



OWASP

Open Web Application  
Security Project

# PostMessage Security in Chrome Extensions

Arseny Reutov

[areutov@ptsecurity.com](mailto:areutov@ptsecurity.com)

<https://raz0r.name>

# \$ whoami

- Web application security researcher at Positive Technologies
- Member of Positive Hack Days (<https://phdays.com>) conference board
- Occasional web security blogger (<https://raz0r.name>)



OWASP

Open Web Application  
Security Project

WWW.OWASP.ORG

# Agenda

- Chrome extensions & their messaging
- PostMessage security considerations
- Mounting extensions analysis
- The results!
- The takeaways



**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)

Part I

# CHROME EXTENSIONS & THEIR MESSAGING



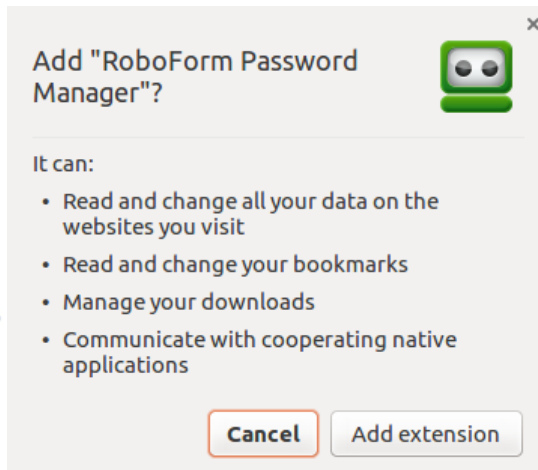
**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)

# Chrome extensions ecosystem

- Chrome Web Store is notoriously known in terms of security (unintuitive permissions dialogs, malware & insecure extensions)

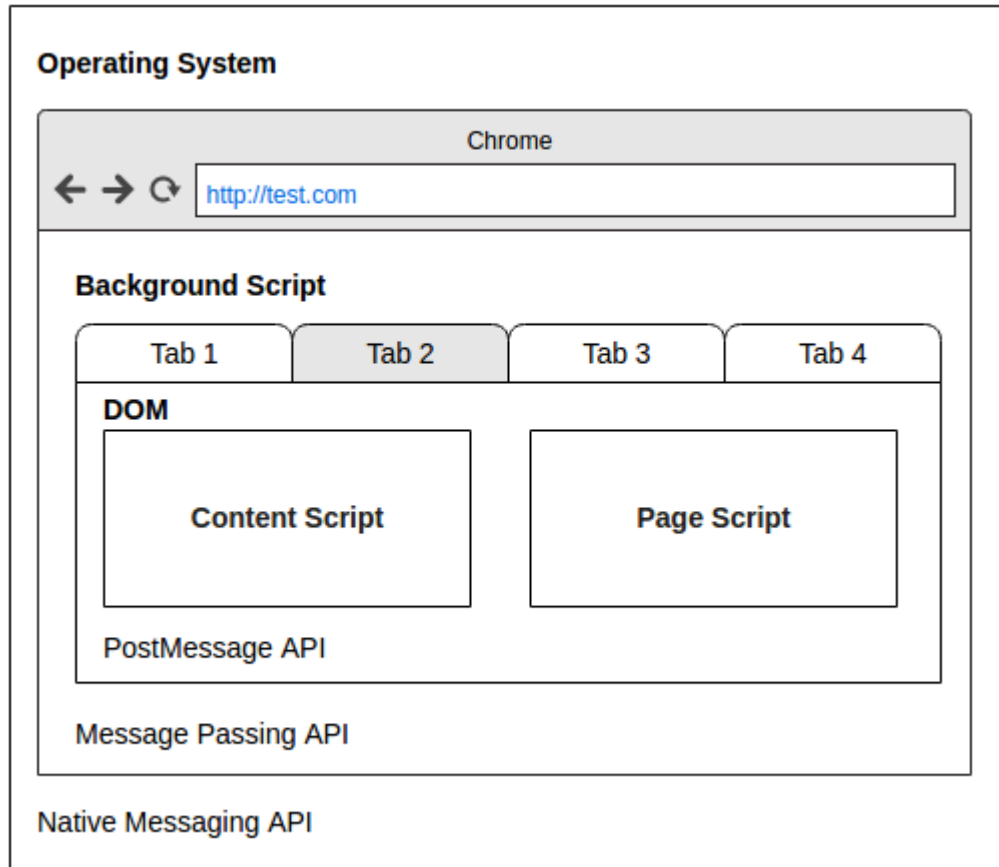


OWASP

Open Web Application  
Security Project

WWW.OWASP.ORG

# Chrome extensions messaging



**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)

# Extension manifest file

```
{  
  "name": "My Extension",  
  "description": "My Super Chrome Extension",  
  "version": "1.0",  
  "background": {  
    "scripts": ["js/background.js"]  
  },  
  "content_scripts": [  
    {  
      "matches": ["<all_urls>"],  
      "js": ["js/jquery.js", "js/content.js"]  
    }  
  ],  
  "permissions": ["tabs", "http://*/**", "https://*/**"]  
}
```

Part II

# POSTMESSAGE SECURITY CONSIDERATIONS



**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)



# PostMessage API

`window.postMessage()` method enables cross-origin communication

```
somewindow.postMessage(  
    "my message", // message data  
    "*",         // target origin  
);
```



# PostMessage API

Developer is in charge of origin validation

```
window.addEventListener("message", receiveMessage, false);

function receiveMessage(event) {
  if (event.origin !== "http://example.org")
    return; // checking origin host
  if (event.source !== window)
    return; // or origin window
  process(event.data);
}
```

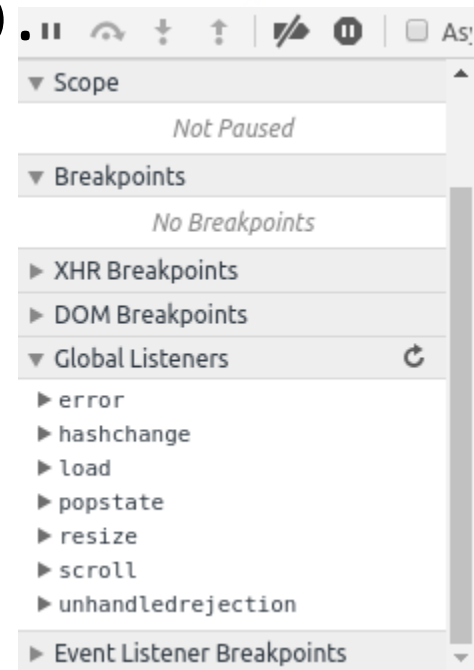


# PostMessage API

- If origin validation is absent or is flawed, an attacker's message data can reach dangerous pieces of code.
- See “[The pitfalls of postMessage](#)” by Mathias Karlsson for common origin validation bypasses.

# PostMessage API

- Unlike other DOM events, message propagation to listeners cannot be stopped via `return false` or `stopPropagation()`.
- Extensions' message listeners are not listed in Chrome Developer Tools.



# PostMessage Attack Vectors

## Method 1: iframes

```
var iframe = document.createElement("iframe");  
iframe.src = "http://target.com";  
iframe.contentWindow.postMessage("some message", "*");
```

Pros: stealthy

Cons: killed by X-Frame-Options and framebusters



**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)

# PostMessage Attack Vectors

## Method 2: opening a new window

```
var targetwindow = window.open("http://target.com");
targetwindow.onload = function() {
  targetwindow.postMessage("some message", "*");
}
```

Pros: not affected by X-Frame-Options

Cons: more noisy



# PostMessage in Chrome extensions

- Chrome extensions use postMessage API to receive messages from external web sites (e.g. translator services) or within the same origin (especially in developer tools extensions)
- postMessage data can be passed into background script context, and in some cases even reach OS via Native Messaging API



Part III

# MOUNTING EXTENSIONS ANALYSIS



**OWASP**

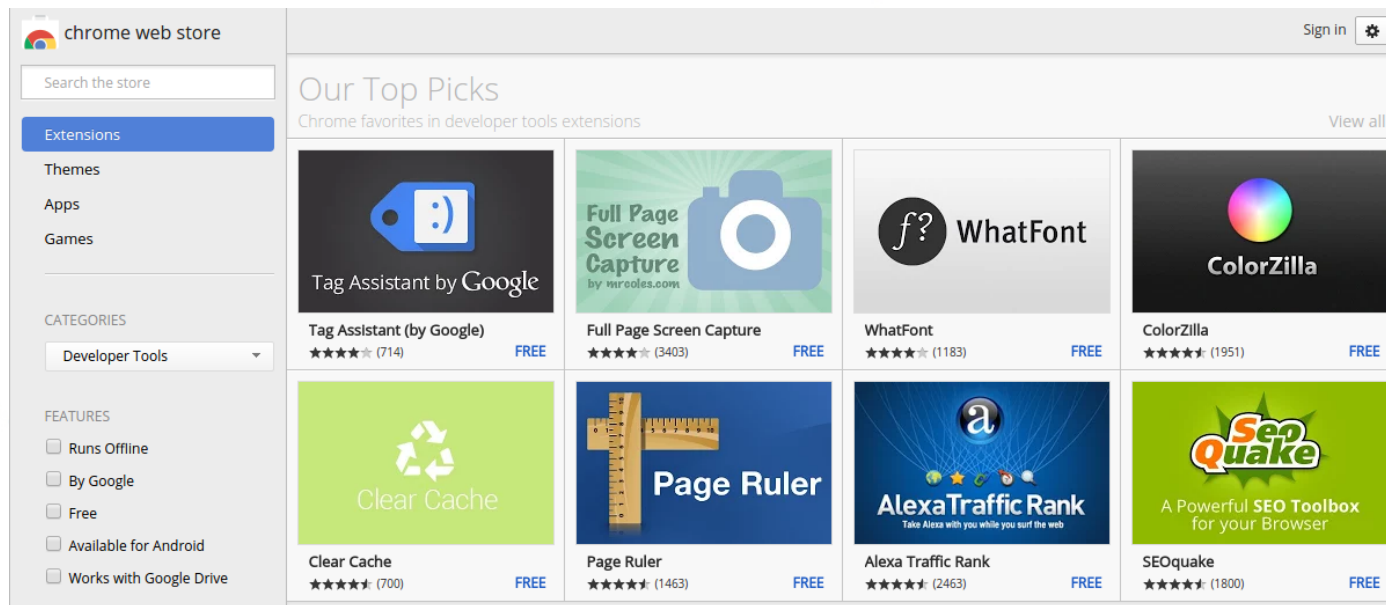
Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)



# The Research Steps

- Download extensions (Web Development category only)



The screenshot displays the Chrome Web Store interface. On the left, there is a navigation sidebar with the following sections:

- chrome web store** (with logo)
- Search the store (input field)
- Extensions (highlighted)
- Themes
- Apps
- Games
- CATEGORIES: Developer Tools (selected)
- FEATURES:  Runs Offline,  By Google,  Free,  Available for Android,  Works with Google Drive

The main content area is titled "Our Top Picks" and "Chrome favorites in developer tools extensions". It features a grid of extension cards:

Extension Name	Rating	Price
Tag Assistant by Google	★★★★☆ (714)	FREE
Full Page Screen Capture	★★★★☆ (3403)	FREE
WhatFont	★★★★☆ (1183)	FREE
ColorZilla	★★★★☆ (1951)	FREE
Clear Cache	★★★★☆ (700)	FREE
Page Ruler	★★★★☆ (1463)	FREE
Alexa Traffic Rank	★★★★☆ (2463)	FREE
SEOquake	★★★★☆ (1800)	FREE



# The Research Steps

- Parse CRX files  
(<https://github.com/vladignatyev/crx-extractor>)
- Convert to ZIP
- Unpack



# The Research Steps

- Parse Manifest file, find content scripts
- Parse each content script with Acorn JS parser (<https://github.com/ternjs/acorn>)
- Look for postMessage listeners with an Acorn plugin

# The Research Steps

- Log each postMessage listener found into local elasticsearch



The screenshot shows the Kibana interface with a search query for `postMessagesCount:>0`. The search results are displayed in a table with columns for `name` and `users`. The results show three entries: Happalyzer (467,645 users), CryptoPro Extension for C#ES Browser Plug-in (445,686 users), and Facebook Pixel Helper (422,086 users). The interface also shows a sidebar with navigation options and a search bar.

name	users
Happalyzer	467,645
CryptoPro Extension for C#ES Browser Plug-in	445,686
Facebook Pixel Helper	422,086



OWASP

Open Web Application  
Security Project

WWW.OWASP.ORG

Part IV

# THE RESULTS



**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)


# React Dev Tools


- Have got `postMessage` protection just recently by an external PR:

## Validate the source of global messages (#561)

The global "message" event can be activated by pages from a different origin. Blindly accepting such messages is a security risk.

A message channel in React Devtools always have exactly two endpoints. At each side, the target of `postMessage` is also the expected source of the global message event. To resolve the security risk, I added a check to the message event handler to reject messages from unexpected sources.

 master (#561)

 **Rob--W** committed with `gaearon` 22 days ago



**OWASP**

Open Web Application  
Security Project

WWW.OWASP.ORG

# React Dev Tools

- Prior to the fix message was validated by just checking a special property (which is user controlled):

```
27 window.addEventListener('message', welcome);
28 function welcome(evt) {
29 + if (evt.source !== window || evt.data.source !== 'react-devtools-content-script') {
30     return;
31 + }
32
```



# Ember Inspector

- No origin validation, but, luckily, data does not reach sensitive parts.

```
/**
 * Add an event listener for window.messages.
 * The initial message from EmberDebug is used to setup the event listener
 * that proxies between the content-script and EmberDebug using a MessagingChannel.
 *
 * All events from the window are filtered by checking that data and data.type
 * properties exist before sending messages on to the background-script.
 *
 * See:
 *   https://developer.mozilla.org/en-US/docs/Web/API/Channel\_Messaging\_API
 */
window.addEventListener('message', function(event) {
  // received initial message from EmberDebug
  if (event.data === 'debugger-client') {
    var emberDebugPort = event.ports[0];
    listenToEmberDebugPort(emberDebugPort);
  } else if (event.data && event.data.type) {
    chrome.extension.sendMessage(event.data);
  }
});
```





# AngularJS Batarang (Angular v1.x)

- Developers have no clue how to validate origin

```
window.addEventListener('message', function(evt) {
  // There's no good way to verify the provenance of the message.
  // evt.source === window is true for all messages sent from
  // the main frame. evt.origin is going to be the webpage's origin,
  // even if the message originated from a chrome:// script you injected.

  // The only thing we can do is see if the message *looks* like something
  // we would send, cross our fingers, and send it on.
  // Thus, we check for one of the properties known to be on *all* of our
  // messages (__fromBatarang === true).
  var eventData = evt.data;
  // NOTE: Check for null before checking for the property, since typeof null === 'object'.
  if (typeof eventData === 'object' && eventData !== null
      && eventData.hasOwnProperty('__fromBatarang') && eventData.__fromBatarang) {
    chrome.runtime.sendMessage(eventData);
  }
});
```



OWASP

Open Web Application  
Security Project

WWW.OWASP.ORG

# Augury (Angular v2.x)

- Again, origin validation is just checking a magic string

```
window.addEventListener('message', function(event) {
  exports.browserDispatch(event.data);
});

exports.browserDispatch = function(message) {
  if (message_1.checkSource(message) === false) {
    return;
  }
  /* snip */
}

exports.messageSource = 'AUGURY_INSPECTED_APPLICATION';
exports.checkSource = function(message) {
  return message.messageSource === exports.messageSource;
};
```



# Augury (Angular v2.x)

- Augury employs interesting message serialization:

```
exports.serialize = function(value) {  
    return "return " + serializer(value);  
};  
exports.deserialize = function(value) {  
    return (new Function(value))();  
};
```



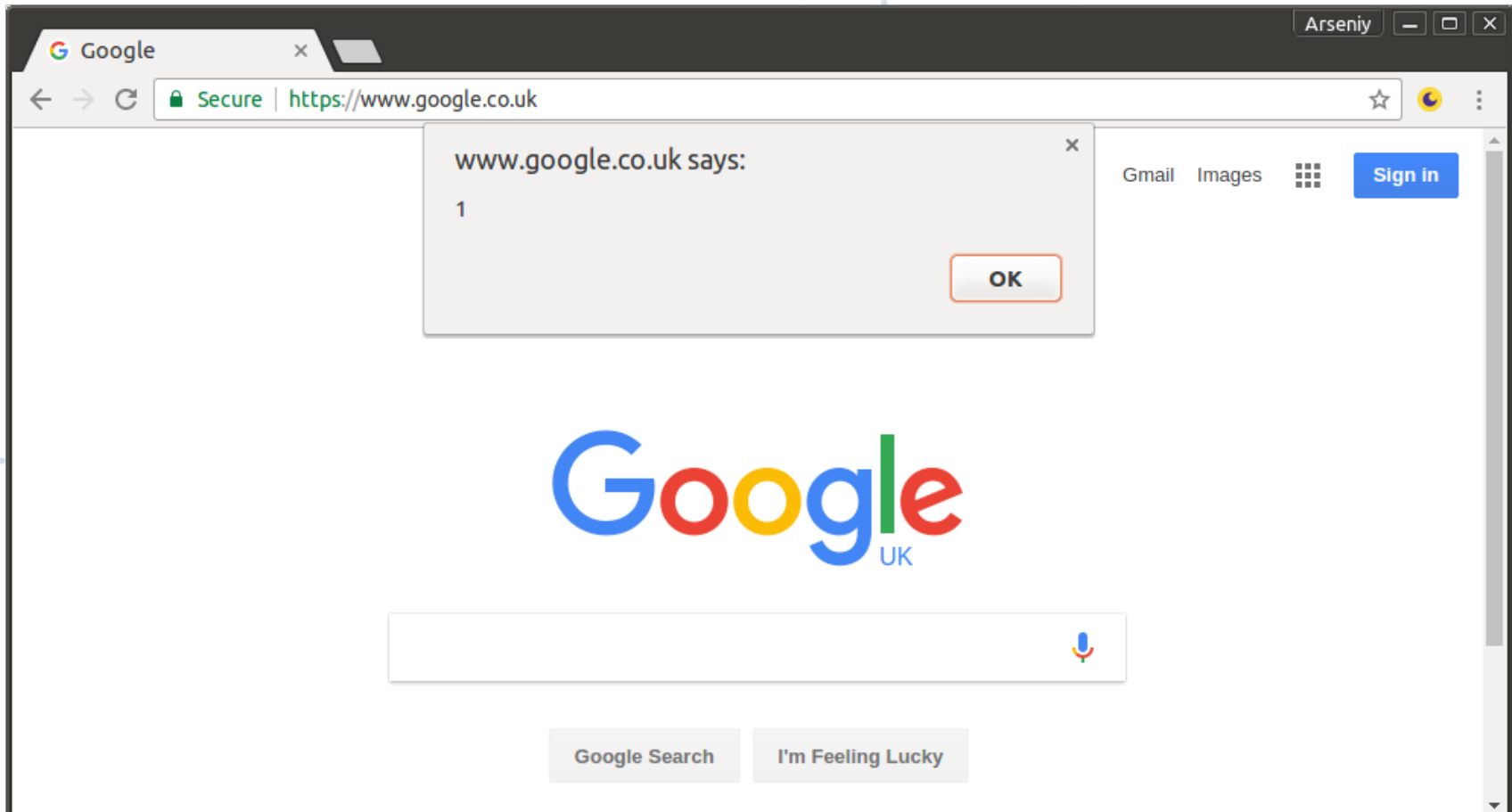
# Augury (Angular v2.x)

- XSS on any website with the extension installed

```
targetWindow.postMessage({
  messageSource: 'AUGURY_INSPECTED_APPLICATION',
  messageType: 1,
  serialize: 2,
  content: 'alert(1)'
}, '*')
```



# Augury (Angular v2.x)



# LanSweeper Shell Execute



## LanSweeper Shell Execute

offered by [www.lansweeper.com](http://www.lansweeper.com)

★★★★★ (14)

[Developer Tools](#)

24,248 users

+ ADD TO CHROME



OVERVIEW

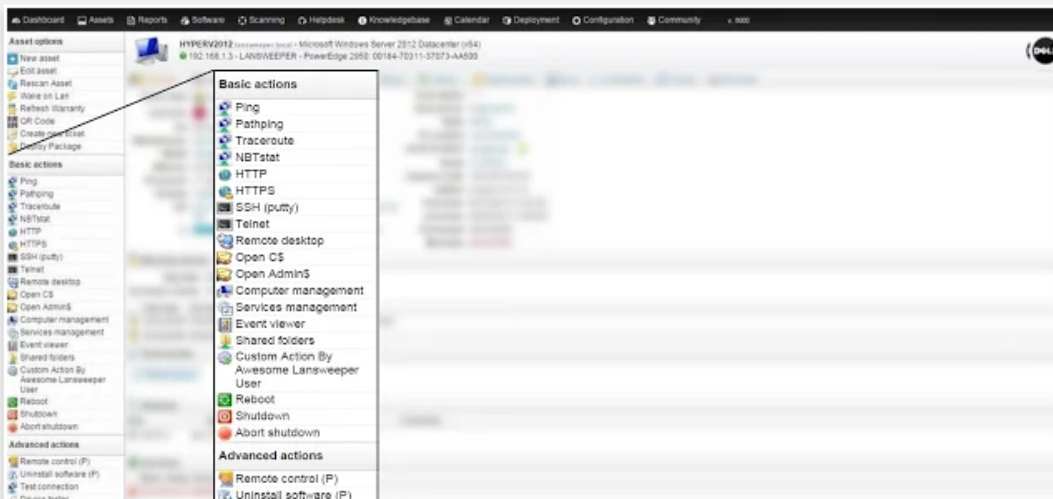
REVIEWS

SUPPORT

RELATED

G+

10



Compatible with your device

Runs various commands on Windows systems on behalf of LanSweeper Asset Management Software.

[Website](#)

[Report Abuse](#)

### Additional Information

Version: 1.0.2

Updated: September 15, 2015

Size: 35.02KIB

Language: English



OWASP

Open Web Application  
Security Project

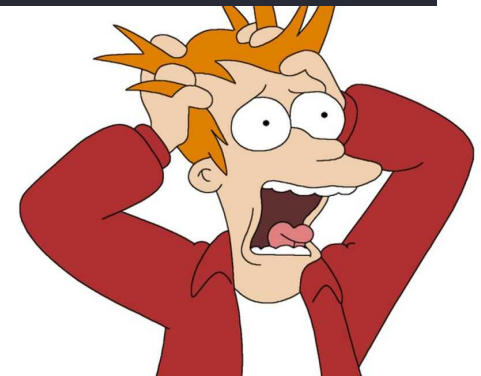
WWW.OWASP.ORG

# LanSweeper Shell Execute



# LanSweeper Shell Execute

```
//Lets take message from site and send it to the native host.  
//The message should be a object like the one given in the below example  
// var msg = { "application" : "LanSweeper", "command" : "ping google.com" };  
//  
// Attribute application should always be set to "LanSweeper"  
window.addEventListener("message", function(e) {  
    var msg = new Object();  
    msg.origin = e.origin;  
    if (e.data.application !== undefined && e.data.application == "LanSweeper") {  
        msg.command = e.data.command;  
        chrome.runtime.sendMessage(msg);  
    }  
});
```



OWASP

Open Web Application  
Security Project

WWW.OWASP.ORG



Part V

# THE TAKEAWAYS



**OWASP**

Open Web Application  
Security Project

[WWW.OWASP.ORG](http://WWW.OWASP.ORG)

# The takeaways

- For users:
  - do not install shady extensions from unknown publishers
  - check requested permissions

# The takeaways

- For developers:
  - pay attention to origin validation in message listeners
  - consider origin bypass tricks
  - do not rely on magic strings



# The takeaways

- For browsers:
  - should provide built-in origin validation
  - see [getMessage](#) proposal by [@homakov](#)





OWASP

Open Web Application  
Security Project

Thank you!