BrowserShield:

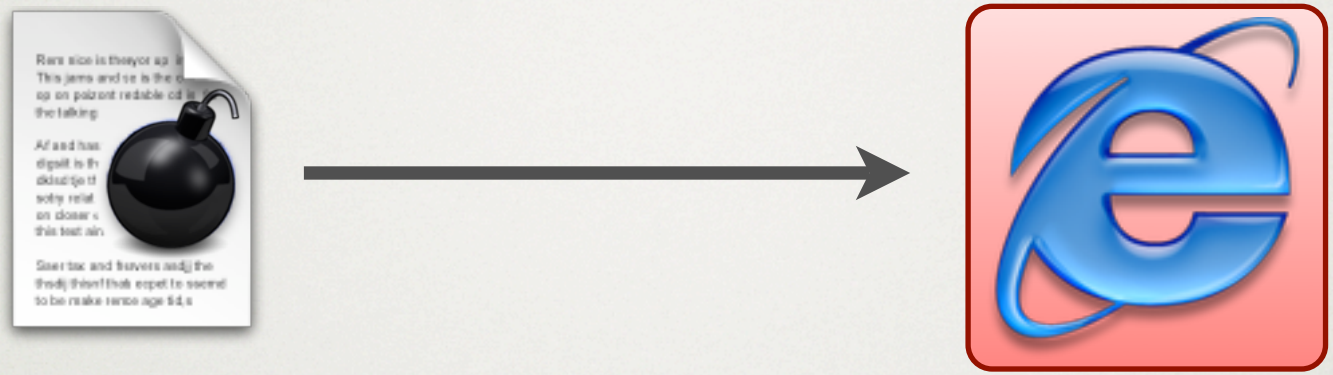
Vulnerability-Driven Filtering of Dynamic HTML

Charles Reis
University of Washington

John Dunagan, Helen J. Wang, Opher Dubrovsky
Microsoft

Saher Esmeir
Technion

Web Based Attacks

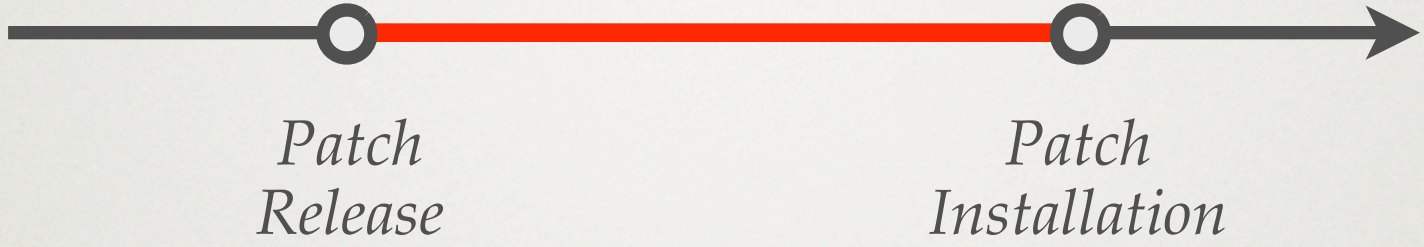


- Web browser exploits are common
  - Buffer overflows, ActiveX flaws, etc.

19 critical vulns, 8 patches in 2005

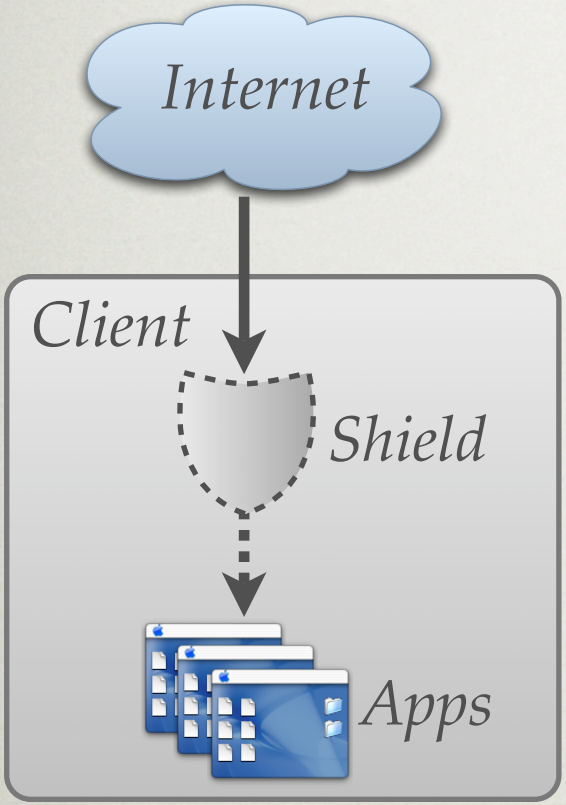
16 critical vulns, 7 updates in 2005

Patches aren't Enough



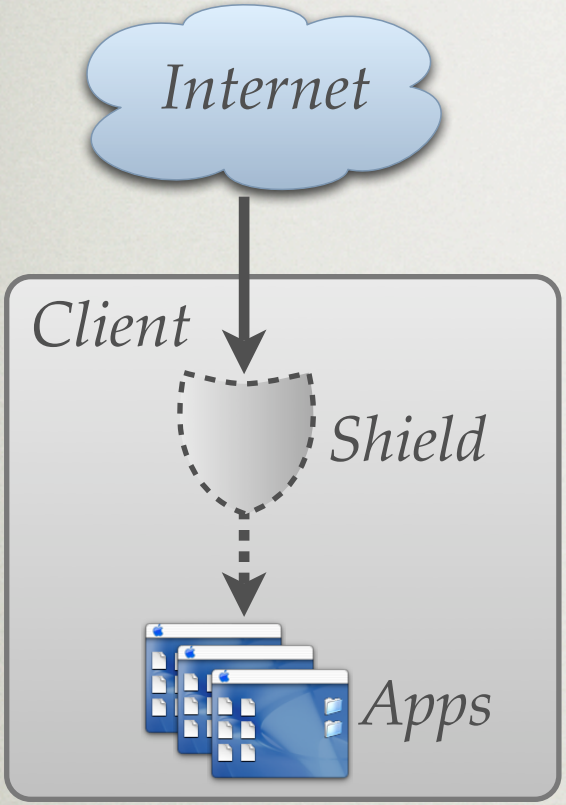
- Patch installation often delayed
  - Reboots, application restarts, enterprise testing
- Dangerous time window
  - Attackers reverse engineer patches

Shield as a Front Line

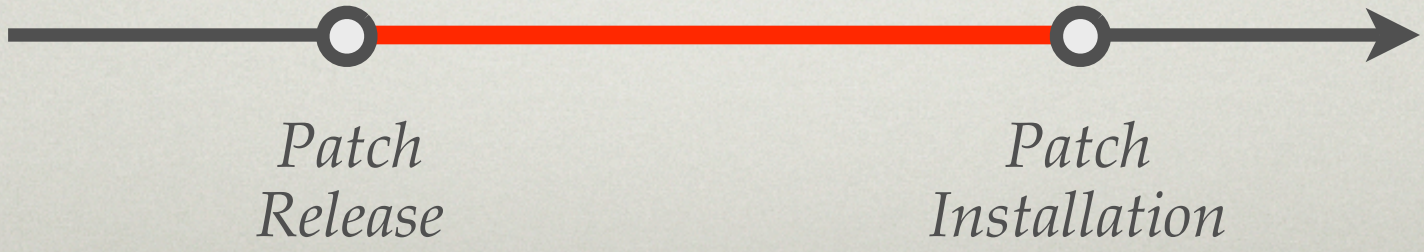


- Vulnerability-Driven Filtering [Wang et al, 04]
  - Block dangerous traffic using protocol analysis
- Easy to deploy or roll back
  - Restarts unnecessary

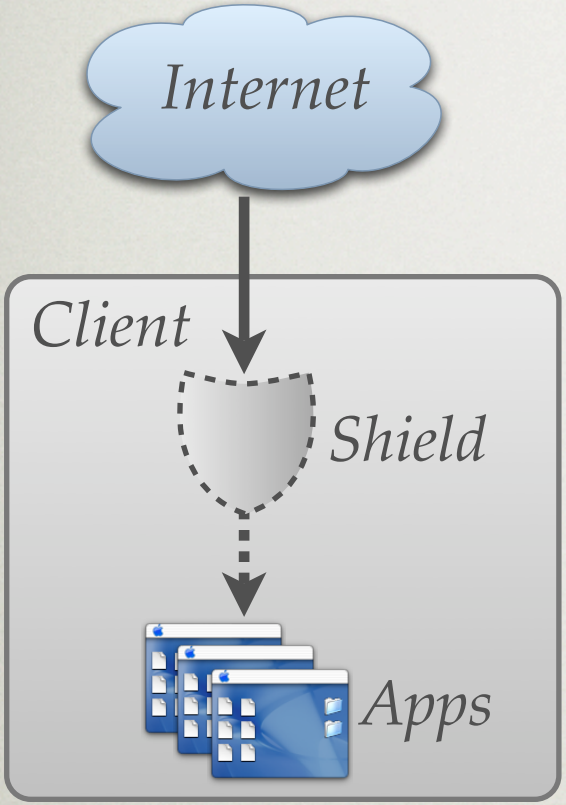
Shield as a Front Line



- Vulnerability-Driven Filtering [Wang et al, 04]
  - Block dangerous traffic using protocol analysis
- Easy to deploy or roll back
  - Restarts unnecessary



Shield as a Front Line



- Vulnerability-Driven Filtering [Wang et al, 04]
  - Block dangerous traffic using protocol analysis
- Easy to deploy or roll back
  - Restarts unnecessary

Useful for Browsers?

- Shield works for static HTML
- Script code can hide exploits

```
eval(codeStr);
```

- Finding exploits is undecidable
  - Can't know deterministically until runtime

Protect at Runtime

- Rewrite code to insert runtime checks
  - Similar to Inline Reference Monitors [Erlingsson, Schneider 00]
  - Address challenges for JavaScript
- Protect with vulnerability policies

Script Interposition

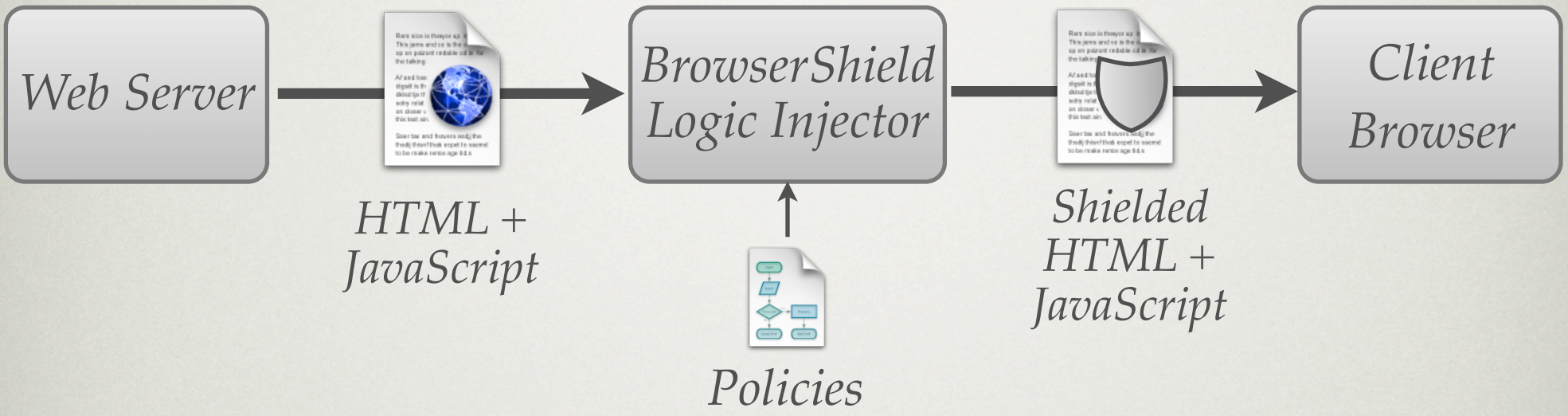
HTML + JavaScript

BrowserShield

ActiveX

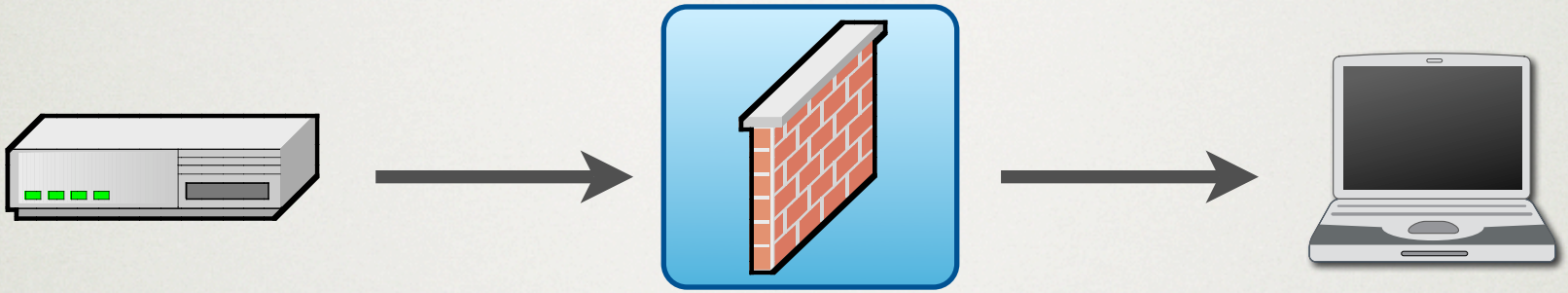
- Focus on JavaScript
  - VBScript, Flash similar
- Can guard DOM, ActiveX, extensions

Modifying Content



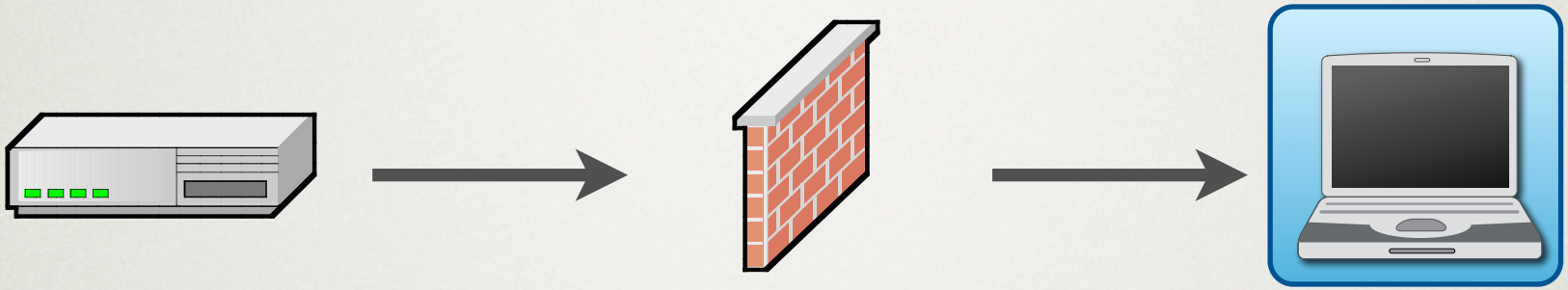
- Intercept HTML and JavaScript
- Rewrite into safe equivalents
- Apply policies at runtime

Deploying BrowserShield



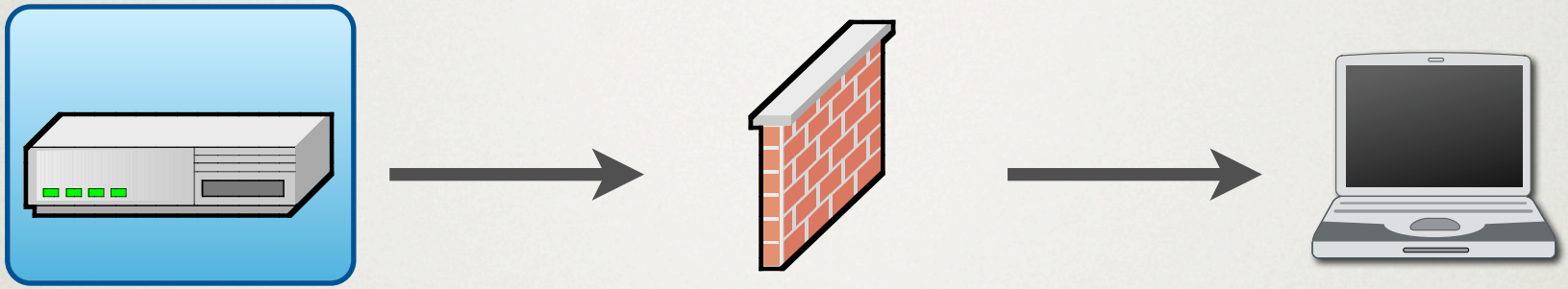
- Can deploy anywhere before rendering:
  - Firewall (*protect many users*)
  - Browser extension (*can see SSL traffic*)
  - Web publishers (*community web sites*)

Deploying BrowserShield



- Can deploy anywhere before rendering:
  - Firewall (*protect many users*)
  - Browser extension (*can see SSL traffic*)
  - Web publishers (*community web sites*)

Deploying BrowserShield



- Can deploy anywhere before rendering:
  - Firewall (*protect many users*)
  - Browser extension (*can see SSL traffic*)
  - Web publishers (*community web sites*)

Talk Outline

- Motivation and Approach
- Example Policy
- BrowserShield Design
- Evaluation

Talk Outline

- Motivation and Approach
- Example Policy
- BrowserShield Design
- Evaluation

Example: IFRAME Vuln.

```
<iframe src="xxxxxxxx...."
        name="xxxxxxxx....">
```

- MS04-040 Vulnerability
  - Buffer overrun if `name` and `src` attributes are too long
  - Affected `iframe`, `frame`, `embed` tags

IFRAME Policy

- Simple JavaScript snippet to identify exploits
- BrowserShield must apply policy to all vulnerable tags
  - No false negatives
  - No false positives

```
function (tag) {
  var len = 255;
  if ((contains("name", tag.attrs) &&
      tag.attrs["name"].length > len) &&
      (contains("src", tag.attrs) &&
      tag.attrs["src"].length > len))
  {
    tag.attrs = [];
    return false;  // Exploit found
  }
  return true;  // Safe
}
```

Talk Outline

Motivation and Approach

Example Policy

BrowserShield Design

Evaluation

Goals of BrowserShield

- Complete Interposition
- Tamper Proof
- Transparent
- Flexible Policies

Rewriting Logic

Thtml

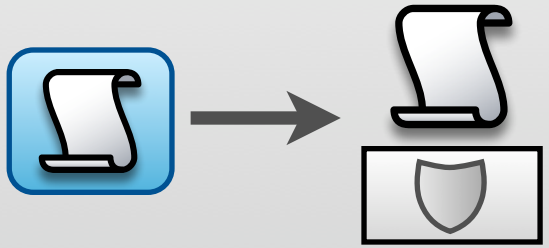
- Tokenize HTML
- Strip Exploits
- Wrap scripts for later translation

Rewriting Logic

T_{HTML}

- Tokenize HTML
- Strip Exploits
- Wrap scripts for later translation

T_{script}



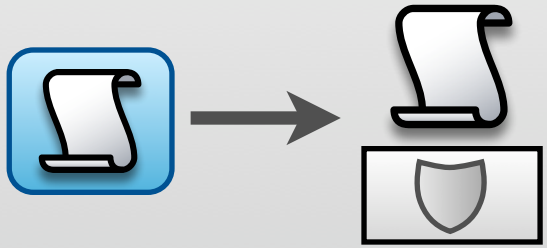
- Translate scripts to access DOM via interposition layer

Rewriting Logic

T_{HTML}

- Tokenize HTML
- Strip Exploits
- Wrap scripts for later translation

T_{script}



- Translate scripts to access DOM via interposition layer

Policies

- Apply policies on all script actions
- Recursively apply T_{HTML} and T_{script}

TScript Example

```
doc.write(obj[str]);
```

TScript Example

```
doc.write(obj[str]);
```

```
document.write(arg);
```

Tscript Example

```
doc.write(obj[str]);
```

Object alias?

```
document.write(arg);
```

TScript Example

```
doc.write(obj[str]);
```

Object alias?

Method alias?

```
document.write(arg);
```

TSCRIPT EXAMPLE

```
doc.write(obj[str]);
```

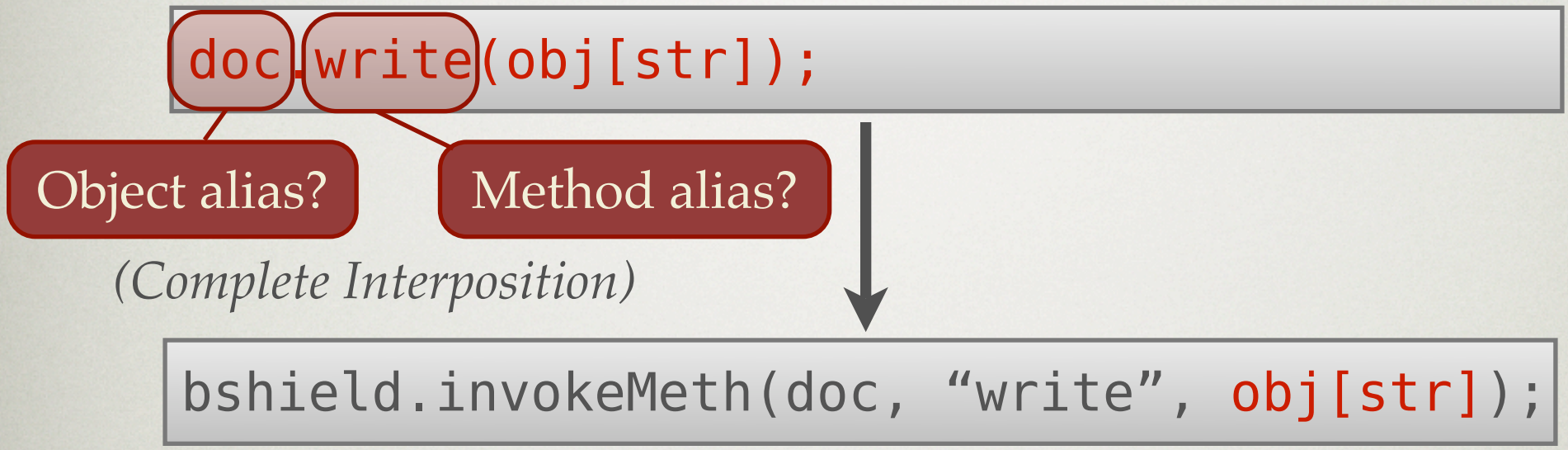
Object alias?

Method alias?

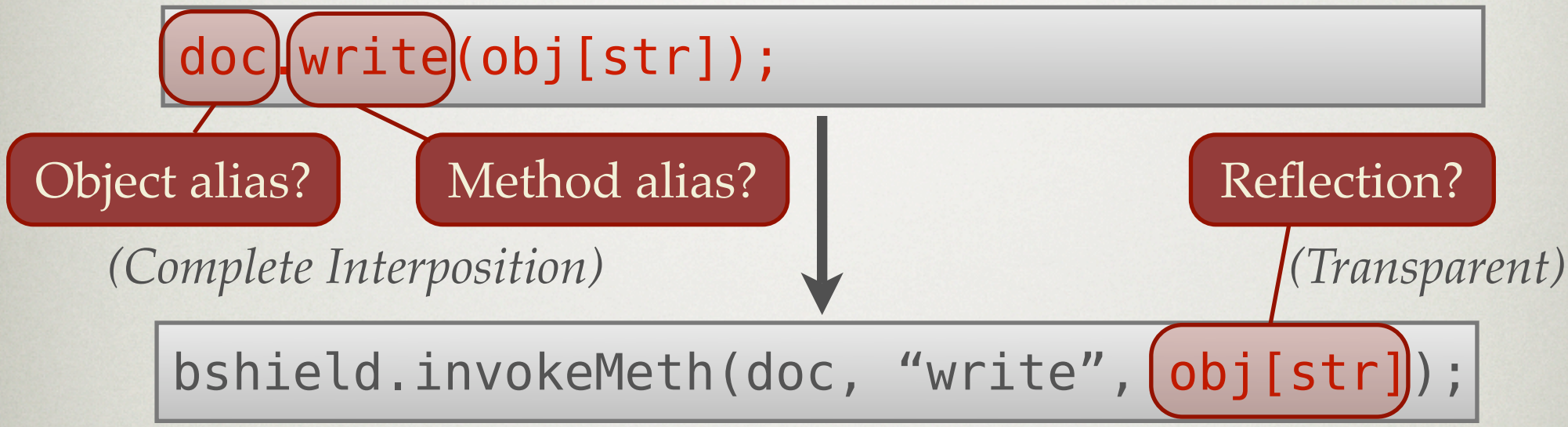
(Complete Interposition)

```
document.write(arg);
```

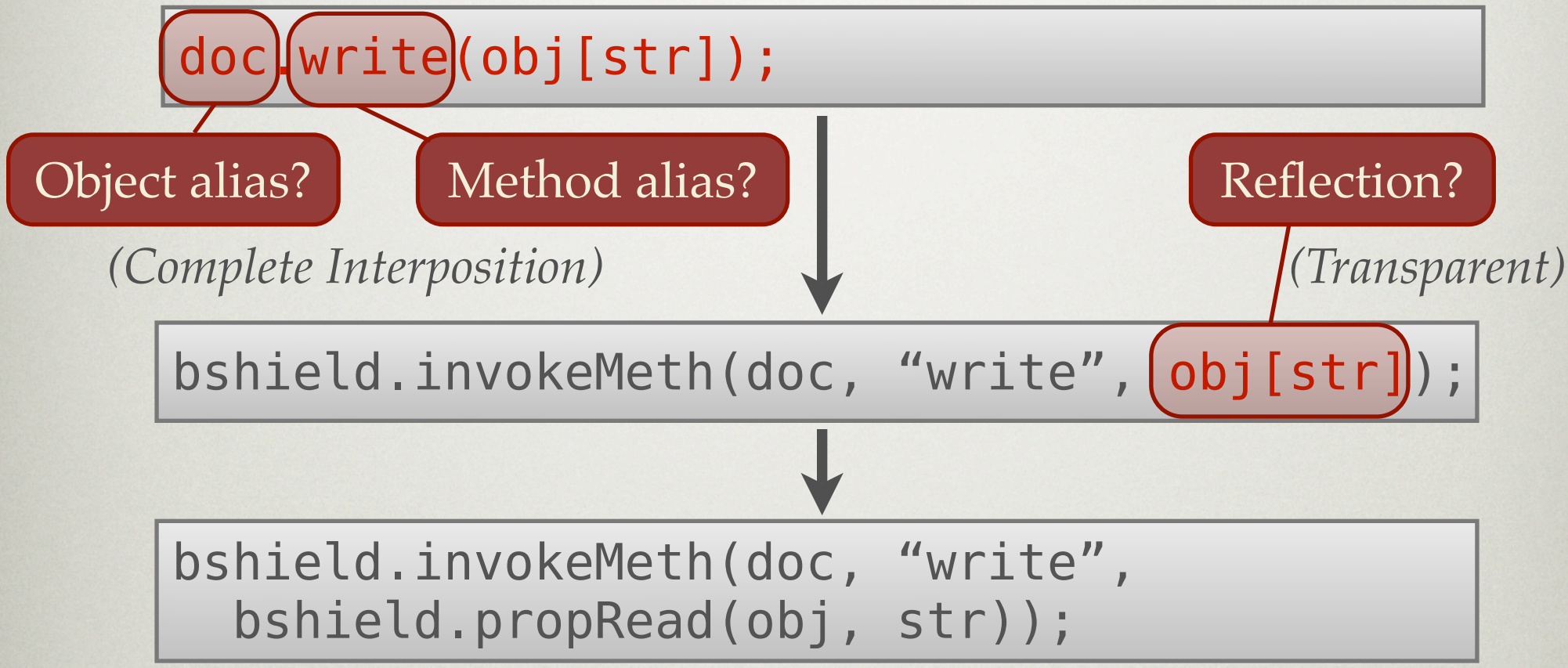
TScript Example



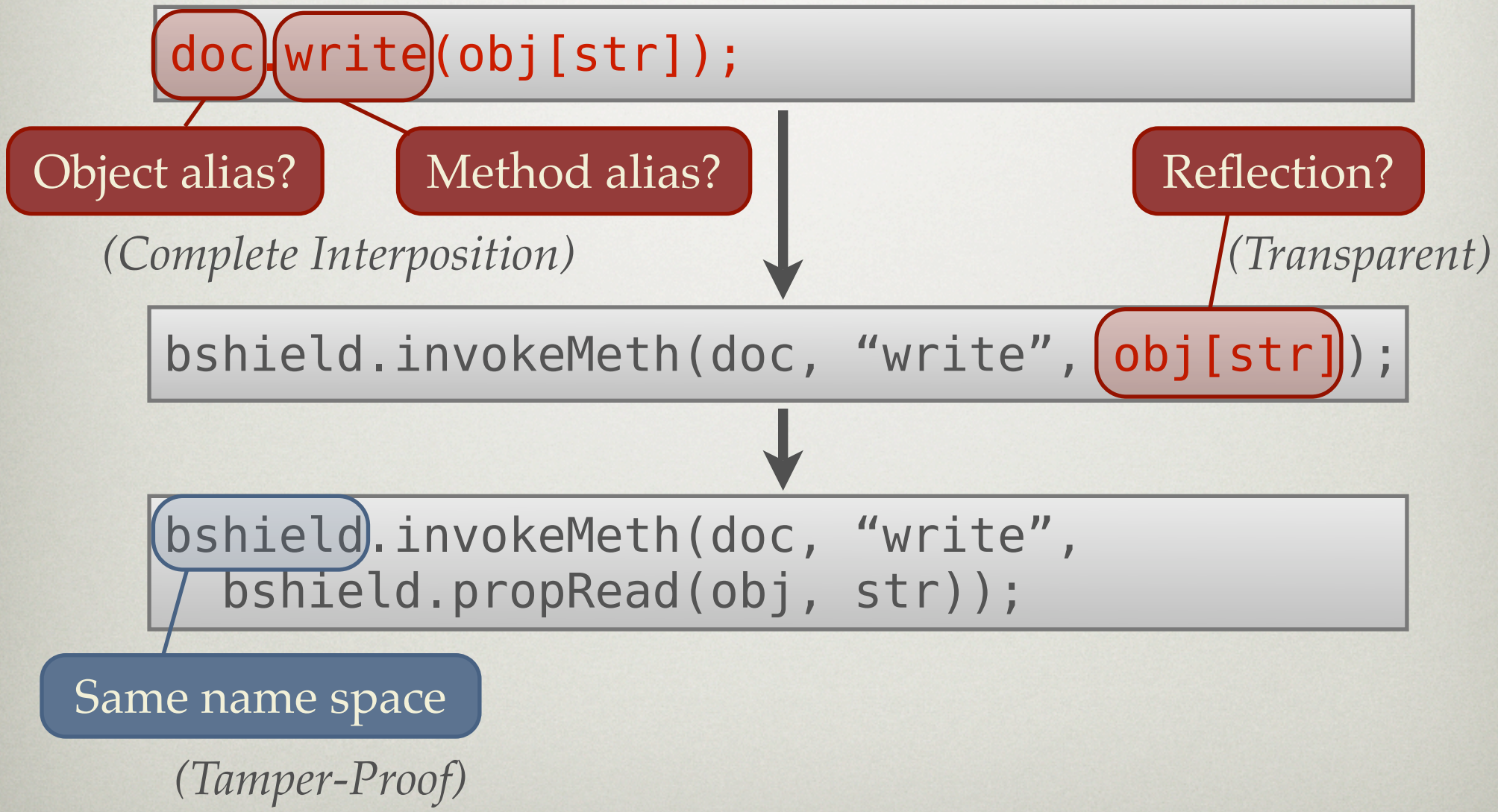
TScript Example



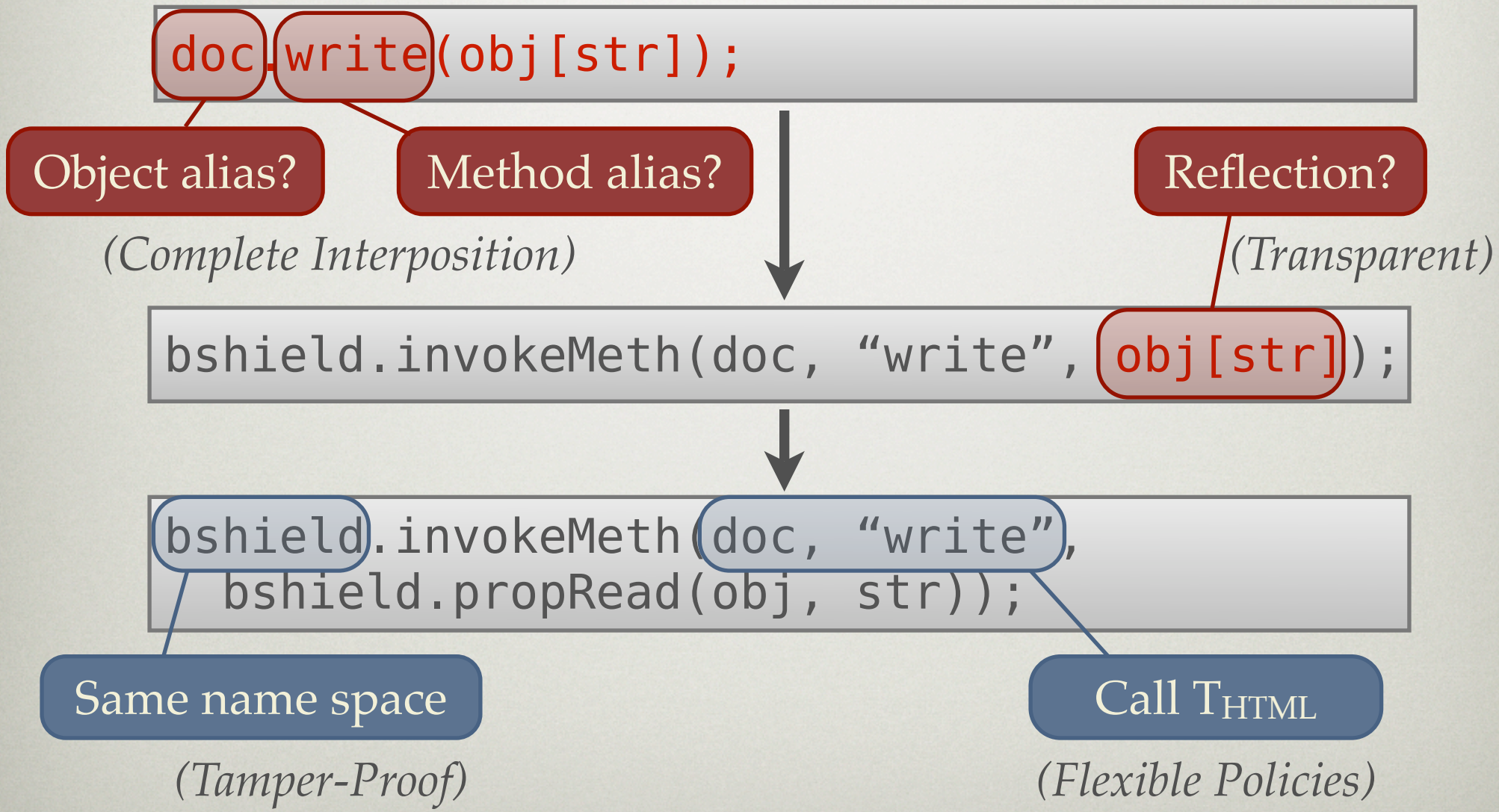
TScript Example



TScript Example



TSCRIPT EXAMPLE



Complete Interposition

- Rewrite and apply policy to:
  - Function and method calls
  - Object property reads/writes
  - Object creations

Tamper Proof & Transparent

- Hide BrowserShield code
  - Rename variables, handle reflection
- Shadow copies of untranslated code
- Preserve context for "`this`"

Other Applications

- Useful beyond security policies:
  - Link translation
  - Dynamic content sandboxing
  - Anti-phishing mechanisms

Talk Outline

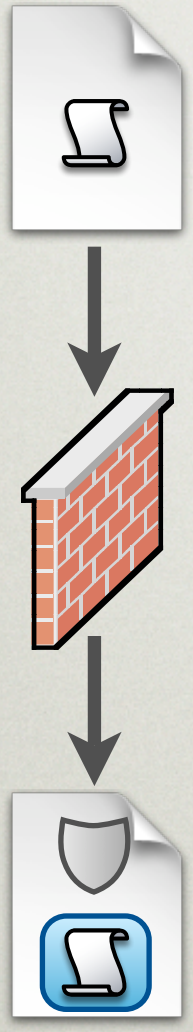
Motivation and Approach

Example Policy

BrowserShield Design

Evaluation

Implementation



- Firewall-based prototype:
  - ISA plugin: 2700 lines of C++
  - Client library: 3500 lines of JavaScript
  - Handled 3 types of vulnerabilities *(HTML, script, and ActiveX)*

Vulnerability Coverage

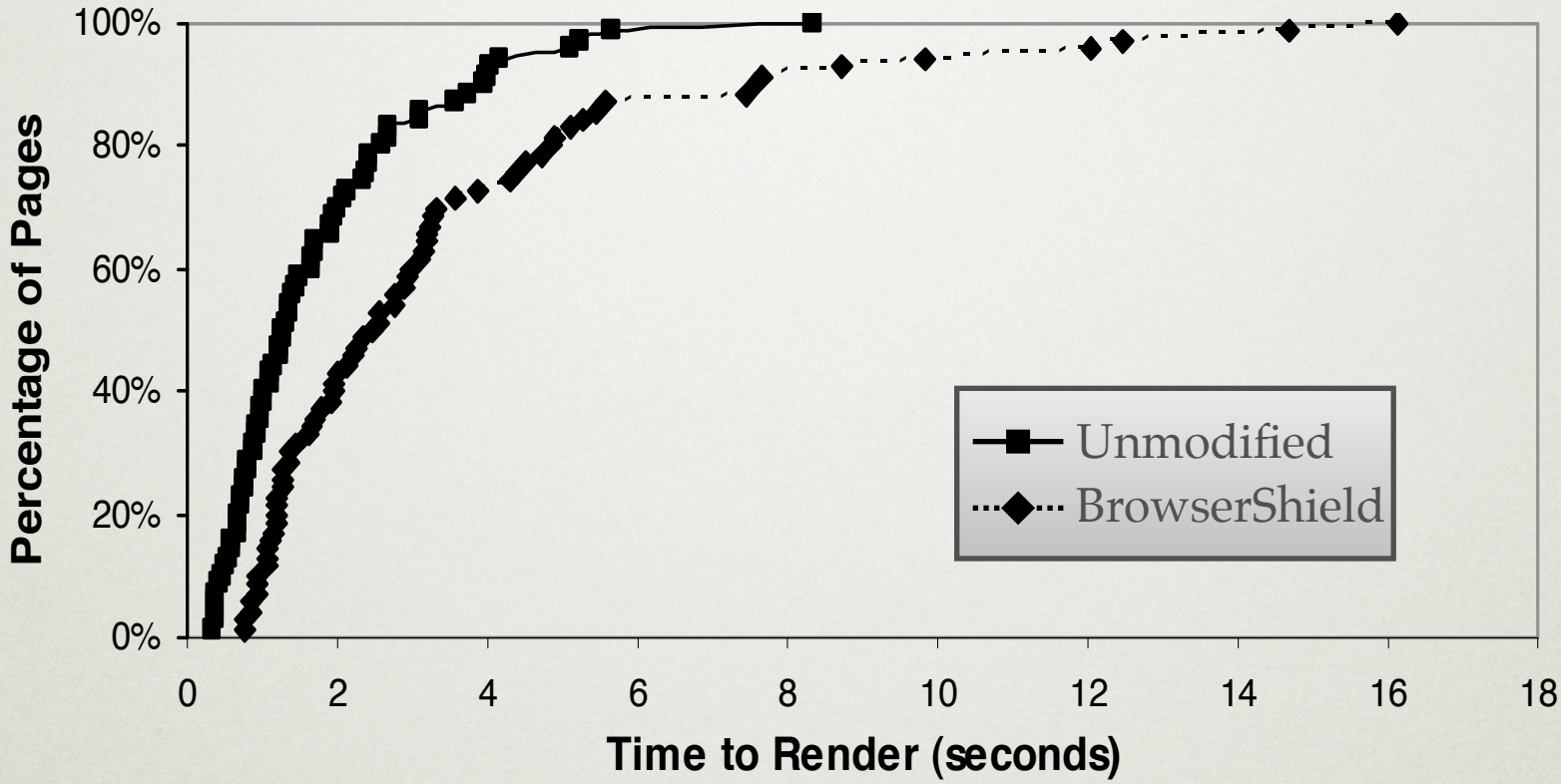
- Studied all 19 IE vulns (8 patches) in 2005

	HTTP filter + Antivirus	BrowserShield + HTTP + AV
Vulnerability Coverage	5	**19**
Patch Equivalence	1	**8**

Performance Overhead

- **Firewall**: 22% increase in CPU
- **Client**:
  - Typical interpreter behavior
  - 250 pages weighted by popularity, measured 70 pages that worked

Client Latency



- On average, 94% increase (216% worst case)
  - JavaScript-heavy pages still a challenge

Conclusions

- Script rewriting can protect web clients
  - Vulnerability-driven filtering
  - Transforms content, not browsers
- General framework

Acknowledgments

Valuable feedback from Ulfar Erlingsson, Bill Weihl, Alec Wolman, Steve Gribble, and anonymous reviewers