<https://hackerone.com/reports/23386>

Works on Chrome and Firefox.



# Opening links in new tab

Parent window

```
<a href="website.com" target="_blank"> </a>
```

New tab (Can be malicious)

```
window.opener.location.reload('phishing-page.com')
```

**window.opener** should be null always and should not be accessible by another Cross-Domain window

# Finding anchor tags with target=\_blank

Easy to find on static HTML pages.

In modern apps, usually anchor tags are dynamically inserted in to the DOM.

Hookish! finds these after the DOM is rendered and all anchor tags are populated.

**Not a serious issue most of the times, but depends on where you have these new links.**

# 4. Custom templating engines

```
var data = {'name': 'mark', 'age': '23'}
```

Welcome to this page, **<%- data.mark %>**

# How would some one write a templating engine using JavaScript?

1. Load the template **data** object and encode it.
2. Find the template pattern
3. Use `string.replace(pattern, matching_data)`

# A simple templating code

```
var inputHTML = "<img src='PLACEHOLDER'>";

function doTemplating(){

    var input = document.getElementById('id_input').value;

    input = filterInput(input);

    var finalHTML = inputHTML.replace("PLACEHOLDER", input);

    console.log(finalHTML);

    document.write('Your input: </br>' + input);

    document.write(finalHTML);

}
```

# The bypass

```
$` onerror=alert(1);//
```

# String.prototype.replace

ECMAScript's String.replace is the culprit

<http://www.ecma-international.org/ecma-262/5.1/#sec-15.5.4.11>

Table 22 — Replacement Text Symbol Substitutions

Characters	Replacement text
\$\$	\$
\$&	The matched substring.
\$`	The portion of <i>string</i> that precedes the matched substring.
\$'	The portion of <i>string</i> that follows the matched substring.
\$n	The $n^{\text{th}}$ capture, where $n$ is a single digit in the range 1 to 9 and $\$n$ is not followed by a decimal digit. If $n \leq m$ and the $n^{\text{th}}$ capture is <b>undefined</b> , use the empty String instead. If $n > m$ , the result is implementation-defined.
\$nn	The $nn^{\text{th}}$ capture, where $nn$ is a two-digit decimal number in the range 01 to 99. If $nn \leq m$ and the $nn^{\text{th}}$ capture is <b>undefined</b> , use the empty String instead. If $nn > m$ , the result is implementation-defined.



# Work in progress

Patching chromium to have V8 level tainting and enable overriding of Objects that are not possible now.

# Thanks

More questions

@skeptic\_fx