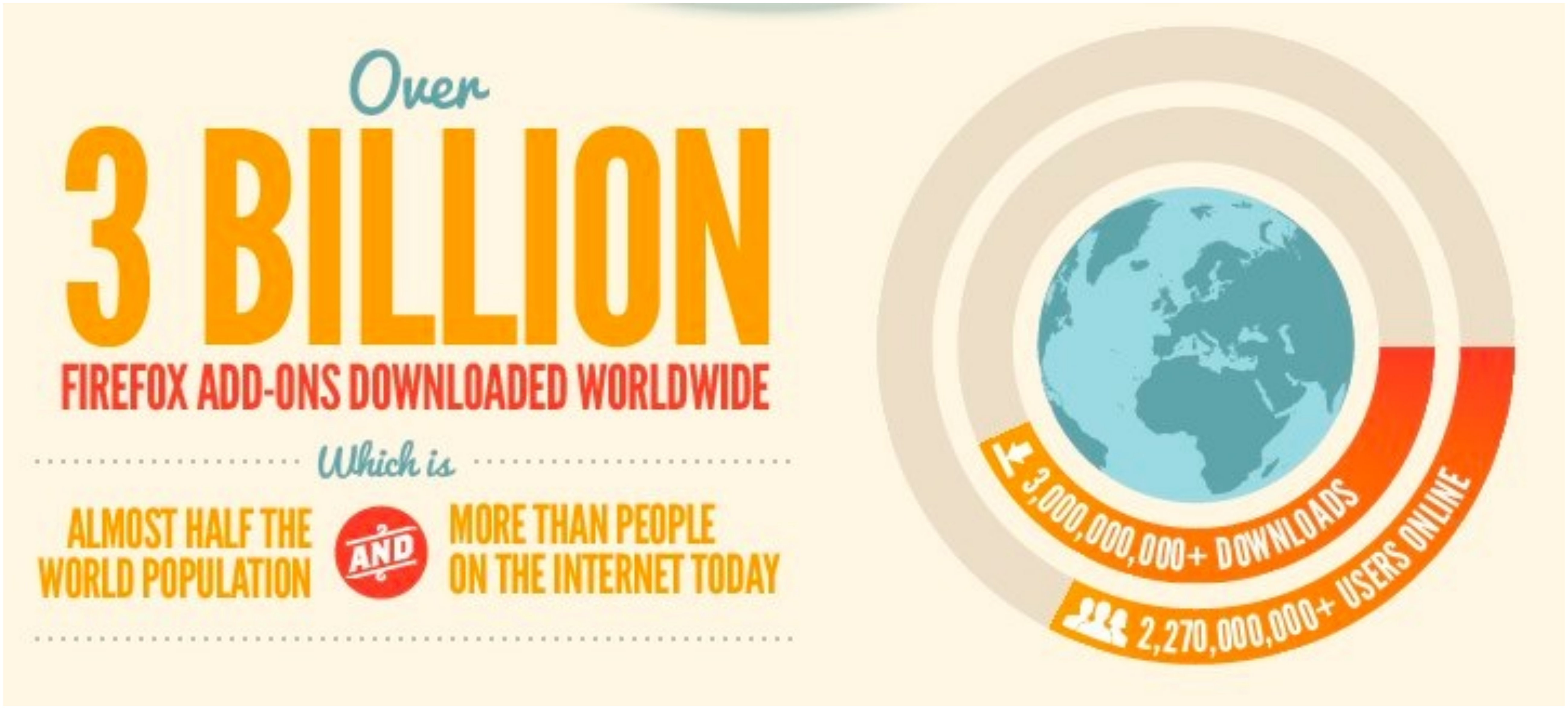
Security Signature Inference for JavaScript-based Browser Addons

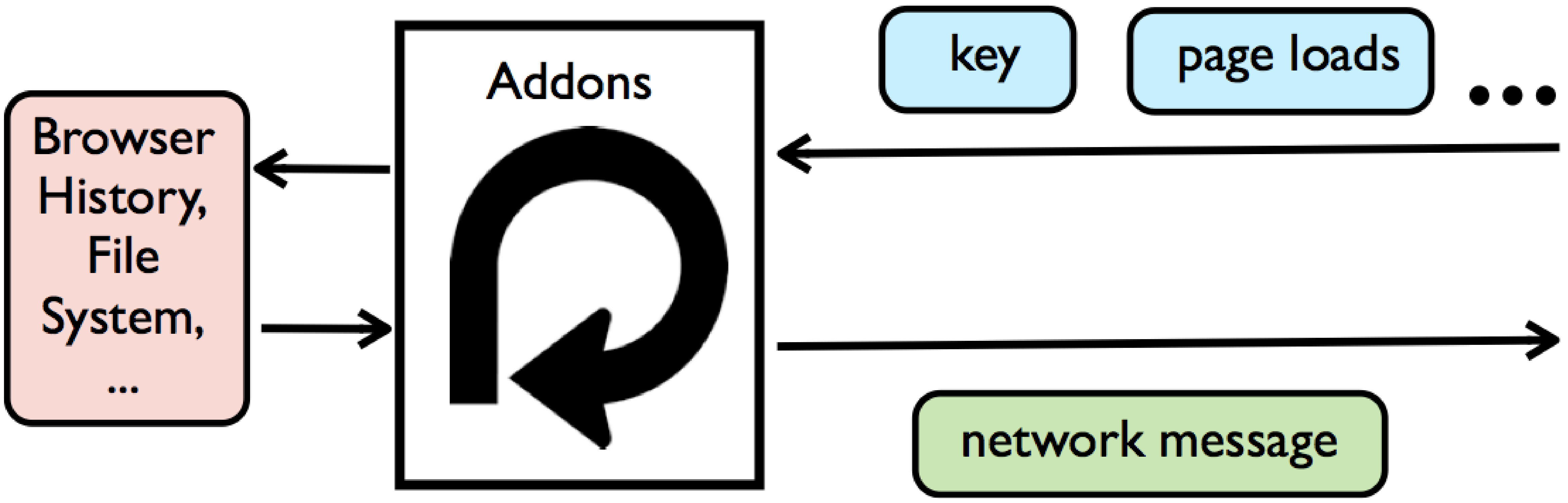
Vineeth Kashyap, Ben Hardekopf
University of California Santa Barbara

CGO 2014

JavaScript-based Browser Addons


Over
3 BILLION
FIREFOX ADD-ONS DOWNLOADED WORLDWIDE
Which is
ALMOST HALF THE WORLD POPULATION
AND
MORE THAN PEOPLE ON THE INTERNET TODAY
3,000,000,000+ DOWNLOADS
2,270,000,000+ USERS ONLINE

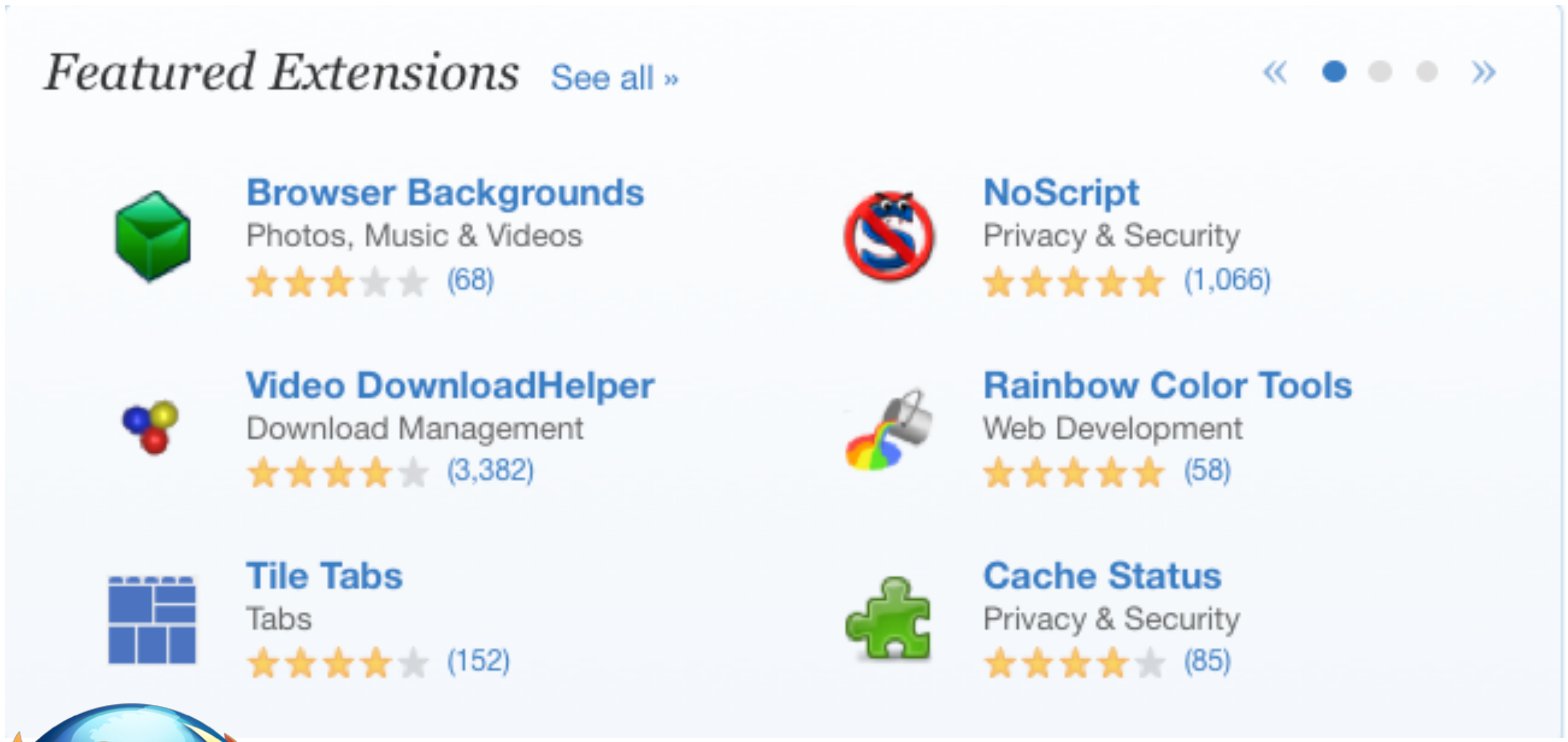
Addons: JavaScript with High Privileges



Urging Security Concern

- Proof of concept exploits
    - FFSniff, a configurable password stealer
- Unintentional vulnerabilities
    - Wikipedia Toolbar allowed arbitrary privileged code execution
- Intentionally malicious
    - Key loggers

Curated Repositories



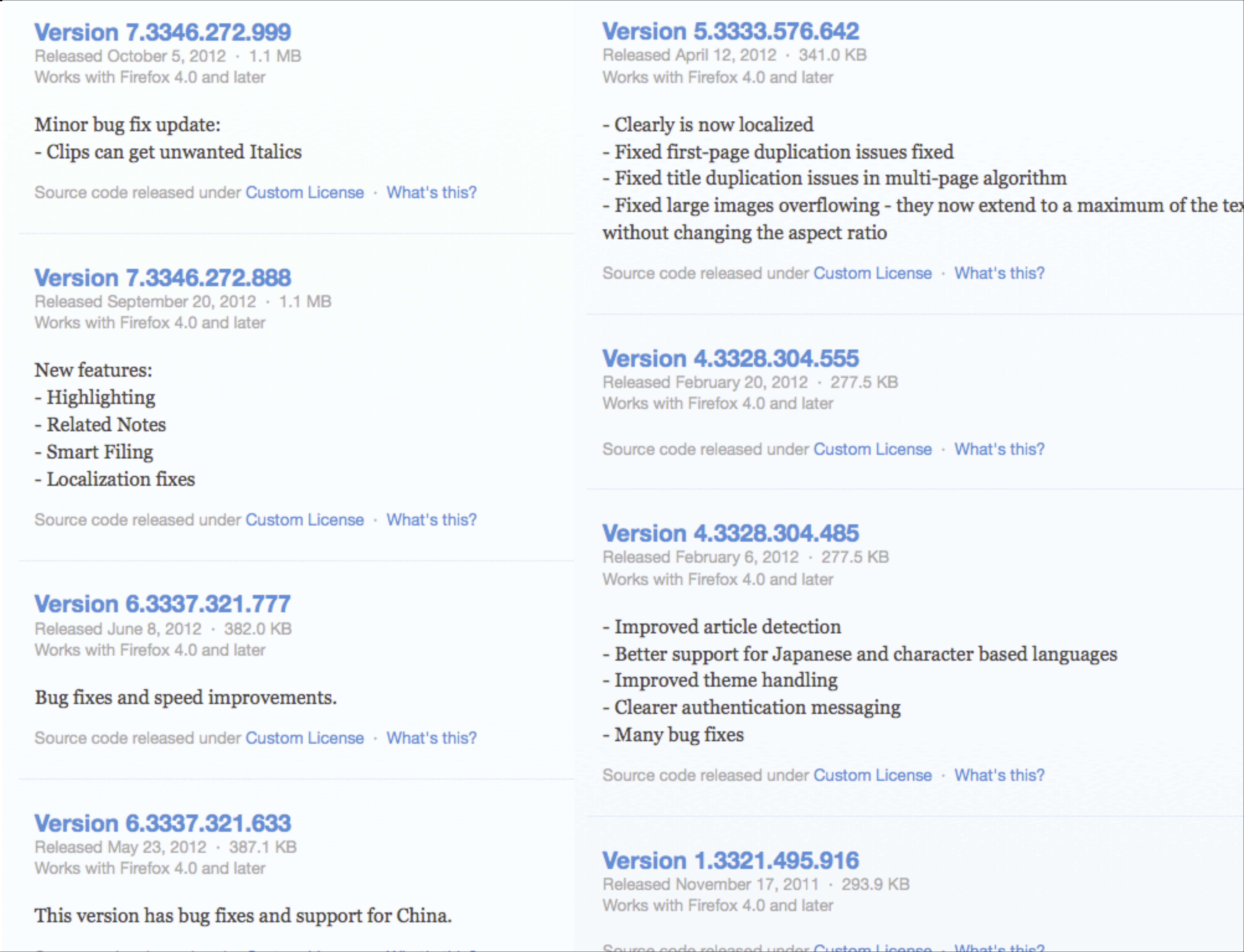
Featured Extensions See all »

« ● ● ● »

Browser Backgrounds Photos, Music & Videos ★★★☆☆ (68)	**NoScript** Privacy & Security ★★★★★ (1,066)
Video DownloadHelper Download Management ★★★★☆ (3,382)	**Rainbow Color Tools** Web Development ★★★★★ (58)
Tile Tabs Tabs ★★★★☆ (152)	**Cache Status** Privacy & Security ★★★★☆ (85)

```
    this.unsafeContentWin = unsafeContentWin;
    this.chromeWindow = chromeWindow;
}

// this function gets called by user scripts in content security scope to
// start a cross-domain xmlhttp request.
//
// details should look like:
// {method,url,onload,onerror,onreadystatechange,headers,data}
// headers should be in the form {name:value,name:value,etc}
surfcanyon_xmlhttpRequester.prototype.contentStartRequest = function(details) {
    var url = details.url;
    this.chromeWindow.setTimeout(
        surfcanyon_gmCompiler.hitch(this, "chromeStartRequest", url, details), 0);
}

// this function is intended to be called in chrome's security context, so
// that it can access other domains without security warning
surfcanyon_xmlhttpRequester.prototype.chromeStartRequest=function(url, details) {
    var req = new this.chromeWindow.XMLHttpRequest();

    this.setupRequestEvent(this.unsafeContentWin, req, "onload", url, details);
    this.setupRequestEvent(this.unsafeContentWin, req, "onerror", url, details);
    this.setupRequestEvent(this.unsafeContentWin, req, "onreadystatechange", url, details)

    req.open(details.method, url);

    if (details.mimeType) {
        req.overrideMimeType(details.mimeType);
    }

    if (details.headers) {
        for (var prop in details.headers) {
            req.setRequestHeader(prop, details.headers[prop]);
    if (details.headers) {
        for (var prop in details.headers) {
            req.setRequestHeader(prop, details.headers[prop]);
        }
    }

    req.send(details.data);
}

// arranges for the specified 'event' on xmlhttprequest 'req' to call the
// method by the same name which is a property of 'details' in the content
// window's security context.
surfcanyon_xmlhttpRequester.prototype.setupRequestEvent =
function(unsafeContentWin, req, event, url, details) {
    if (details[event]) {
        req[event] = function() {
            var responseHeaders = '';
            var status = 0;
            var statusText = '';

            if (req.readyState == 4) {
                try {
                    responseHeaders = req.getAllResponseHeaders();
                    status = req.status;
                    statusText = req.statusText;
                } catch (e) {
                }
            }

            var responseState = {
                url: url,
                responseText: req.responseText,
                readyState: req.readyState,
                responseHeaders: responseHeaders,
                status: status,
getUrlContents: function(aUrl){
// getUrlContents adapted from Greasemonkey Compiler
// http://www.letitblog.com/code/python/greasemonkey.py.txt
// used under GPL permission
//
// most everything else below based heavily off of Greasemonkey
// http://greasemonkey.mozdev.org/
// used under GPL permission
    var ioService=Components.classes["@mozilla.org/network/io-service;1"]
        .getService(Components.interfaces.nsIIOService);
    var scriptableStream=Components
        .classes["@mozilla.org/scriptableinputstream;1"]
        .getService(Components.interfaces.nsIScriptableInputStream);

    var channel=ioService.newChannel(aUrl, null, null);
    var input=channel.open();
    scriptableStream.init(input);
    var str=scriptableStream.read(input.available());
    scriptableStream.close();
    input.close();

    return str;
},

contentLoad: function(e) {
    try {
    var unsafeWin=e.target.defaultView;
    if (unsafeWin.wrappedJSObject) {
        unsafeWin=unsafeWin.wrappedJSObject;
    }

    var unsafeLoc=new XPCNativeWrapper(unsafeWin, "location").location;
    var href=new XPCNativeWrapper(unsafeLoc, "href").href;

    if (/^http/.test(href)) {
```

```
    try {
        var statusNode = doc.getElementById('surfcanyon-status');
        if (statusNode) {
            var disabled;
            try {
                disabled = prefsBranch.getBoolPref('disabled');
            } catch (e1) {
            }

            var statusBarIconDisabled;
            try {
                statusBarIconDisabled = prefsBranch.getBoolPref('status_bar_icon_disabled
            } catch (e2) {
            }

            statusNode.style.visibility = statusBarIconDisabled ? "collapse" : "visible";
            statusNode.setAttribute('status', (disabled ? '0' : '1'));
        }
    } catch (e3) {
    }

    try {
        var urlBarNode = doc.getElementById('surfcanyon-urlbar-main');

        var urlBarIconDisabled;
        try {
            urlBarIconDisabled = prefsBranch.getBoolPref('url_bar_icon_disabled');
        } catch (e4) {
        }

        urlBarNode.style.display = urlBarIconDisabled ? 'none' : 'block';
    } catch (e5) {
    }

    var hrefStart = href.substring(7, 27);
```



Version 7.3346.272.999

Released October 5, 2012 · 1.1 MB
Works with Firefox 4.0 and later

Minor bug fix update:
- Clips can get unwanted Italics

Source code released under Custom License · What's this?

Version 7.3346.272.888

Released September 20, 2012 · 1.1 MB
Works with Firefox 4.0 and later

New features:
- Highlighting
- Related Notes
- Smart Filing
- Localization fixes

Source code released under Custom License · What's this?

Version 6.3337.321.777

Released June 8, 2012 · 382.0 KB
Works with Firefox 4.0 and later

Bug fixes and speed improvements.

Source code released under Custom License · What's this?

Version 6.3337.321.633

Released May 23, 2012 · 387.1 KB
Works with Firefox 4.0 and later

This version has bug fixes and support for China.

Version 5.3333.576.642

Released April 12, 2012 · 341.0 KB
Works with Firefox 4.0 and later

- Clearly is now localized
- Fixed first-page duplication issues fixed
- Fixed title duplication issues in multi-page algorithm
- Fixed large images overflowing - they now extend to a maximum of the tex without changing the aspect ratio

Source code released under Custom License · What's this?

Version 4.3328.304.555

Released February 20, 2012 · 277.5 KB
Works with Firefox 4.0 and later

Source code released under Custom License · What's this?

Version 4.3328.304.485

Released February 6, 2012 · 277.5 KB
Works with Firefox 4.0 and later

- Improved article detection
- Better support for Japanese and character based languages
- Improved theme handling
- Clearer authentication messaging
- Many bug fixes

Source code released under Custom License · What's this?

Version 1.3321.495.916

Released November 17, 2011 · 293.9 KB
Works with Firefox 4.0 and later

Source code released under Custom License · What's this?

Manual JavaScript Addon Vetting is Difficult

- Ad-hoc
- Tedious
- Error-prone

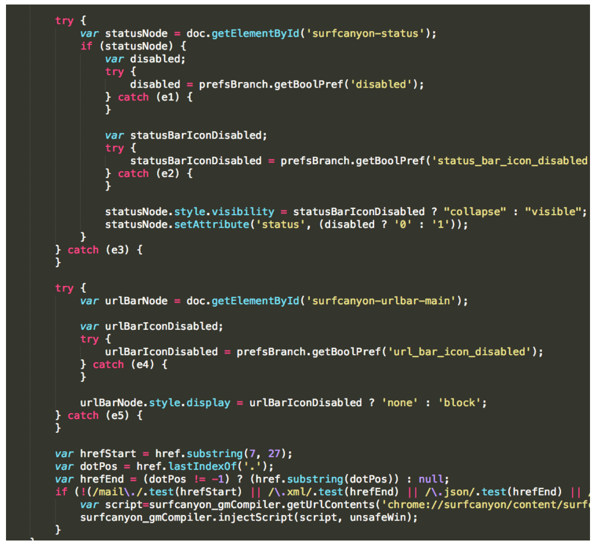
Our Goal: Help Automate the Vetting Process

- Automatically infer **security signatures**
- Summarize interesting information flows and critical API usages

Our Goal: Help Automate the Vetting Process

- Automatically infer **security signatures**

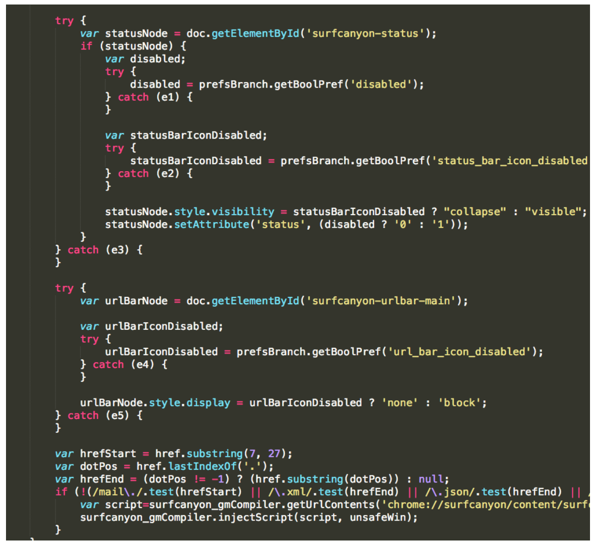
- Summarize interesting information flows and critical API usages



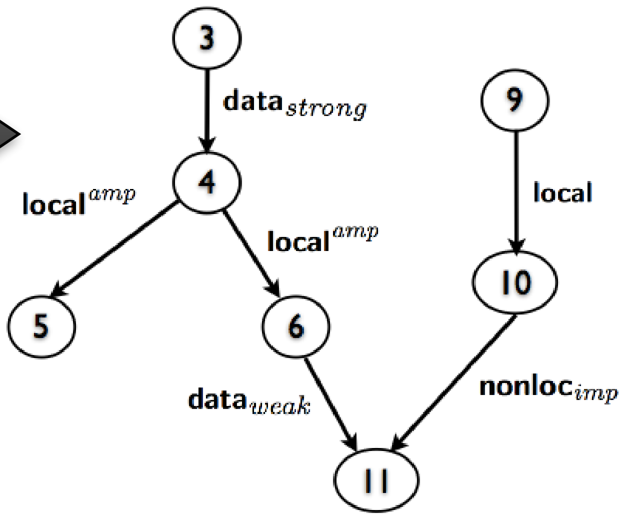
```
try {
    var statusNode = doc.getElementById('surfcanyon-status');
    if (statusNode) {
        var disabled;
        try {
            disabled = prefsBranch.getBoolPref('disabled');
        } catch (e1) {
        }

        var statusBarIconDisabled;
        try {
            statusBarIconDisabled = prefsBranch.getBoolPref('status_bar_icon_disabled
        } catch (e2) {
        }

        statusNode.style.visibility = statusBarIconDisabled ? "collapse" : "visible";
        statusNode.setAttribute('status', (disabled ? '0' : '1'));
    }
} catch (e3) {
}

try {
    var urlBarNode = doc.getElementById('surfcanyon-urlbar-main');

    var urlBarIconDisabled;
    try {
        urlBarIconDisabled = prefsBranch.getBoolPref('url_bar_icon_disabled');
    } catch (e4) {
    }

    urlBarNode.style.display = urlBarIconDisabled ? 'none' : 'block';
} catch (e5) {
}

var hrefStart = href.substring(7, 27);
var dotPos = href.lastIndexOf('.');
var hrefEnd = (dotPos != -1) ? (href.substring(dotPos)) : null;
if (!(/mail\./.test(hrefStart) || /\.xml/.test(hrefEnd) || /\.json/.test(hrefEnd) ||
    var script=surfcanyon_gmCompiler.getUrlContents('chrome://surfcanyon/content/surf
    surfcanyon_gmCompiler.injectScript(script, unsafeWin);
}
```

Our Goal: Help Automate the Vetting Process

- Automatically infer **security signatures**
- Summarize interesting information flows and critical API usages

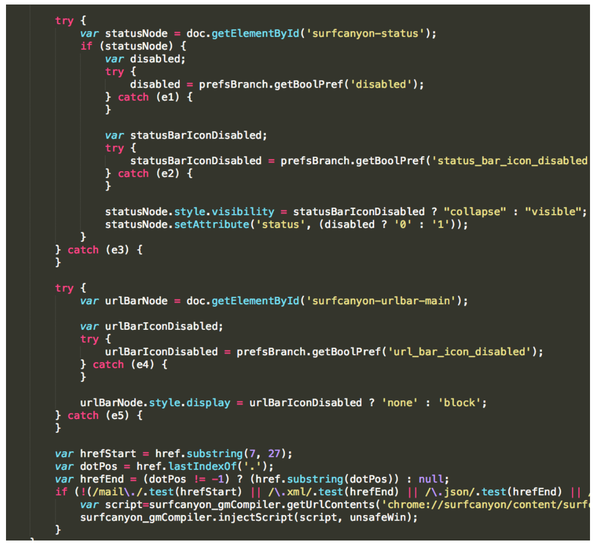


```
try {
    var statusNode = doc.getElementById('surfcanyon-status');
    if (statusNode) {
        var disabled;
        try {
            disabled = prefsBranch.getBoolPref('disabled');
        } catch (e1) {
        }

        var statusBarIconDisabled;
        try {
            statusBarIconDisabled = prefsBranch.getBoolPref('status_bar_icon_disabled
        } catch (e2) {
        }

        statusNode.style.visibility = statusBarIconDisabled ? "collapse" : "visible";
        statusNode.setAttribute('status', (disabled ? '0' : '1'));
    }
} catch (e3) {
}

try {
    var urlBarNode = doc.getElementById('surfcanyon-urlbar-main');

    var urlBarIconDisabled;
    try {
        urlBarIconDisabled = prefsBranch.getBoolPref('url_bar_icon_disabled');
    } catch (e4) {
    }

    urlBarNode.style.display = urlBarIconDisabled ? 'none' : 'block';
} catch (e5) {
}

var hrefStart = href.substring(7, 27);
var dotPos = href.lastIndexOf('.');
var hrefEnd = (dotPos != -1) ? (href.substring(dotPos)) : null;
if (!(/mail\./.test(hrefStart) || /\.xml/.test(hrefEnd) || /\.json/.test(hrefEnd) ||
    var script=surfcanyon_gmCompiler.getUrlContents('chrome://surfcanyon/content/surf
    surfcanyon_gmCompiler.injectScript(script, unsafeWin);
}
```

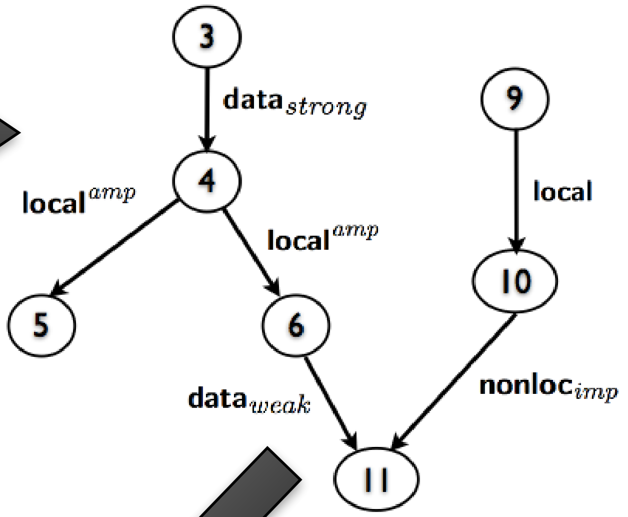


Our Goal: Help Automate the Vetting Process

- Automatically infer **security signatures**
- Summarize interesting information flows and critical API usages



```
try {
    var statusNode = doc.getElementById('surfcanyon-status');
    if (statusNode) {
        var disabled;
        try {
            disabled = prefsBranch.getBoolPref('disabled');
        } catch (e1) {
        }

        var statusBarIconDisabled;
        try {
            statusBarIconDisabled = prefsBranch.getBoolPref('status_bar_icon_disabled
        } catch (e2) {
        }

        statusNode.style.visibility = statusBarIconDisabled ? "collapse" : "visible";
        statusNode.setAttribute('status', (disabled ? '0' : '1'));
    }
} catch (e3) {
}

try {
    var urlBarNode = doc.getElementById('surfcanyon-urlbar-main');

    var urlBarIconDisabled;
    try {
        urlBarIconDisabled = prefsBranch.getBoolPref('url_bar_icon_disabled');
    } catch (e4) {
    }

    urlBarNode.style.display = urlBarIconDisabled ? 'none' : 'block';
} catch (e5) {
}

var hrefStart = href.substring(7, 27);
var dotPos = href.lastIndexOf('.');
var hrefEnd = (dotPos != -1) ? (href.substring(dotPos)) : null;
if (!(/mail\./.test(hrefStart) || /\.xml/.test(hrefEnd) || /\.json/.test(hrefEnd) || 
    var script=surfcanyon_gmCompiler.getUrlContents('chrome://surfcanyon/content/surf
    surfcanyon_gmCompiler.injectScript(script, unsafeWin);
}
```



url $\xrightarrow{\textbf{amplified local control flow}}$ **send** (www.evil.com)

Key Challenges

- **Flexible security policies**
  - No single policy applies for all addons
- **Classifying Information Flows**
  - Binary result (secure or insecure) is not enough
- **Inferring Network Domains**
  - Critical to reason about addon’s network communication

Our Solution

- Construct **annotated Program Dependence Graphs** (PDG)
- Use annotated PDGs to generate **security signatures**
- Use **prefix string analysis** to infer network domains communicated with

Our Solution

- Construct **annotated Program Dependence Graphs** (PDG)
- Use annotated PDGs to generate **security signatures**
- Use **prefix string analysis** to infer network domains communicated with

UCSB

Our Solution

- Construct **annotated Program Dependence Graphs** (PDG)
- Use annotated PDGs to generate **security signatures**
- Use **prefix string analysis** to infer network domains communicated with

Our Solution

- Construct **annotated Program Dependence Graphs** (PDG)
- Use annotated PDGs to generate **security signatures**
- Use **prefix string analysis** to infer network domains communicated with

Automatically summarize API usages, interesting information flows (classified based on the type of flow)

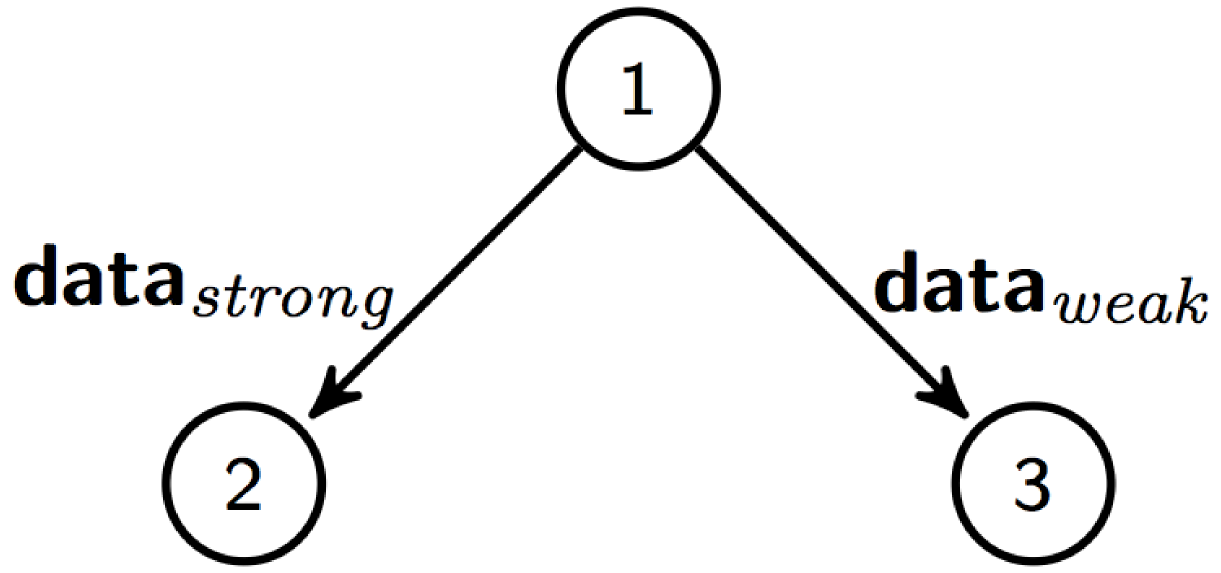
Annotated Program Dependence Graph

- Use JSAI[†] to construct a PDG
- Annotate the edges of PDG with the type of dependency

† JSAI is a sound and efficient JavaScript abstract interpreter we developed.

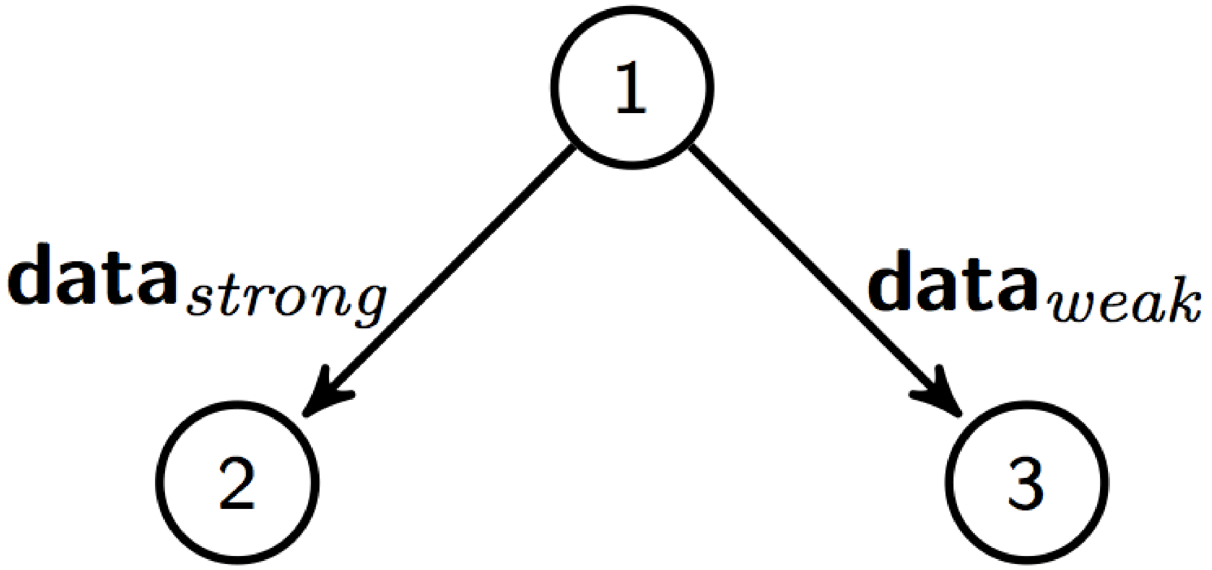
Strong vs. Weak Data Dependency

```
1 var data = {loc: url, other: 1}
2 send(data["loc"]);
3 send(data[getString()]);
```



Strong vs. Weak Data Dependency

```
① var data = {loc: url, other: 1}
② send(data["loc"]);
③ send(data[getString()]);
```



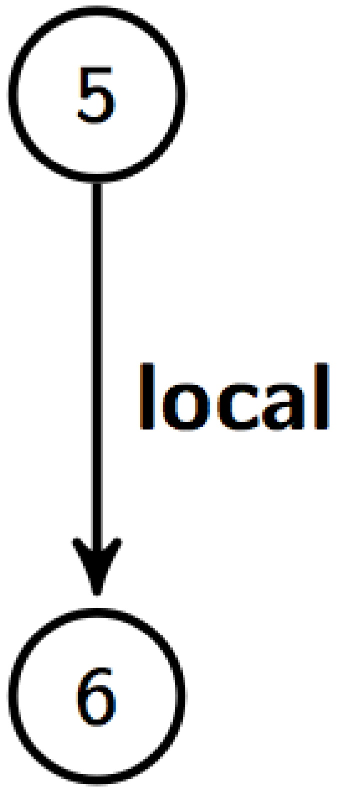
Strong vs. Weak Data Dependency

```
① var data = {loc: url, other: 1}
② send(data["loc"]);
③ send(data[getString()]);
```

$$\begin{tikzcd}
 & 1 \arrow[dl, "\mathbf{data}_{strong}"'] \arrow[dr, "\mathbf{data}_{weak}"] & \\
2 & & 3
\end{tikzcd}$$

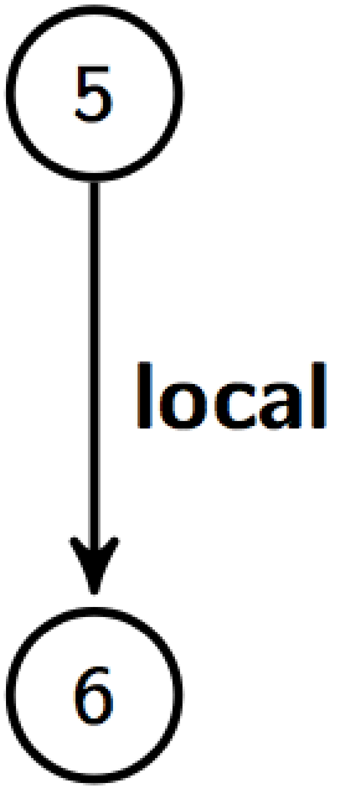
Local Control Dependency

```
5 if (url == "secret.com")
6   send(null);
```


$$\begin{tikzcd} 5 \arrow[d, "local"] \\ 6 \end{tikzcd}$$

Local Control Dependency

```
5 if (url == "secret.com")
6   send(null);
```



Syntax-obvious Non-local Control Dependency

```
13 try {
14   if (url != "hush-hush.com")
15     throw "irrelevant";
16   send(null);
17 } catch(x) {};
```

$$\begin{tikzcd}
14 \arrow[d, "\textbf{nonloc}_{exp}"] \\
16
\end{tikzcd}$$

Syntax-obvious Non-local Control Dependency

```
13 try {
14    if (url != "hush-hush.com")
15       throw "irrelevant";
16    send(null);
17 } catch(x) {};
```

$$\begin{tikzcd}
14 \arrow[d, "\mathbf{nonloc}_{exp}"] \\
16
\end{tikzcd}$$

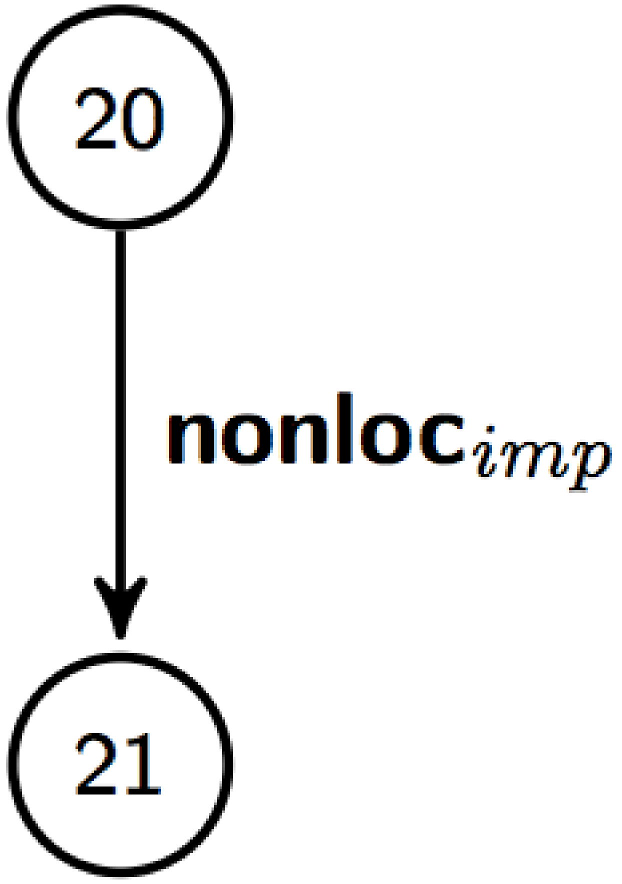
Non-obvious Non-local Control Dependency

```
18 try {
19   if (url != "mystic.com")
20     obj.prop = 1;
21   send(null);
22 } catch(x) {}
```

$$\begin{tikzcd}
20 \arrow[d, "\mathbf{nonloc}_{imp}"] \\
21
\end{tikzcd}$$

Non-obvious Non-local Control Dependency

```
18 try {
19   if (url != "mystic.com")
20     obj.prop = 1;
21   send(null);
22 } catch(x) {}
```



Amplified vs. Simple Control Dependencies

```
7  var arr = ["covert.com", "priv.com"/*,..*/];
8  var i=0, count=0;
9  while (arr[i] && url != arr[i]) {
10     i++;
11     count++;
   } // end while
12 send(count);
```

$$\begin{tikzcd}
9 \arrow[d, "\text{local}^{amp}"] \\
11 \arrow[d, "\text{data}_{weak}"'] \\
12
\end{tikzcd}$$

Amplified vs. Simple Control Dependencies

```
7  var arr = ["covert.com", "priv.com"/*,..*/];
8  var i=0, count=0;
9  while (arr[i] && url != arr[i]) {
10     i++;
11     count++;
   } // end while
12 send(count);
```

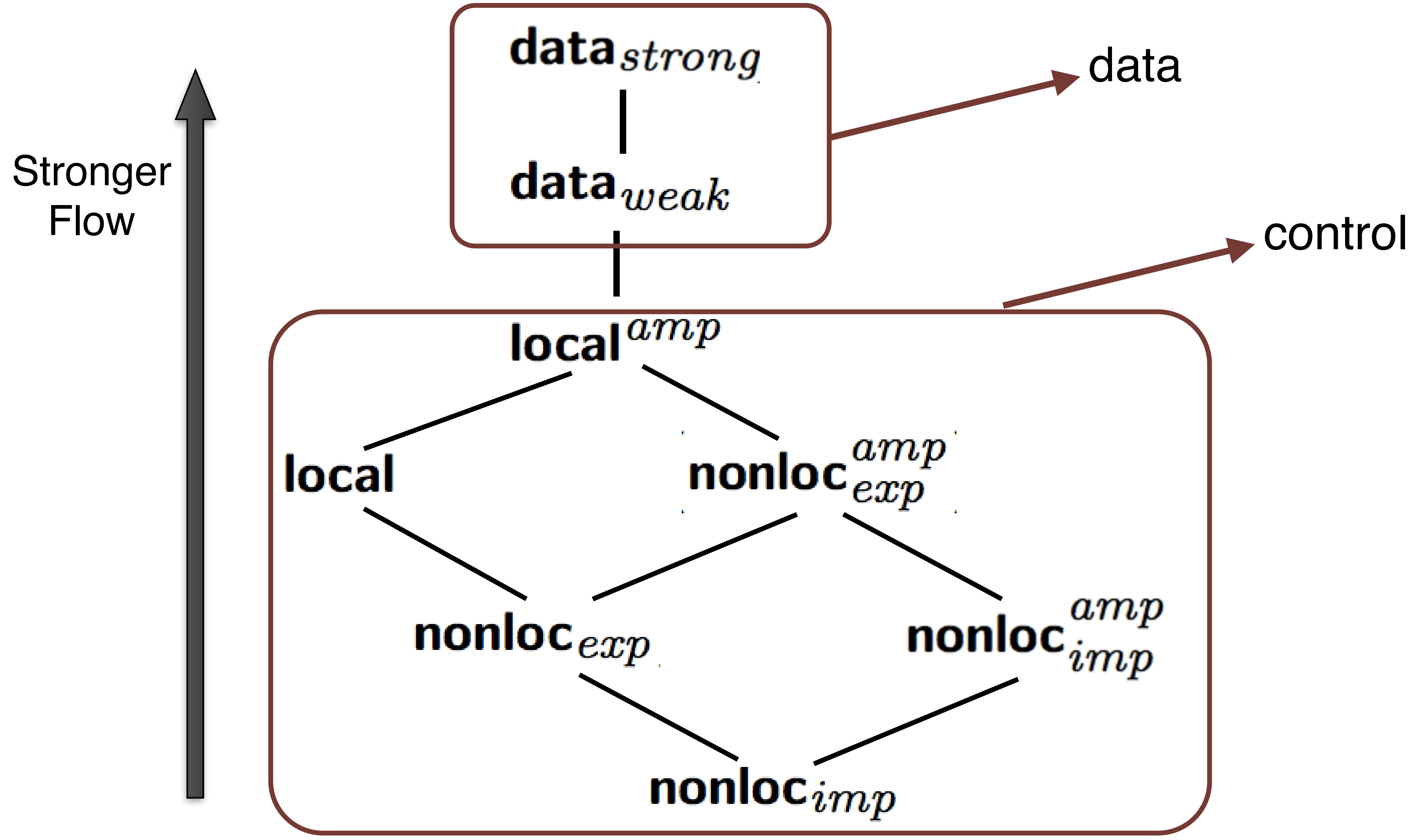
$$\begin{tikzcd}
9 \arrow[d, "\mathbf{local}^{amp}"] \\
11 \arrow[d, "\mathbf{data}_{weak}"'] \\
12
\end{tikzcd}$$

Lattice of Perceived Flow Strength

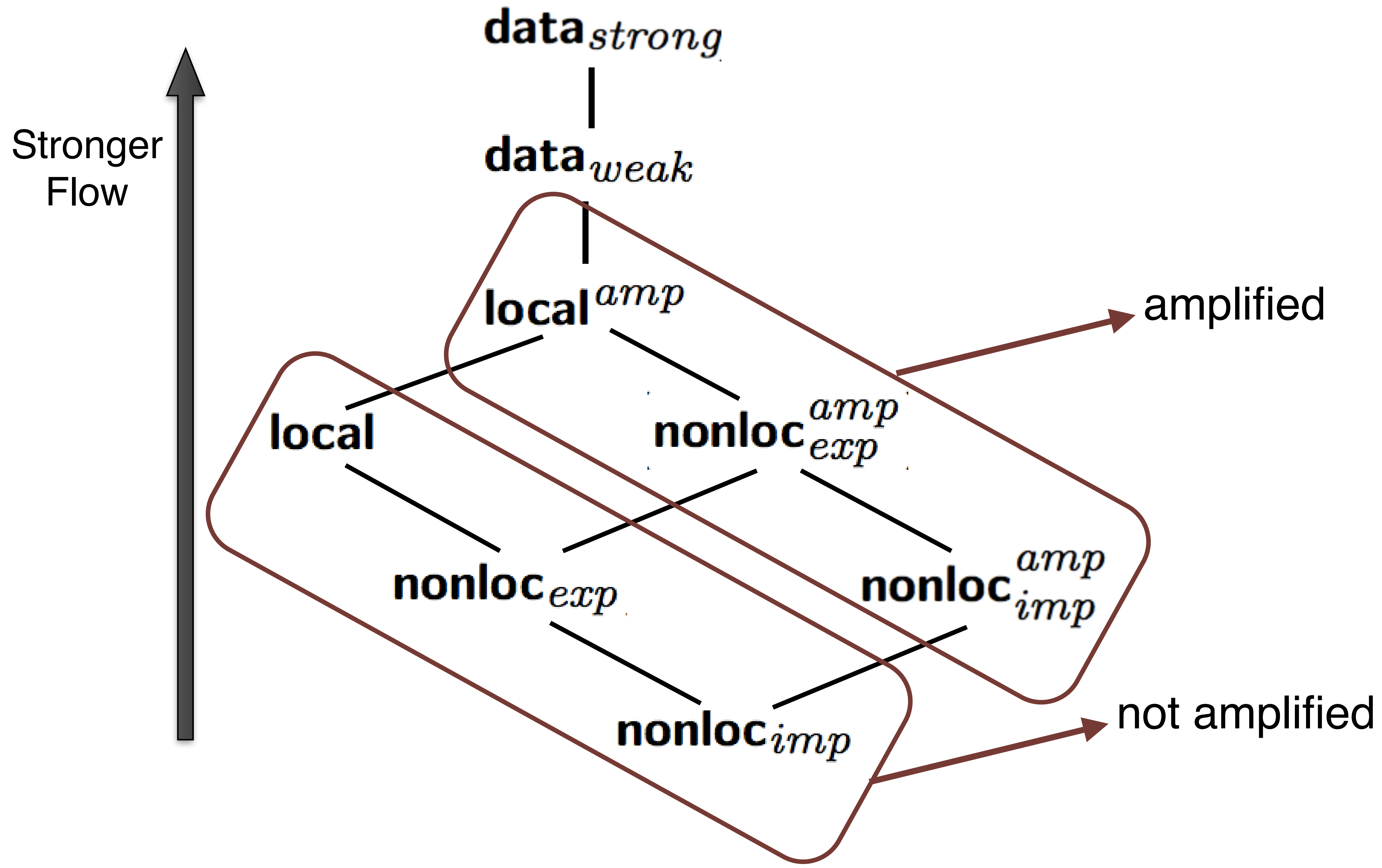
Stronger
Flow

$$\begin{tikzcd}
 & \mathbf{data}_{strong} \arrow[d, dash] & & \\
 & \mathbf{data}_{weak} \arrow[d, dash] & & \\
 & \mathbf{local}^{amp} \arrow[dl, dash] \arrow[dr, dash] & & \\
\mathbf{local} \arrow[dr, dash] & & \mathbf{nonloc}^{amp}_{exp} \arrow[dl, dash] \arrow[dr, dash] & \\
 & \mathbf{nonloc}_{exp} \arrow[dr, dash] & & \mathbf{nonloc}^{amp}_{imp} \arrow[dl, dash] \\
 & & \mathbf{nonloc}_{imp} &
\end{tikzcd}$$

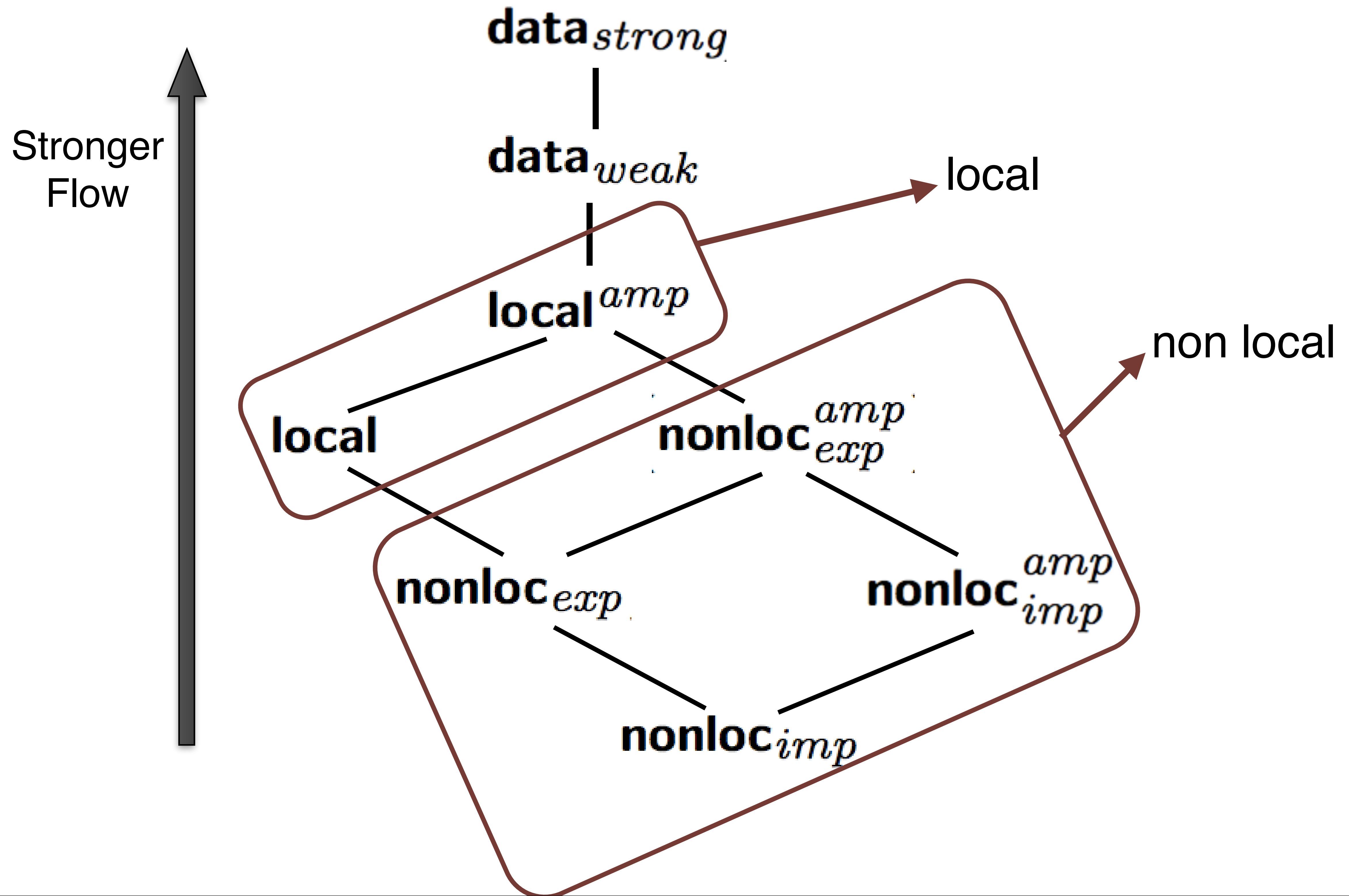
Lattice of Perceived Flow Strength



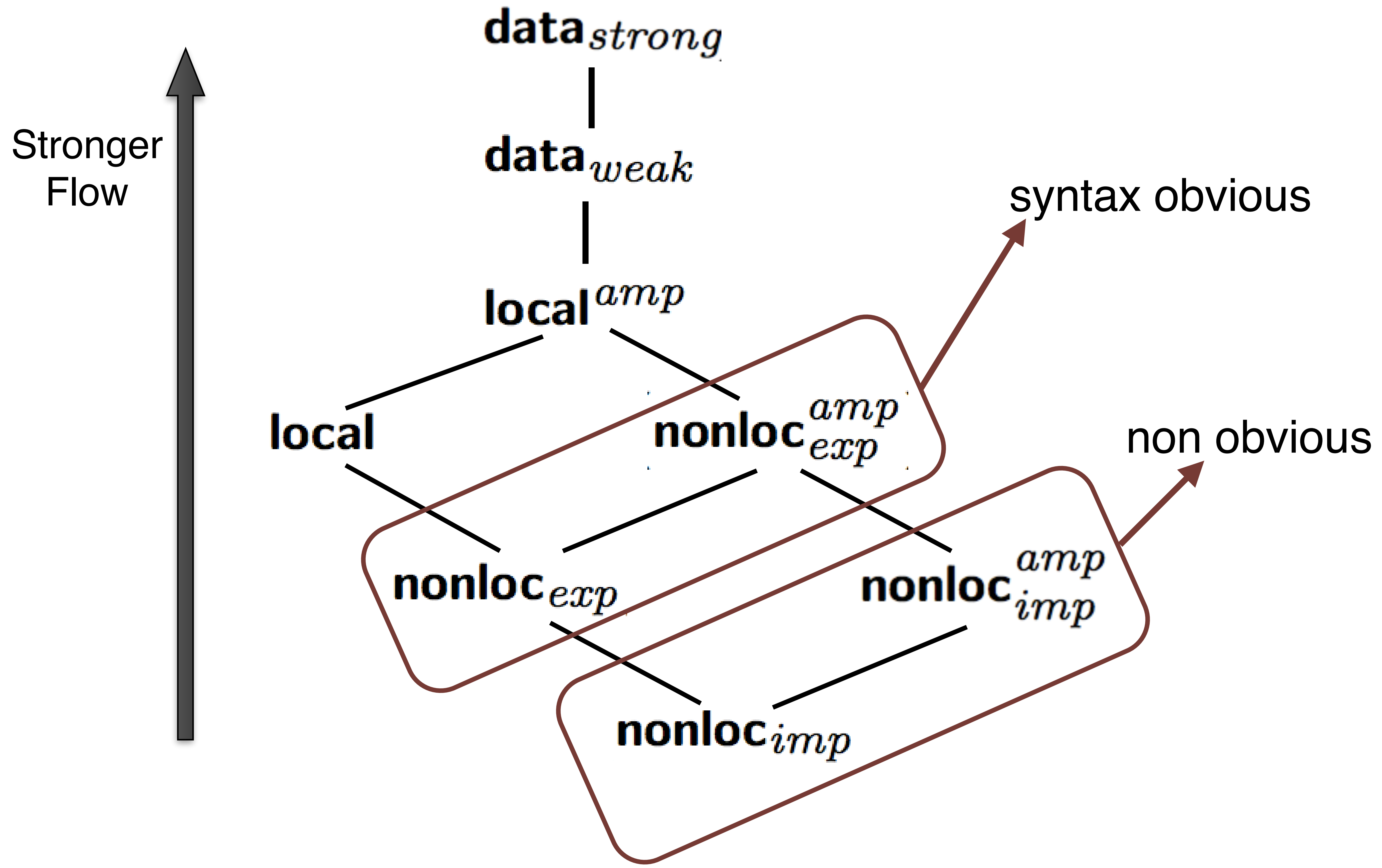
Lattice of Perceived Flow Strength



Lattice of Perceived Flow Strength



Lattice of Perceived Flow Strength



Generating Security Signatures

- Use the PDG to reason about information flow in addons
- Use PDG annotations to classify flows
- Output a signature summarizing relevant flows

$$\boxed{entry \in Entry ::= src \xrightarrow{type} sink \mid sink}$$

Generating Security Signatures

- Use the PDG to reason about information flow in addons
- Use PDG annotations to classify flows
- Output a signature summarizing relevant flows

$$\boxed{entry \in Entry ::= src \xrightarrow{type} sink \mid sink}$$

$$\textbf{url} \xrightarrow{\textbf{amplified local control flow}} \textbf{send}\ \text{(www.evil.com)}$$

Generating Security Signatures

```
1   xhr.open("GET", "www.evil.com");
2   var dom = ["a.com", "b.com", ...];
3   var i = 0, count = 0;
4   while (dom[i] && url != dom[i]) {
5     i++;
6     count++;
7   }
8   try {
9     if (url != "c.com")
10      obj.prop = 1;
11    xhr.send(count);
12  } catch(x) {}
```

$$\begin{tikzcd}
 & 3 \arrow[d] & & & 9 \arrow[d] \\
 & 4 \arrow[dl] \arrow[dr] & & & 10 \arrow[ddl] \\
5 & & 6 \arrow[dr] & & \\
 & & & 11 &
\end{tikzcd}$$

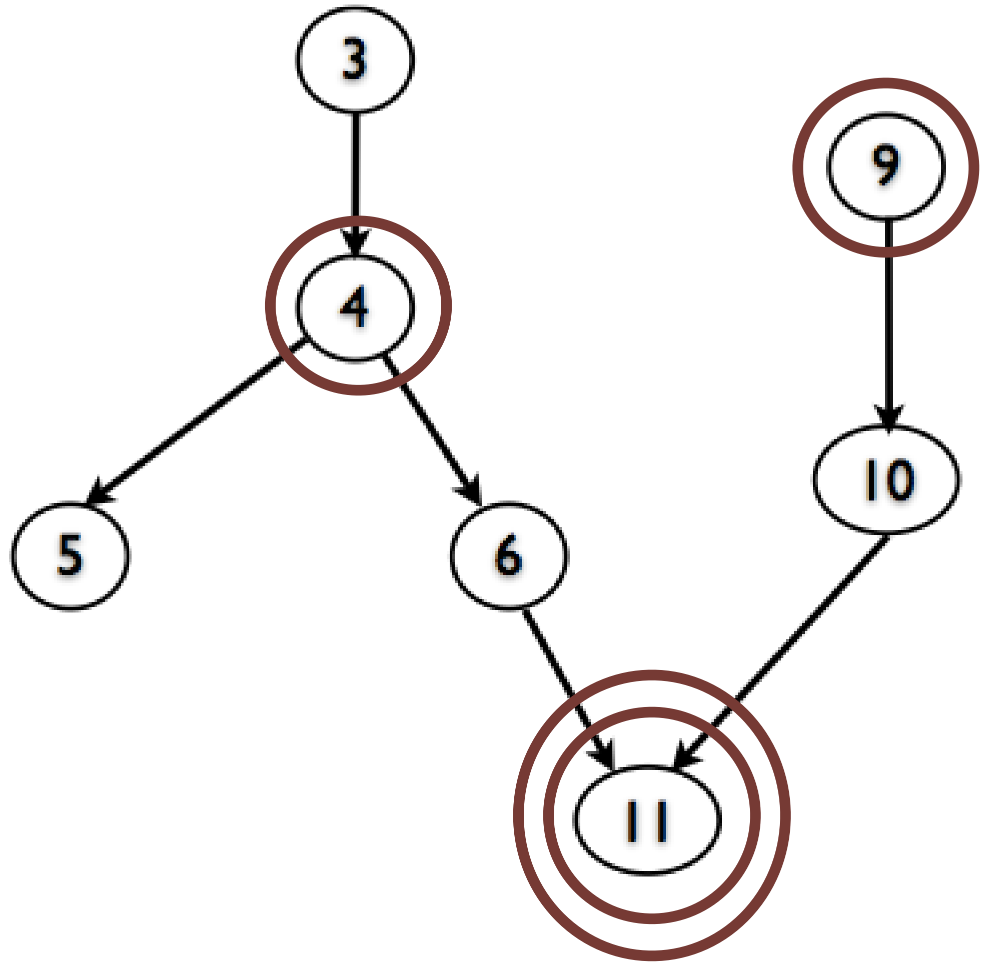
Generating Security Signatures

```
1  xhr.open("GET", "www.evil.com");
2  var dom = ["a.com", "b.com", ...];
3  var i = 0, count = 0;
4  while (dom[i] && url != dom[i]) {
5    i++;
6    count++;
7  }
8  try {
9    if (url != "c.com")
10     obj.prop = 1;
11   xhr.send(count);
12 } catch(x) {}
```

$$\begin{tikzcd}
 & 3 \arrow[d] & & & 9 \arrow[dd] \\
 & 4 \arrow[dl] \arrow[dr] & & & \\
5 & & 6 \arrow[dr] & & 10 \arrow[dl] \\
 & & & 11 &
\end{tikzcd}$$

Generating Security Signatures

```
1   xhr.open("GET", "www.evil.com");
2   var dom = ["a.com", "b.com", ...];
3   var i = 0, count = 0;
4   while (dom[i] && url != dom[i]) {
5     i++;
6     count++;
7   }
8   try {
9     if (url != "c.com")
10      obj.prop = 1;
11    xhr.send(count);
12  } catch(x) {}
```

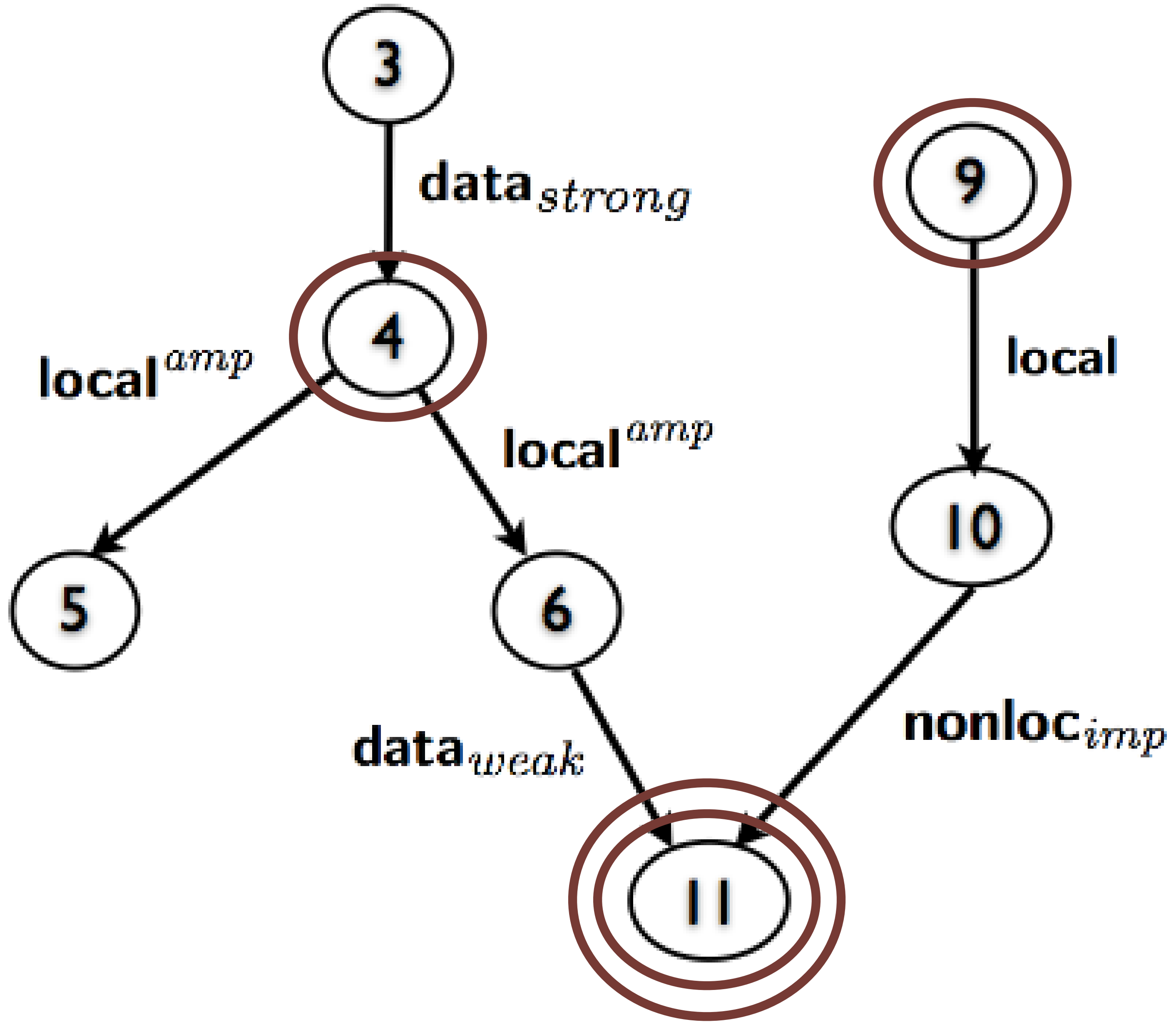


Generating Security Signatures

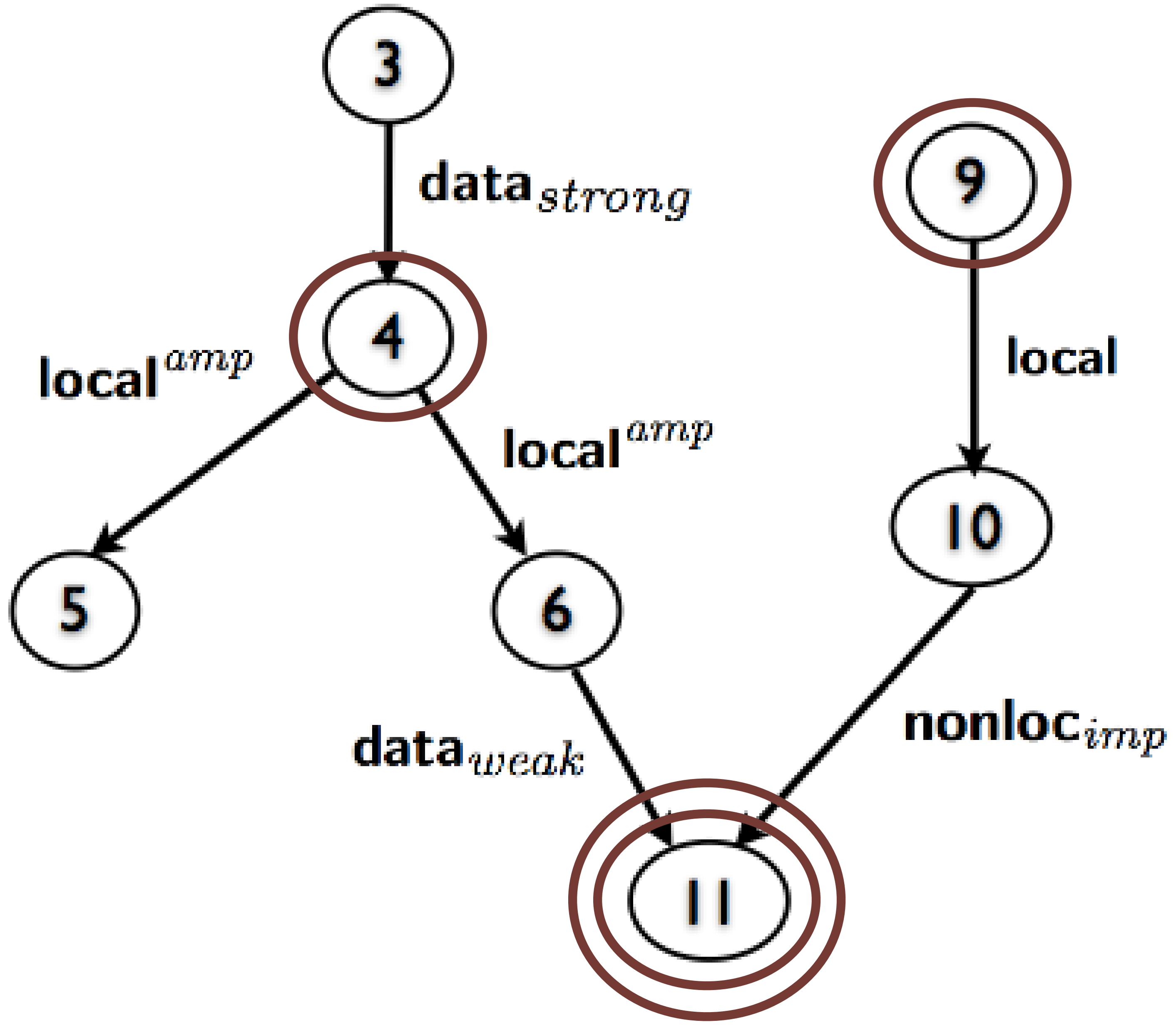
```
1   xhr.open("GET", "www.evil.com");
2   var dom = ["a.com", "b.com", ...];
3   var i = 0, count = 0;
4   while (dom[i] && url != dom[i]) {
5     i++;
6     count++;
7   }
8   try {
9     if (url != "c.com")
10      obj.prop = 1;
11    xhr.send(count);
12  } catch(x) {}
```

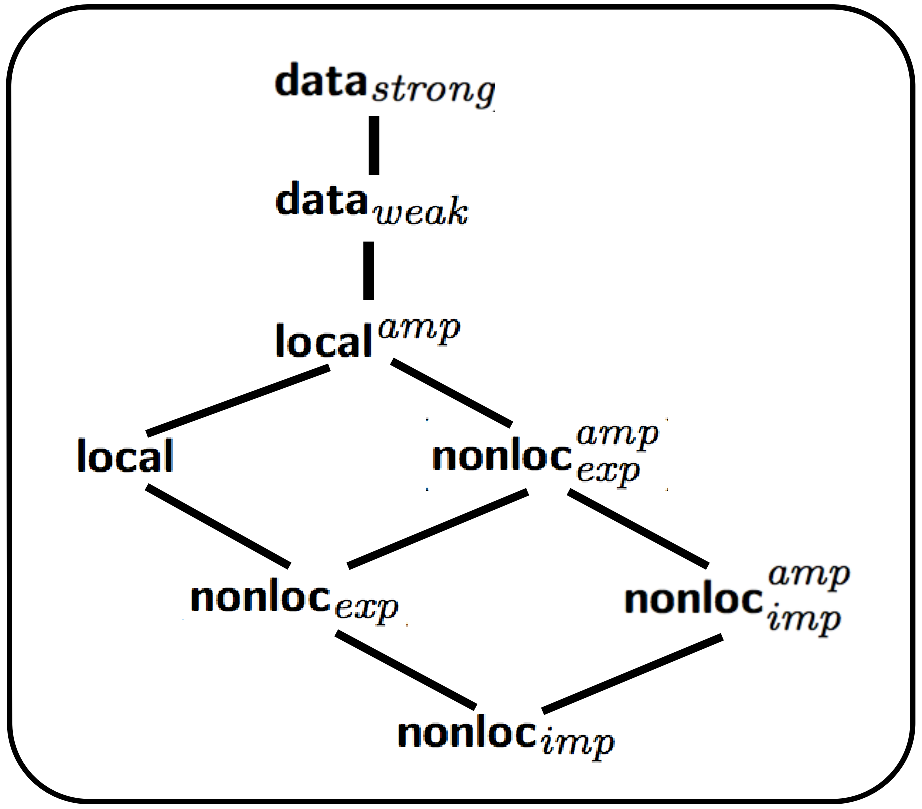
$$\begin{tikzcd}
 & 3 \arrow[d, "\mathsf{data}_{strong}"] & & 9 \arrow[dd, "\mathsf{local}"] \\
 & 4 \arrow[dl, "\mathsf{local}^{amp}"'] \arrow[dr, "\mathsf{local}^{amp}"] & & \\
5 & & 6 \arrow[d, "\mathsf{data}_{weak}"'] & 10 \arrow[dl, "\mathsf{nonloc}_{imp}"] \\
 & & 11 &
\end{tikzcd}$$

Generating Security Signatures

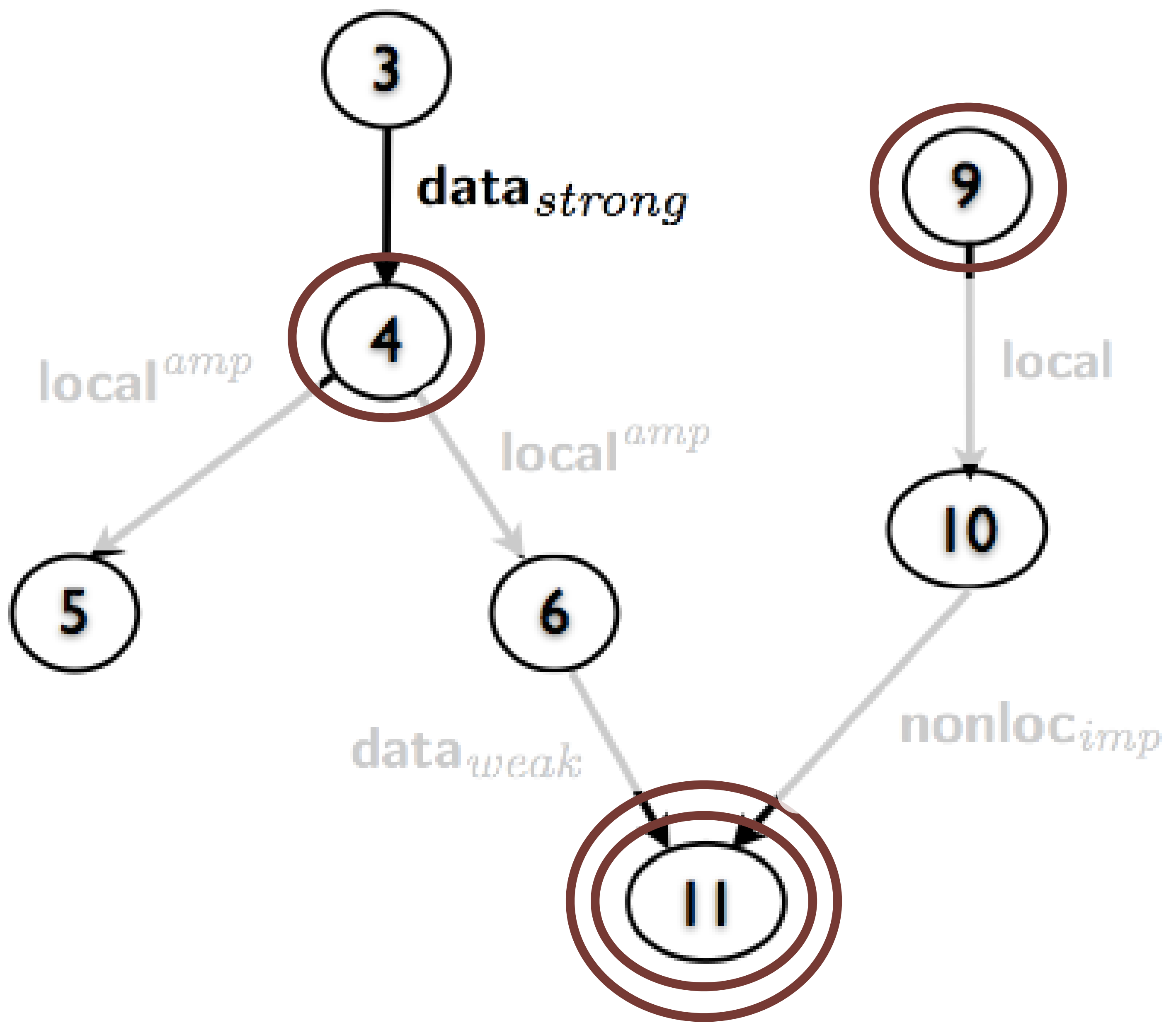


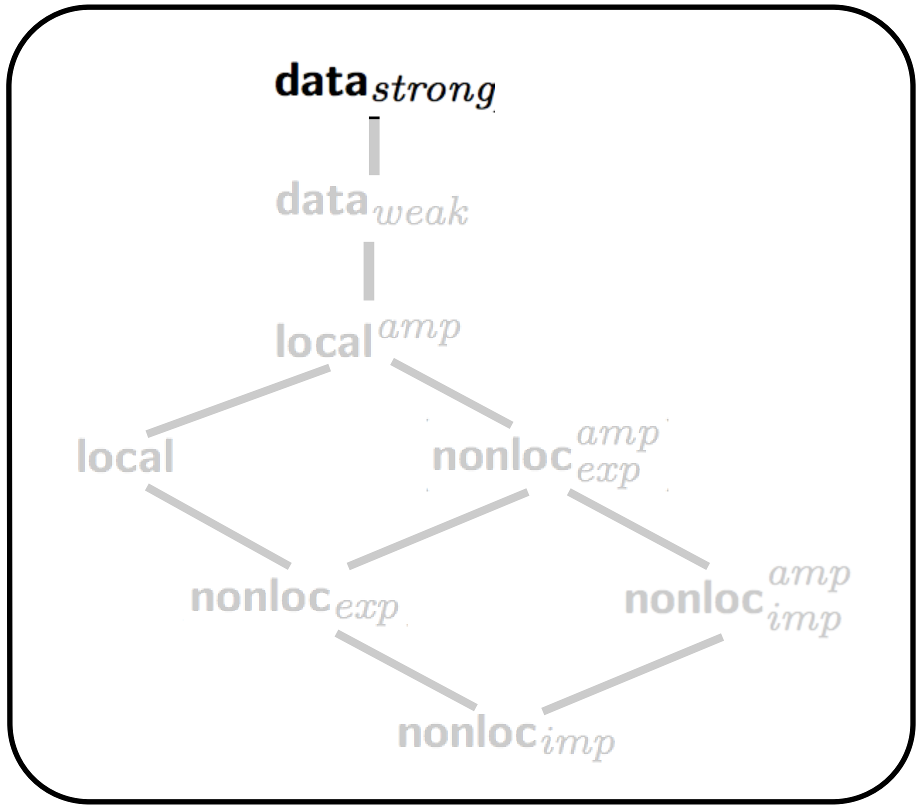
Generating Security Signatures



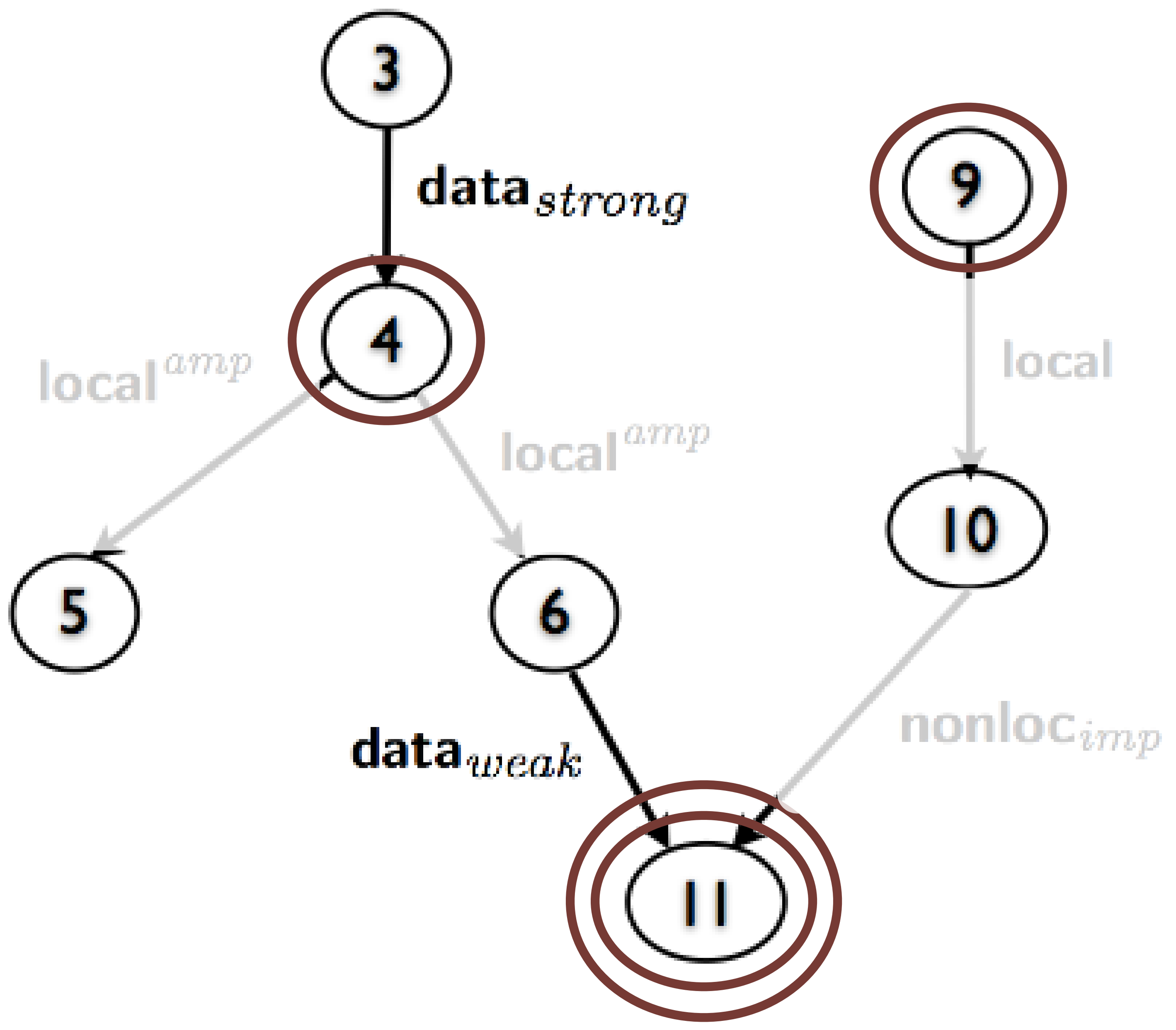


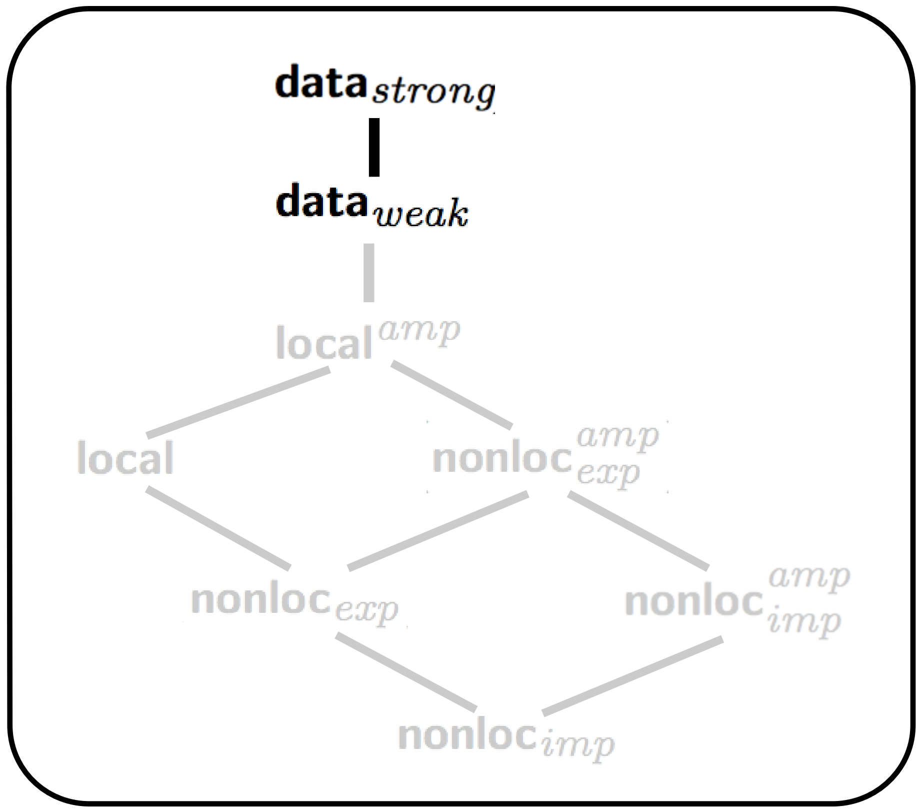
Generating Security Signatures



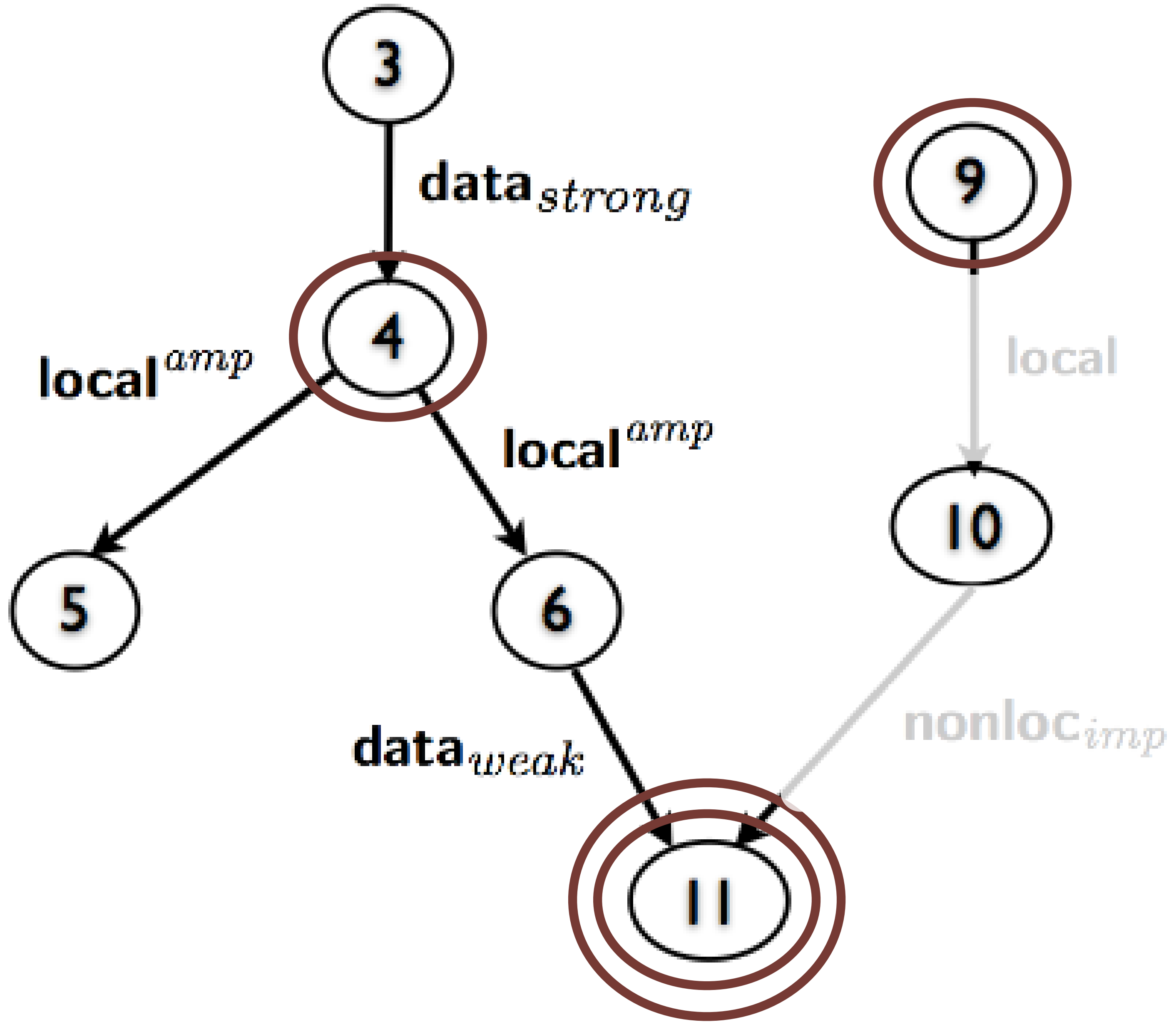


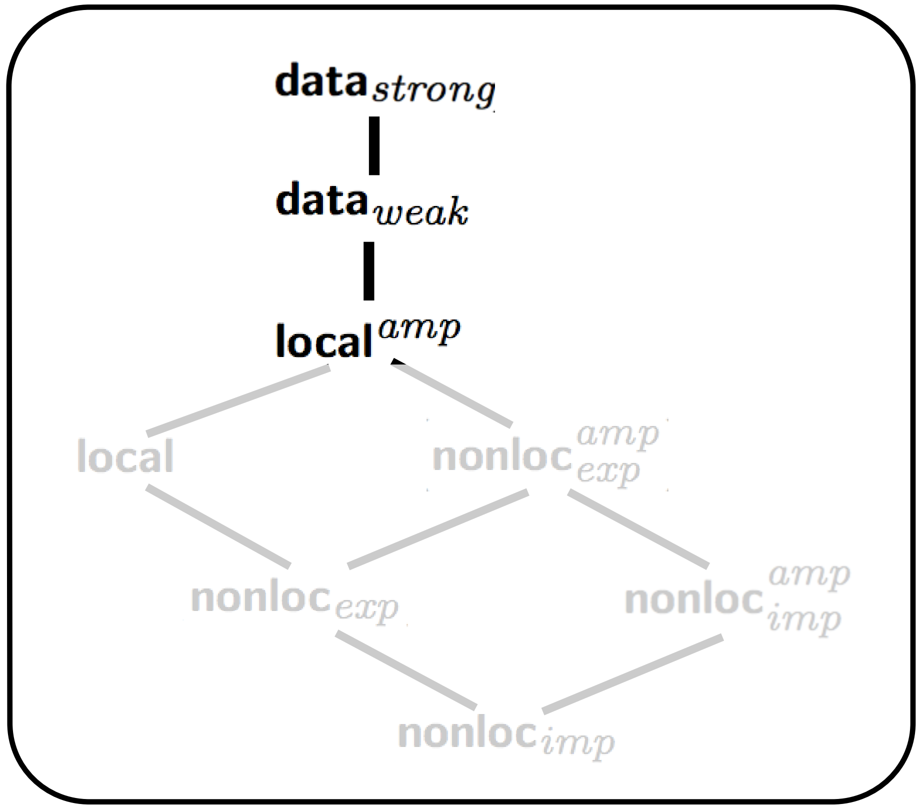
Generating Security Signatures





Generating Security Signatures

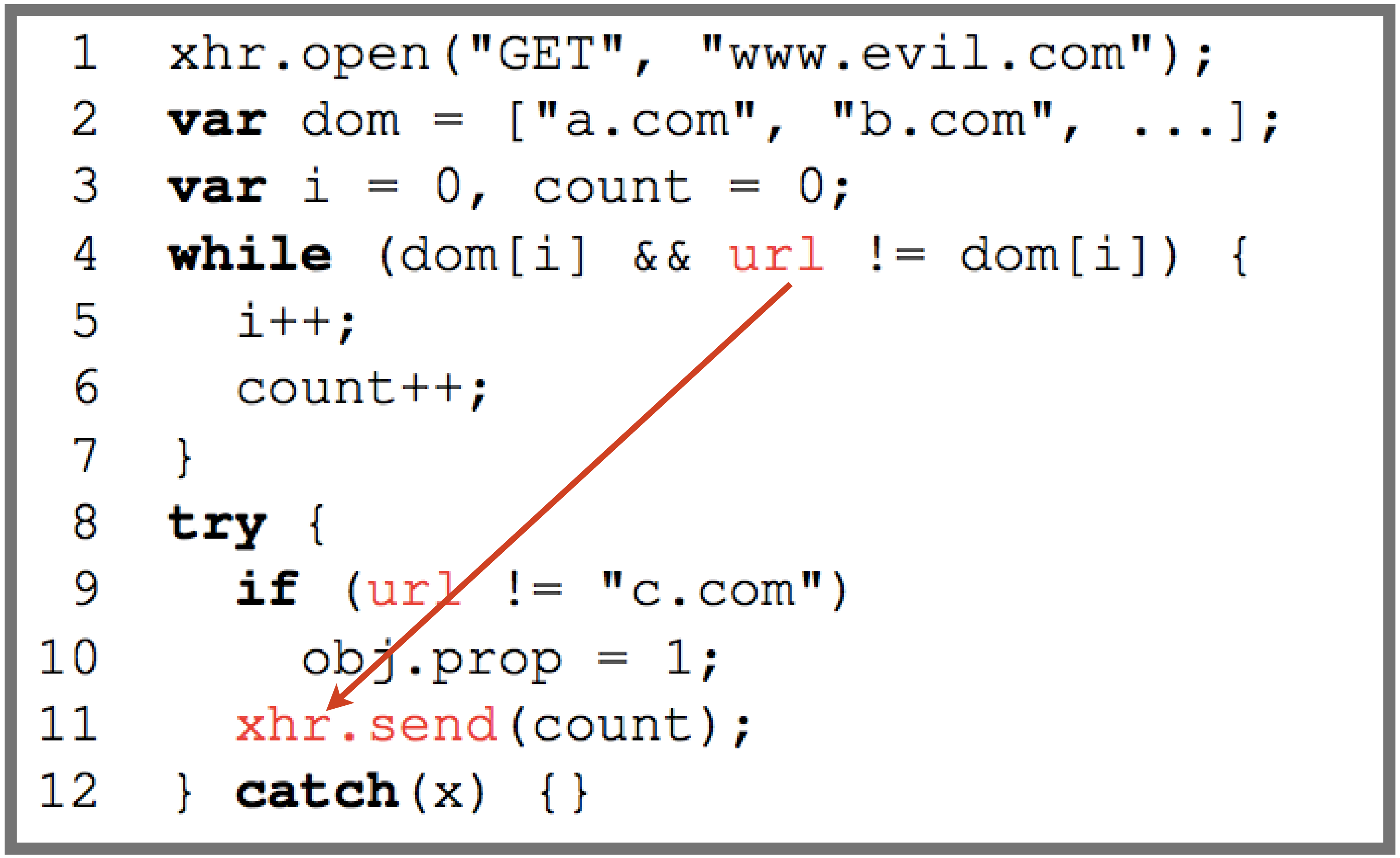




Generating Security Signatures

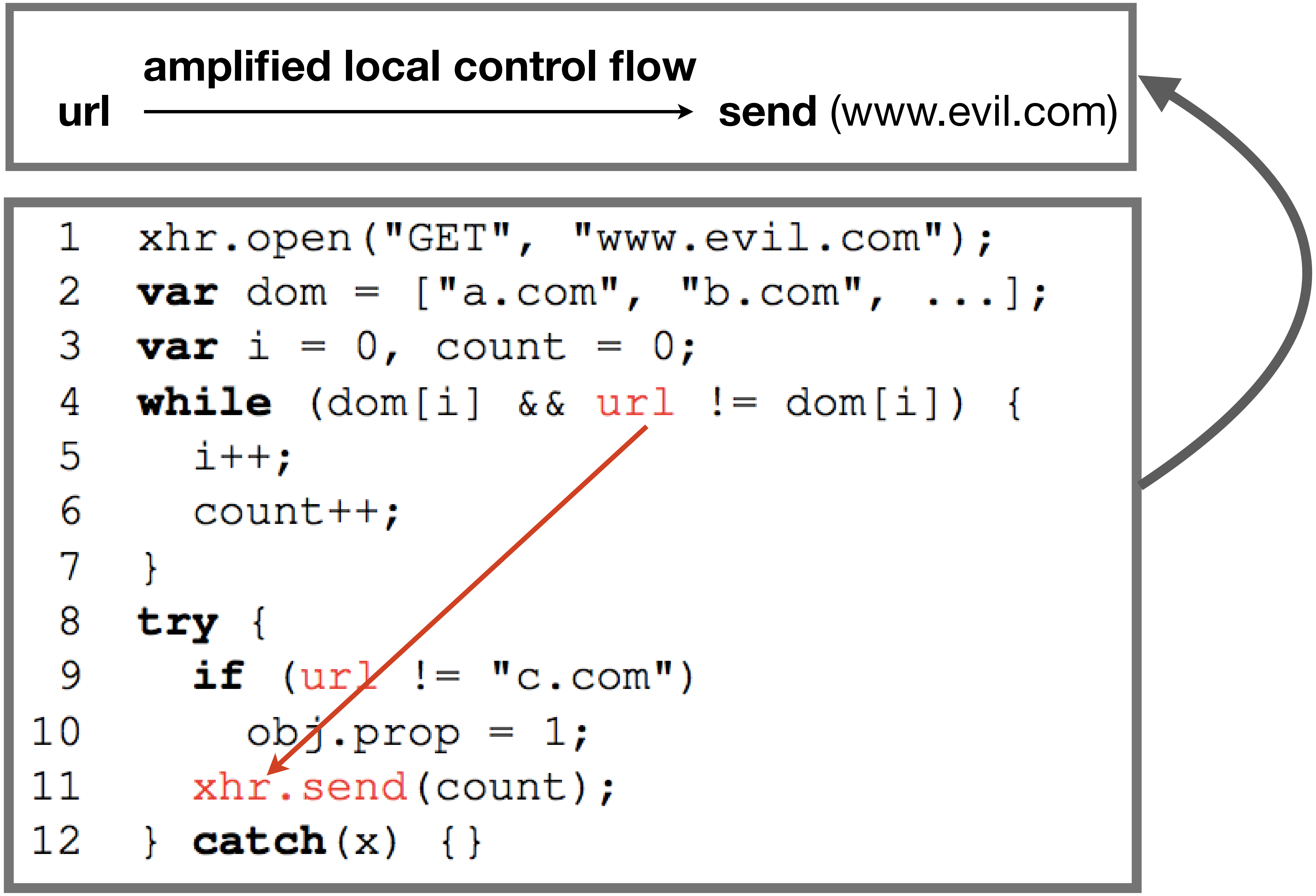
```
xhr.open("GET", "www.evil.com");
var dom = ["a.com", "b.com", ...];
var i = 0, count = 0;
while (dom[i] && url != dom[i]) {
  i++;
  count++;
}
try {
  if (url != "c.com")
    obj.prop = 1;
  xhr.send(count);
} catch(x) {}
```

Generating Security Signatures



```
xhr.open("GET", "www.evil.com");
var dom = ["a.com", "b.com", ...];
var i = 0, count = 0;
while (dom[i] && url != dom[i]) {
  i++;
  count++;
}
try {
  if (url != "c.com")
    obj.prop = 1;
  xhr.send(count);
} catch(x) {}
```

Generating Security Signatures



```
xhr.open("GET", "www.evil.com");
var dom = ["a.com", "b.com", ...];
var i = 0, count = 0;
while (dom[i] && url != dom[i]) {
  i++;
  count++;
}
try {
  if (url != "c.com")
    obj.prop = 1;
  xhr.send(count);
} catch(x) {}
```

Evaluation

- Evaluated analysis on 10 real addons from Mozilla repository
- Manually created security signatures based on submitted addon description
- Ran the analysis to get inferred signature, compared against our manual signature
- Possible experimental outcomes:
    - **pass** (no unexpected information flow)
    - **fail** (false unexpected information flow)
    - **leak** (true unexpected information flow)

Results

Addon Name	Result	|AST|	Time(s)
LivePagerank	pass	3,900	46.7
HyperTranslate	pass	3,576	40.8
Chess.comNotifier	pass	1,079	3.0
CoffeePodsDeals	pass	1,670	3.2
oDeskJobWatcher	pass	609	1.4
LessSpamPlease	fail†	3,696	28.1
VKVideoDownloader	fail†	2,016	9.5
YoutubeDownloader	leak	3,755	35.8
PinPoints	leak	2,146	20.6
GoogleTransliterate	leak	4,270	12.8

† In all these cases, the failure was due to insufficient precision in the string domain.

Conclusion

- Browser addon vetting is hard, needs automation
- Security signatures are useful to understand security behavior of addons

Implementation available under the Downloads link at
http://www.cs.ucsb.edu/~pllab

Acknowledgements

- Tommy Ashmore and Ben Wiedermann (Harvey Mudd College)
- Dave Herman (Mozilla Research)
- Mozilla Addon Vetting Team

Questions?

vineeth@cs.ucsb.edu