Run-Time Monitoring and Formal Analysis of Information Flows in Chromium

Lujo Bauer, Shaoying Cai*, Limin Jia, Timothy Passaro, Michael Stroucken, and Yuan Tian

Carnegie Mellon University
* Institute for Infocomm Research
Websites increasingly host sensitive services

Passwords
Bank account numbers
Emails
......
Confidential data could be revealed to …

Passwords
Bank account numbers
Emails
……
Browser architecture & security mechanisms

Dynamic entities

- tab
- main page
- DOM tree
- iframe page
- image
- form field
- scripts
- same-origin
- 3rd-party scripts
- extension
- content scripts
- extension core

Static entities

- cookies
- bookmarks
- history
- API
Browser architecture & security mechanisms

Same origin policy (SOP)
Browser architecture & security mechanisms

Process isolation

Isolated worlds

tab

main page

DOM tree

image form field

……

iframed page

same-origin scripts

3rd-party scripts

extension content scripts

same-origin scripts

3rd-party scripts

event

extension content scripts

event

extension core

extension core
Permissions and content security policy (CSP)
Risks to users’ data remain

Evil Extension
(Masquerading as a translation extension)
Proposed solutions

Carnegie Mellon University
CyLab

JSFlow, …
[ Arden et al. 2012,
Austin and Flanagan 2012,
Bichhawat et al. 2014,
Chugh et al. 2009,
Hedin et al. 2014,
Hedin and Sabelfeld 2012]

COWL, BFlow
[ Stefan et al. 2014, Yip et al.
2009 ]

FlowFox
[ Groef et al. 2012 ]
Our approach: Run-time information-flow control

- Uses coarse-grained dynamic taint tracking
- Encompasses wide range of browser entities
- Supports rich policy specification
- Formalized and proved noninterference
- Functional prototype implementation on Chromium
Our approach

Dynamic entities

- Labels represent policy
- Communications are mediated
- Labels change with tainting
Example walkthrough

Carnegie Mellon

Evil Extension

Password Manager

Example walkthrough

1. current secrecy level
2. ceiling

integrity label:
API permissions

secrecy label:
two sets of secrecy tags

ID:
Password:

Log in

{{cnn}, {cnn.*}},
{};
{+network}
Example: before injecting scripts

ID: 
Password: 
Log in

{{cnn}, {cnn.*}}, 
{}, 
{+network}
Example: content scripts injected

- `{[]},{cnn.pwd},
  [],
  []
- `{[]},{cnn.eve},
  [],
  []
- `{[]},{cnn.pwd, ebay.pwd},{{.pwd}},
  [],
  {*.pwd, +localStorage}
- `{[]},{cnn.pwd},
  [],
  []
- `{[]},{cnn.eve},
  [],
  []
- `{[]},{{.pwd}},
  [],
  []
- `{[]},{{.eve}},
  [],
  []
- `{[]},{{.pwd}},
  [],
  []
- `{[]},{{.evil}}
  [],
  []
- `{[]},{{.network}}
  [],
  []

Log in

Password:

ID:
Example: password sent to content script

{cnn.pwd, ebay.pwd} \subseteq \{cnn.pwd\}

ID: CNN
Password: (cnn.pwd, ebay.pwd)
Log in

Password Manager

Evil Extension

{{cnn}, {cnn.*}},
{},
{+network}

{{ },{*.evil}}
{},
{+network}
Example: password sent to content script

{cnn.pwd, ebay.pwd}

\*Declassify ebay.pwd\*

{cnn.pwd}

ID: 
Password: 
Log in

Password Manager

Evil Extension

{{cnn}, {cnn.*}},
{},
{+network}

{{}, {*.*}},
{},
{-.*.pwd, +localStorage}
Example: password sent to content script

\{\{\{cnn.pwd\}\},
{},
{}\}

\{cnn.pwd\}
\subseteq ?

\{cnn.pwd\}

{{}, {cnn.pwd}},
{},
{}

Evil Extension

Password Manager

ID:
Password:
Log in

{{}, {cnn.pwd}},
{},
{+network}

{{ }, {*.evil}}
{},
{+network}

cnn.com

CS

CS
Example: password sent to content script
Example: password filled in

```
{{cnn.pwd}, {cnn.pwd}},
{},
{}
```

```
{{}, {cnn.eve}},
{},
{}
```

```
{{cnn.pwd, ebay.pwd}, {*.pwd}},
{},
{*-pwd +localStorage}
```

```
{{cnn.pwd}, {cnn.eve}},
{},
{}
```

```
{{cnn.pwd, {cnn.pwd},
{},
{+network}
```

```
{cnn.pwd}
\subseteq
{cnn.*}
```

```
{{ }, {*.evil}}
{},
{+network}
```
Example: password filled in

```
{{cnn.pwd}, {cnn.pwd}},
{},
{}
```

```
{{}, {cnn.eve}},
{},
{}
```

```
{{cnn.pwd, ebay.pwd}, {*.pwd}},
{},
{*.pwd, +localStorage}
```

```
{{}, {*.evil}}
```

```
{{cnn.pwd}, {cnn.*}},
{},
{+network}
```

```
{{cnn.pwd}, {cnn.*}, {*.evil}}
```

```
{{}, {+network}}
```

Carnegie Mellon University
CyLab
Example: password stealing blocked

Carnegie Mellon

Evil Extension

Example: password stealing blocked

{\{cnn.pwd\}, \{cnn.pwd\}},
{},
{}

Password Manager

ID: CNN
Password: CNN
Log in

{\{cnn.pwd\}, \{cnn.*\}},
{},
{+network}

Evil Extension

{} \subseteq \{cnn.evil\}

{\{cnn.pwd, ebay.pwd\}, \{*.*pwd\}},
{},
{+.*pwd

{\{ },{*.evil}}

{} \subseteq \{cnn.evil\}

{+network}
Approximating existing browser policies

- SOP
- CSP
- postMessage
- iframe policies
- Domain relaxation

- Interesting composition issues when representing them all in one framework
  - E.g., conflicting policies of iframed page and parent page
Formal proof of security

- Model enforcement mechanism
  - In an extended version of Chromium

- Specify security property – noninterference
  - Attacker cannot learn any information about secrets prohibited by policies

- Proof of noninterference
  - Provides assurance of the model’s correctness
## Limitations

- **Trace-based noninterference**
  - Attacker may have more knowledge than traces
  - Allows certain implicit flows

- **To achieve stronger formal security guarantees:**
  - Make scheduler less predictable
  - Non-determinism or probabilistic execution
  - Secure multi-execution
  - Stronger notions of noninterference
  - …
Prototype implementation

- Built on Chromium version 32.0.1660.0
- Front pages of Alexa global top-10 web sites (40 runs each)
- 29% overhead to page load time added (unoptimized)
  - E.g., Google.com: 6 web requests, 28 label checks, 17% overhead
  - E.g., Amazon.com: 212 web requests, 639 label checks, 25% overhead
Summary

Dynamic entities
- Investigated coarse-grained dynamic tainting for enforcing information-flow policies
- Encompassed many entities in browser
- Identified interesting composition issues
- Our approach and model strike a balance between practicality and formal guarantees

Static entities

Lujo Bauer, Shaoying Cai, Limin Jia, Timothy Passaro, Michael Stroucken, and Yuan Tian