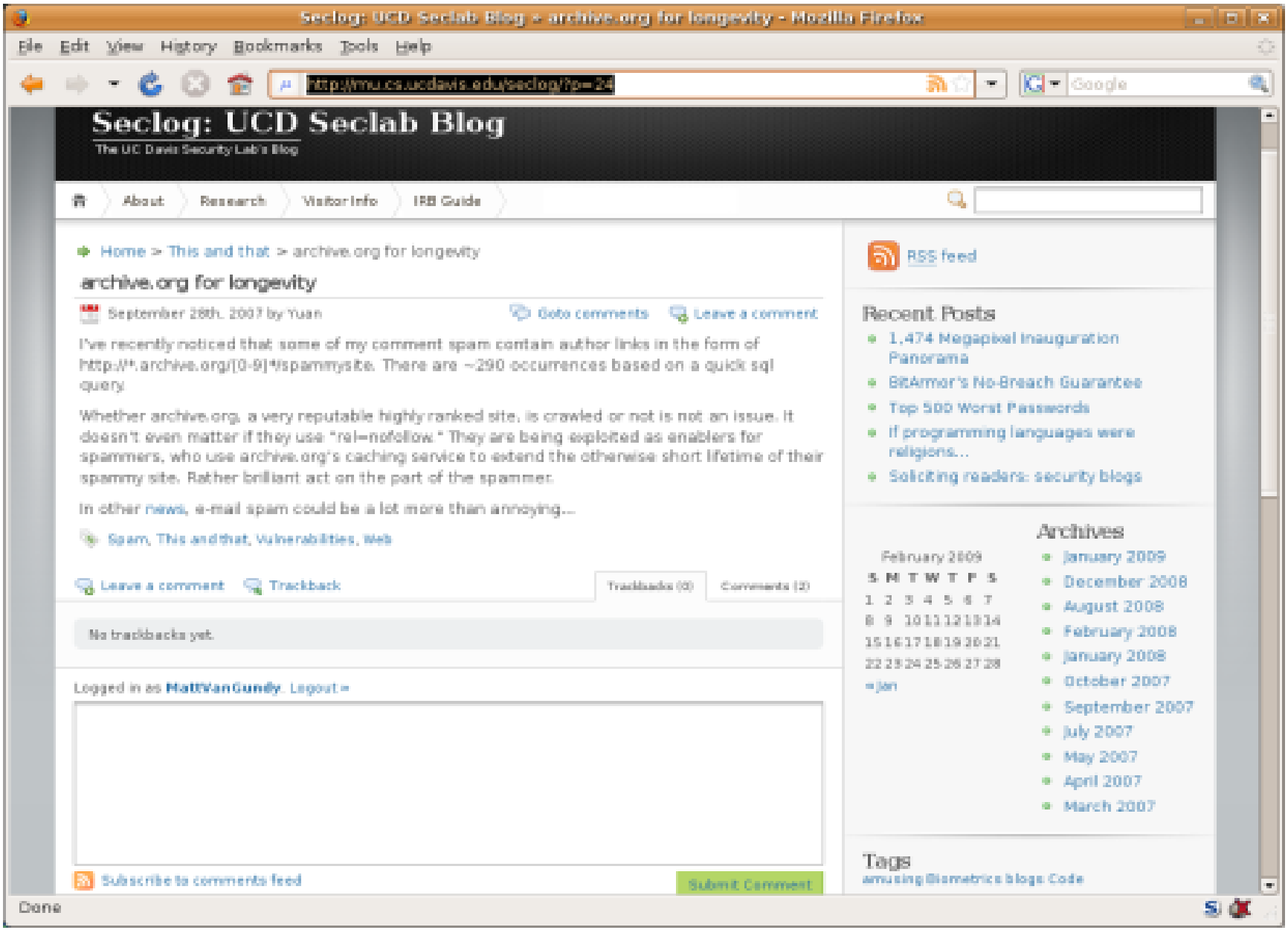
Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks

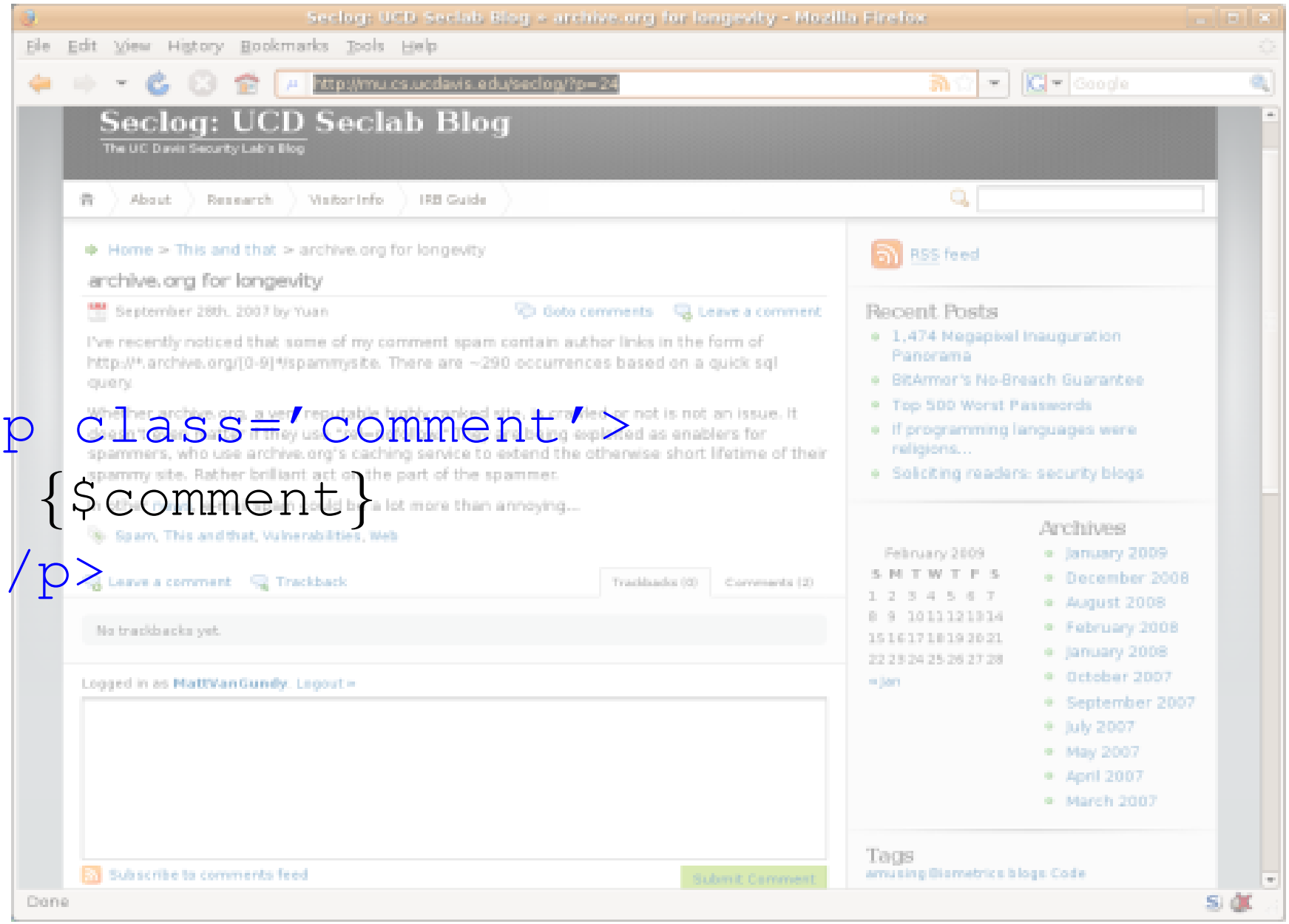
Matthew Van Gundy and Hao Chen
University of California, Davis

16th Annual Network & Distributed System Security Symposium

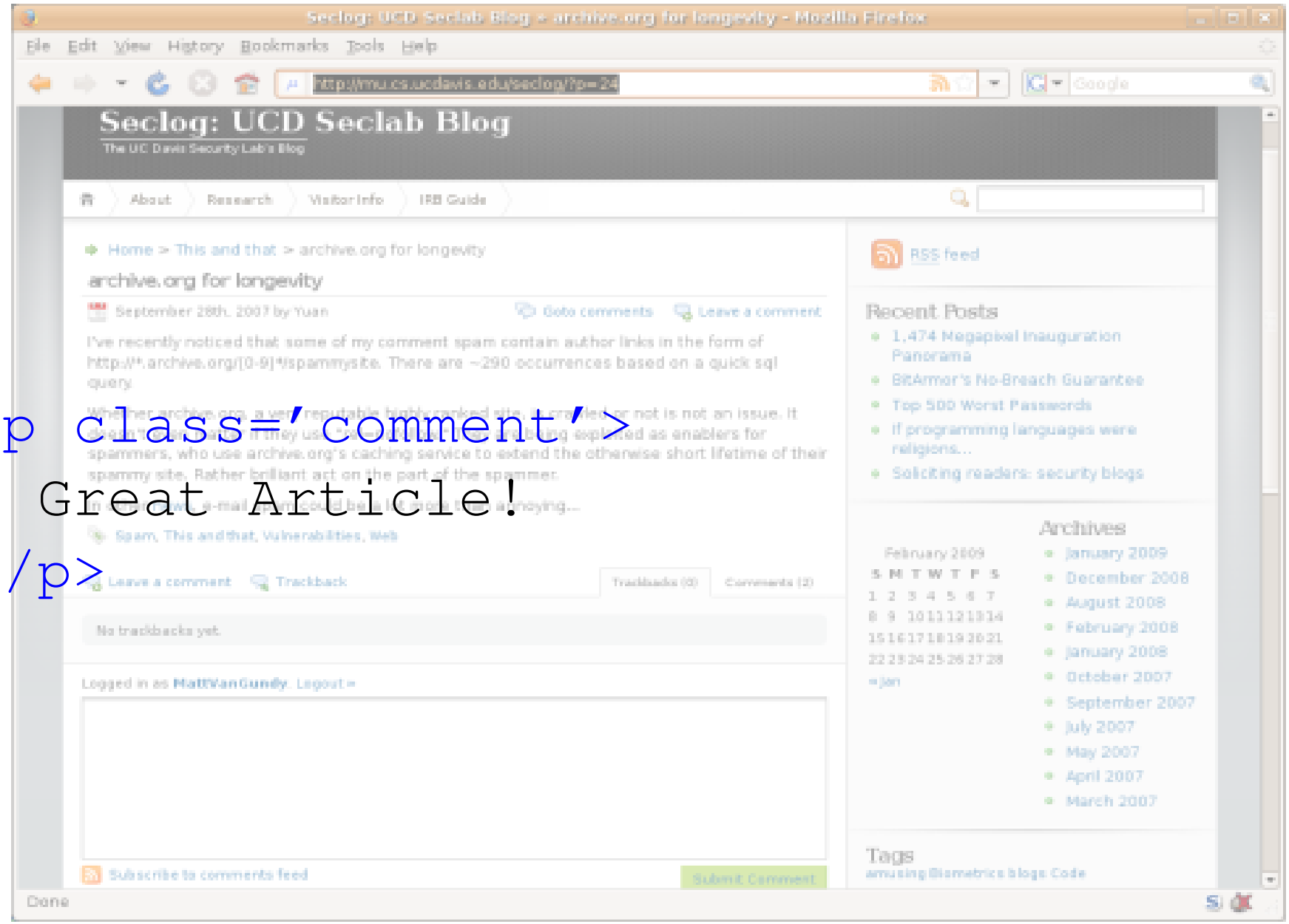
Cross-Site Scripting (XSS) Vulnerabilities



Cross-Site Scripting (XSS) Vulnerabilities

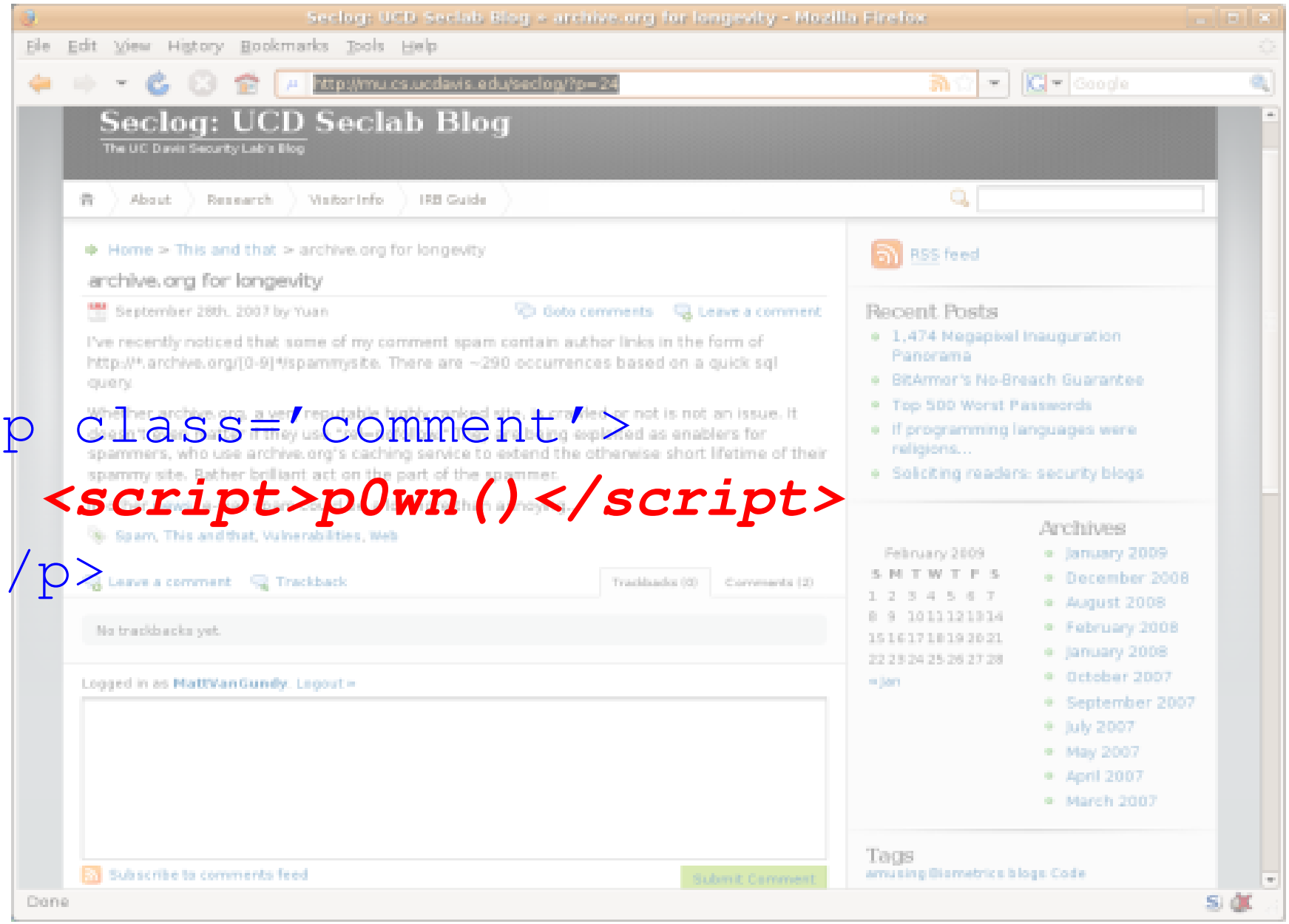


Cross-Site Scripting (XSS) Vulnerabilities

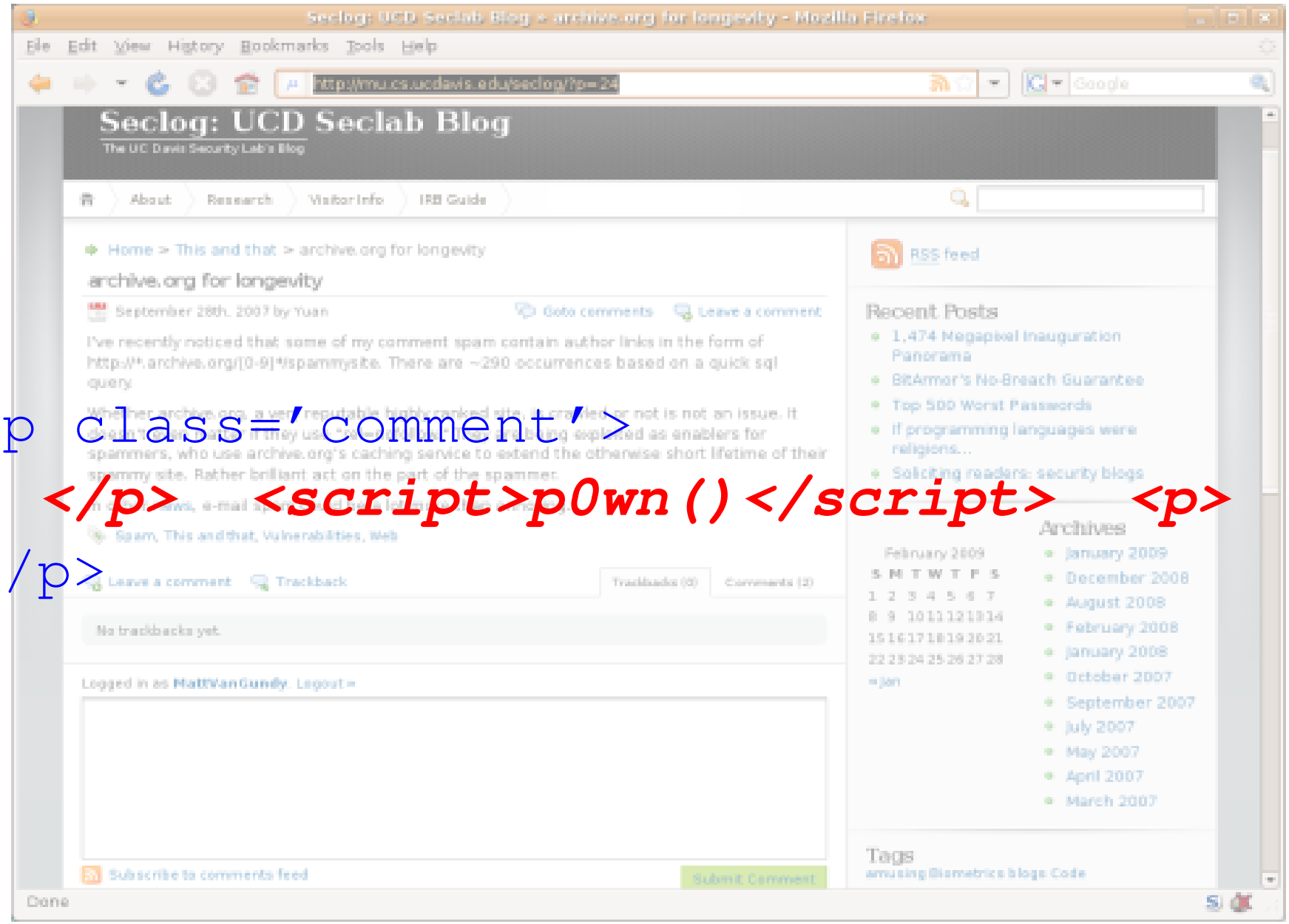



```
<p class='comment'>
  Great Article!
</p>
```

Cross-Site Scripting (XSS) Vulnerabilities



Cross-Site Scripting (XSS) Vulnerabilities




```
<p class='comment'>
  </p> <script>p0wn()</script>  <p>
</p>
```

Threat Model

- An attacker can submit arbitrary content to XSS-vulnerable applications
- An attacker cannot compromise web server or browser directly
- Malicious content must contain XHTML tags and attributes

Limitations of Existing Solutions

Server-side

- Server *sanitizes* untrusted data before sending it to the client
- Client may interpret data in an unexpected way
- E.g. Server replaces `"<script>"` with `""`
  But attacker injects `<script/xss>`

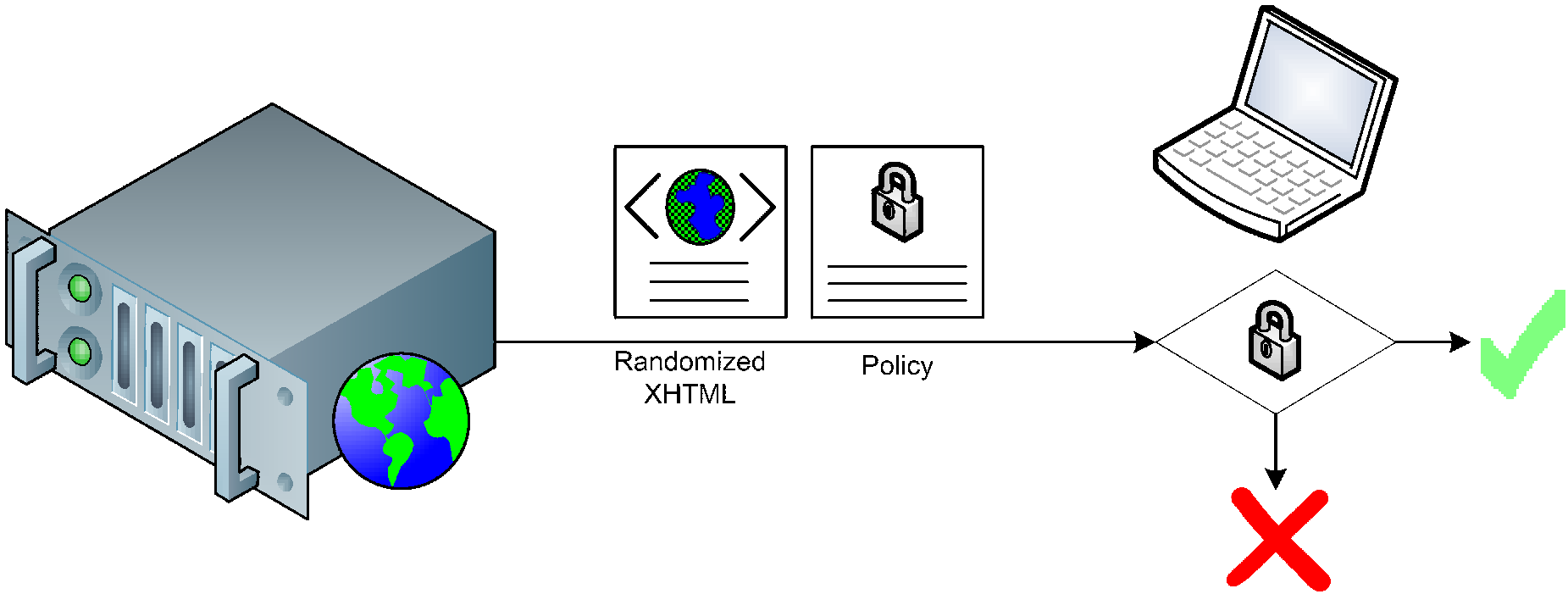
Client-side

- Client enforces a server-specified policy

Challenges

- The client must know whether to trust content
- Attacker must not be able to forge trust metadata

Noncespaces Architecture



- Server partitions content into *trust classes*
- Server randomizes document to prevent forging of trust classification
- Server specifies policy of content permitted for each trust class
- Client displays the document only if it conforms to the policy

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)
- `<x:q xmlns:x="http://www.w3.org/1999/xhtml">`

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)
- `<`$\underbrace{\texttt{x}}$`:`$\underbrace{\texttt{q}}$ `xmlns:x="`$\underbrace{\texttt{http://www.w3.org/1999/xhtml}}_{\text{NamespaceURI}}$`">`

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)
- `<`x`:`q `xmlns:x="`http://www.w3.org/1999/xhtml`">`
  (x: prefix; http://www.w3.org/1999/xhtml: NamespaceURI)

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)
- `<`$\underbrace{\texttt{x}}_{\text{prefix}}$`:`$\underbrace{\texttt{q}}_{\text{name}}$ `xmlns:x="`$\underbrace{\texttt{http://www.w3.org/1999/xhtml}}_{\text{NamespaceURI}}$`">`

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)
- `<`x (prefix) `:` q (name) `xmlns:x="`http://www.w3.org/1999/xhtml (NamespaceURI)`">`
- `<f:q xmlns:f="urn:FAQML">`

Namespaces in XML

- In (X)HTML: `<q>` = quote, `<a>` = anchor
- In FAQML: `<q>` = question, `<a>` = answer
- XHTML quote = (`"http://www.w3.org/1999/xhtml"`, `"q"`)
- FAQML question = (`"urn:FAQML"`, `"q"`)
- `<`x (prefix) `:` q (name) `xmlns:x="`http://www.w3.org/1999/xhtml (NamespaceURI)`">`
- `<f:q xmlns:f="urn:FAQML">`
- `<faq:q xmlns:faq="urn:FAQML">`

Defeating Node Splitting

- `<x:a>...</x:a>`

Defeating Node Splitting

- `<x:a>...</x:a>`
- `<x:a>...</a>`

Defeating Node Splitting

- `<x:a>...</x:a>`
- `<x:a>...</a>`
- `<x:a>...</y:a>`

Encoding Trust Classifications

- Trusted `<a>`

Encoding Trust Classifications

- Trusted `<a>` ⇒ `<t:a>`

Encoding Trust Classifications

- Trusted `<a>` ⇒ `<t:a>`
- Untrusted `<a>`

Encoding Trust Classifications

- Trusted `<a>` $\Rightarrow$ `<t:a>`
- Untrusted `<a>`
- Randomly choose trusted prefixes to prevent forgery

Web Page Before Noncespaces

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>nile.com : ++Shopping</title>
</head>
<body>
<h1 id="title">{$item->name}</h1>

<h2>Reviews</h2>
<p class='review'>
  {$review}
</p>
</body>
</html>
```

Node Splitting Attack After Noncespaces

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<r617:html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:r617="http://www.w3.org/1999/xhtml">
<r617:head>
  <r617:title>nile.com : ++Shopping</r617:title>
</r617:head>
<r617:body>
<r617:h1 r617:id="title">Useless Do-dad</r617:h1>

<r617:h2>Reviews</r617:h2>
<r617:p r617:class='review'>
  </p>  <script>p0wn()</script>  <p>
</r617:p>
</r617:body>
</r617:html>
```

XSS Attack After Noncespaces

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<r617:html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:r617="http://www.w3.org/1999/xhtml">
<r617:head>
  <r617:title>nile.com : ++Shopping</r617:title>
</r617:head>
<r617:body>
<r617:h1 r617:id="title">Useless Do-dad</r617:h1>

<r617:h2>Reviews</r617:h2>
<r617:p r617:class='review'>
  <script src='http://badguy.com/p0wn.js' />
</r617:p>
</r617:body>
</r617:html>
```

Need for a client-side policy

Innocuous Input

```
<b>WARNING:</b>
```

Need for a client-side policy

Innocuous Input

```
<b>WARNING:</b>

<em>very</em> important
```

Need for a client-side policy

Innocuous Input

```
<b>WARNING:</b>

<em>very</em> important

<a href='http://useful.com/'>[1]</a>
```

Need for a client-side policy

Innocuous Input

```
<b>WARNING:</b>

<em>very</em> important

<a href='http://useful.com/'>[1]</a>
```

Malicious Input

```
<b onmouseover='...'>WARNING:</b>
```

Need for a client-side policy

Innocuous Input

```
<b>WARNING:</b>

<em>very</em> important

<a href='http://useful.com/'>[1]</a>
```

Malicious Input

```
<b onmouseover='...'>WARNING:</b>

<em onclick='...'>very</em> important
```

Need for a client-side policy

Innocuous Input

```
<b>WARNING:</b>
<em>very</em> important
<a href='http://useful.com/'>[1]</a>
```

Malicious Input

```
<b onmouseover='...'>WARNING:</b>
<em onclick='...'>very</em> important
<a href='javascript:...'>[1]</a>
```

Need for a client-side policy

XHTML

```
<b>

<em>

<a href='http:...'>
```

Policy

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	
`<a href='http:...'>`	

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	`allow //untrusted:em`
`<a href='http:...'>`	

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	`allow //untrusted:em`
`<a href='http:...'>`	`allow //untrusted:a/@untrusted:href[ starts-with(normalize-space(.), "http:")]`

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	`allow //untrusted:em`
`<a href='http:...'>`	`allow //untrusted:a/@untrusted:href[ starts-with(normalize-space(.), "http:")]`
`<b onmouseover=''>`	
`<em onclick=''>`	
`<a href='java...'>`	

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	`allow //untrusted:em`
`<a href='http:...'>`	`allow //untrusted:a/@untrusted:href[ starts-with(normalize-space(.), "http:")]`
`<b onmouseover=''>`	`deny //@untrusted:onmouseover`
`<em onclick=''>`	
`<a href='java...'>`	

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	`allow //untrusted:em`
`<a href='http:...'>`	`allow //untrusted:a/@untrusted:href[ starts-with(normalize-space(.), "http:")]`
`<b onmouseover=''>`	`deny //@untrusted:onmouseover`
`<em onclick=''>`	`deny //@untrusted:*`
`<a href='java...'>`	

Need for a client-side policy

XHTML	Policy
`<b>`	`allow //untrusted:b`
`<em>`	`allow //untrusted:em`
`<a href='http:...'>`	`allow //untrusted:a/@untrusted:href[ starts-with(normalize-space(.), "http:")]`
`<b onmouseover=''>`	`deny //@untrusted:onmouseover`
`<em onclick=''>`	`deny //@untrusted:*`
`<a href='java...'>`	`deny //@untrusted:href[ starts-with(normalize-space(.), "javascript:")]`

Determining Trusted Content

- Design patterns separate presentation and business logic
- Templates contain static HTML (presentation)

- Program creates dynamic content from user input

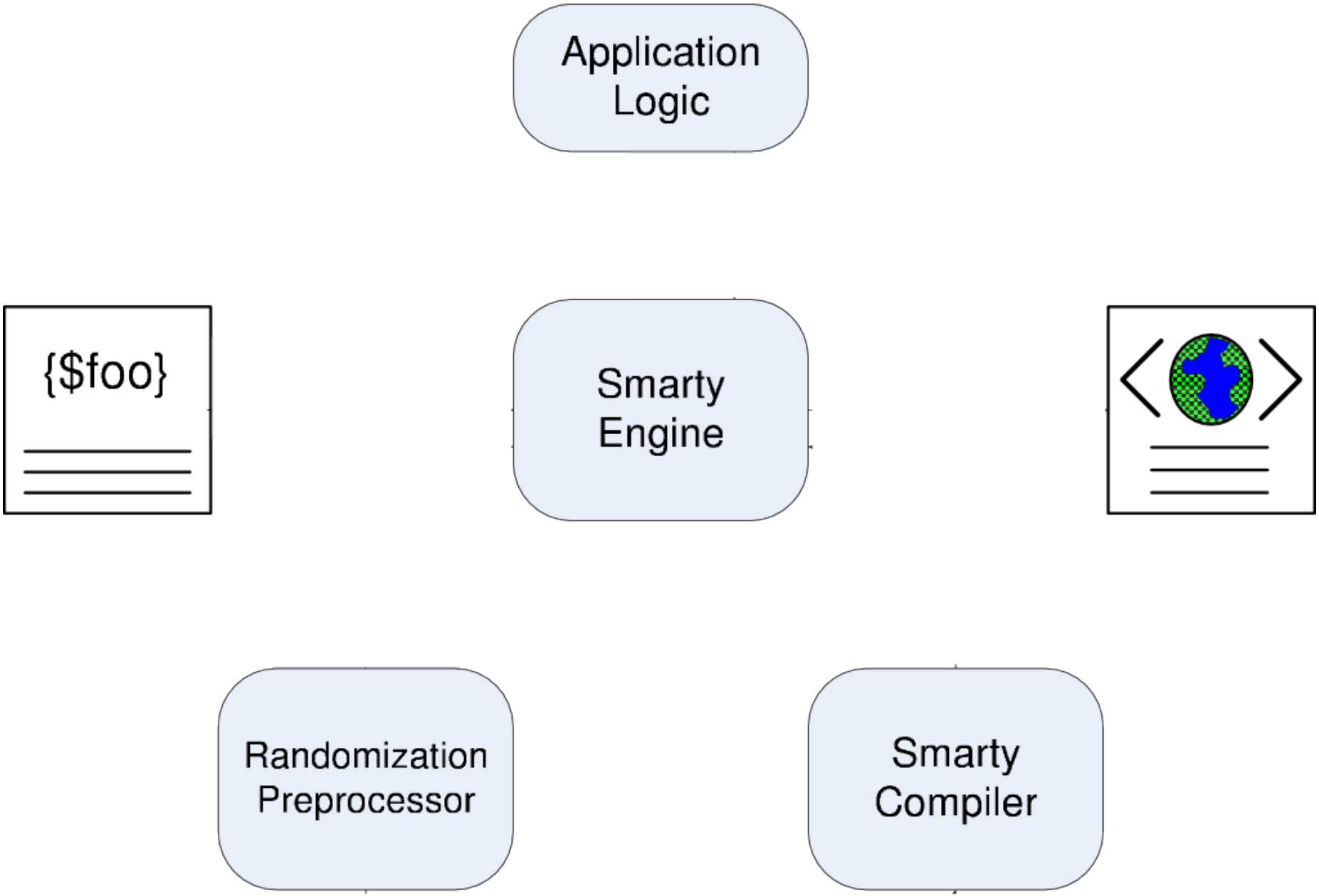
Determining Trusted Content

- Design patterns separate presentation and business logic
- Templates contain static HTML (presentation)
  - Classify as trusted
- Program creates dynamic content from user input

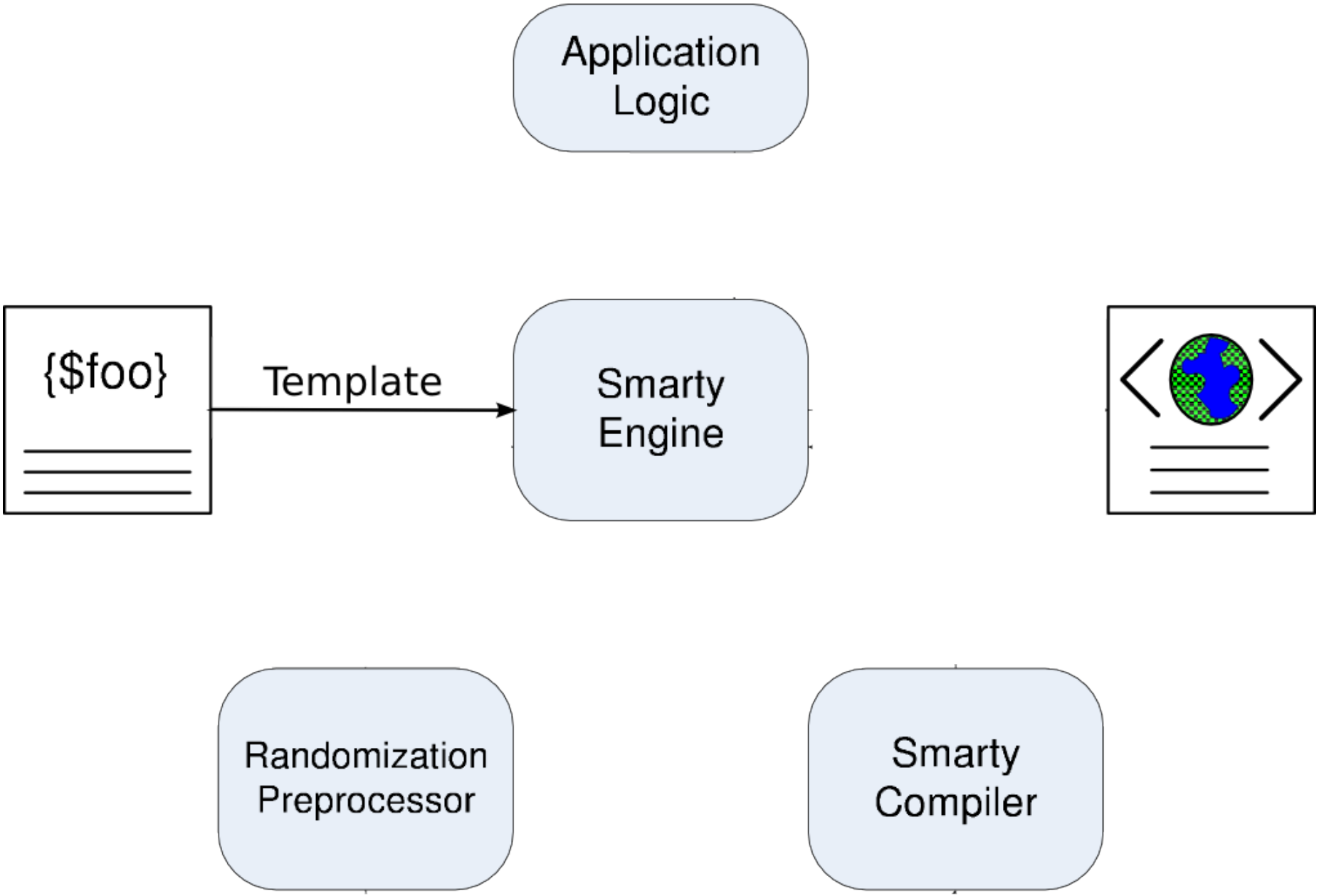
Determining Trusted Content

- Design patterns separate presentation and business logic
- Templates contain static HTML (presentation)
  - Classify as trusted
- Program creates dynamic content from user input
  - Classify as untrusted

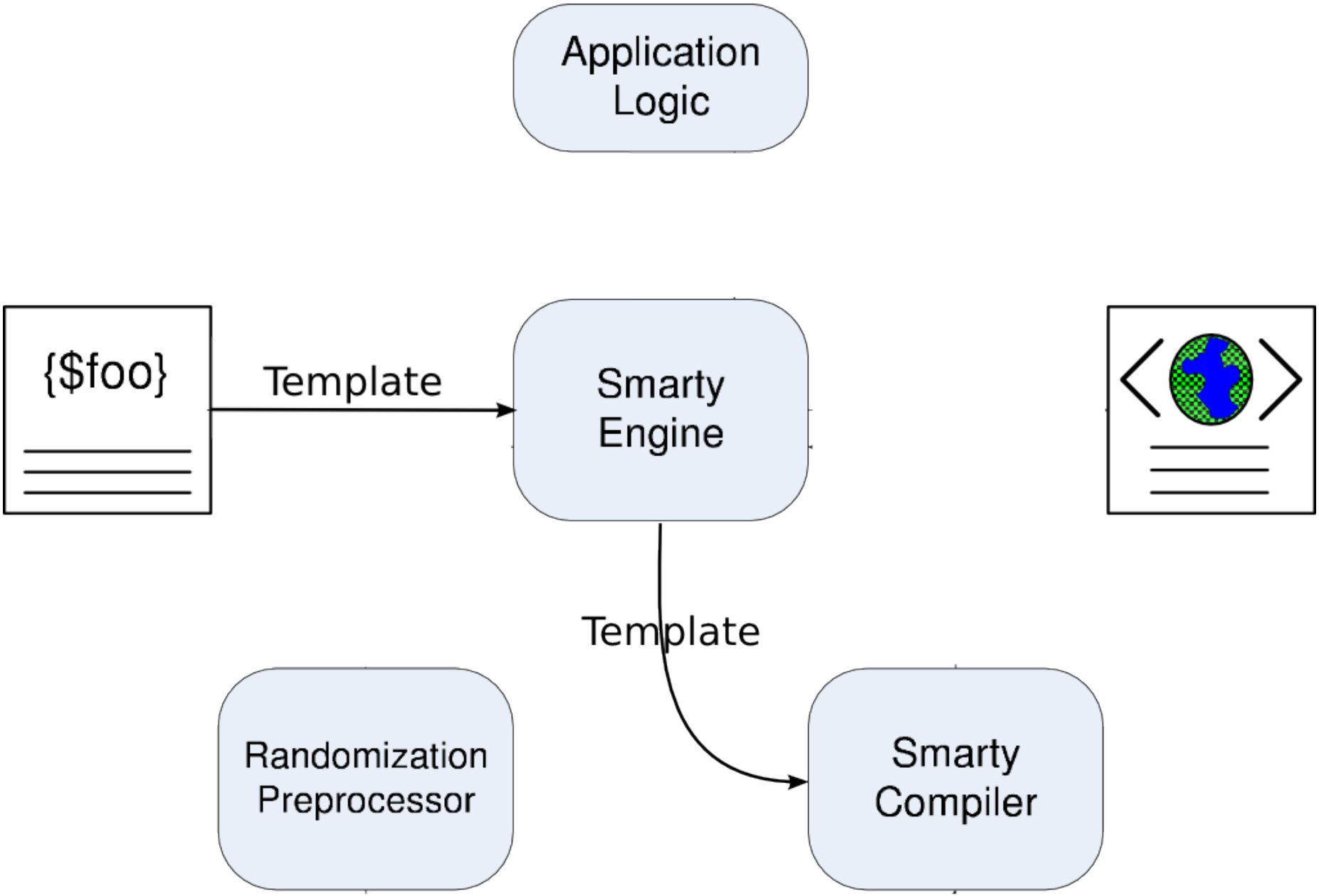
Modifications to Smarty



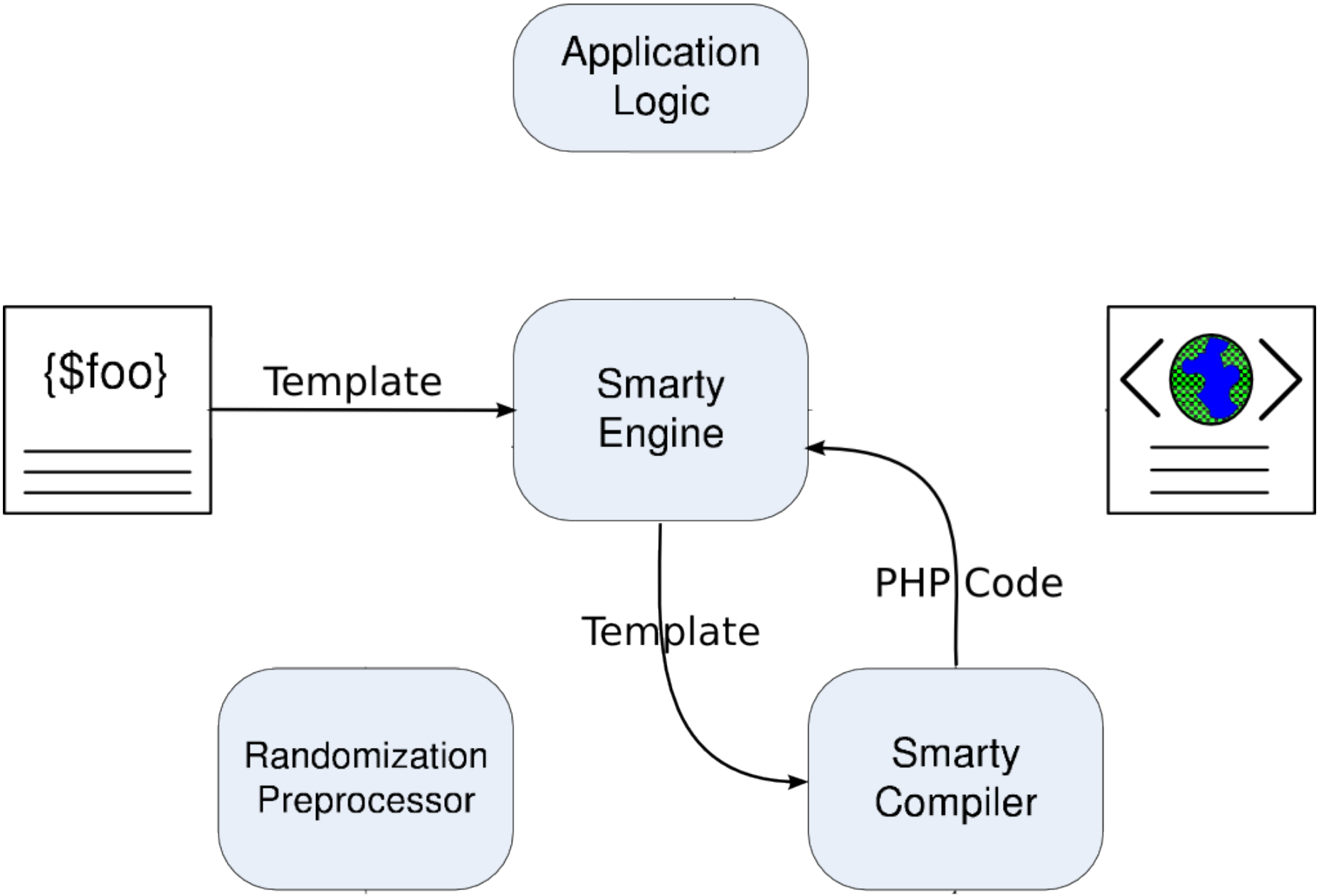
Modifications to Smarty



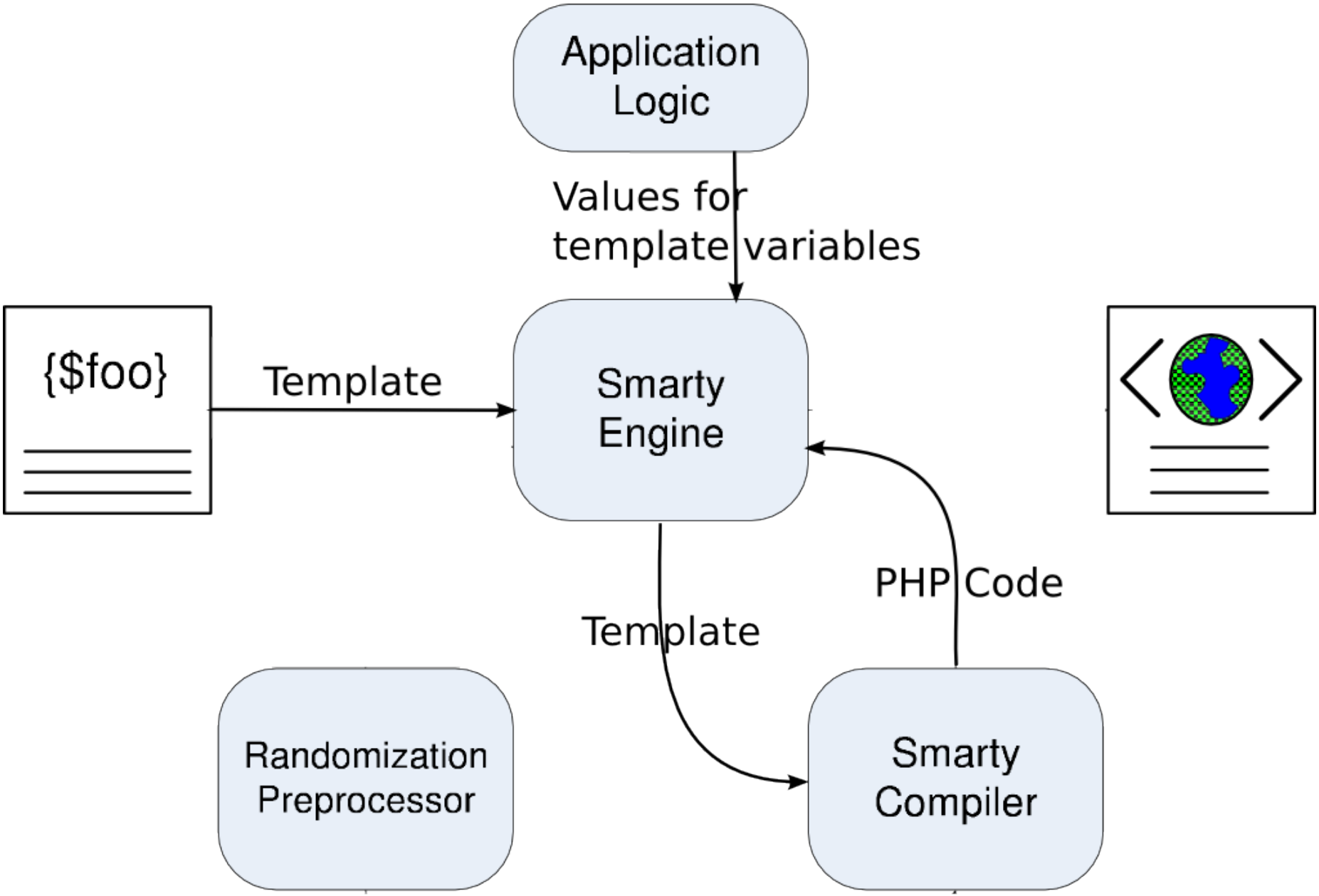
Modifications to Smarty



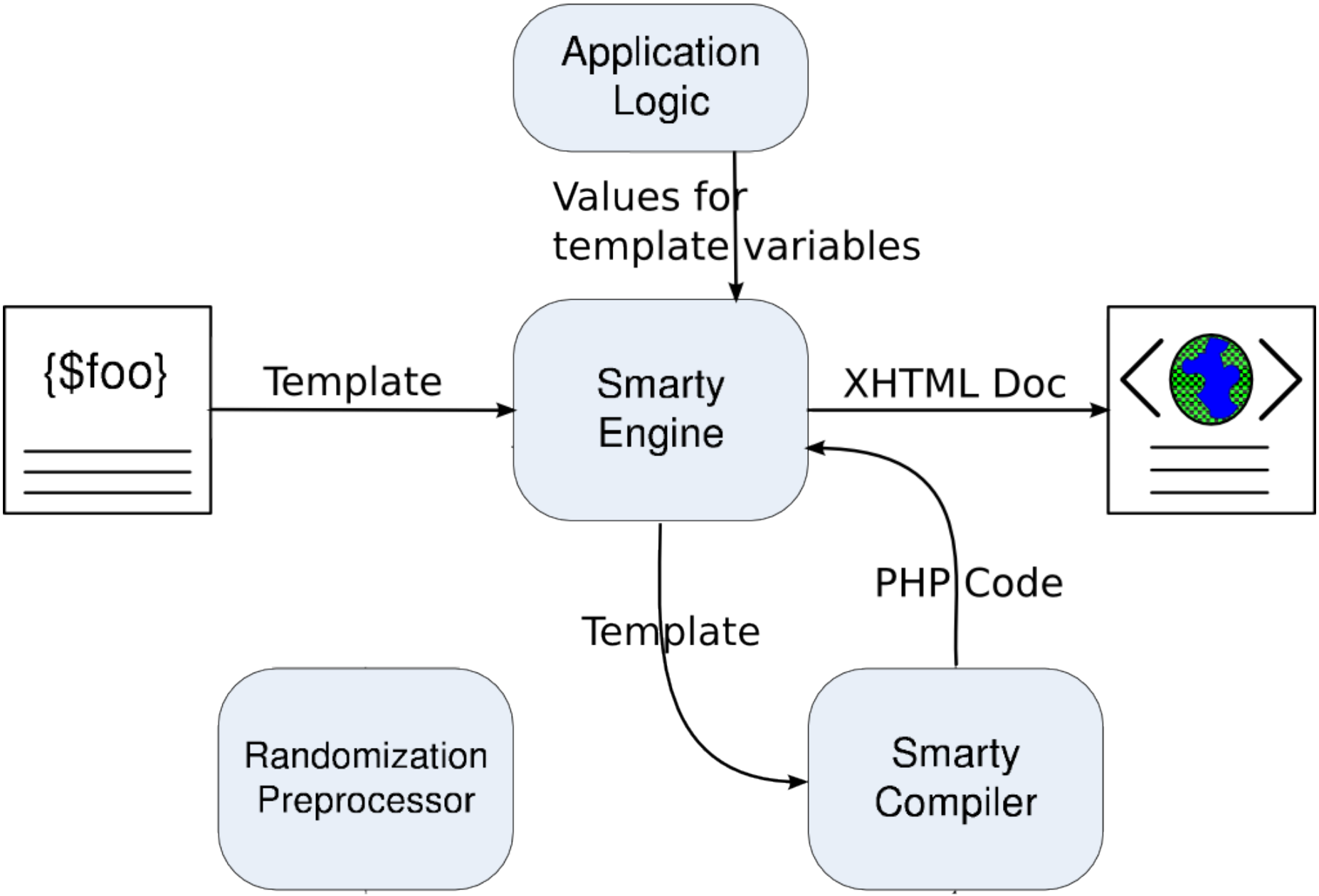
Modifications to Smarty



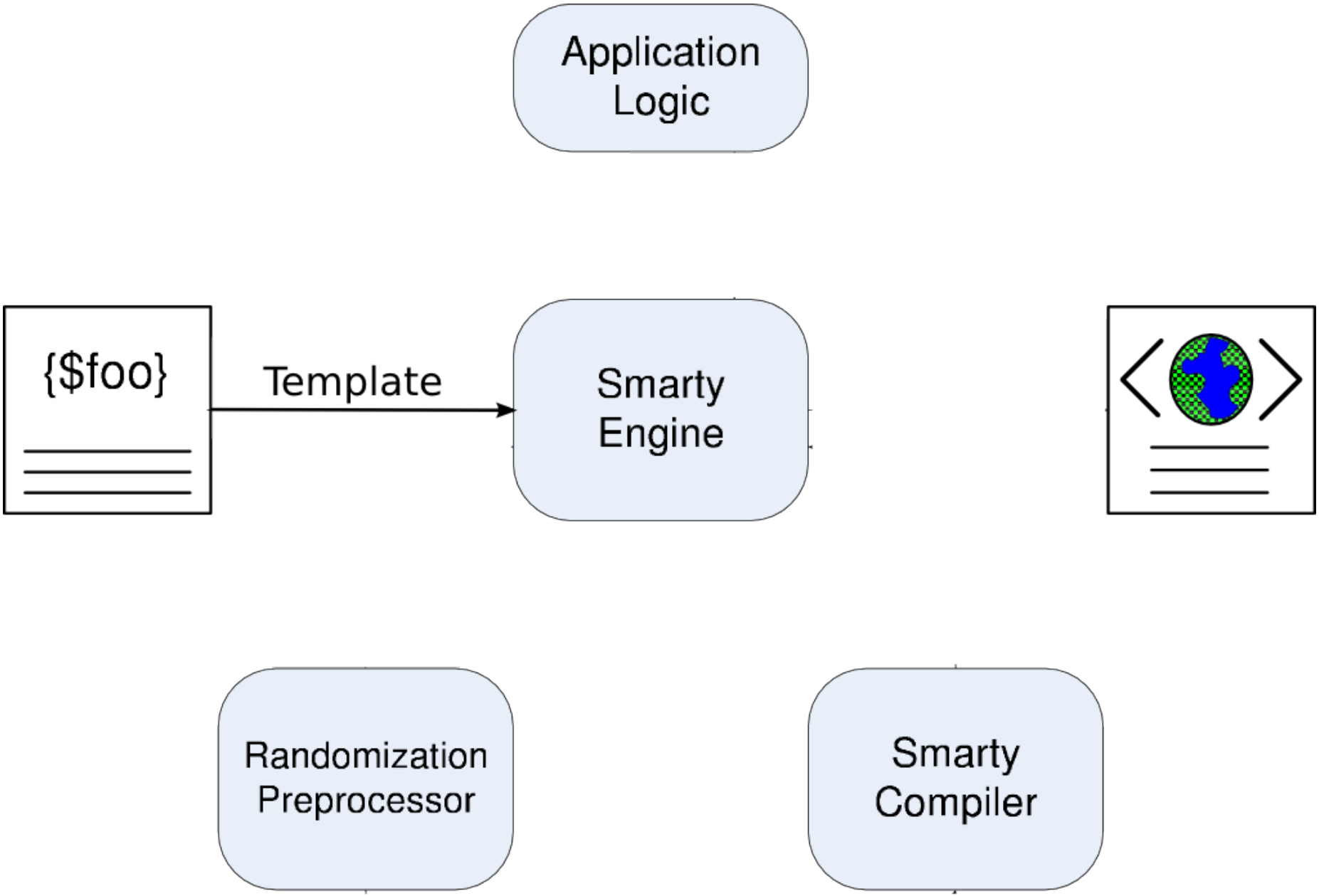
Modifications to Smarty



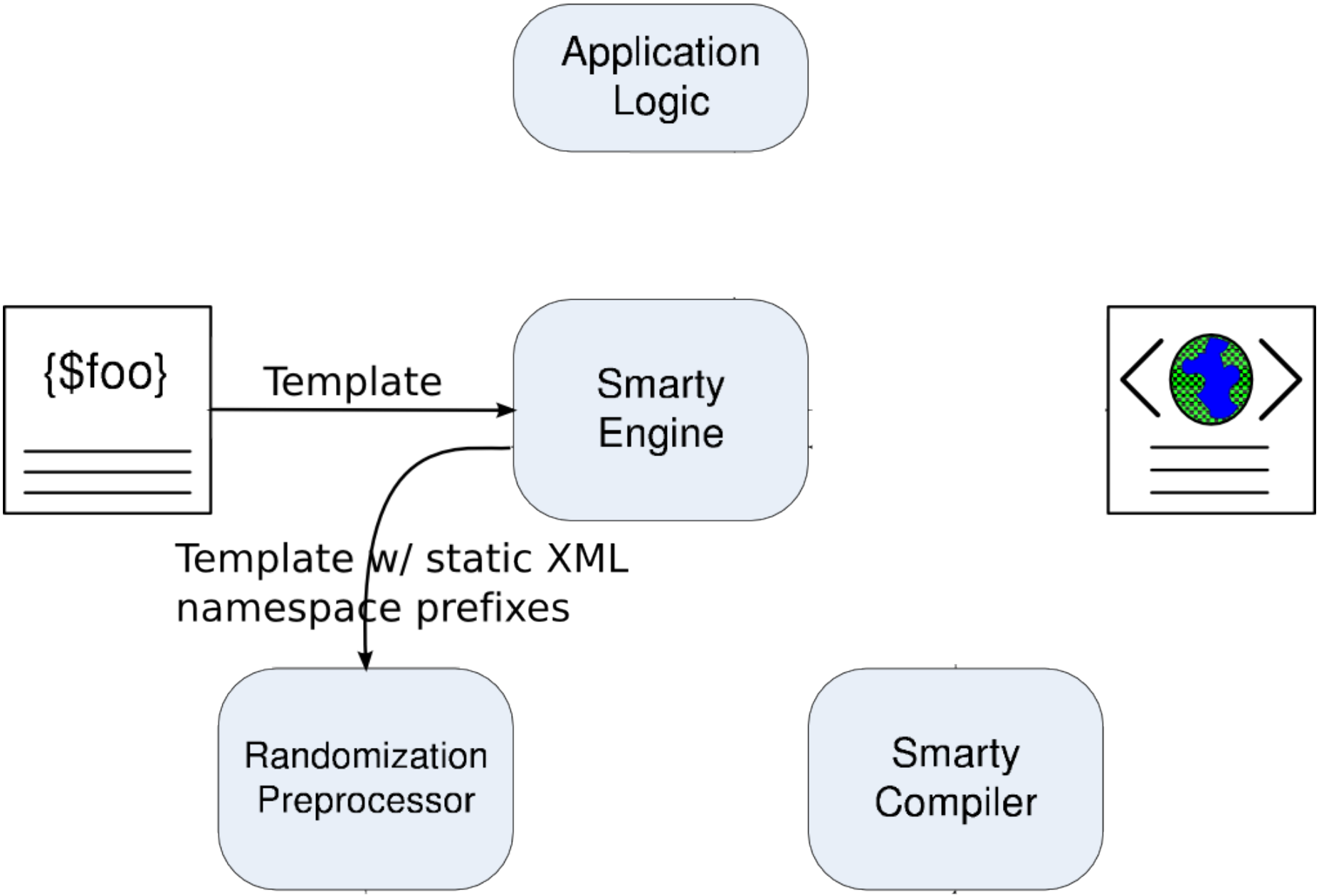
Modifications to Smarty



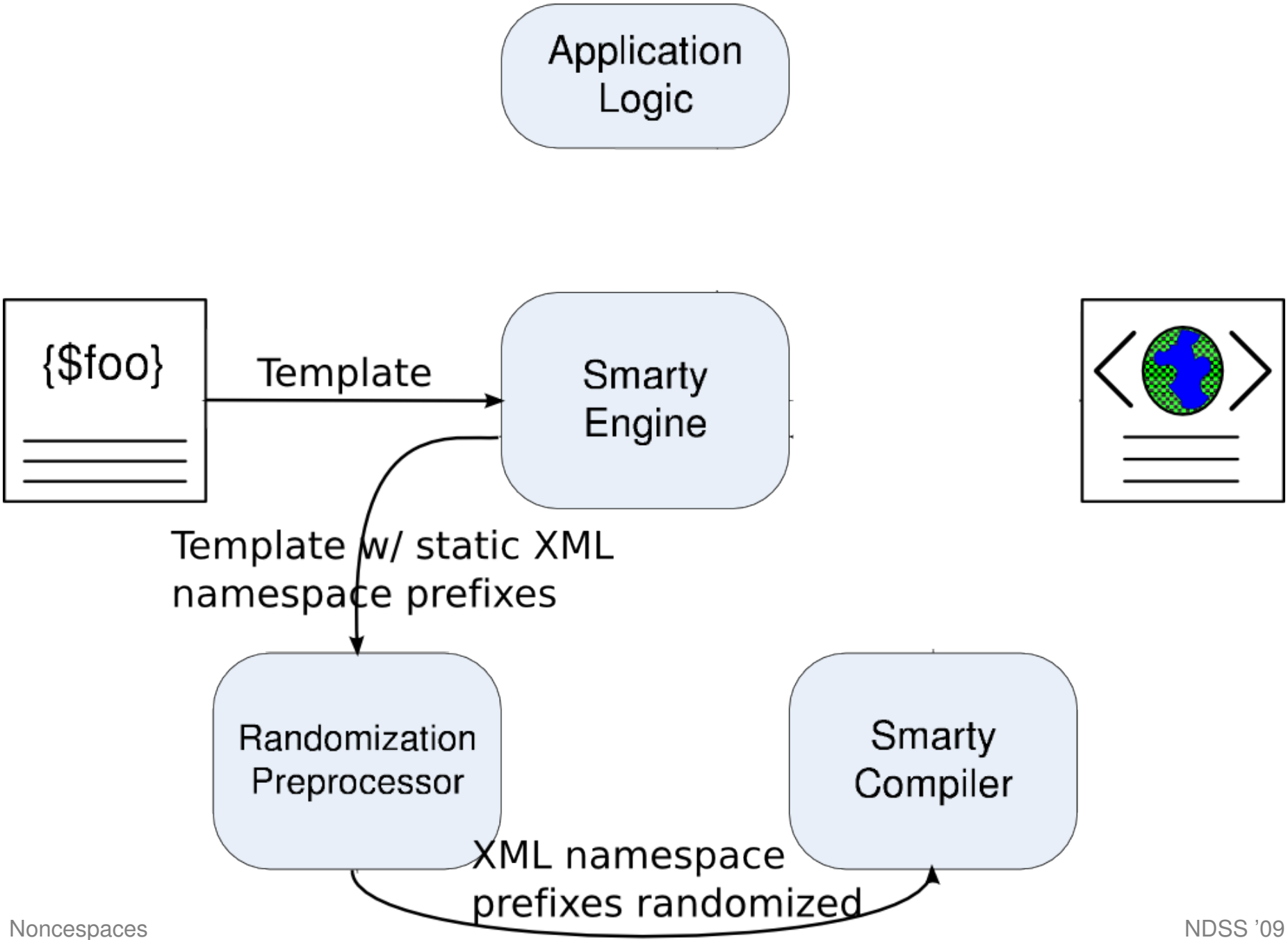
Modifications to Smarty



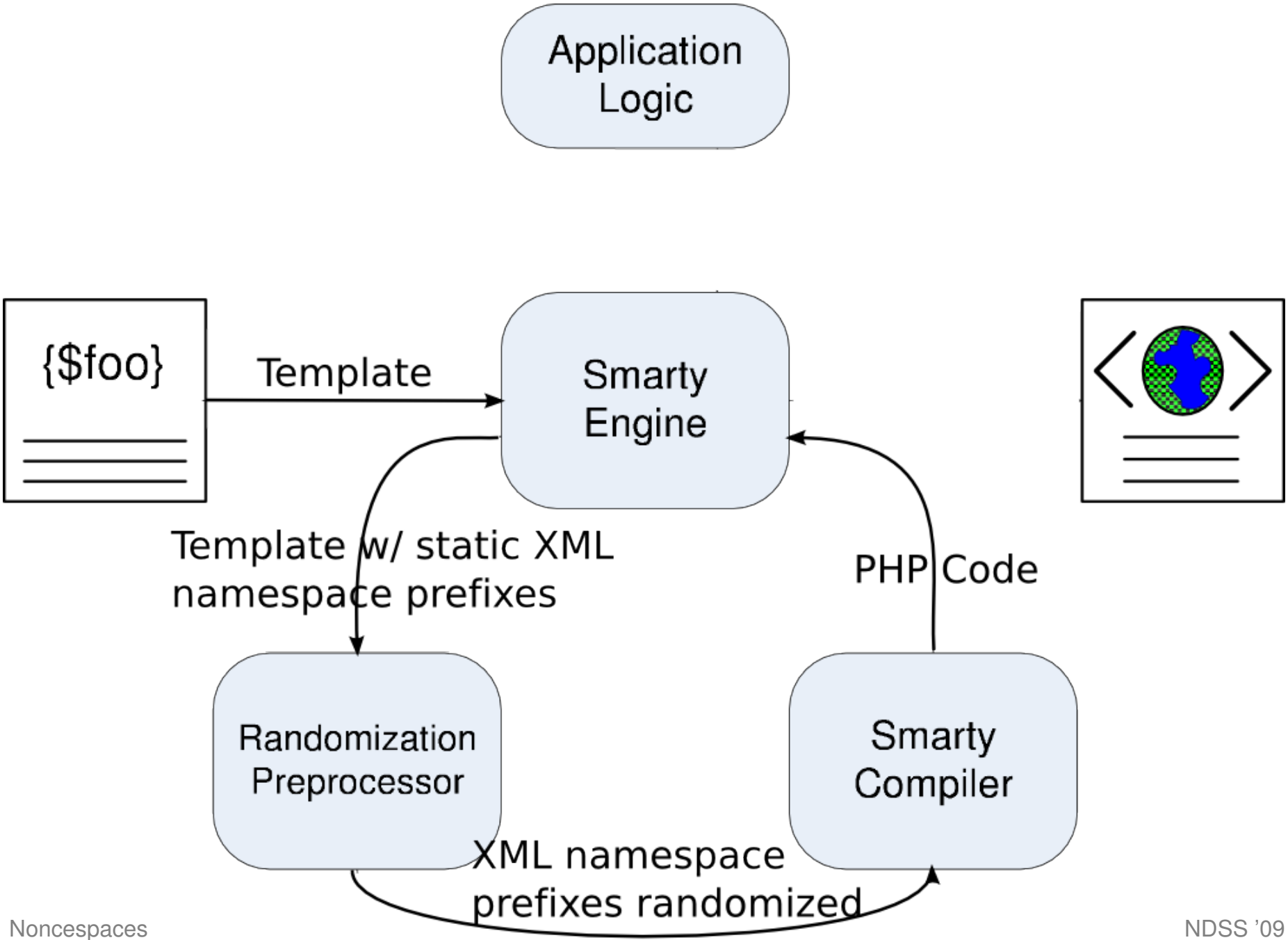
Modifications to Smarty



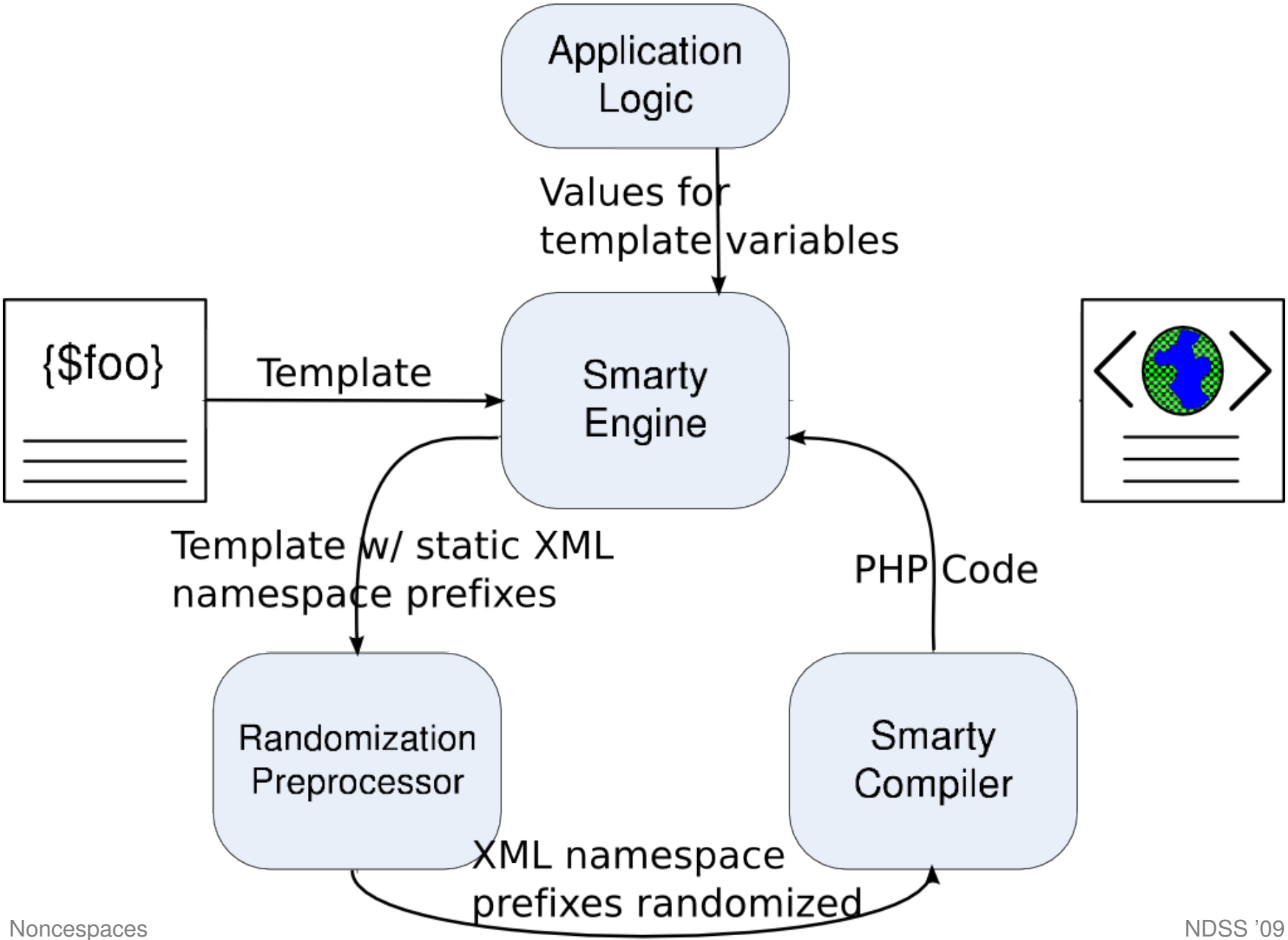
Modifications to Smarty



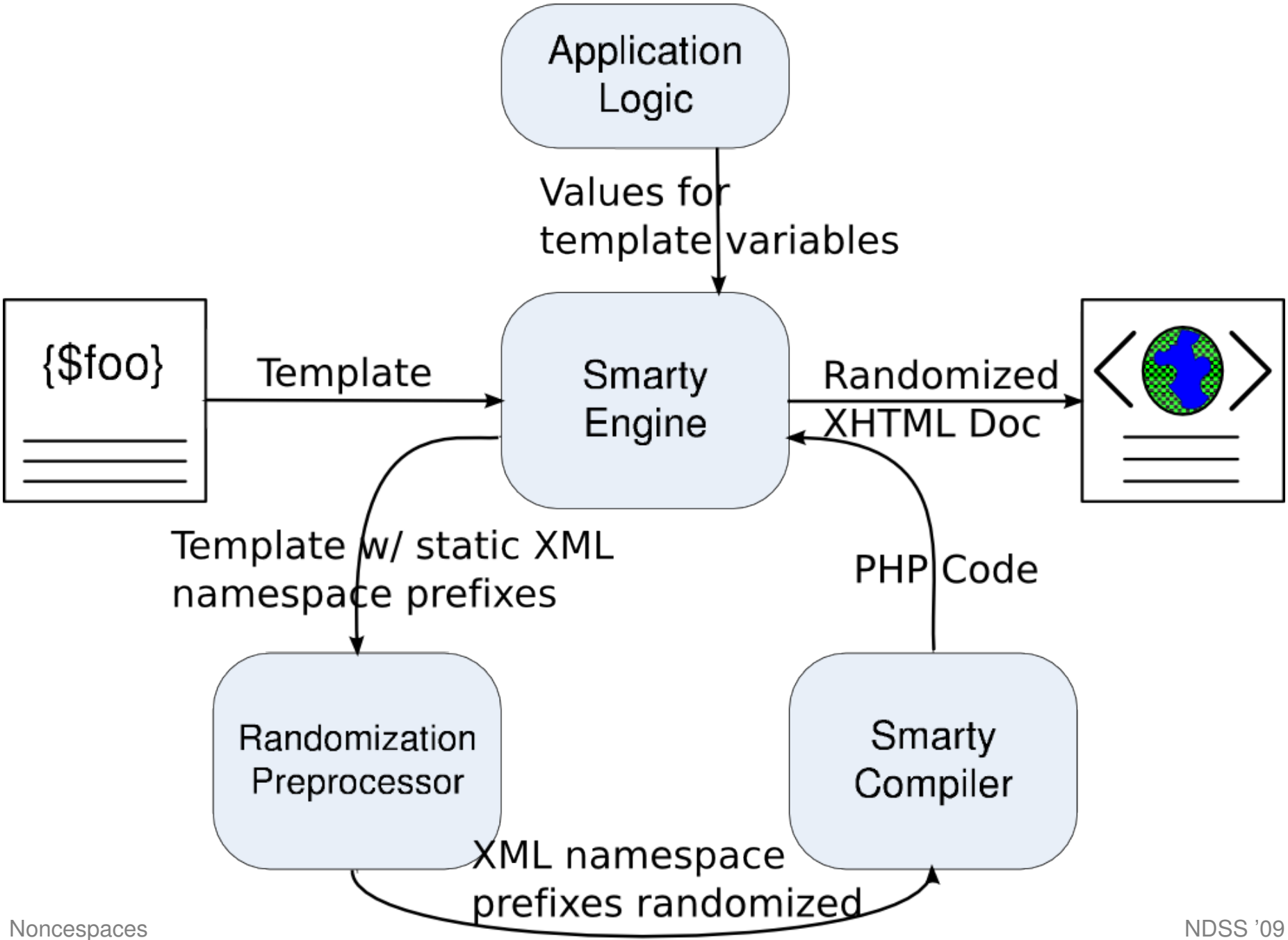
Modifications to Smarty



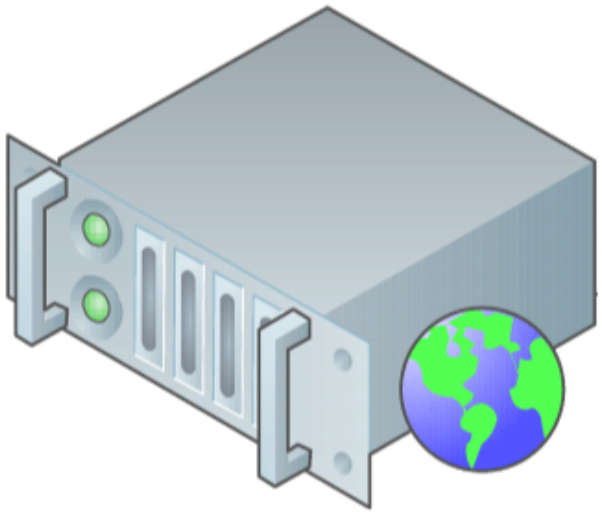
Modifications to Smarty

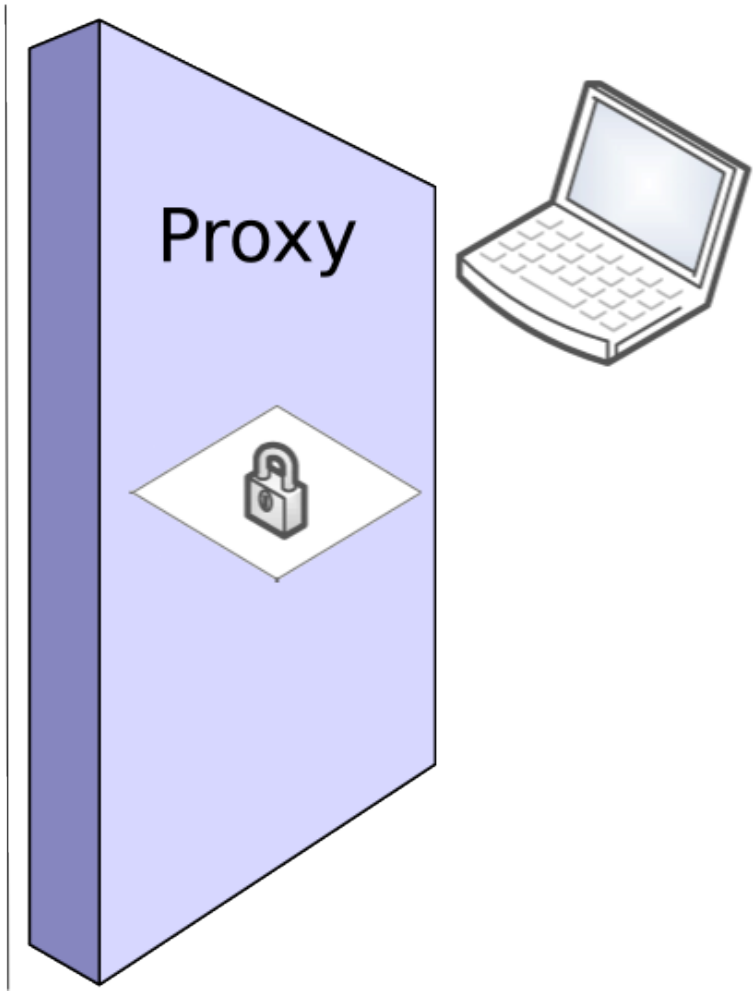


Modifications to Smarty

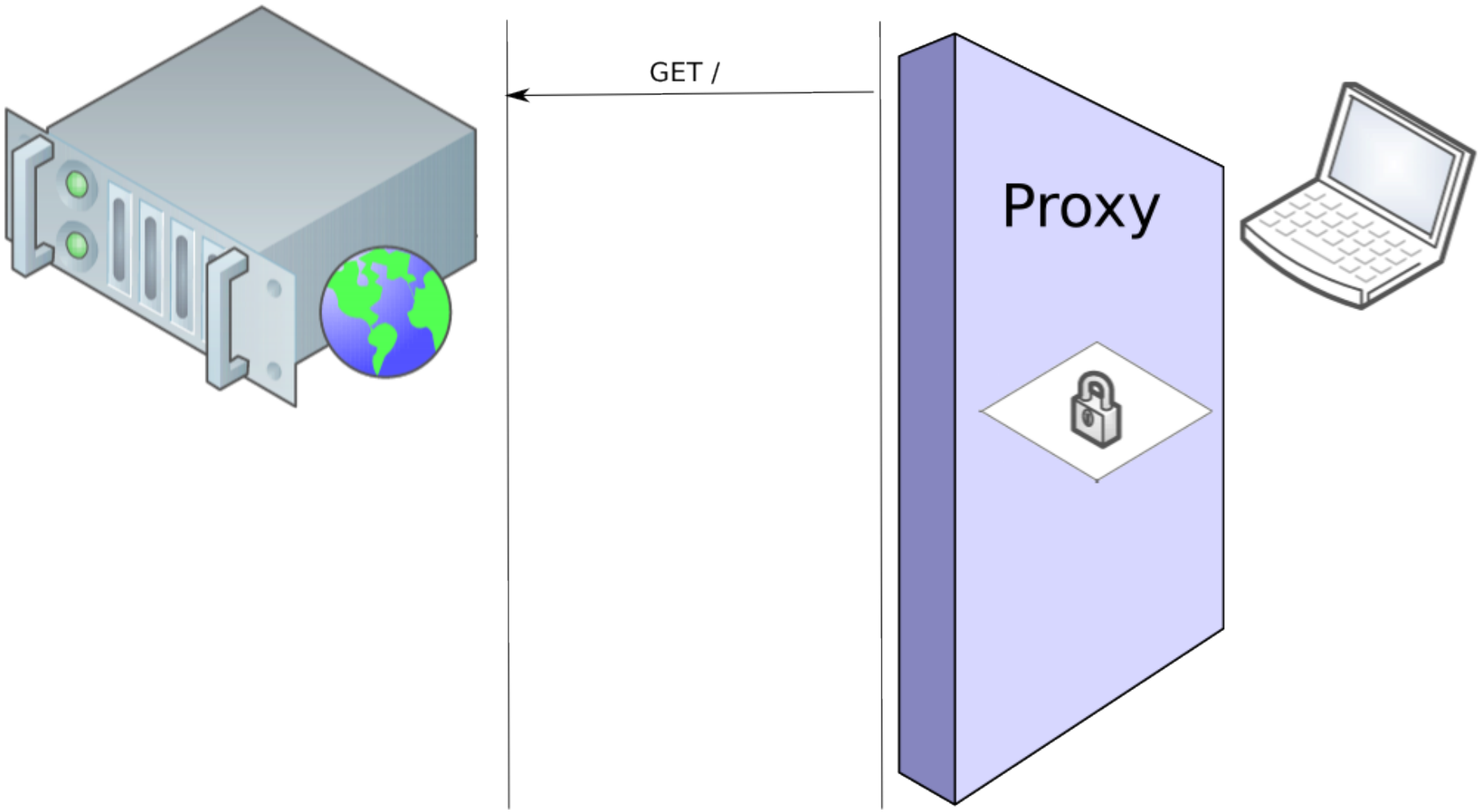


Client-side Modifications

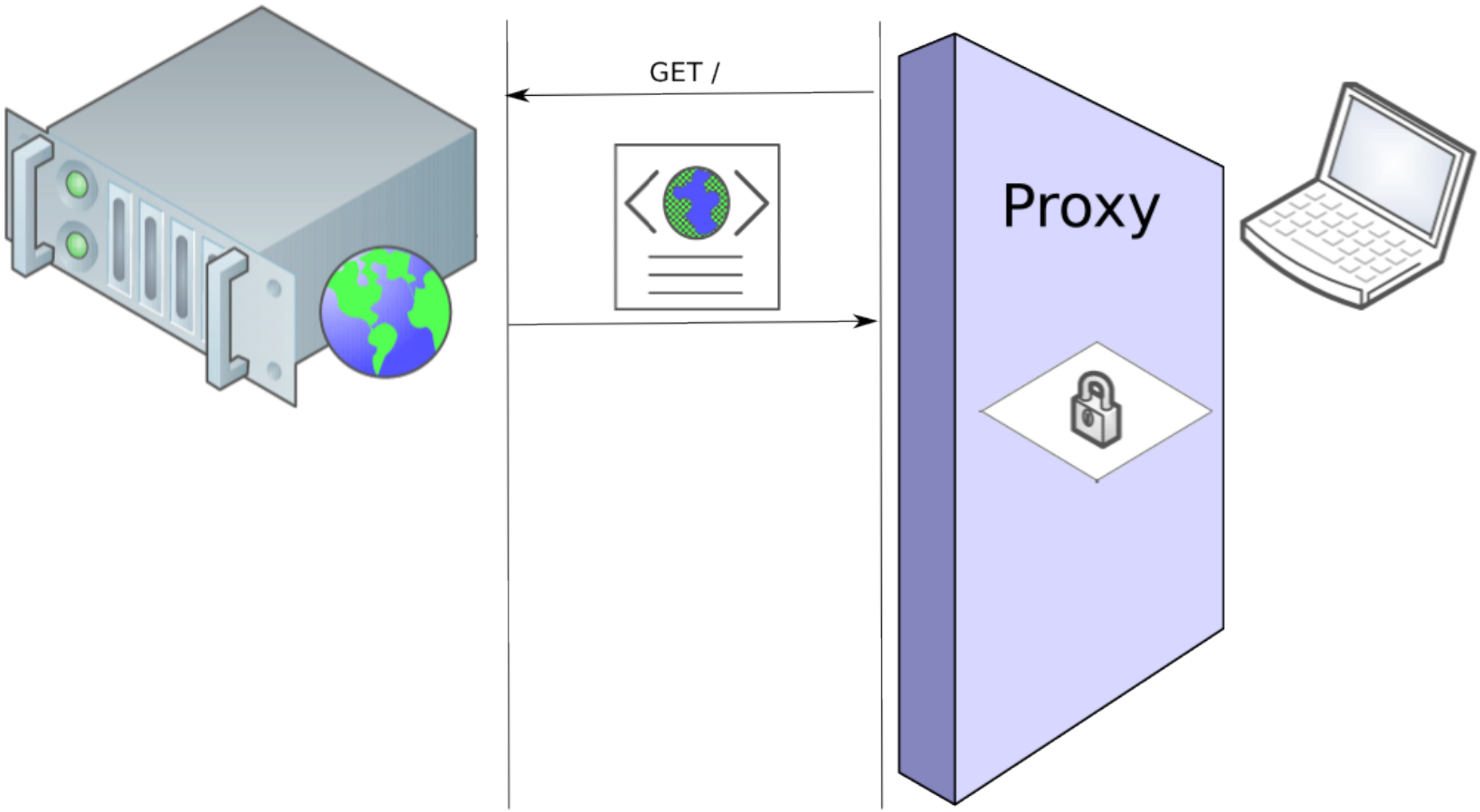




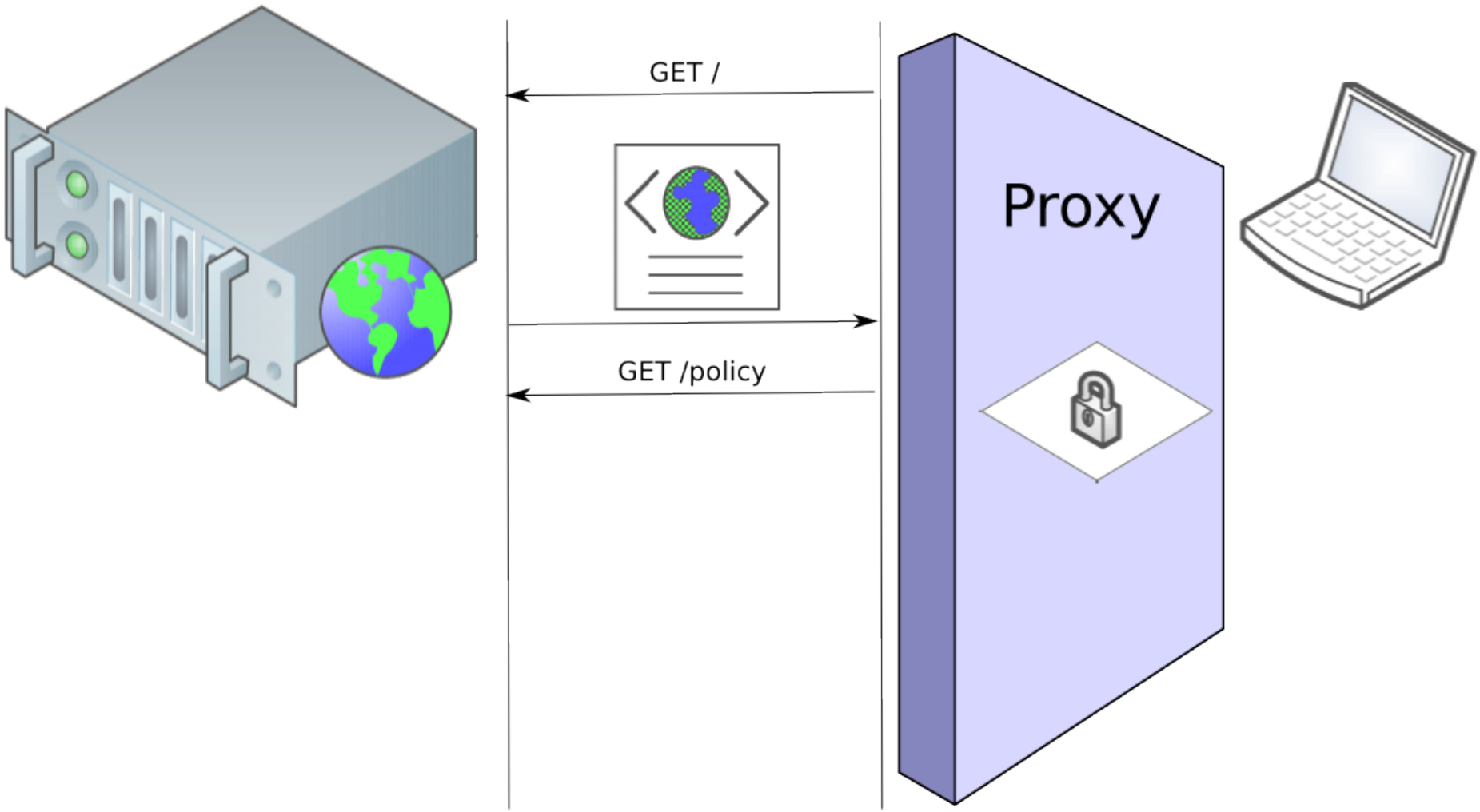
Client-side Modifications



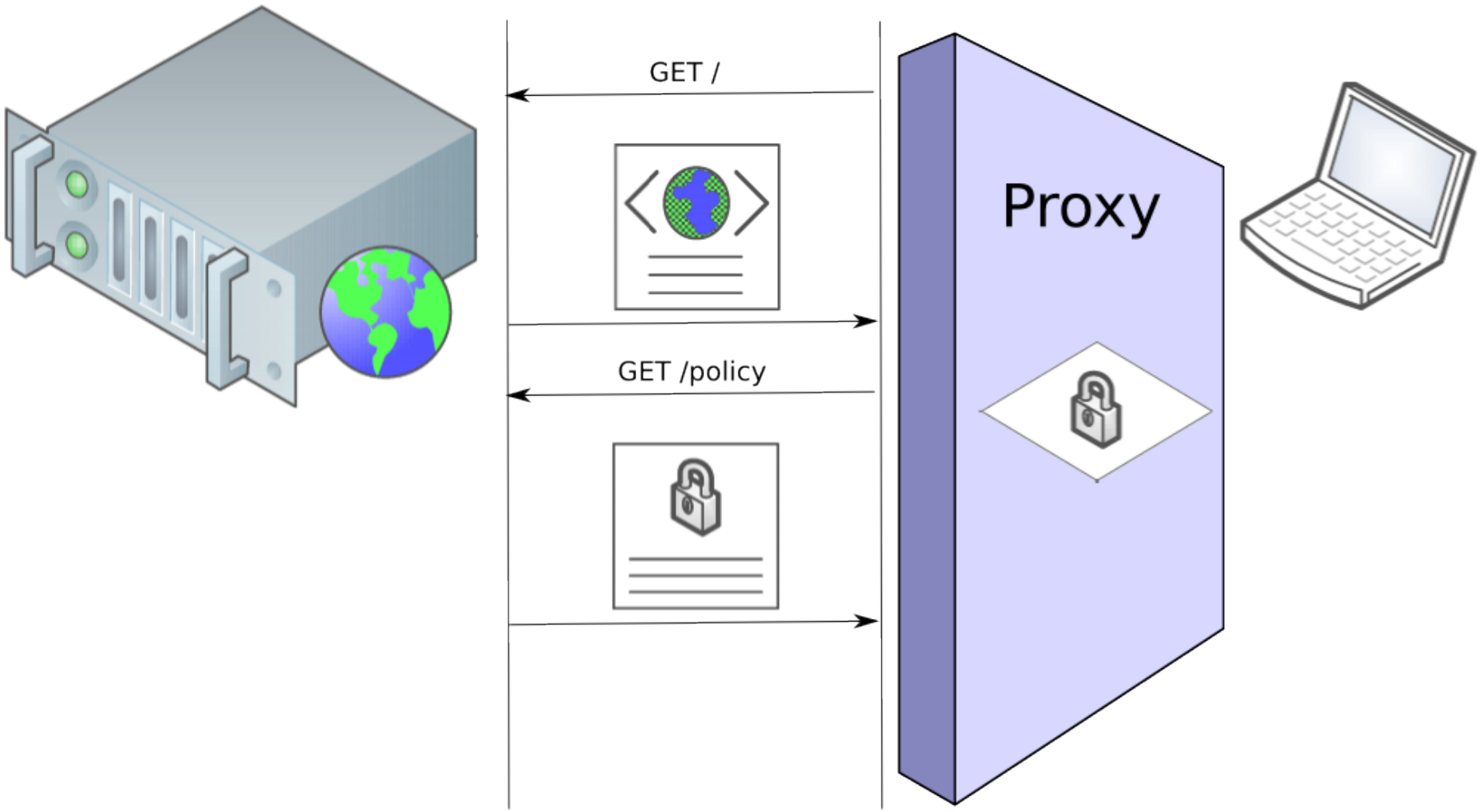
Client-side Modifications



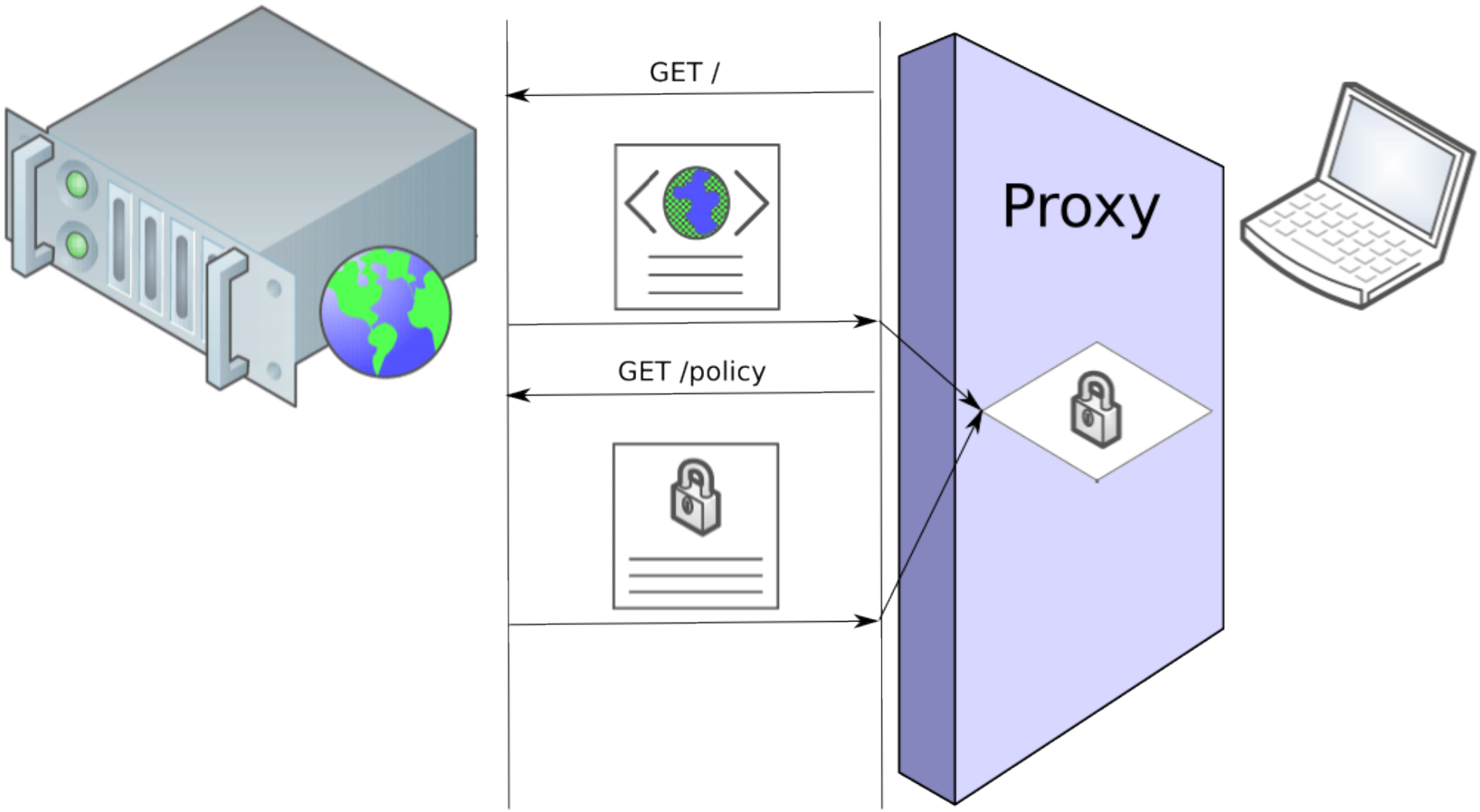
Client-side Modifications



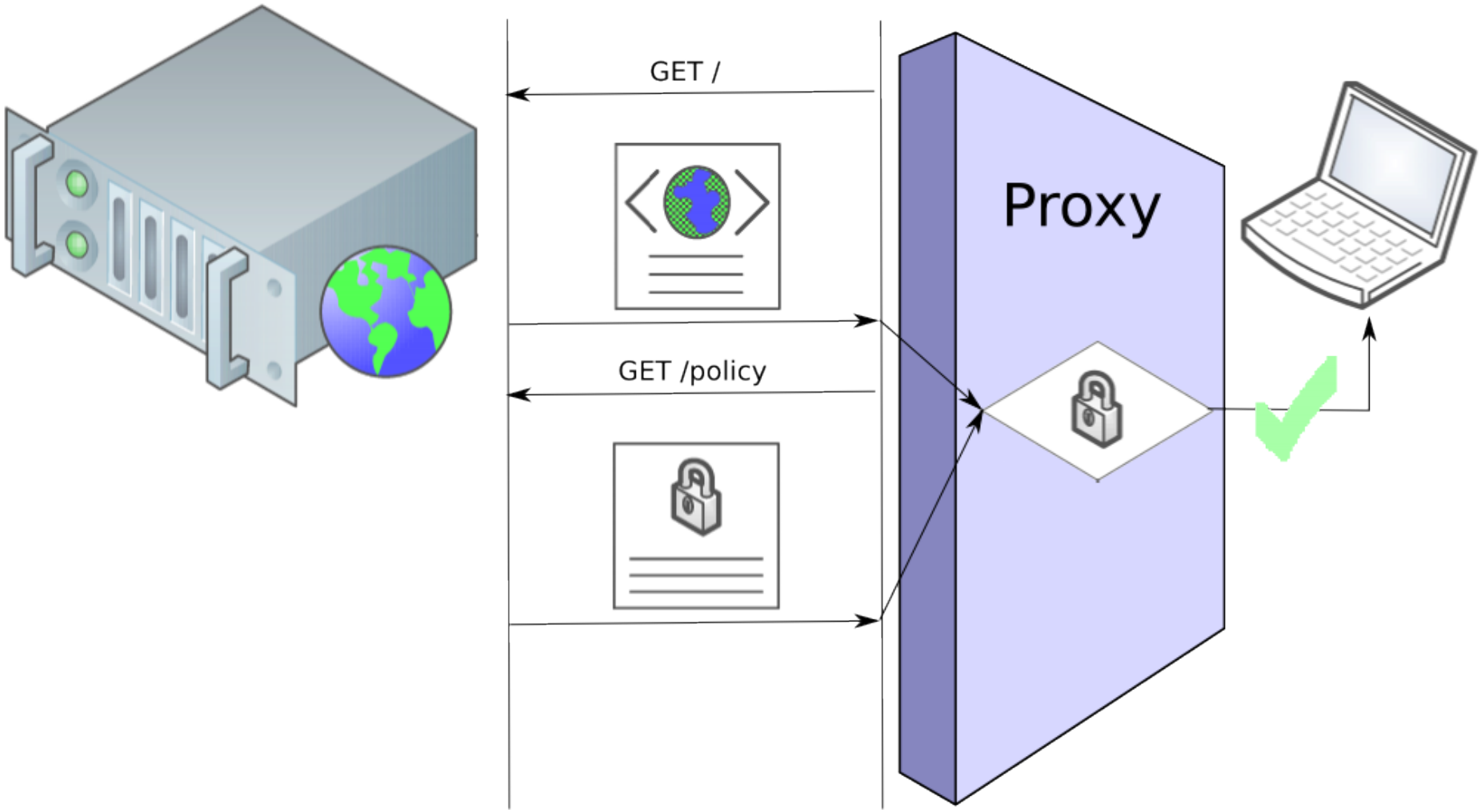
Client-side Modifications



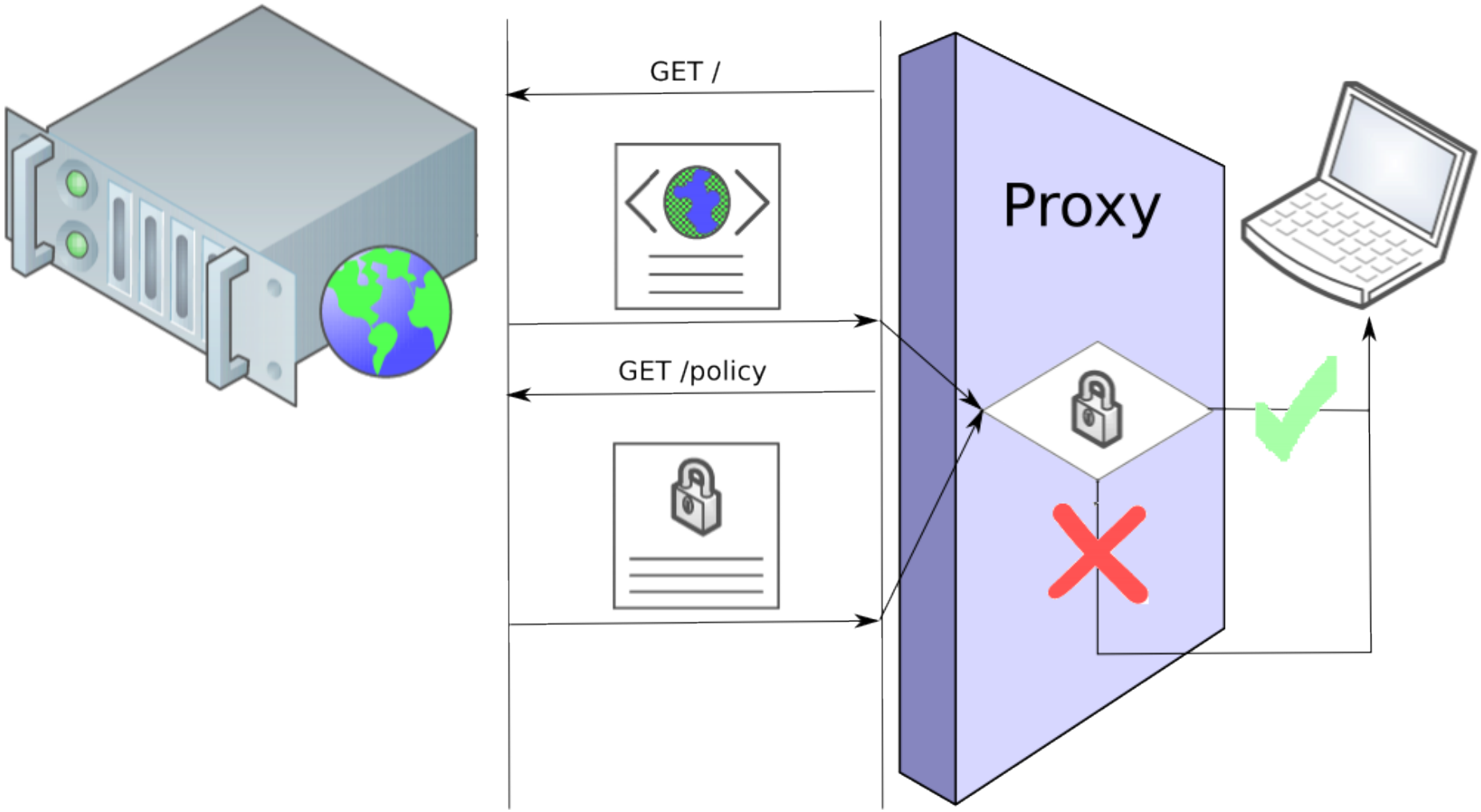
Client-side Modifications



Client-side Modifications



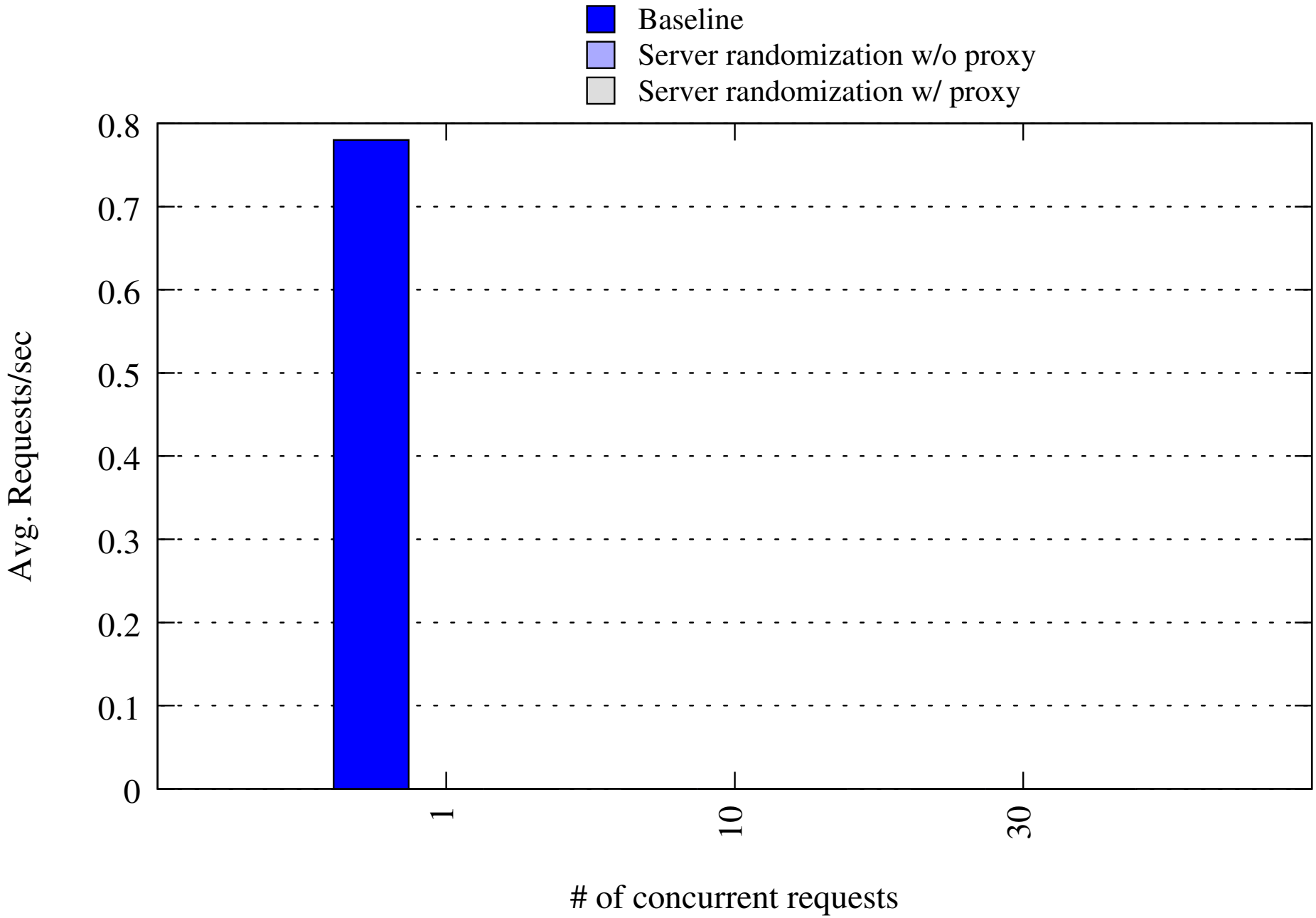
Client-side Modifications



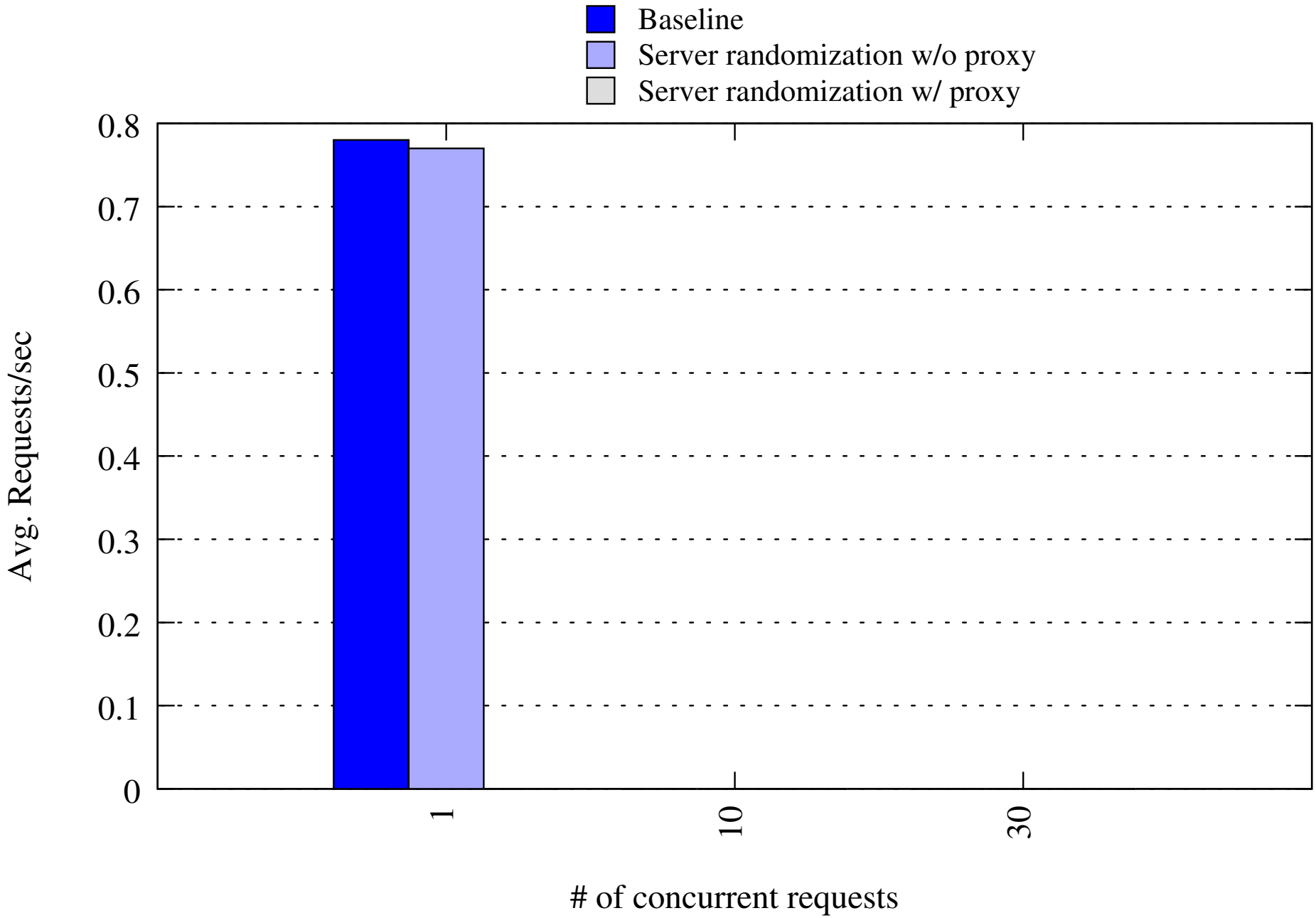
Evaluation

- Tested effectiveness of Noncespaces on 2 applications
- Developed policy for each application
- Ensured that Noncespaces stopped a number of XSS attacks
- Measured performance overhead of both server-side randomization and client-side policy checking

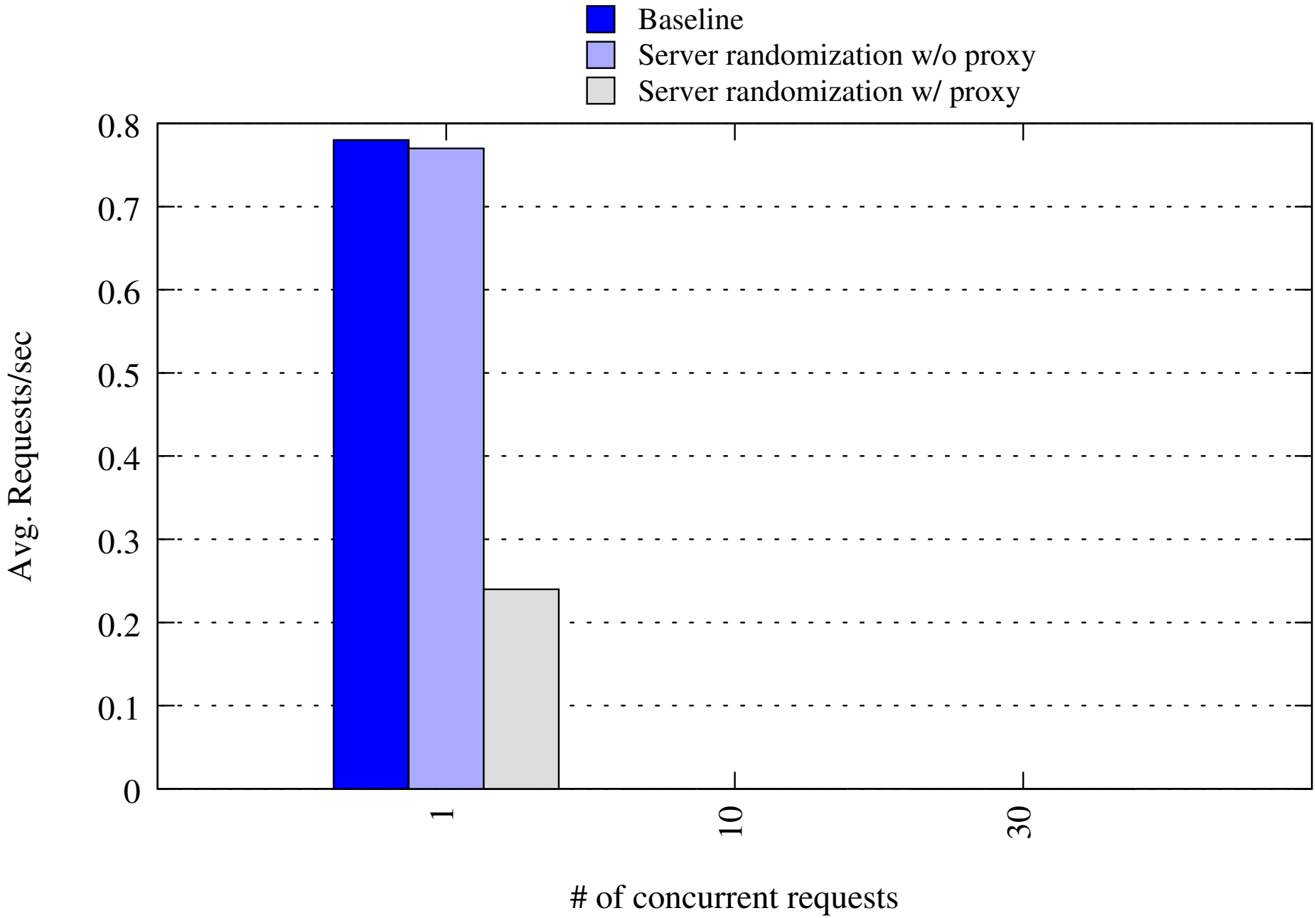
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

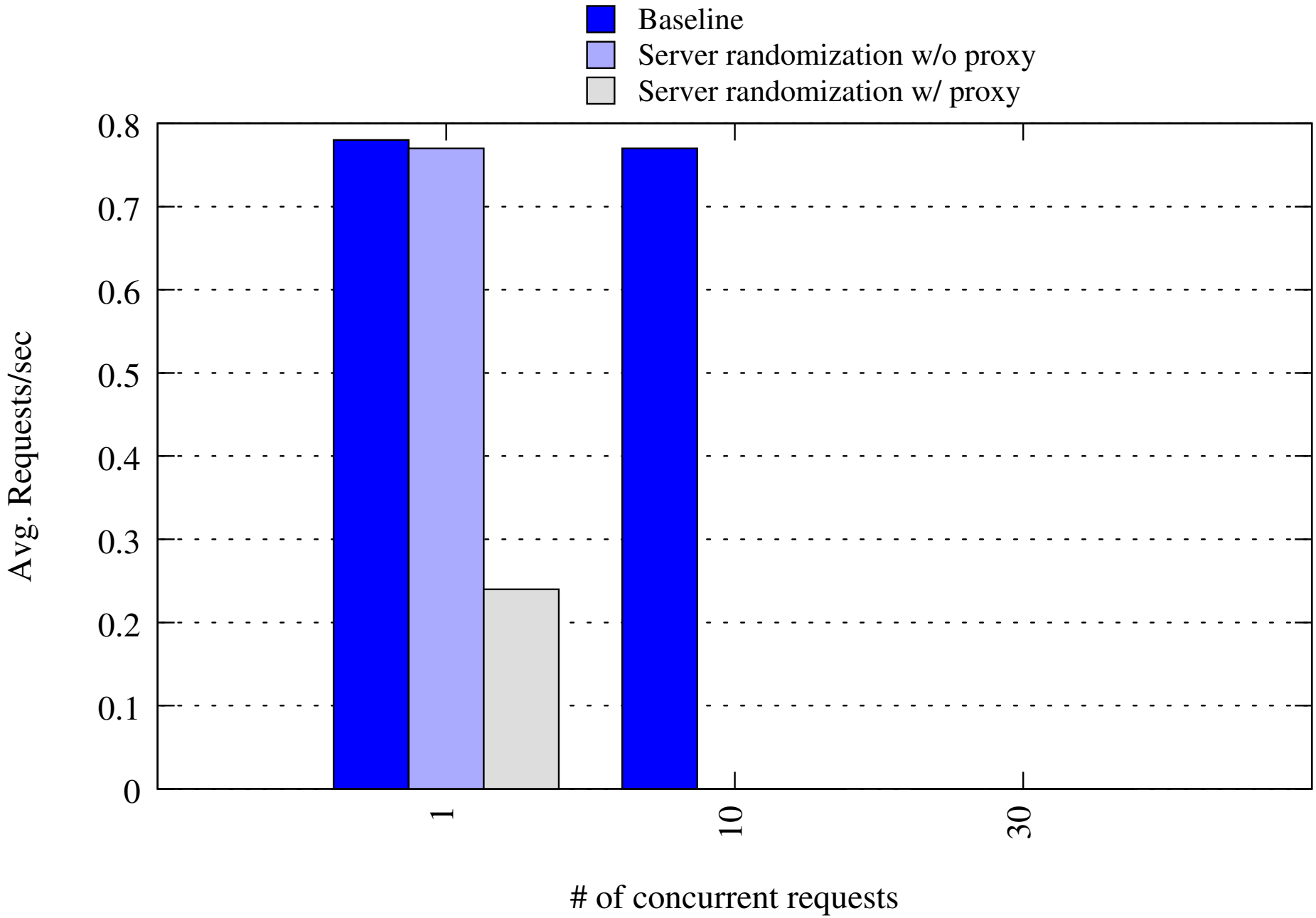
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

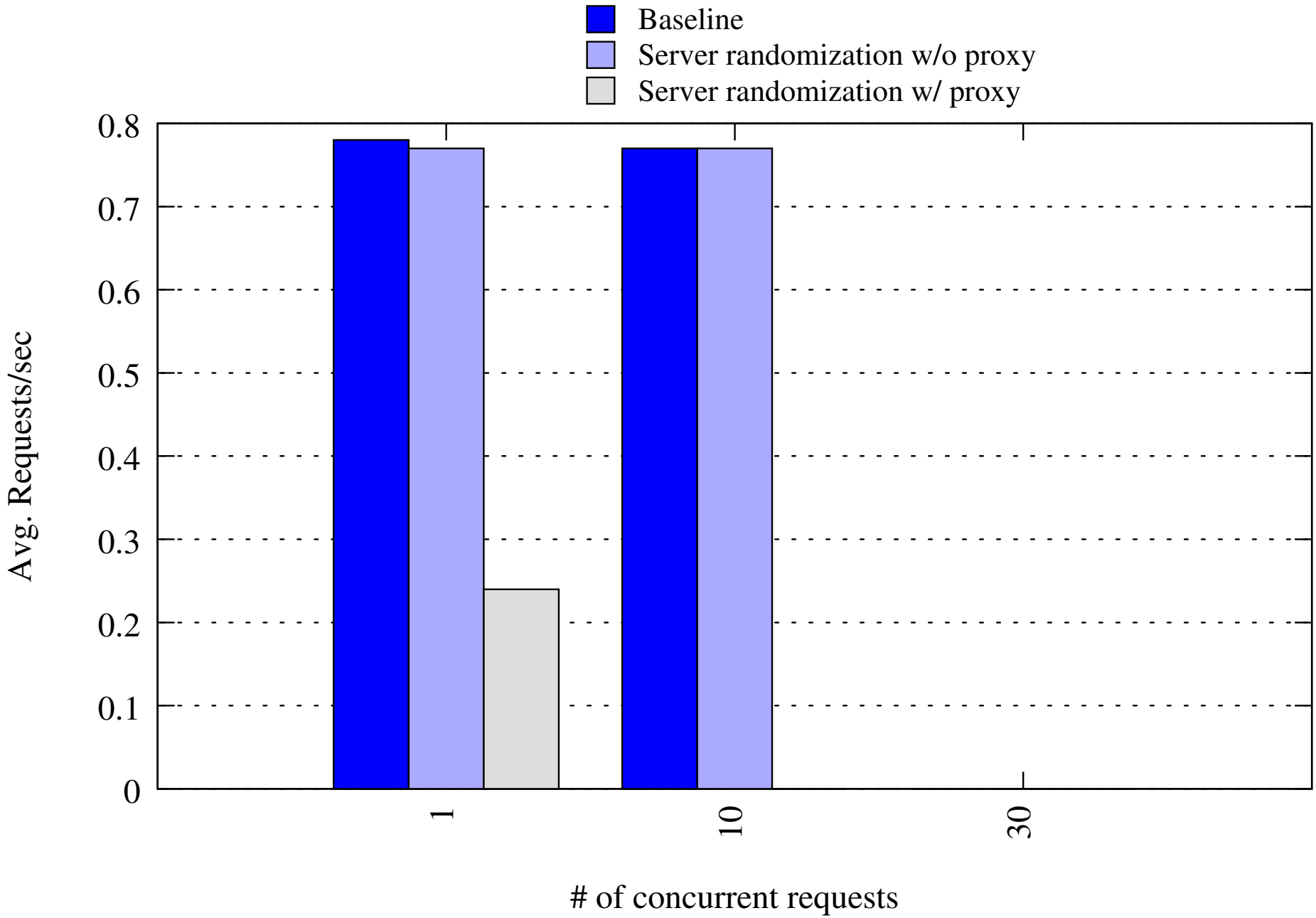
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

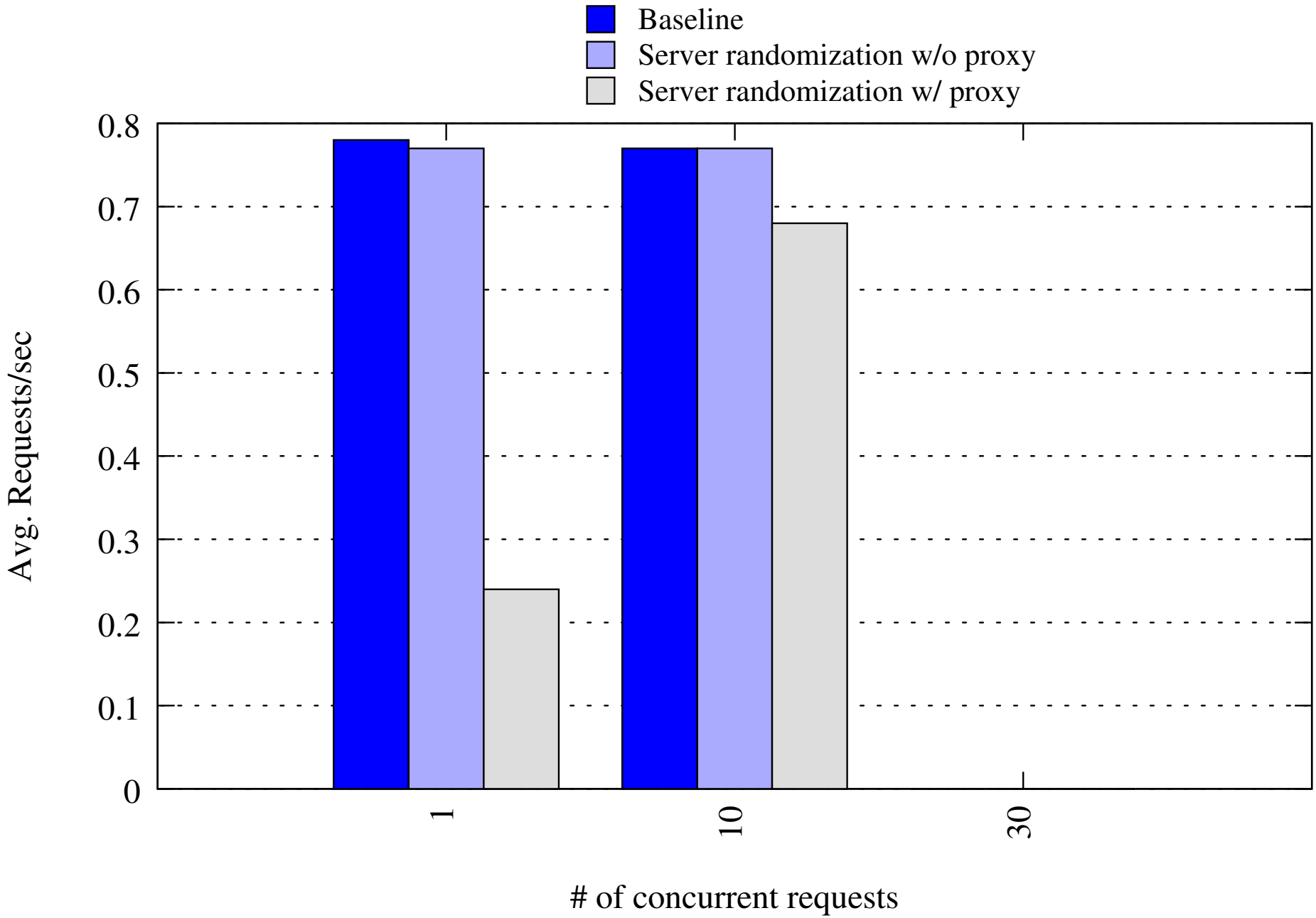
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

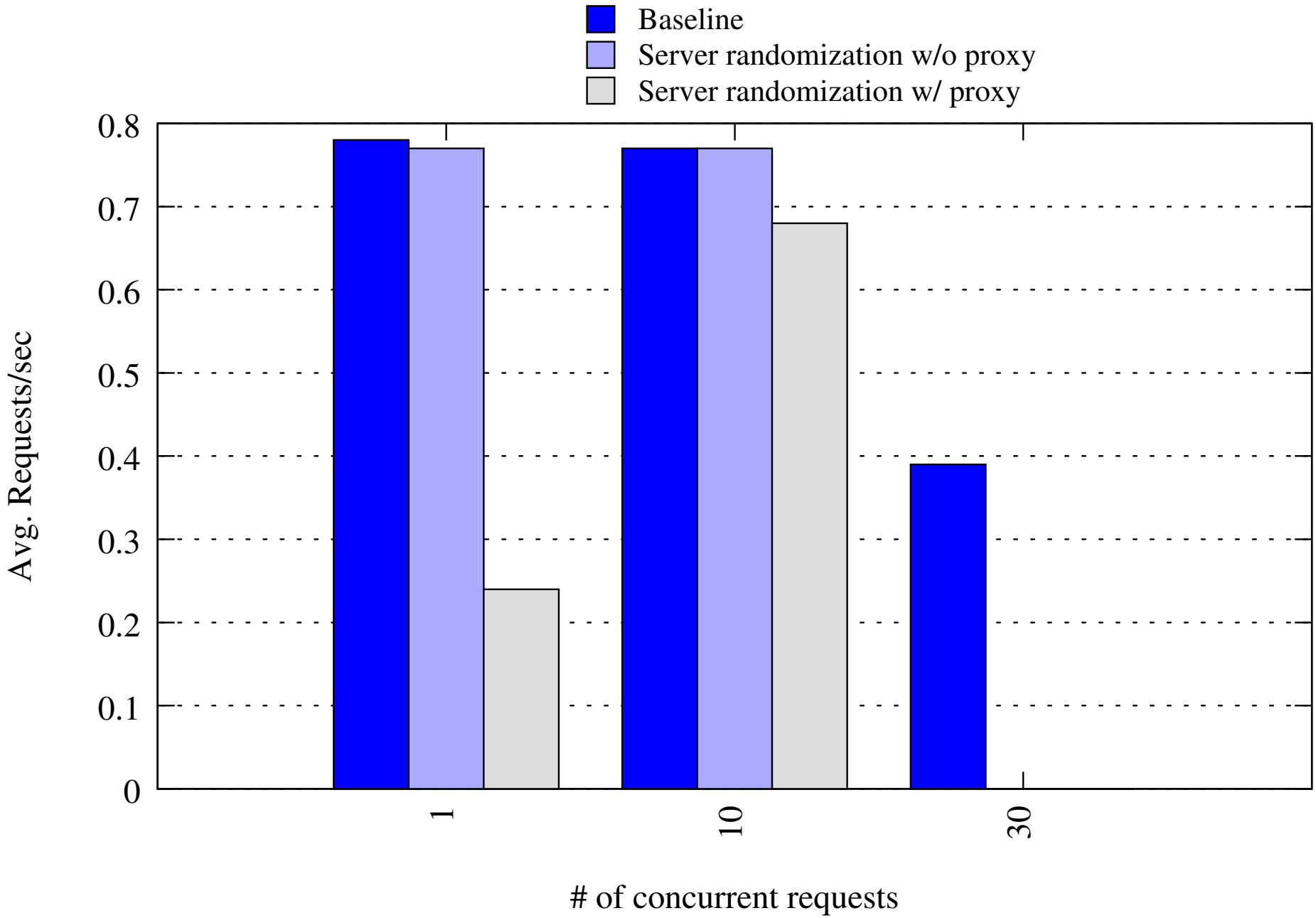
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

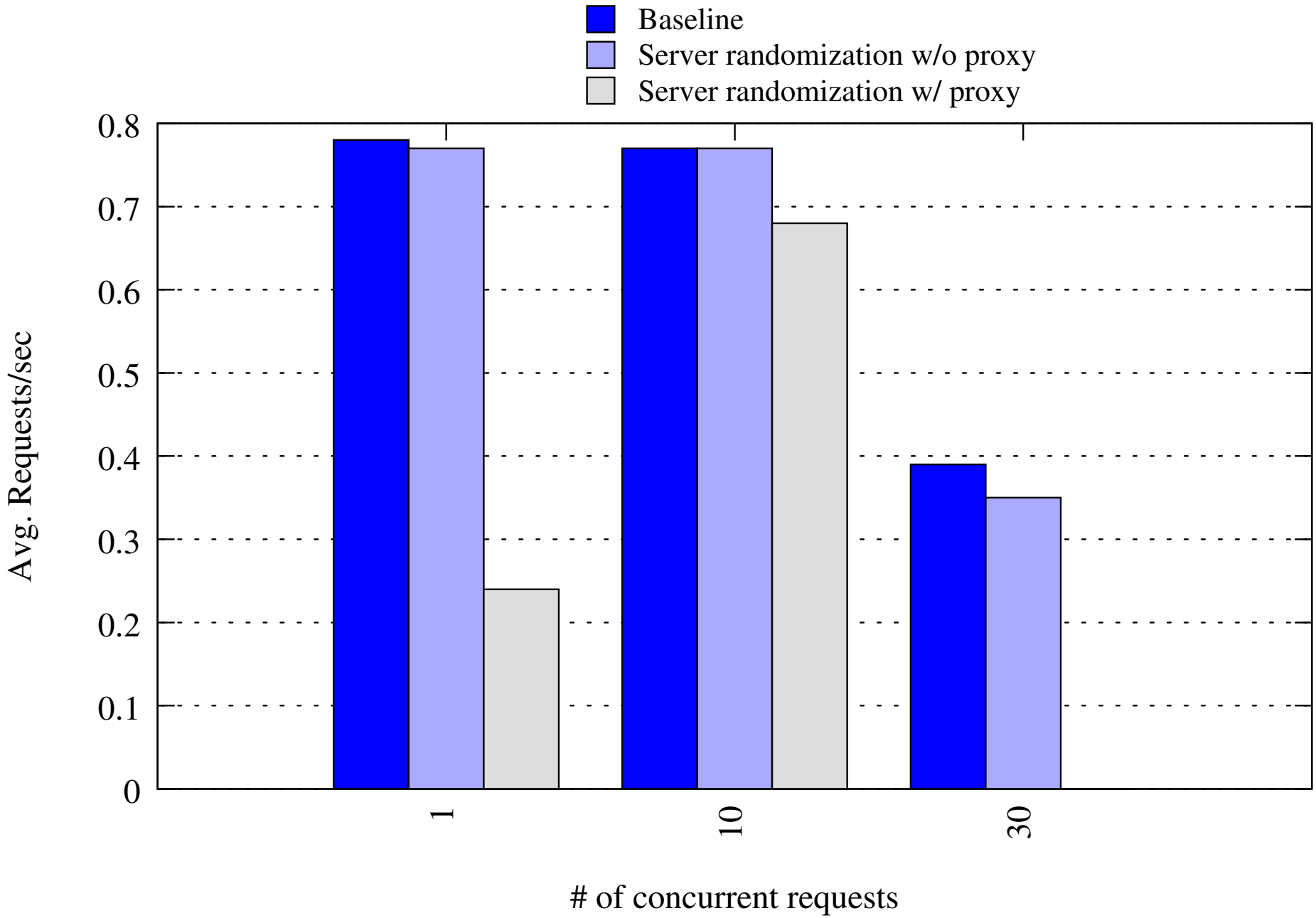
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

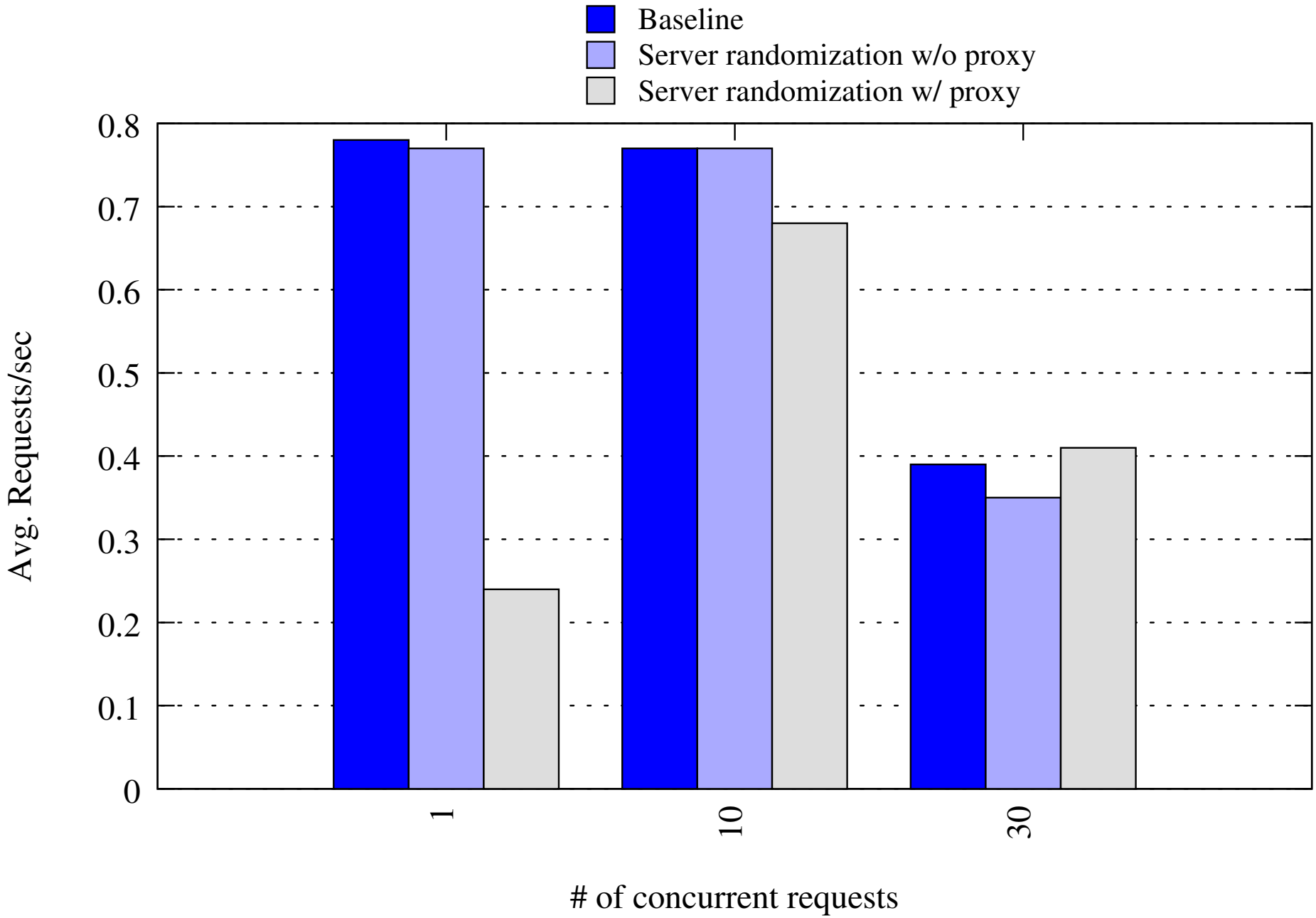
Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

Evaluation


Baseline
Server randomization w/o proxy
Server randomization w/ proxy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Avg. Requests/sec
1
10
30
# of concurrent requests

Related Work

- Instruction Set Randomization (Kc et al., CCS '03) and (Barrantes et al., CCS '03)
- BEEP (Jim et al., WWW '07)
- Mutation Event Transforms (Erlingsson et al., HotOS '07)
- Noxes (Kirda et al., ACM SAC '06)
- Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis (Vogt et al., NDSS '07)

Conclusion

- We can achieve security without data sanitization on the server
  - Servers classify how trustworthy content is
  - Servers convey trust classifications in a tamper resistant way
  - Clients interpreting the content enforce the policy
- Leverage randomization and XML features to thwart XSS attacks
- Leverage design paradigms to determine trust information without dynamic information flow tracking

Questions?

Example Noncespaces Policy

```
1  namespace trusted
2  namespace untrusted
3
4  allow //trusted:*
5  allow //trusted:@*
6
7  allow //untrusted:b
8  allow //untrusted:i
9  allow //untrusted:u
10 allow //untrusted:a
11 allow //untrusted:a/@untrusted:href[
12     starts-with(normalize-space(.), "http:")]
13 allow //untrusted:img
14 allow //untrusted:img/@untrusted:src[
15     starts-with(normalize-space(.), "http:")]
16
17 deny //*
18 deny //@*
```