

## Sabre uses JavaScript level information flow tracking to detect confidentiality and integrity violations in JSEs

### JSEs affect Browser Security

#### Inadequate sandboxing of JavaScript

- JSEs not constrained by Same Origin Policy
- Browsers APIs extend JavaScript functionality

#### Consequence: Malicious JSEs misuse privileges

- JSEs can access sensitive data
  - Cookies, browsing history, form fields, etc.
- JSEs can communicate across domains
  - Can send XMLHttpRequest to *any* domain
  - Cross-domain interfaces (XPCOM) allow access to file system and network

#### Example: FFsniff emails passwords in form fields

```
function do_sniff() {
  var hesla = window.content.document.getElementsByTagName("input");
  data = "";
  for (var i = 0; i < hesla.length; i++) {
    if (hesla[i].value != "") {
      data += hesla[i].type + ":" + hesla[i].name + ":" + hesla[i].value + "\n";
    }
  }
  // the rest of the code sends 'data' via an email message.
}
```

#### Browser and JSE vulnerabilities

- Malicious websites can exploit browser and JSE bugs
  - Involve subtle browser/JSE interactions

#### Consequence: Remote attackers violate confidentiality and integrity

- Attacker can execute JavaScript code with JSE privileges
  - Can access the local file system
  - Can start new process

#### Attack against GreaseMonkey/Firefox bug

```
1. <script type="text/javascript">
2. window._GM_xmlhttpRequest = null;
3. function trapGM(...) {
4.   window._GM_xmlhttpRequest = window.GM_xmlhttpRequest;
5.   ...
6. }
7. function checkGM() {
8.   if (window._GM_xmlhttpRequest) {
9.     window._GM_xmlhttpRequest(
10.    {method: 'GET', url: 'file:///c:/boot.ini',
11.    onload: function(Response) {
12.      document.formname.textfield.value = Response.responseText;
13.    }});
14.   }
15. }
16. if (typeof window.addEventListener != 'undefined') {
17.   window.watch('GM_apis', trapGM);
18.   window.addEventListener('load', checkGM, true);
19. }
20. </script>
```

### Security Architecture for Browser Extensions

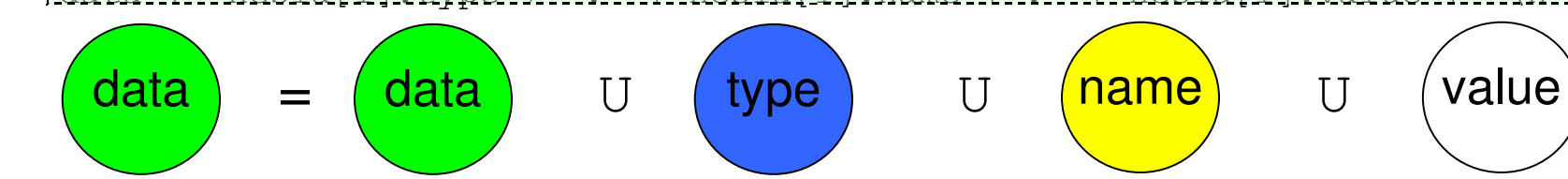
#### Security Labels

- Sabre modifies JavaScript interpreter's object representation to store security labels
  - Ensures all JavaScript objects have labels
  - Eases label propagation
- Separate confidentiality and integrity labels
- Each label consists of
  - Sensitivity level of associated JavaScript object
  - Provenance information
- Objects and properties have individual labels
  - Properties initially inherit parent object's label

#### Label Propagation

- Modified JavaScript interpreter to propagate labels
- Sabre handles assignments, function calls and control structures

```
var hesla = window.content.document.getElementsByTagName("input");
data += hesla[i].type + ":" + hesla[i].name + ":" + hesla[i].value + "\n";
```



- Labeled scopes handle some implicit flows
  - Sabre cannot handle all implicit flows
  - Requires static analysis of JavaScript code

- Sabre uses the provenance of each JS instruction to determine whether an object is modified by a JSE.

#### Sabre detects flows from sensitive sources to low-sensitivity sinks

##### Sources

- High-confidentiality sources
  - DOM, persistent data, file system
- Low-integrity sources
  - User input, network



##### Sinks

- Low-confidentiality sinks
  - File system, network
- High-integrity sinks
  - File system, process



### Sabre Design

#### Design Goals

- Monitor all JavaScript execution
  - including code executed in web applications, JSEs & browser core
- Ease action attribution
- Track information across browser sub-systems

#### Challenges and Optimizations

- Problem:** Browser executes scripts that may violate confidentiality and integrity
- Solution:** Sabre raises alert only when an object is modified by a JSE

- Problem:** JSEs can contain cross domain objects

```
var cookieMgr = Components.classes["@mozilla.org/cookieManager;1"].
  getService(Components.interfaces.nsICookieManager);
var e = cookieMgr.enumerator;
```

- cookieMgr is not a JavaScript object

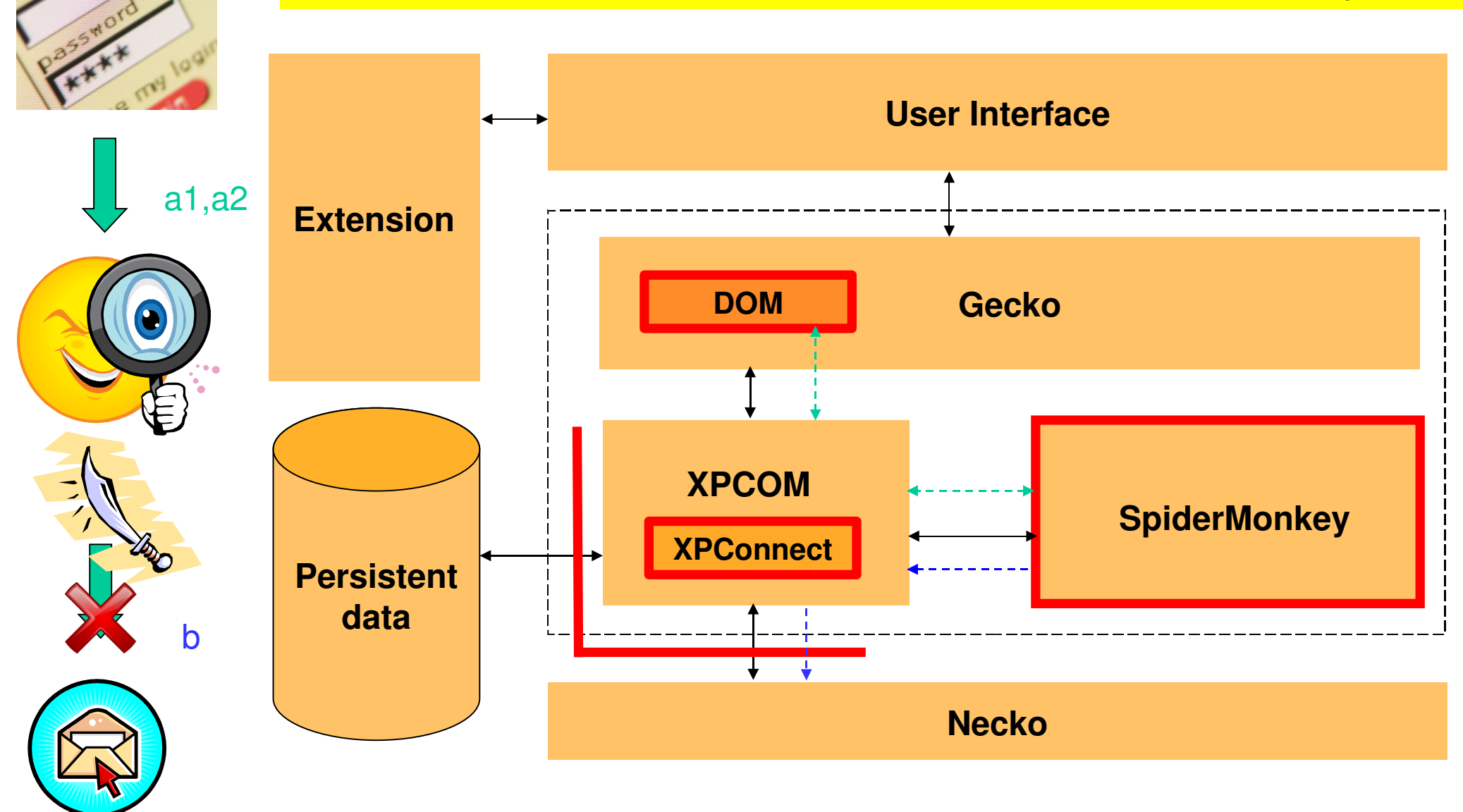
```
sis.init(is); // initializes a nsIScriptableInputStream object (sis)
// using a nsIInputStream object (is).
```

- init() is not a JavaScript method

- Solution:** Sabre has wrappers for cross domain objects and method calls

- Problem:** Benign JSEs may contain information flows violations
- Solution:** Sabre *declassifies* or *endorses* legal flows

#### Sabre monitors & controls information flow across browser sub-systems



### Evaluation

- Used a suite of 24 JSEs, with over 120K lines of JavaScript code

#### Benign JSEs (20 popular Firefox JSEs)

JSE	Advertised Functionality of JSE	1	2	3	4	5
1. Adblock Plus	Prevent page elements, such as ads, from being downloaded	✓	✓	✓	✓	✓
2. All-in-One-Sidebar	Sidebar control to switch between sidebar panels and view dialog windows	✓	✓	✓	✓	✓
3. CoolPreviews	Preview links and images without leaving current page or tab.	✓	✓	✓	✓	✓
4. Download Statusbar	Manage downloads from a tidy statusbar	✓	✓	✓	✓	✓
5. Fast Video Download	Easy download of video files from popular sites	✓	✓	✓	✓	✓
6. Forecastfox	Gets weather forecasts from AccuWeather.com	✓	✓	✓	✓	✓
7. Foxmarks Synchronizer	Keeps bookmarks and passwords backed up and synchronized	✓	✓	✓	✓	✓
8. Ghostery	Alerts user's about web bugs, ad networks and widgets on webpages	✓	✓	✓	✓	✓
9. GooglePreview	Inserts thumbnails and ranks of web sites into Google search results	✓	✓	✓	✓	✓
10. Greasemonkey (0.8.1)	Allows users customize webpages with user scripts	✓	✓	✓	✓	✓
11. NoScript	Restricts executable content to trusted domains	✓	✓	✓	✓	✓
12. PDF Download	Tool for handling, viewing and creating Web-based PDF files	✓	✓	✓	✓	✓
13. Pwdhash	Customizes user passwords to domains to prevent phishing	✓	✓	✓	✓	✓
14. SpeedDial	Easy access to frequently visited websites	✓	✓	✓	✓	✓
15. StumbleUpon	Discovers web sites based on user's interests	✓	✓	✓	✓	✓
16. Stylish	Easy management of user styles to enhance browsing experience	✓	✓	✓	✓	✓
17. Tab Mix Plus	Enhances Firefox's tab browsing capabilities	✓	✓	✓	✓	✓
18. User Agent Switcher	Switches the user agent of the browser	✓	✓	✓	✓	✓
19. Video DownloadHelper	Tool for web content extraction	✓	✓	✓	✓	✓
20. Web-of-Trust	Warns users before they interact with a harmful site	✓	✓	✓	✓	✓

Behavior key: (1) HTML forms; (2) HTTP channels; (3) File system; (4) Loading URLs; (5) JavaScript events.

**Result:** Whitelisted benign flows for all 20 JSEs, no false positive

**Malicious JSEs:** GreaseMonkey v0.3.3, Firebug v1.01, FFsniff and BrowserSPY

**Result:** Precisely identified all flow violations

### Performance Overheads

#### Test Setup

- Integrated Sabre with Firefox 2.0.0.9.
- 2.33Ghz Intel Core2 Duo, 3GB RAM, Ubuntu 7.10

	Sabre	No provenance check
SunSpider	6.10x	77%
V8	2.36x	42%

\* Discounting results of 3d-morph, access-nsieve and bitops-nsieve-bits.

#### Memory Consumption

- 15 minute workload of moderate to heavy JS pages
- Sabre increased memory consumption by 74%

### Future Work

- Automatic placement of declassifiers to aid developers
- Dynamic slicing to automatically identify offending JavaScript and ease action attribution
- Enhance Sabre with static analysis support