ORACLE®

Safe Harbor Statement

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

ORACLE

Static Taint Analysis for Web Applications

Haven't we solved this problem yet?

Francois Gauthier
Oracle Labs Australia
20th November 2015

Introduction and Motivation

Why are we doing this?

Why Static Taint Analysis?

Let’s take a look at the most common flaws in web applications…

Ranking	Defect
1	Injection (SQL, OS, LDAP)
2	Broken authentication and session management
3	Cross-site scripting (XSS)
4	Insecure direct object references
5	Security misconfiguration
6	Sensitive data exposure
7	Missing function level access control
8	Cross-site request forgery (CSRF)
9	Using components with known vulnerabilities
10	Unvalidated redirects and forwards

Why Static Taint Analysis?

Most of them can be detected with taint analysis!

Ranking	Defect
1	**Injection (SQL, OS, LDAP)**
2	Broken authentication and session management
3	**Cross-site scripting (XSS)**
4	**Insecure direct object references**
5	Security misconfiguration
6	**Sensitive data exposure**
7	**Missing function level access control**
8	**Cross-site request forgery (CSRF)**
9	Using components with known vulnerabilities
10	**Unvalidated redirects and forwards**

ORACLE

Common Attack Vector

Phishing emails

From: accounts@company.com

Subject: Your Account Has Been Suspended

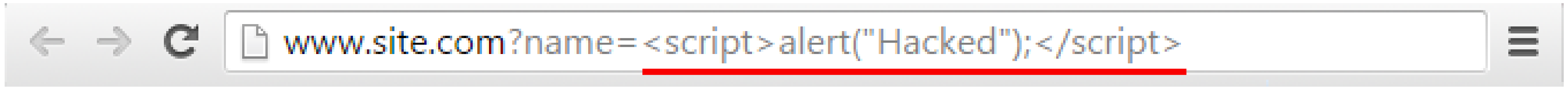
Dear user,

We are sending this email to let you know that your credit card has expired. To update your account information, please visit Your Account.

Best regards,
<Company XYZ>

Understanding XSS Flaws

Detailed example of a reflected cross-site scripting flaw



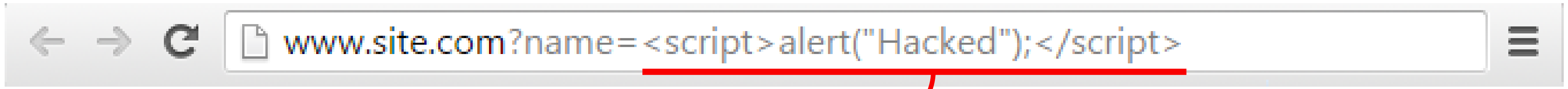
www.site.com?name=<script>alert("Hacked");</script>

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = page.getParameter(request, "xslTitle");
%>
<head>
    <title><%= xslTitle %></title>
```

Understanding XSS Flaws

Detailed example of a reflected cross-site scripting flaw



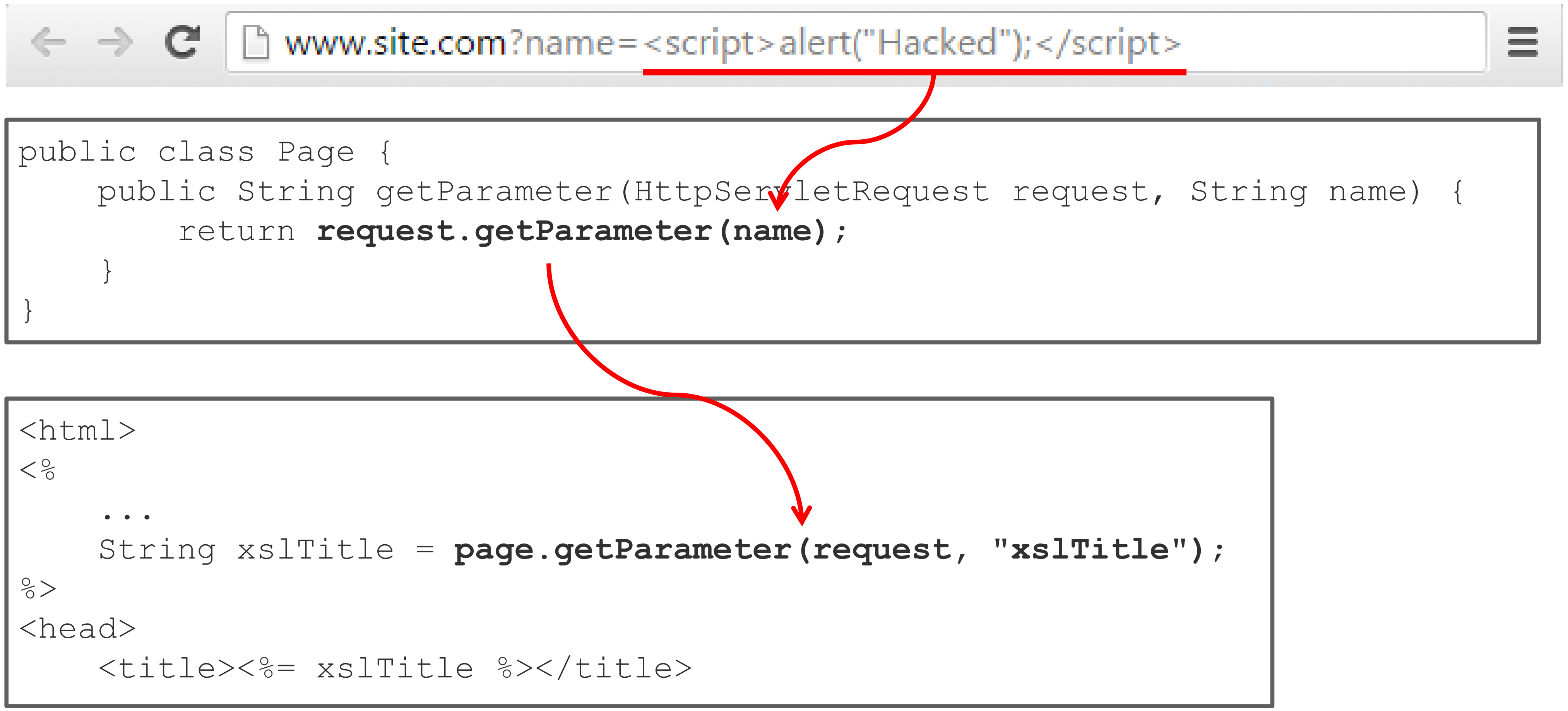
```
www.site.com?name=<script>alert("Hacked");</script>
```

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = page.getParameter(request, "xslTitle");
%>
<head>
    <title><%= xslTitle %></title>
```

Understanding XSS Flaws

Detailed example of a reflected cross-site scripting flaw



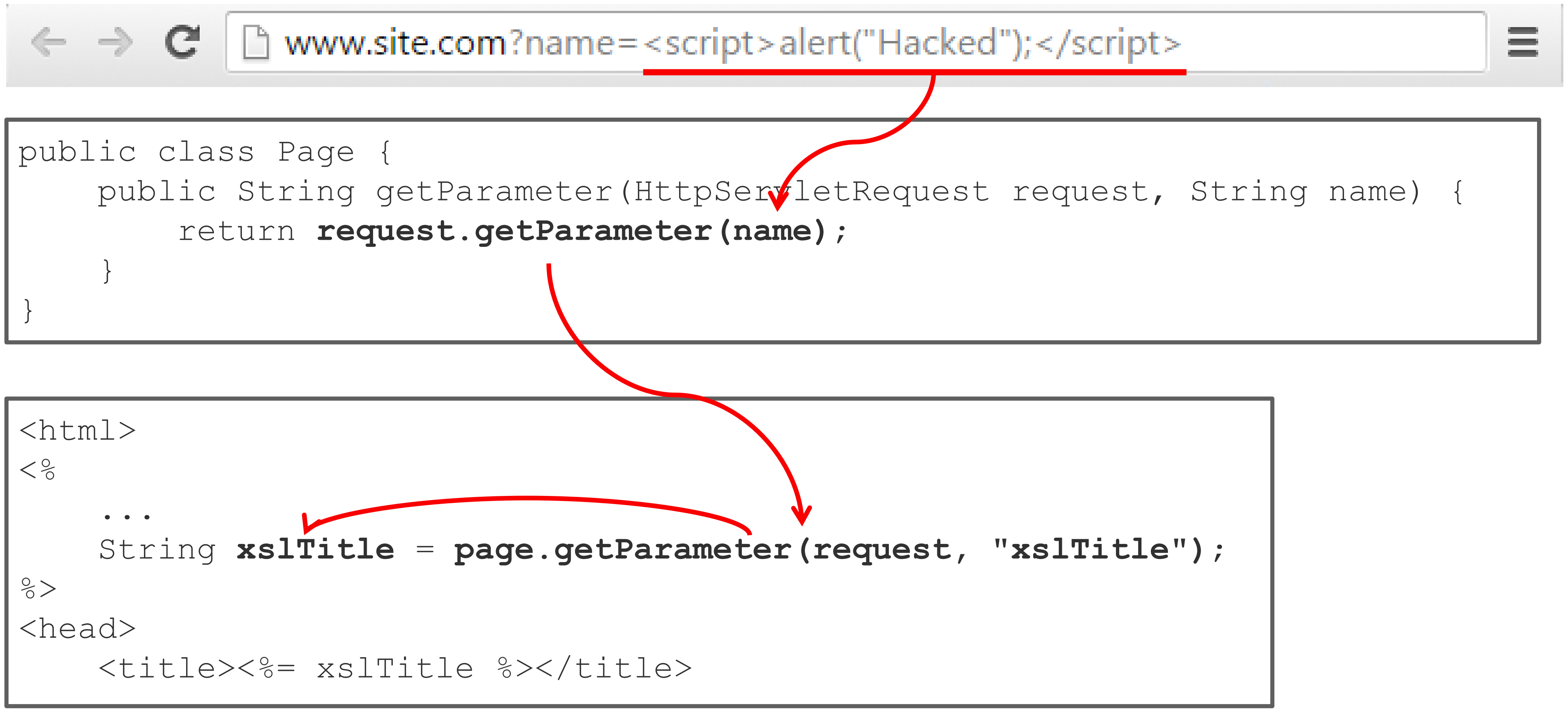
```
www.site.com?name=<script>alert("Hacked");</script>
```

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = page.getParameter(request, "xslTitle");
%>
<head>
    <title><%= xslTitle %></title>
```

Understanding XSS Flaws

Detailed example of a reflected cross-site scripting flaw



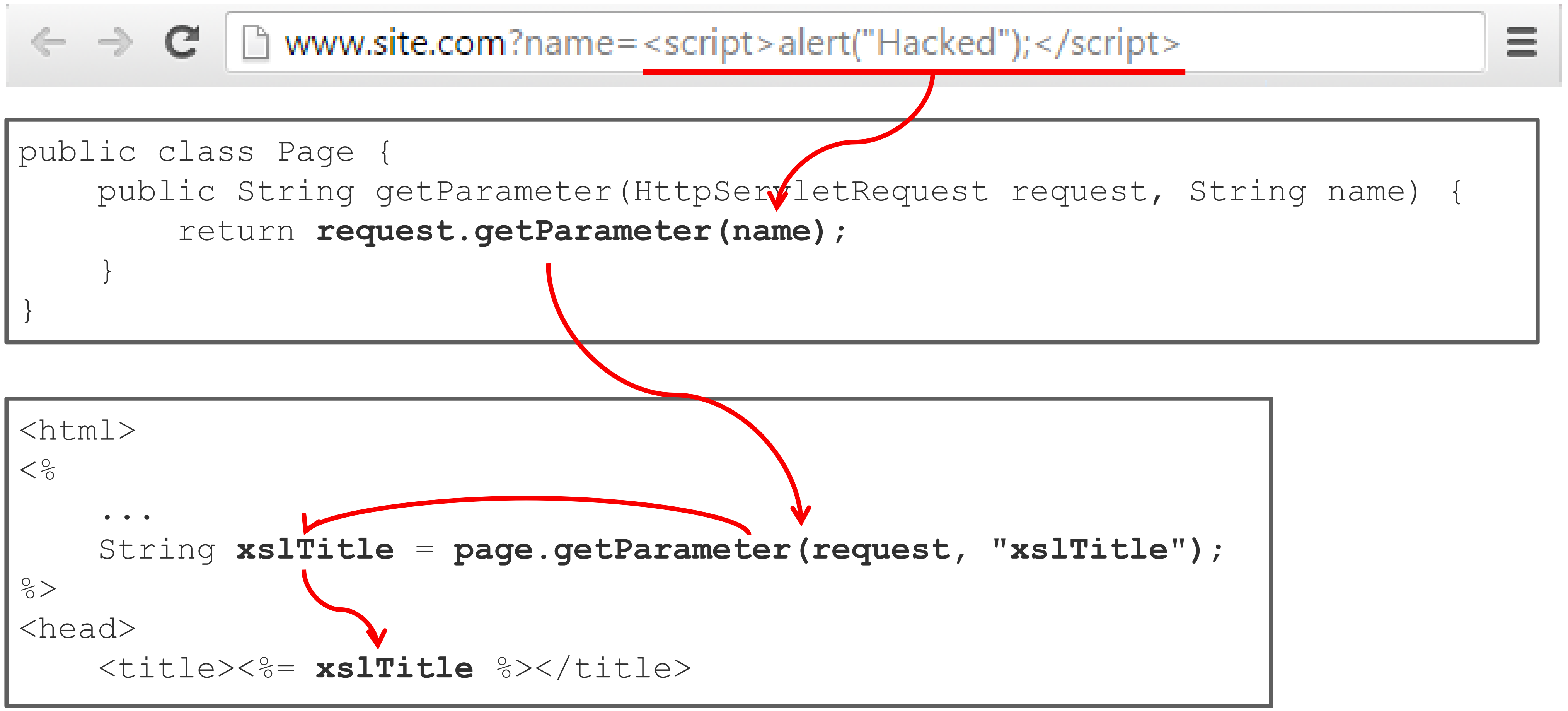
```
www.site.com?name=<script>alert("Hacked");</script>
```

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = page.getParameter(request, "xslTitle");
%>
<head>
    <title><%= xslTitle %></title>
```

Understanding XSS Flaws

Detailed example of a reflected cross-site scripting flaw



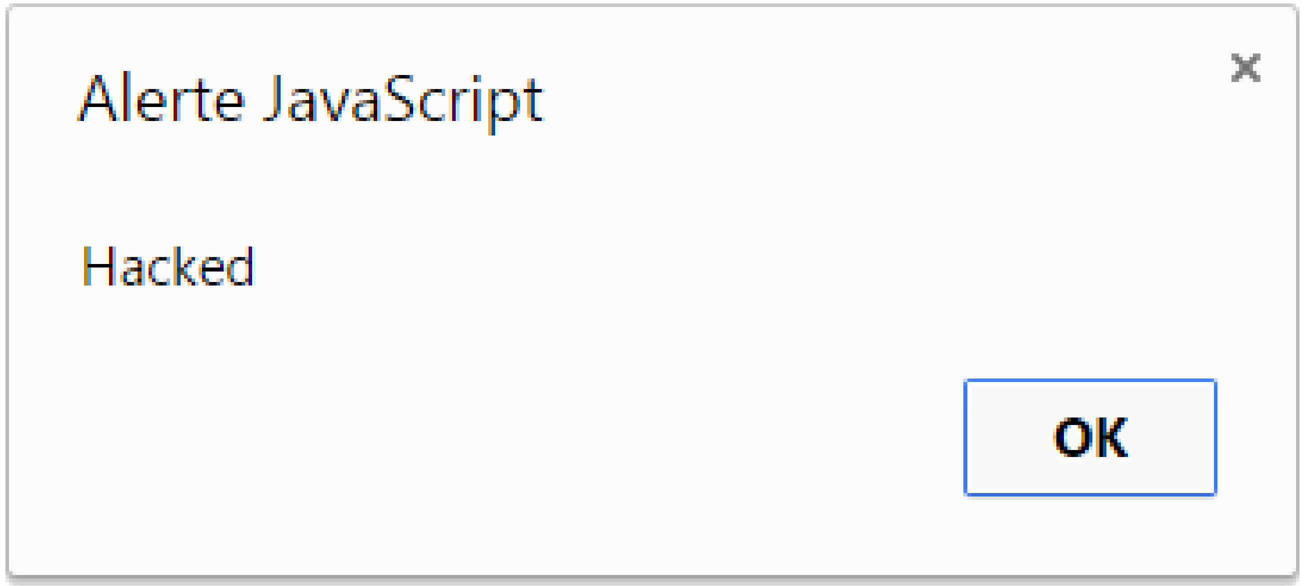
```
www.site.com?name=<script>alert("Hacked");</script>
```

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = page.getParameter(request, "xslTitle");
%>
<head>
    <title><%= xslTitle %></title>
```

Without Sanitization

Malicious script is executed

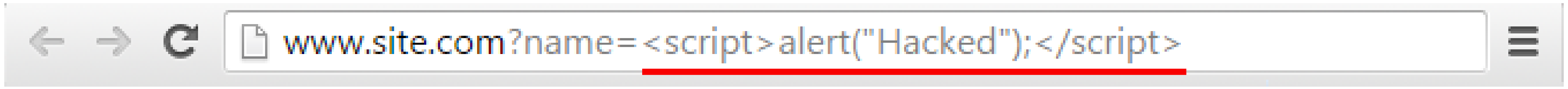


Alerte JavaScript

Hacked

Preventing XSS Flaws

Sanitization is the key

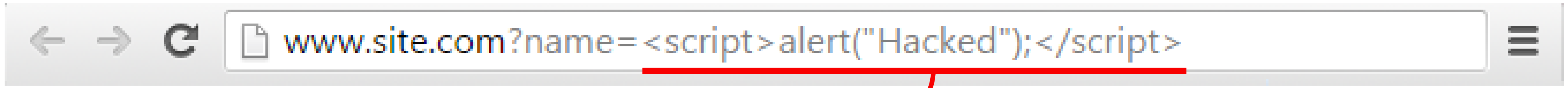

www.site.com?name=<script>alert("Hacked");</script>

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = htmlEncode(page.getParameter(request, "xslTitle"));
%>
<head>
    <title><%= xslTitle %></title>
```

Preventing XSS Flaws

Sanitization is the key



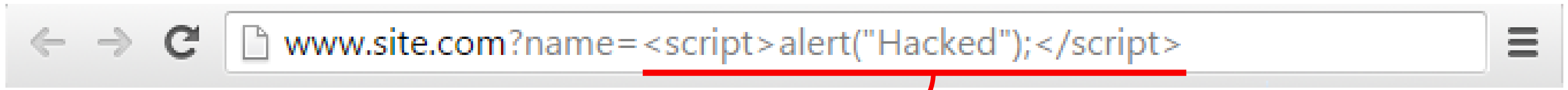
```
www.site.com?name=<script>alert("Hacked");</script>
```

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = htmlEncode(page.getParameter(request, "xslTitle"));
%>
<head>
    <title><%= xslTitle %></title>
```

Preventing XSS Flaws

Sanitization is the key



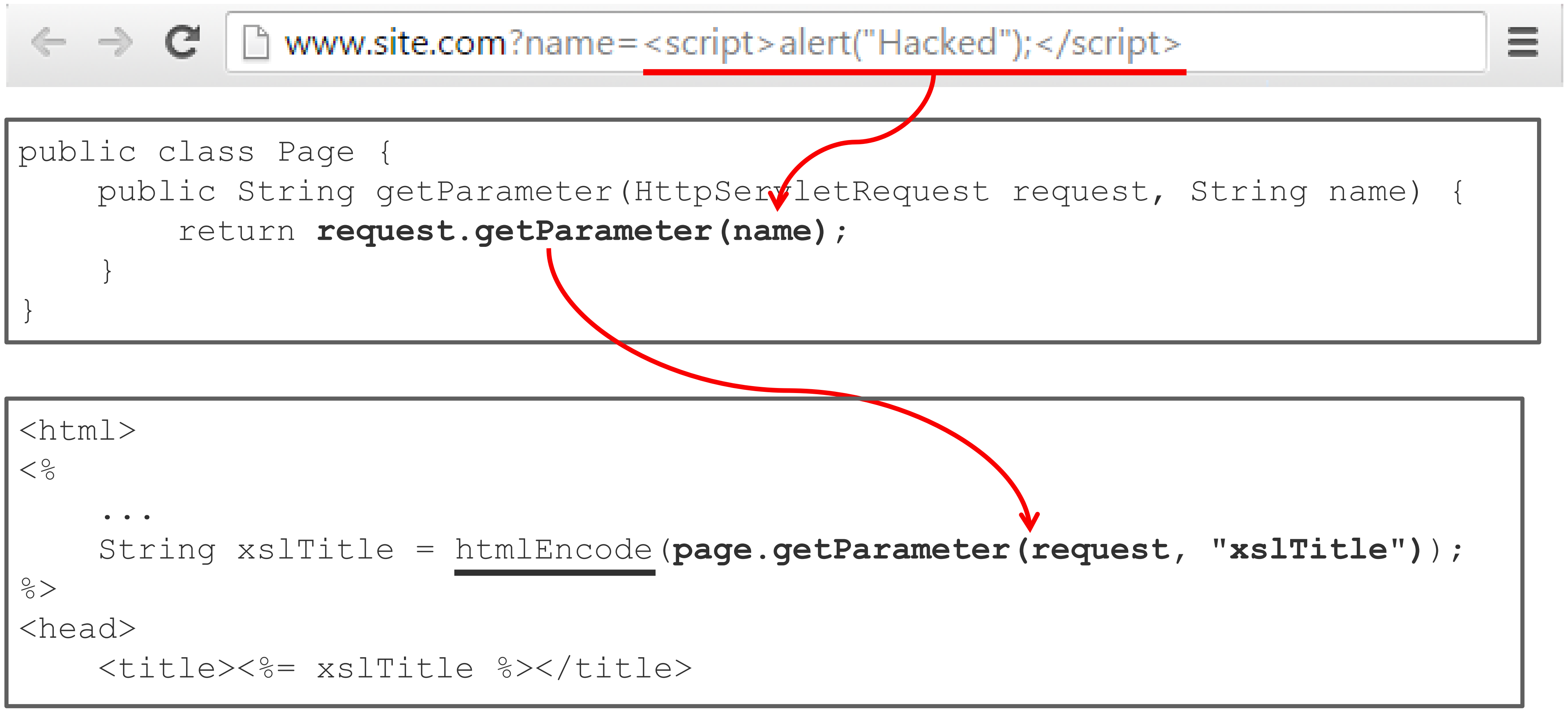
www.site.com?name=<script>alert("Hacked");</script>

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = htmlEncode(page.getParameter(request, "xslTitle"));
%>
<head>
    <title><%= xslTitle %></title>
```

Preventing XSS Flaws

Sanitization is the key



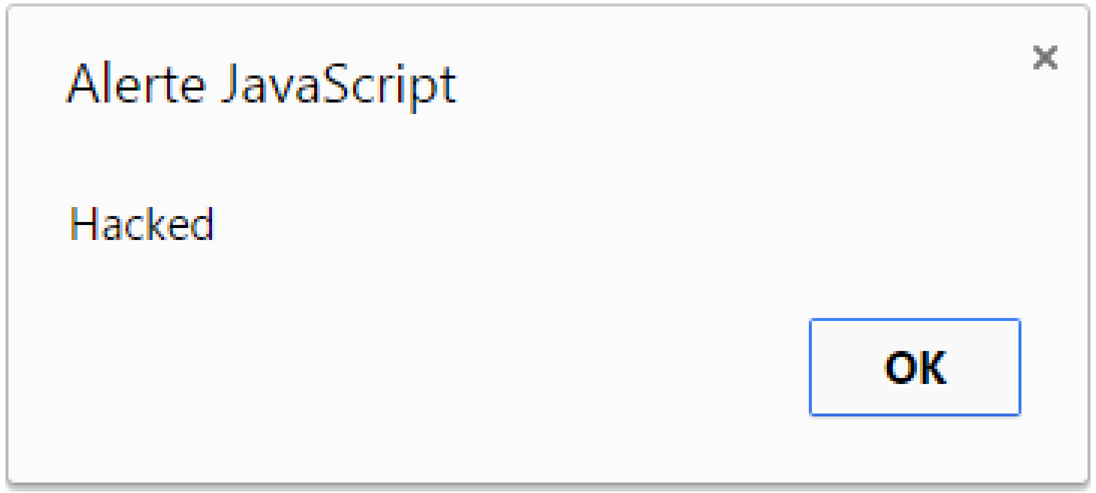
```
www.site.com?name=<script>alert("Hacked");</script>
```

```
public class Page {
    public String getParameter(HttpServletRequest request, String name) {
        return request.getParameter(name);
    }
}
```

```
<html>
<%
    ...
    String xslTitle = htmlEncode(page.getParameter(request, "xslTitle"));
%>
<head>
    <title><%= xslTitle %></title>
```

Effect of Sanitization

Without sanitizer: Successful attack



Hacked

With sanitizer: A string is displayed

<script>alert("Hacked");</script>

Static Taint Analysis of Industrial Web Applications

Experiments with IFDS and points-to analysis

Modelling Web Application Behaviour

Supporting the RequestDispatcher

```
RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
if ("include".equalsIgnoreCase(action)) {
      rd.include(request, response);
} else if ("forward".equalsIgnoreCase(action)) {
      rd.forward(request, response);
}
```

Modelling Web Application Behaviour

Supporting the RequestDispatcher

```
RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
if ("include".equalsIgnoreCase(action)) {
      rd.include(request, response);
} else if ("forward".equalsIgnoreCase(action)) {
      rd.forward(request, response);
}
```

Configuration file analysis needed to map “index.jsp” to an actual servlet.

Modelling Web Application Behaviour

Supporting the RequestDispatcher

```
RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
if ("include".equalsIgnoreCase(action)) {
      rd.include(request, response);
} else if ("forward".equalsIgnoreCase(action)) {
      rd.forward(request, response);
}
```

JEE server injects an instance of the servlet in the RequestDispatcher object.

Modelling Web Application Behaviour

Supporting the RequestDispatcher

```
RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
if ("include".equalsIgnoreCase(action)) {
      rd.include(request, response);
} else if ("forward".equalsIgnoreCase(action)) {
      rd.forward(request, response);
}
```

Def-use analysis needed to track the dispatcher.

Modelling Web Application Behaviour

Supporting the RequestDispatcher

```
RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
if ("include".equalsIgnoreCase(action)) {
      rd.include(request, response);
} else if ("forward".equalsIgnoreCase(action)) {
      rd.forward(request, response);
}
```

`include` and `forward` calls are replaced with calls to the `_jspService()` method of the servlet instance contained in `rd`.

IFDS Framework

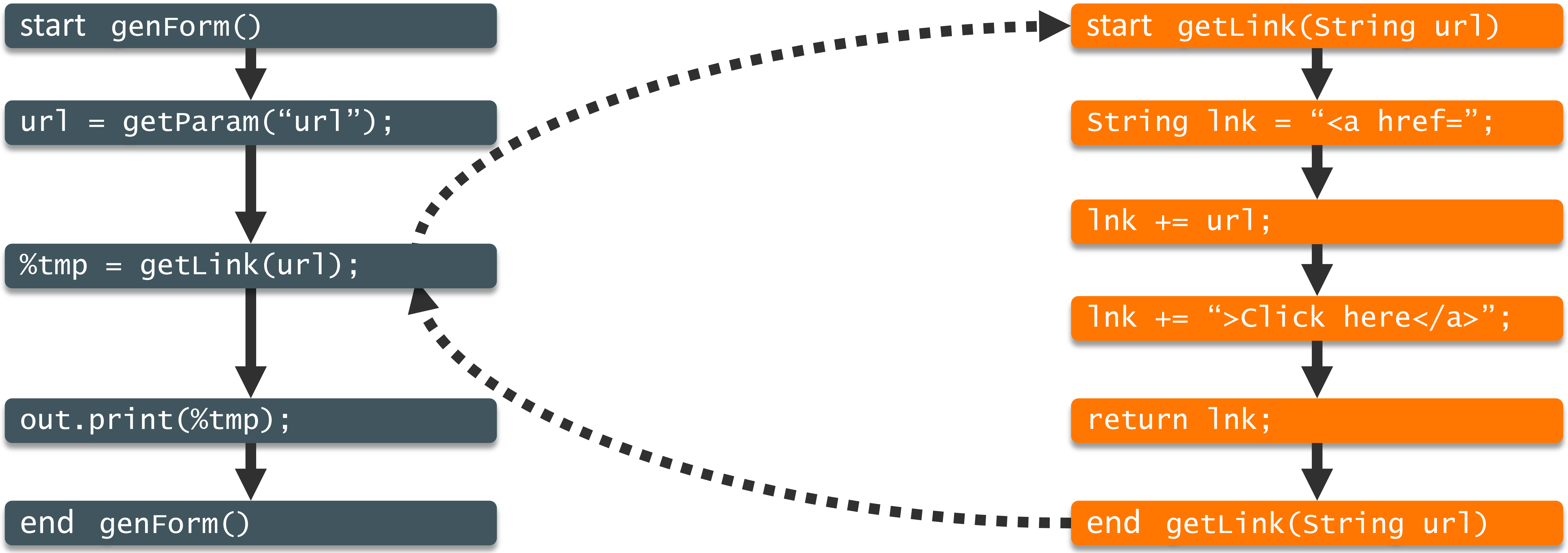
- Framework for solving inter-procedural, finite, distributive, subset (IFDS) problems.
- Flow functions are defined over a finite domain **D** and have to be distributive over the meet operator: $f(a) \wedge f(b) = f(a \wedge b)$.
- IFDS reduces data-flow problems to graph reachability problems.

Working Example

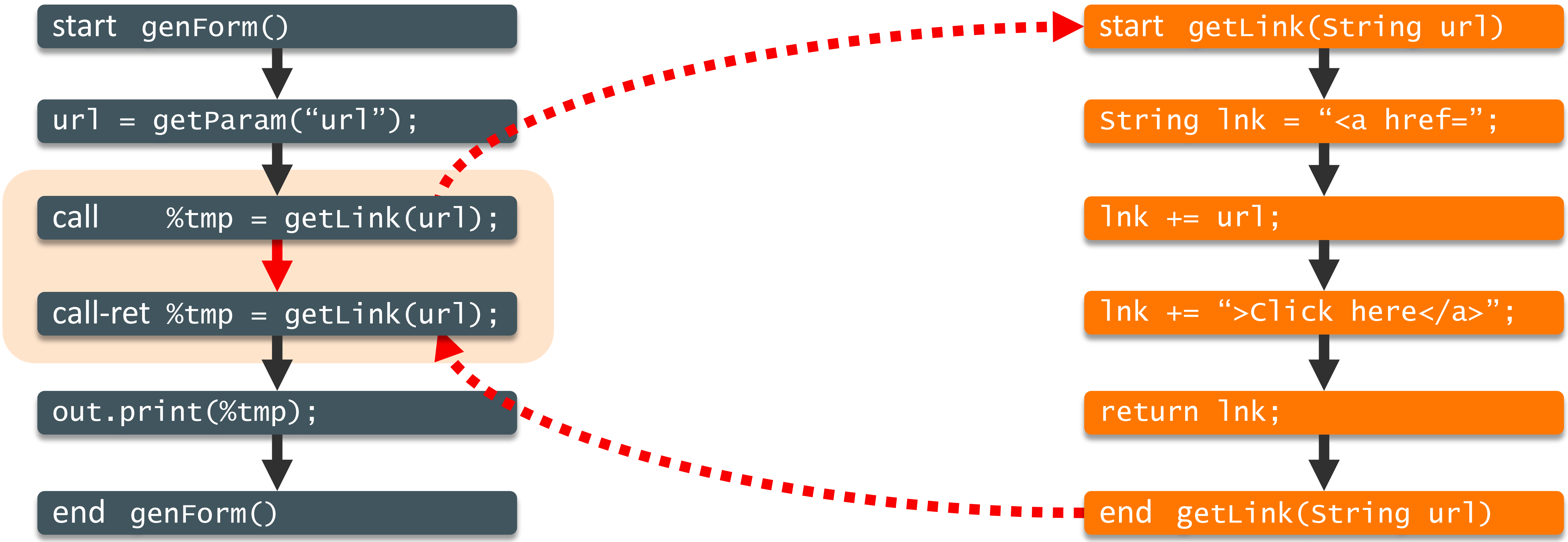
```
public void genForm() {
  ...
  url = getParam(“url”);
  out.print(getLink(url));
  ...
}
```

```
public String getLink(String url) {
  ...
  String lnk = “<a href=”;
  lnk += url;
  lnk += “>Click here</a>”;
  return lnk;
}
```

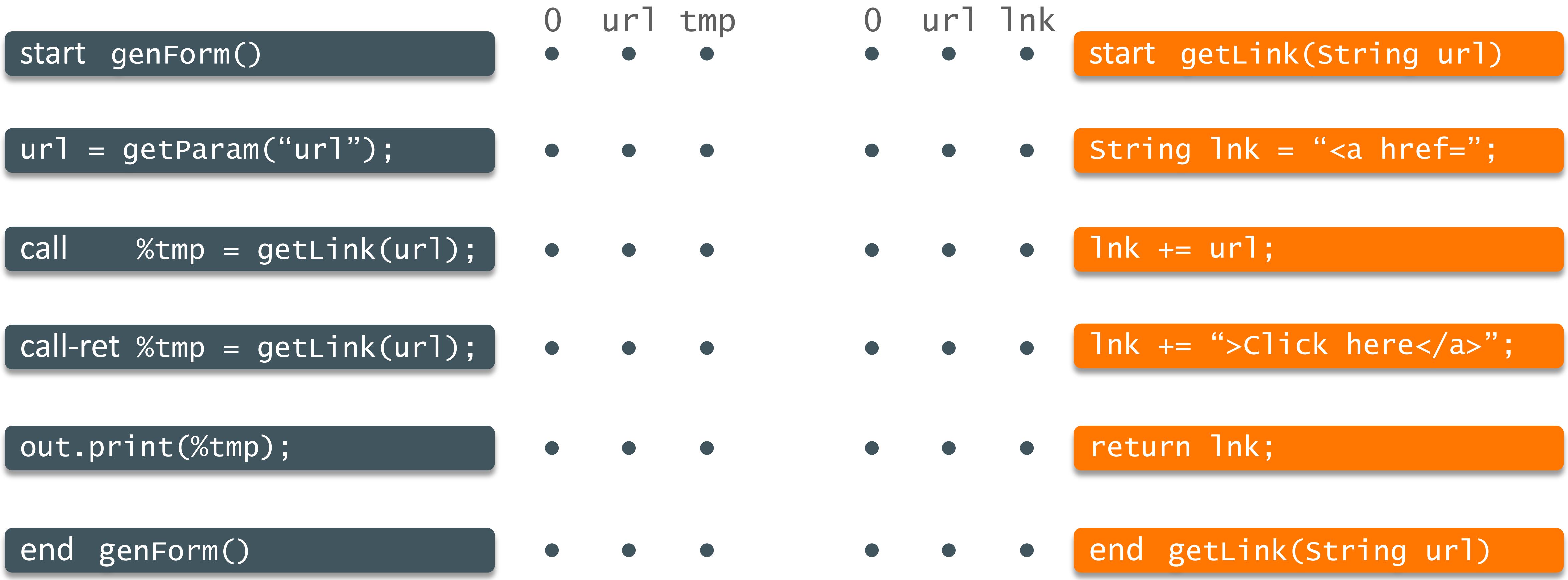
Control Flow Graph



Supergraph


start genForm()
url = getParam("url");
call %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end genForm()
start getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end getLink(String url)

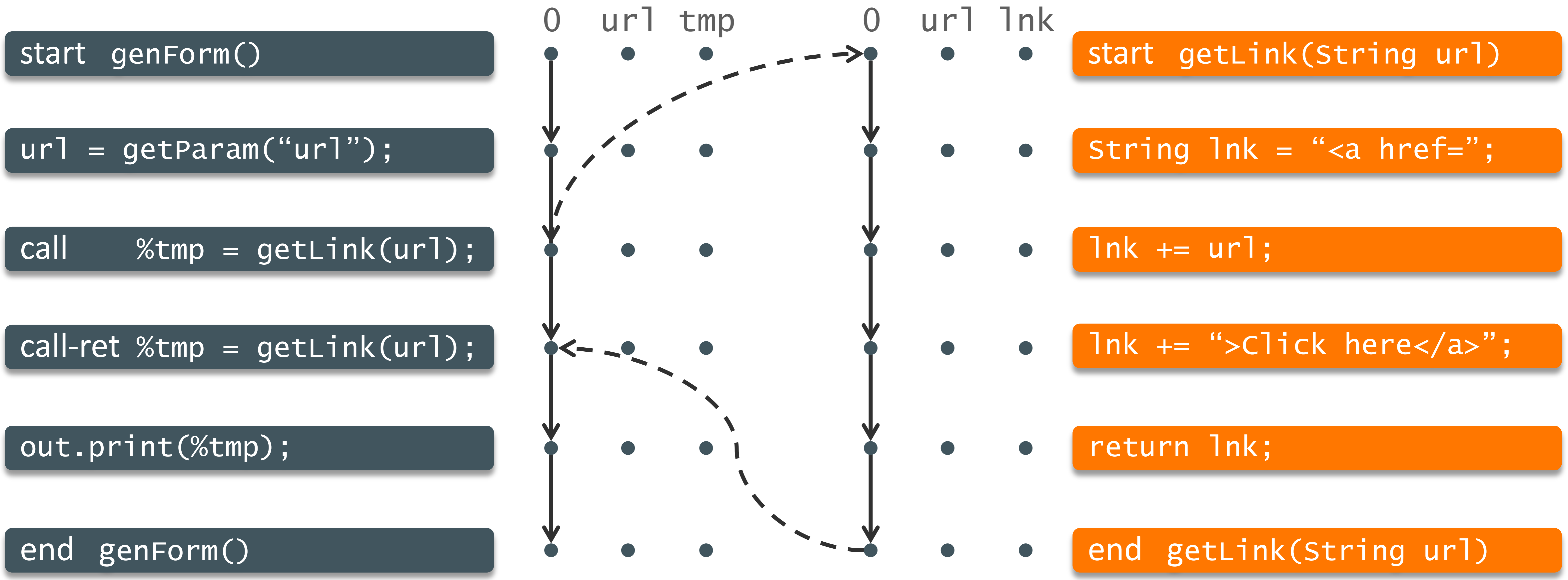
Exploded Supergraph



```
start  genForm()
url  = getParam(“url”);
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = “<a href=”;
lnk += url;
lnk += “>Click here</a>”;
return lnk;
end  getLink(String url)
```

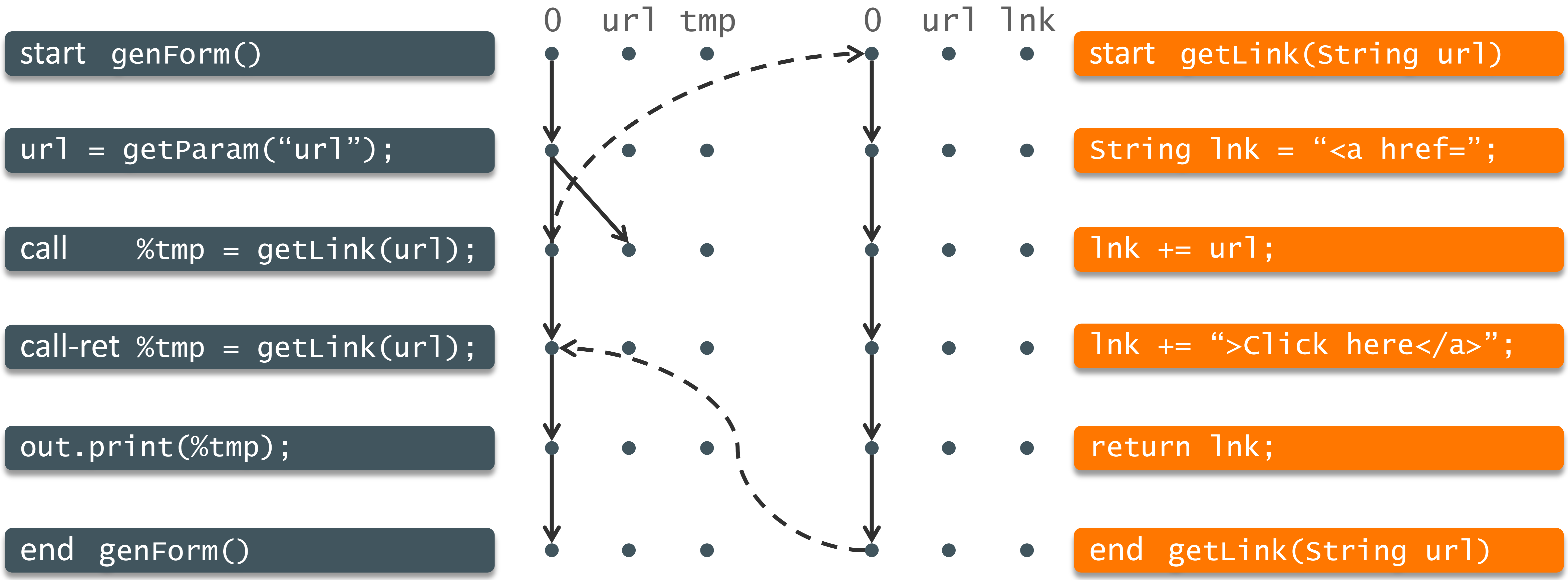
Exploded Supergraph



```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

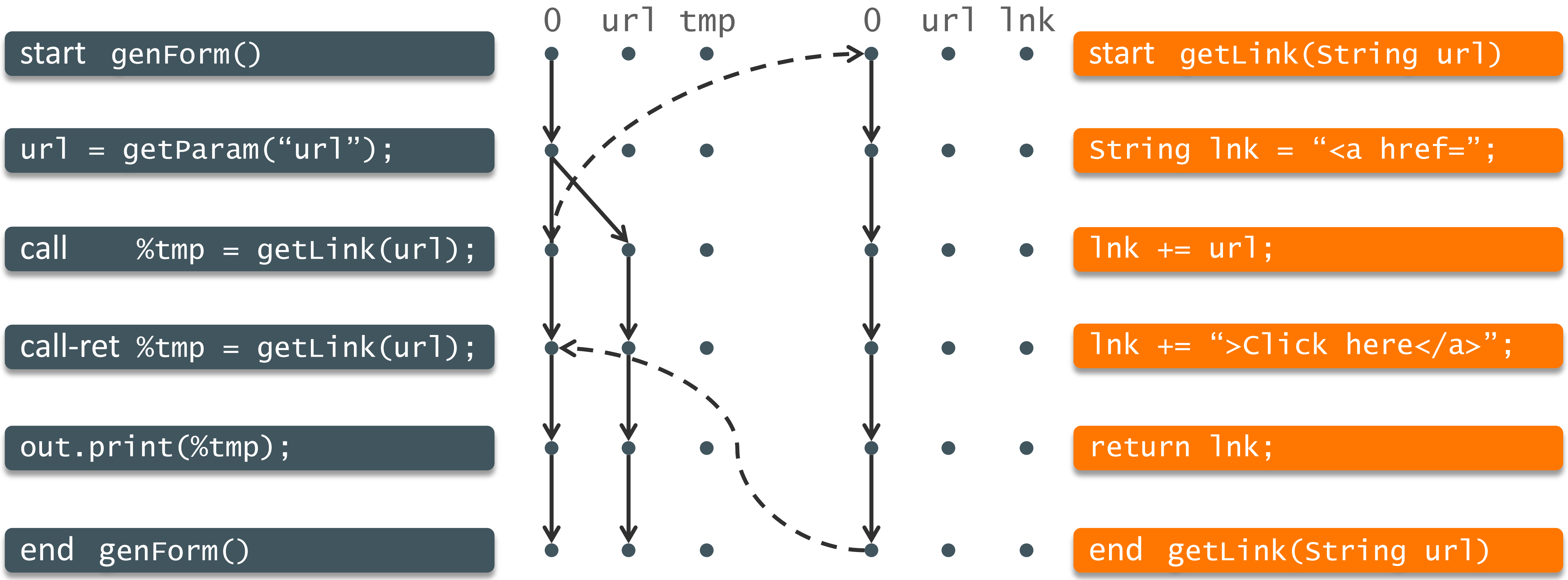
Exploded Supergraph



```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

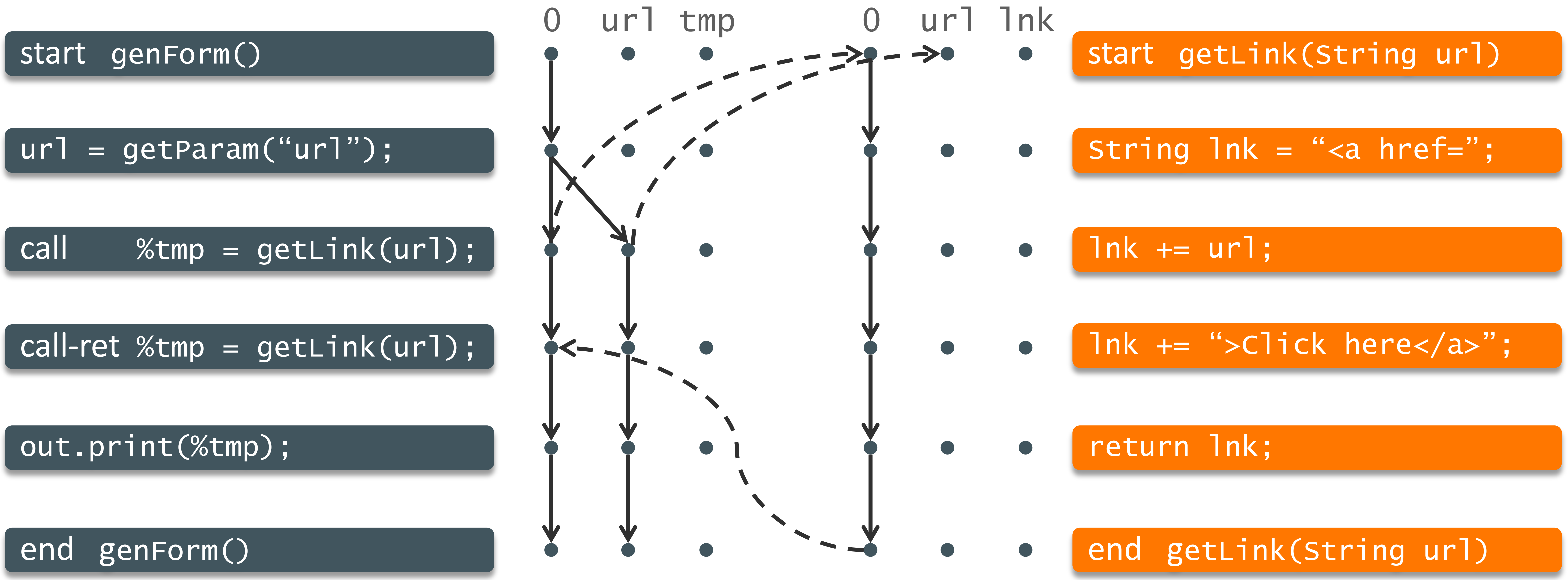
Exploded Supergraph



```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

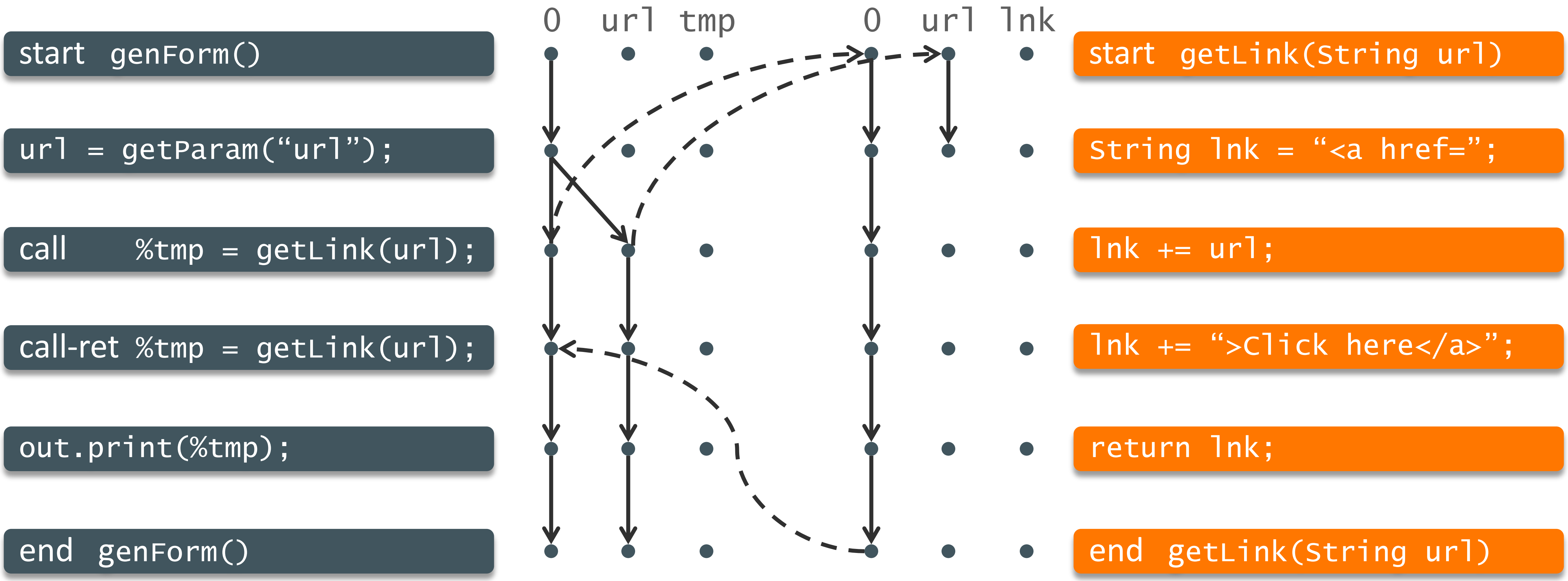
Exploded Supergraph



```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Exploded Supergraph

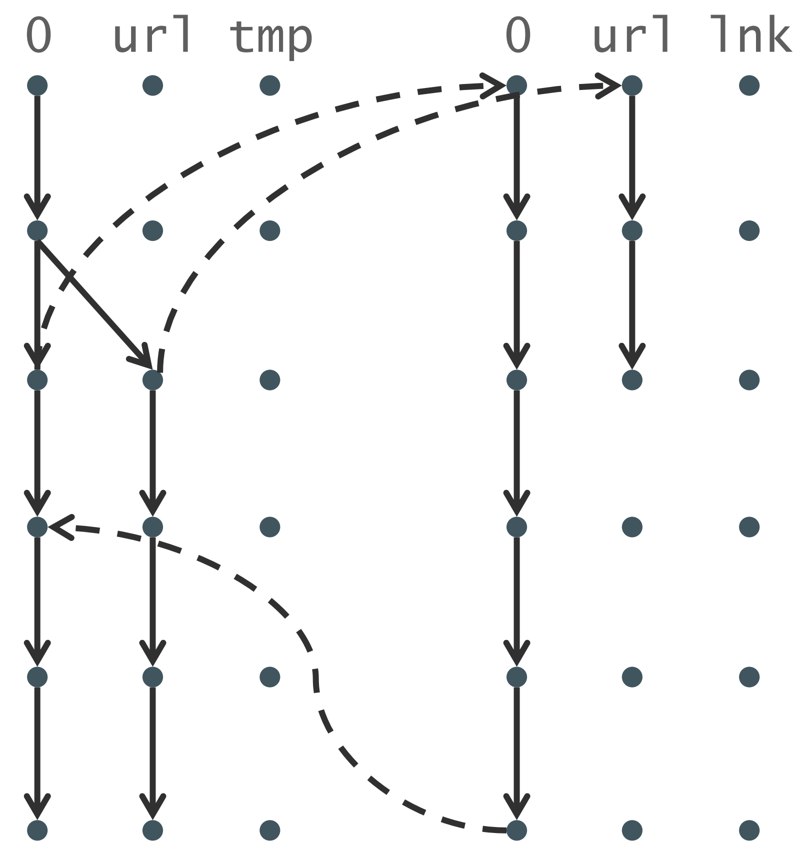


```
start  genForm()
url = getParam(“url”);
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = “<a href=”;
lnk += url;
lnk += “>Click here</a>”;
return lnk;
end  getLink(String url)
```

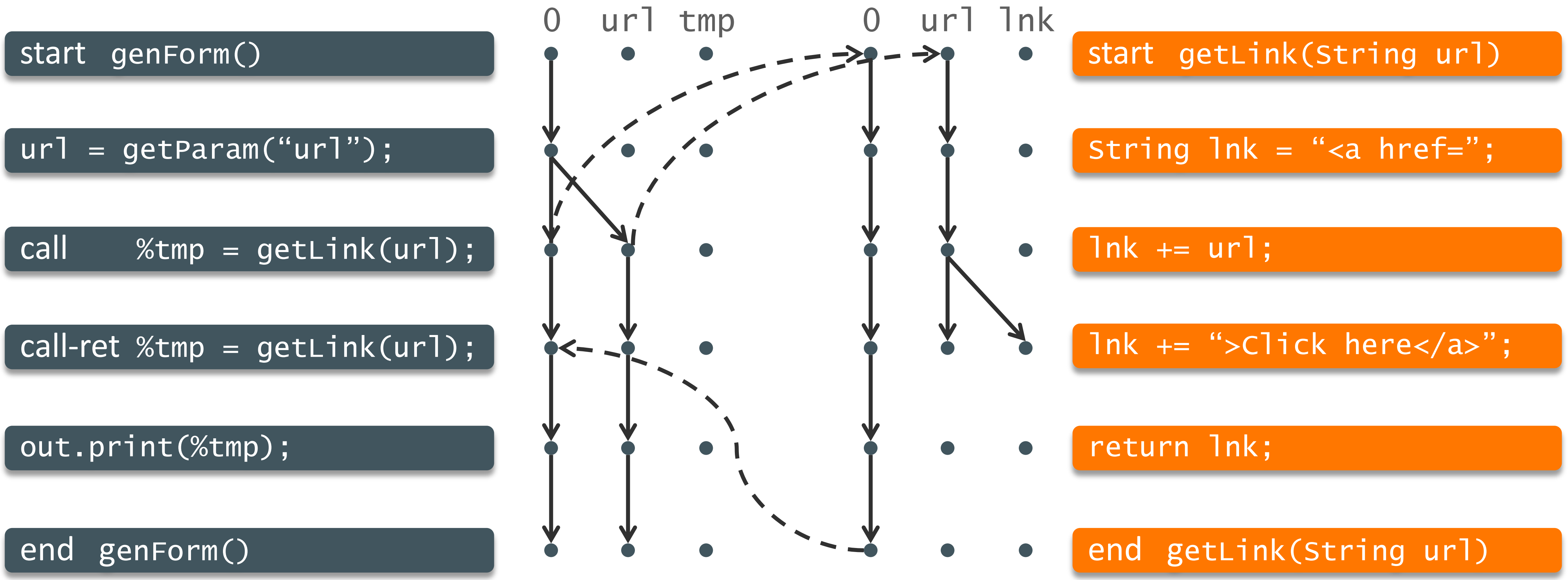
Exploded Supergraph

```
start  genForm()
url = getParam(“url”);
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = “<a href=”;
lnk += url;
lnk += “>Click here</a>”;
return lnk;
end  getLink(String url)
```

Exploded Supergraph

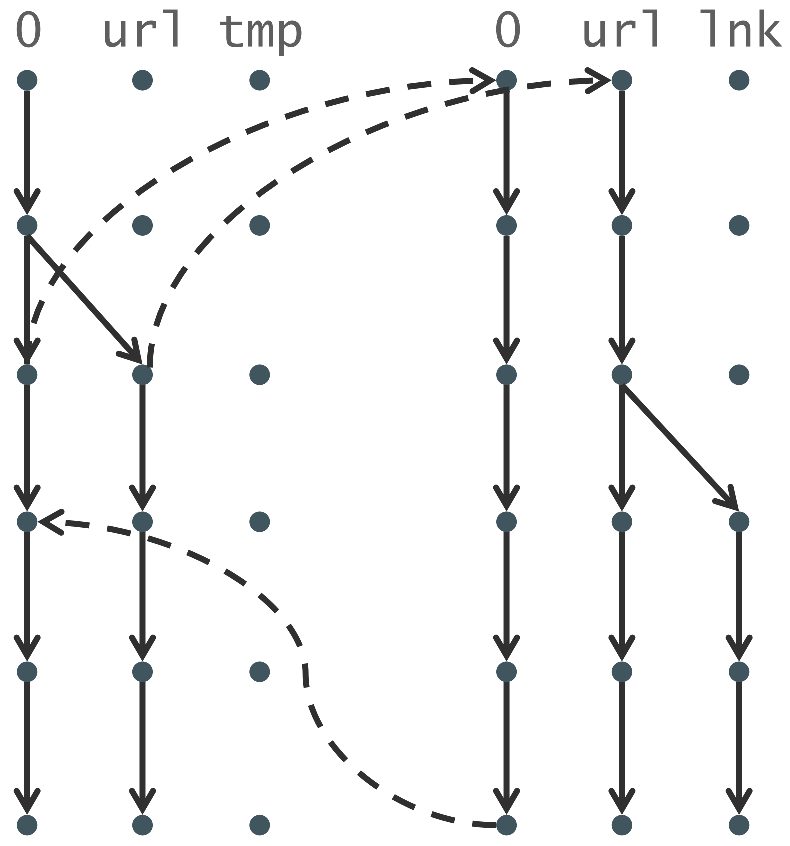


```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Exploded Supergraph

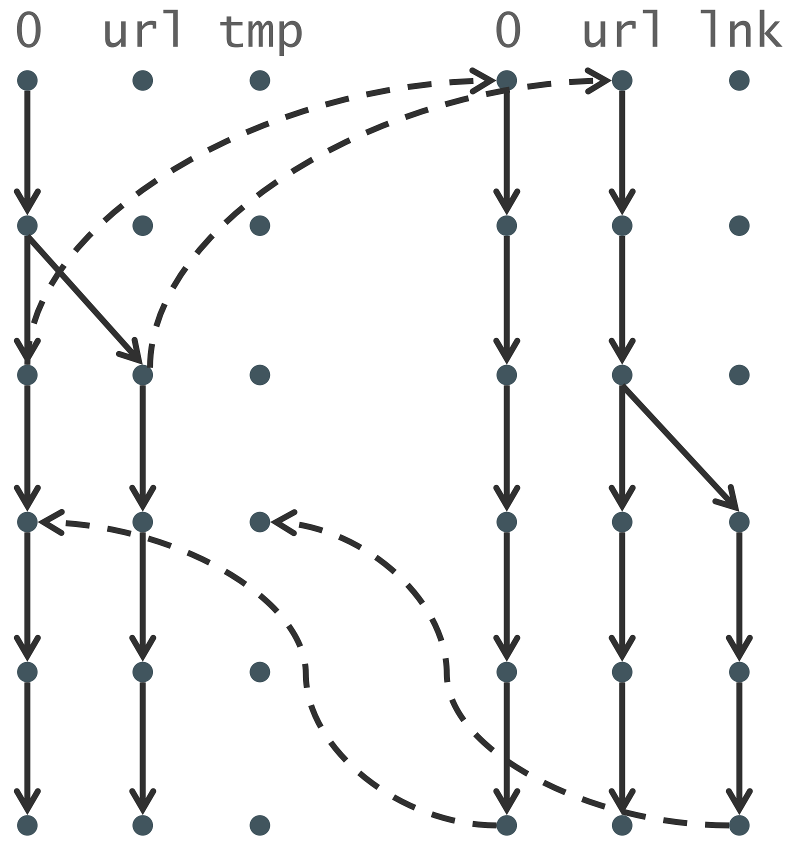
```
start  genForm()

url = getParam("url");

call     %tmp = getLink(url);

call-ret %tmp = getLink(url);

out.print(%tmp);

end  genForm()
```



```
start  getLink(String url)

String lnk = "<a href=";

lnk += url;

lnk += ">Click here</a>";

return lnk;

end  getLink(String url)
```

Exploded Supergraph

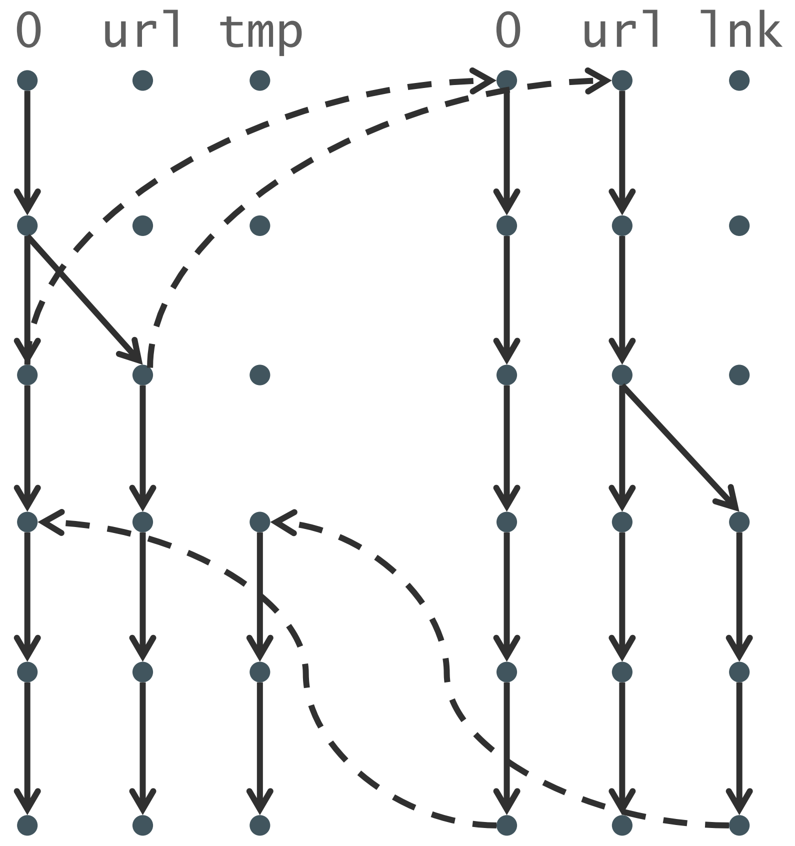
```
start  genForm()
url = getParam("url");
call    %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Exploded Supergraph

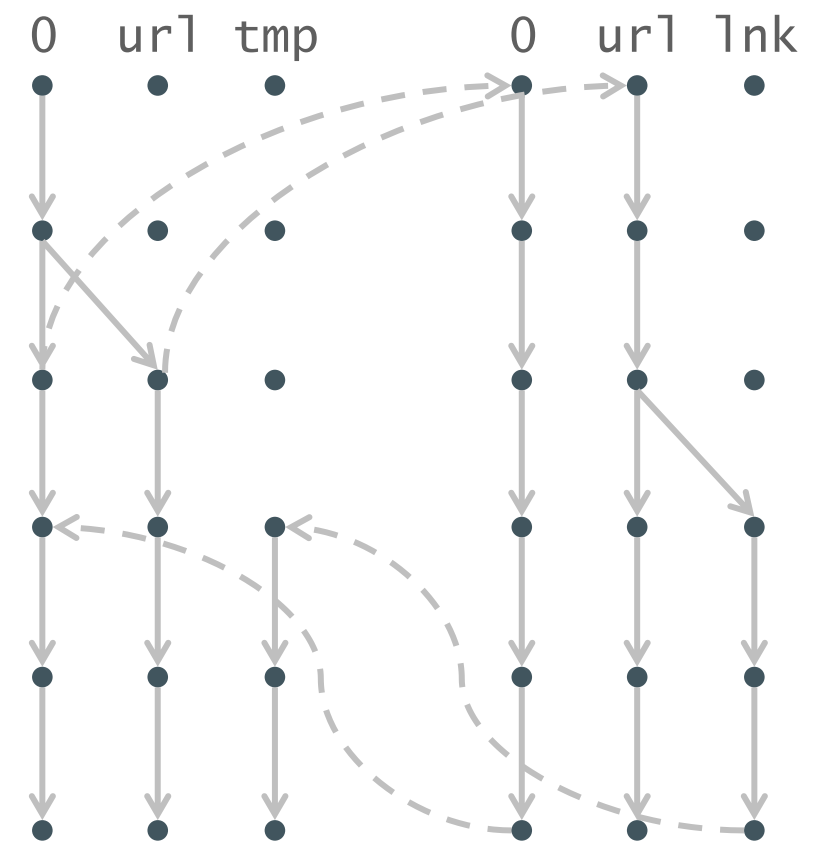
```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Reachability Analysis

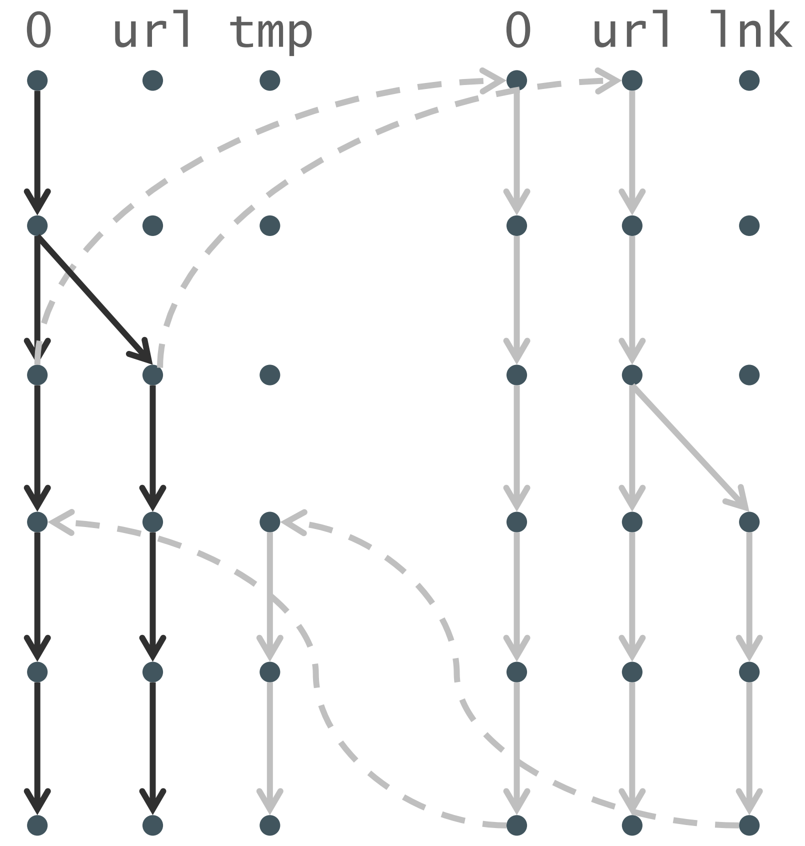
```
start  genForm()
url = getParam(“url”);
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = “<a href=”;
lnk += url;
lnk += “>Click here</a>”;
return lnk;
end  getLink(String url)
```

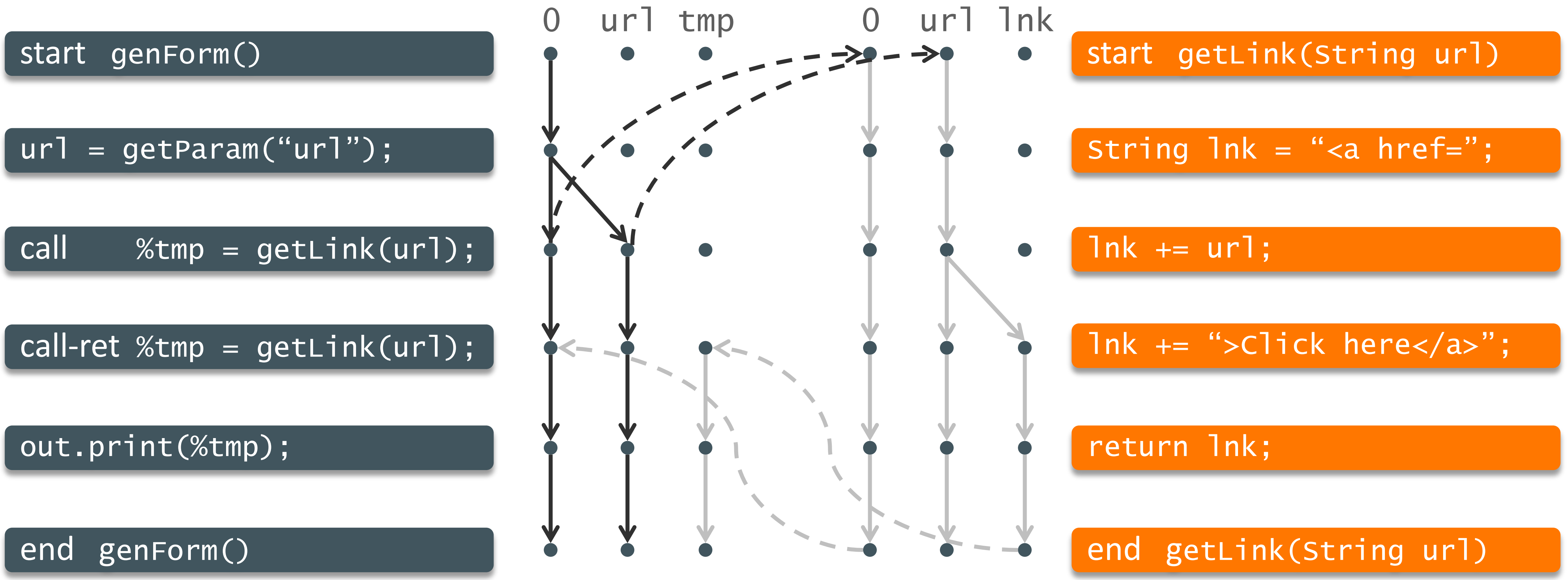
Reachability Analysis

```
start  genForm()
url  = getParam("url");
call     %tmp  = getLink(url);
call-ret %tmp  = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Reachability Analysis

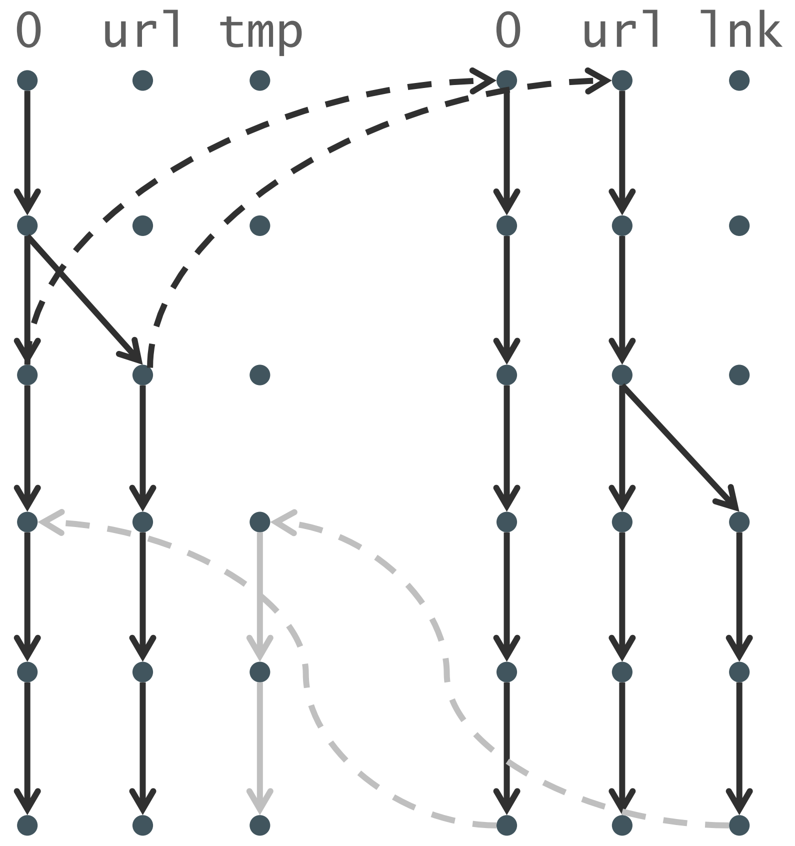


```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Reachability Analysis

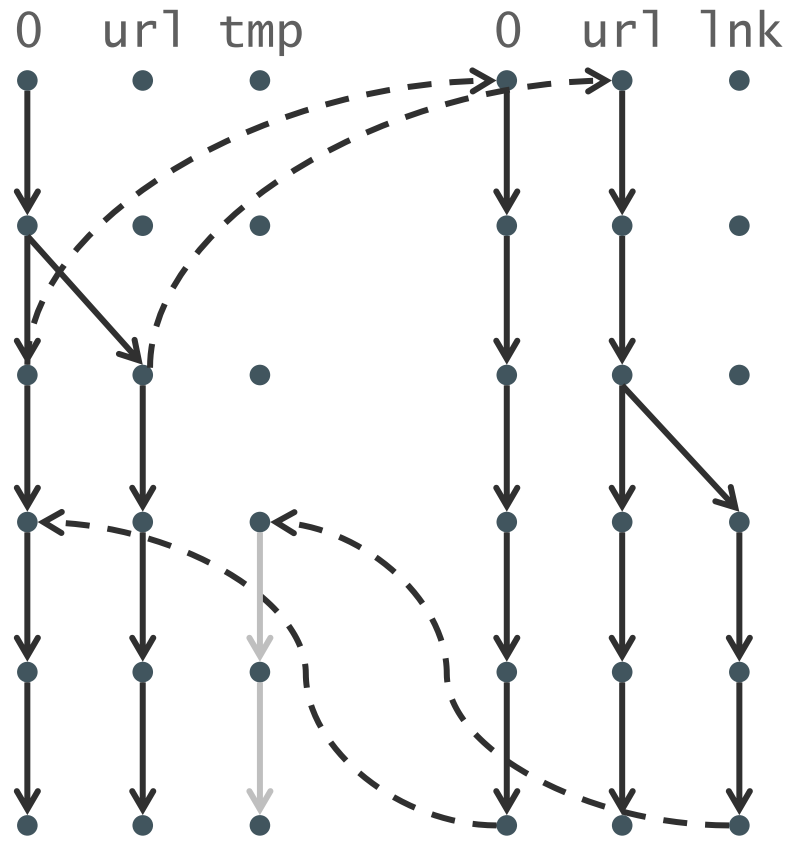
```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Reachability Analysis

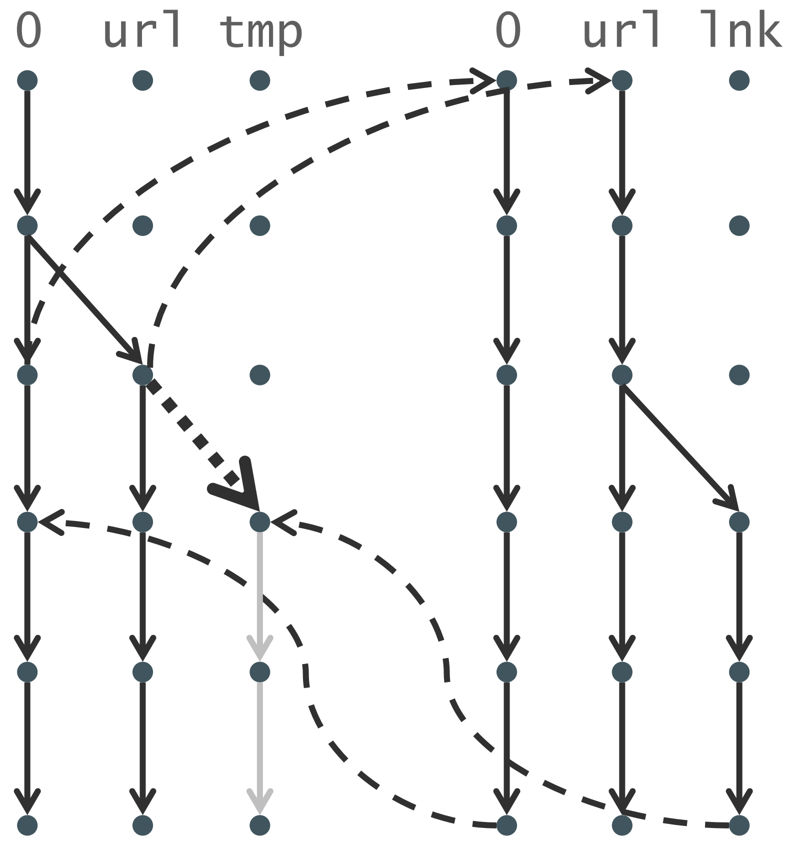
```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Function Summary Computation

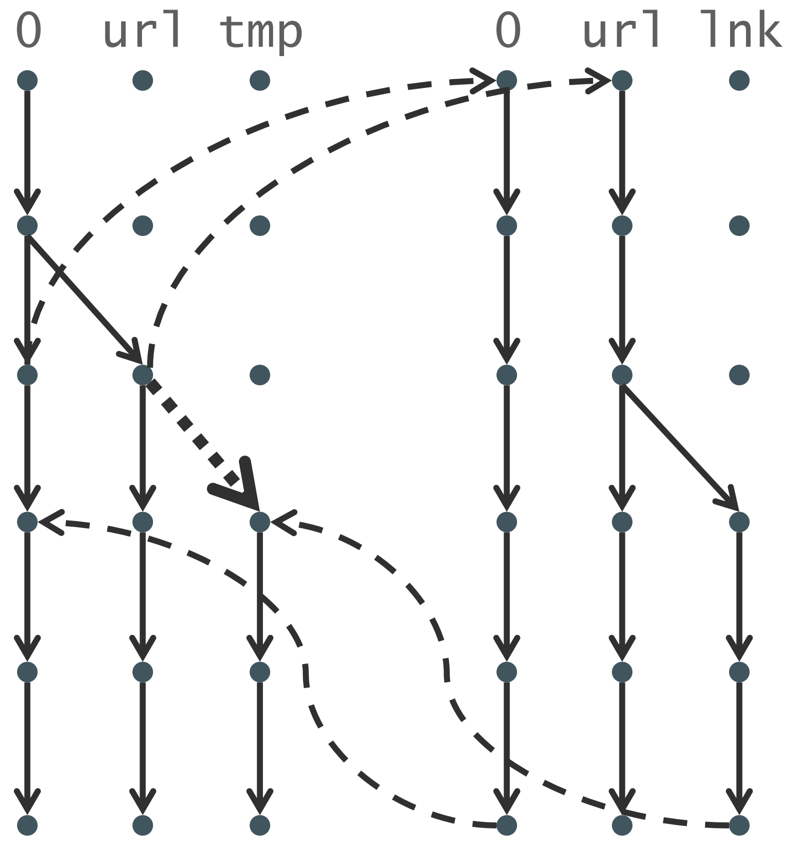
```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

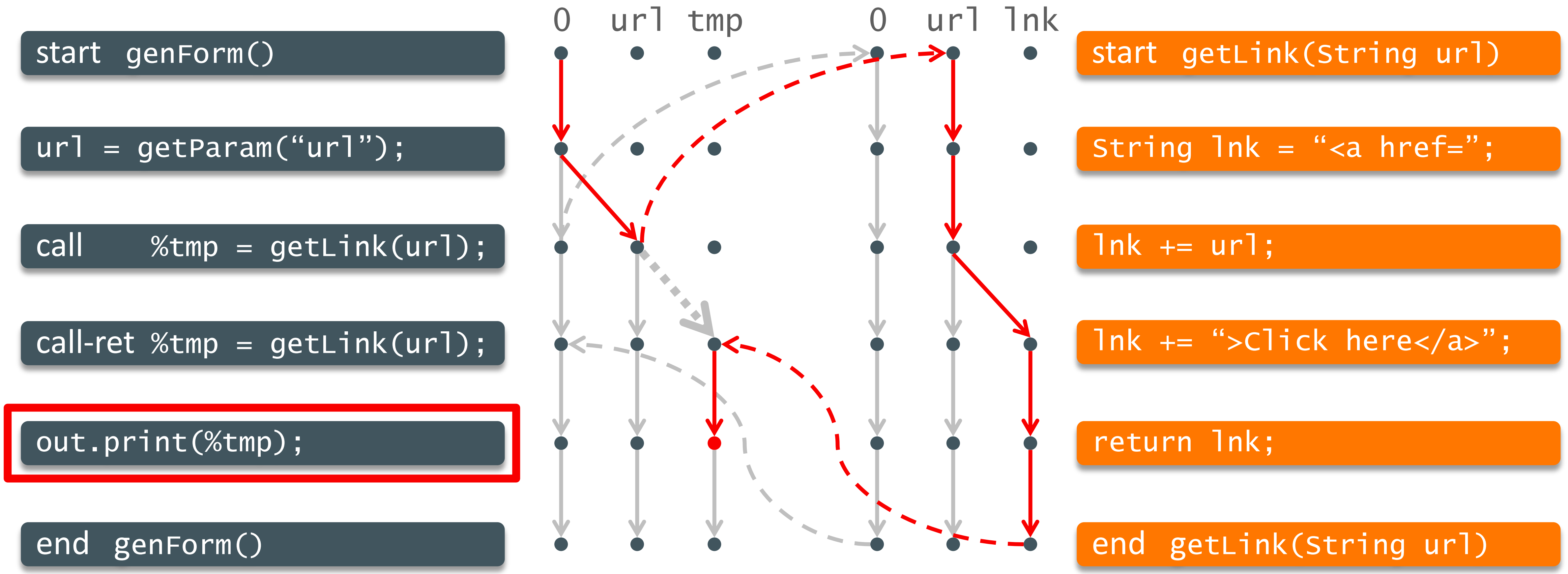
Reachability Analysis

```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Vulnerability Detection



```
start  genForm()
url = getParam("url");
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```

```
start  getLink(String url)
String lnk = "<a href=";
lnk += url;
lnk += ">Click here</a>";
return lnk;
end  getLink(String url)
```

Vulnerability Mitigation Through Sanitization

```
public void genForm() {
  ...
  url = getParam(“url”);
  out.print(getLink(url));
  ...
}
```

```
public String getLink(String url) {
  ...
  String link = “<a href=”;
  link += htmlEncode(url);
  link += “>Click here</a>”;
  return link;
}
```

Before Sanitization

```
start  genForm()
```

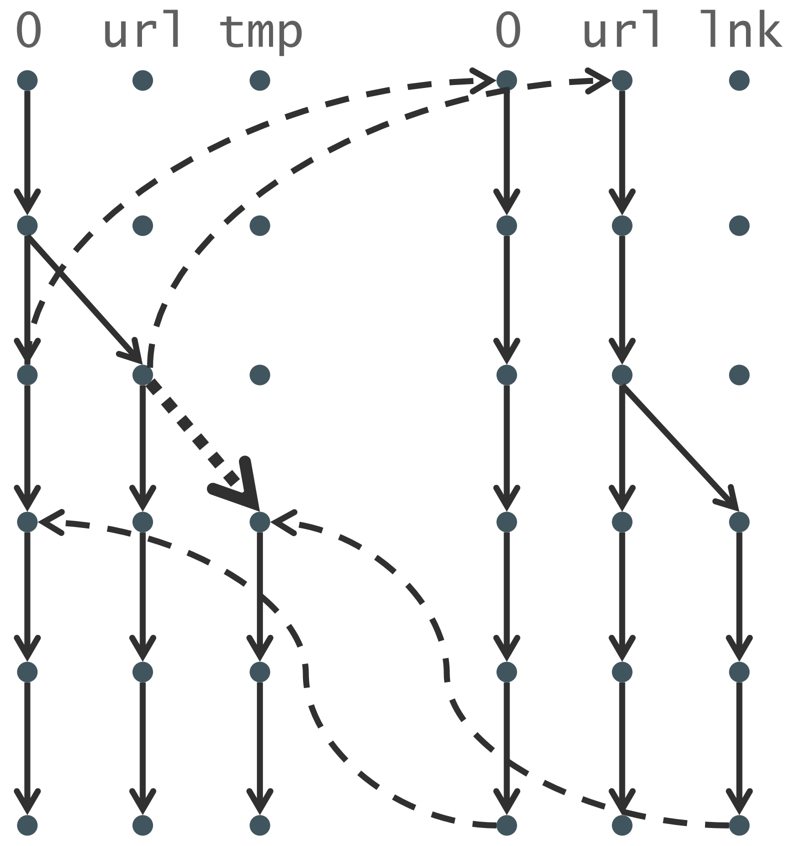
```
url = getParam("url");
```

```
call     %tmp = getLink(url);
```

```
call-ret %tmp = getLink(url);
```

```
out.print(%tmp);
```

```
end  genForm()
```



```
start  getLink(String url)
```

```
String lnk = "<a href=";
```

```
lnk += url;
```

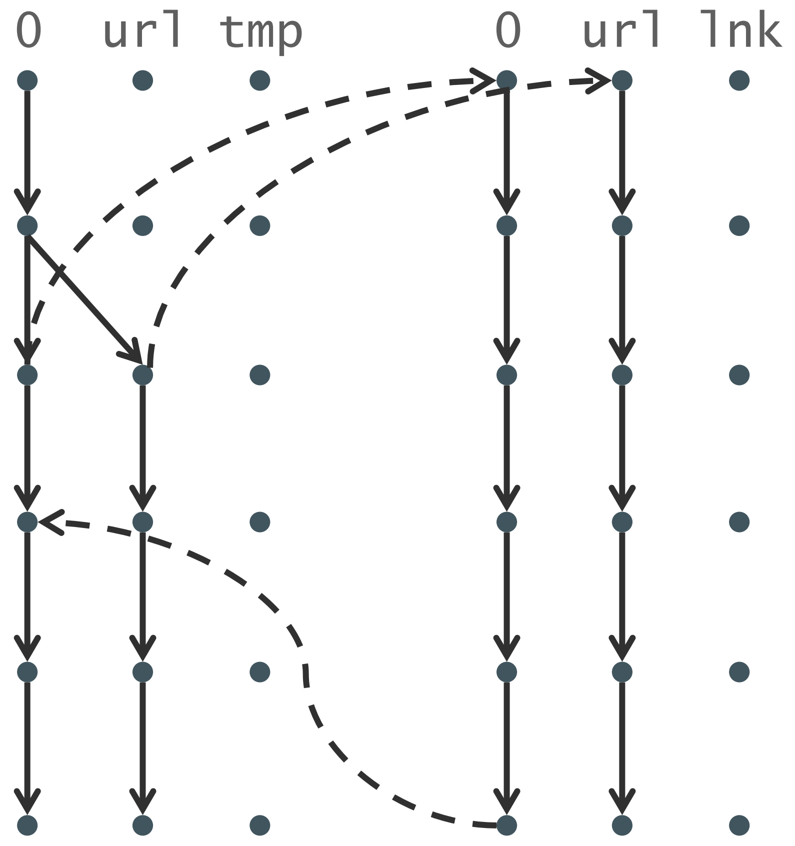
```
lnk += ">Click here</a>";
```

```
return lnk;
```

```
end  getLink(String url)
```

After Sanitization

```
start  genForm()
url = getParam(“url”);
call     %tmp = getLink(url);
call-ret %tmp = getLink(url);
out.print(%tmp);
end  genForm()
```



```
start  getLink(String url)
String lnk = “<a href=”;
lnk += htmlEncode(url);
lnk += “>Click here</a>”;
return lnk;
end  getLink(String url)
```

Adding Points-To Analysis to the Mix

```
public void genForm() {
  ...
  f.url = getParam(“url”);
  out.print(getLink(f));
  ...
}
```

```
public String getLink(Form form) {
  ...
  String link = “<a href=”;
  link += form.url;
  link += “>Click here</a>”;
  return link;
}
```

Points-To Analysis in Soufflé

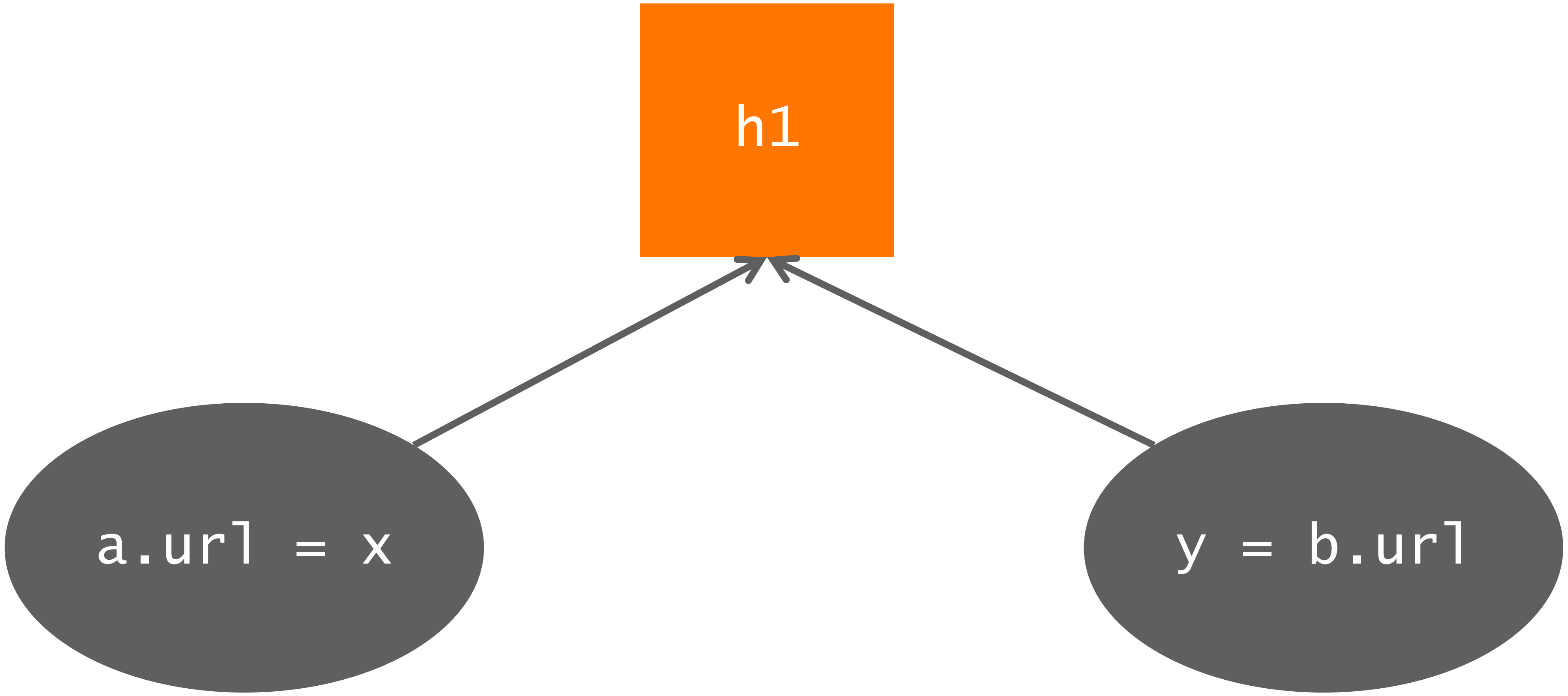
OpenJDK7-b147

Context-insensitive points-to	8m	6.7Gb
Context-sensitive points-to (minus swing)	34m	26.8Gb
Context-sensitive points-to (full)	7h	874Gb

ORACLE

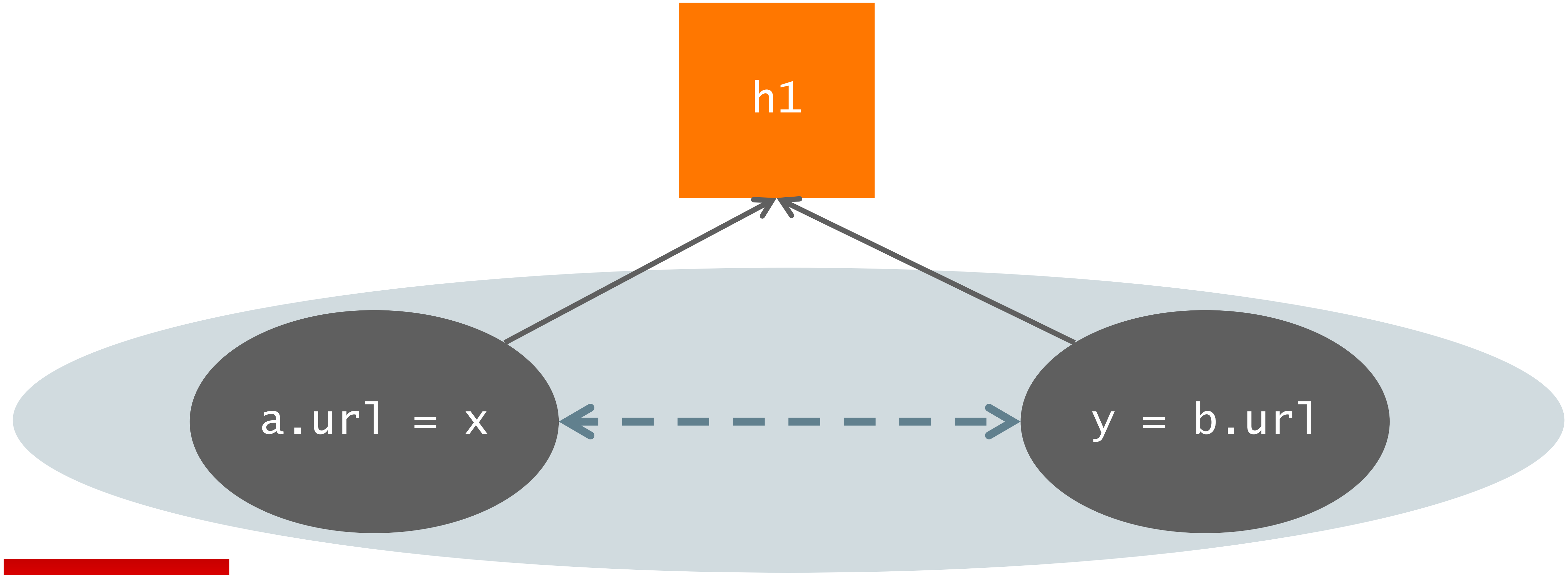
Matching Loads and Stores

Eliminating heap abstraction in points-to analysis



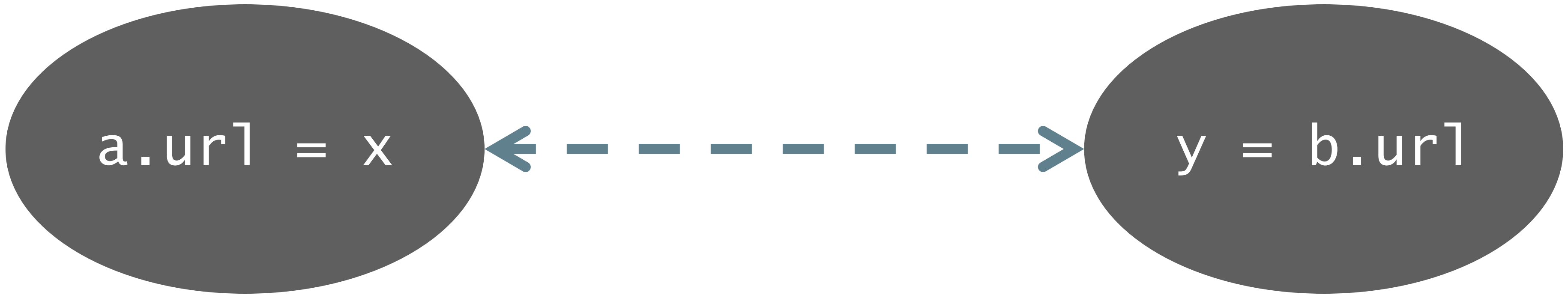
Matching Loads and Stores

Eliminating heap abstraction in points-to analysis

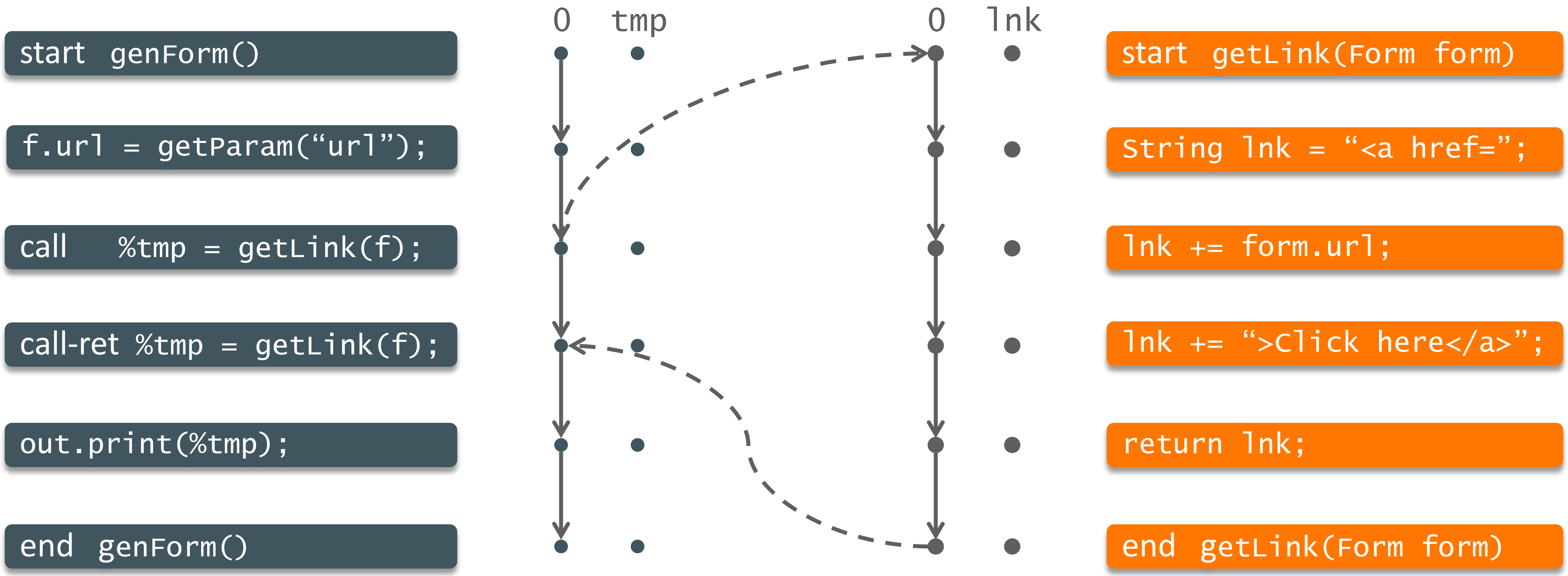


Matching Loads and Stores

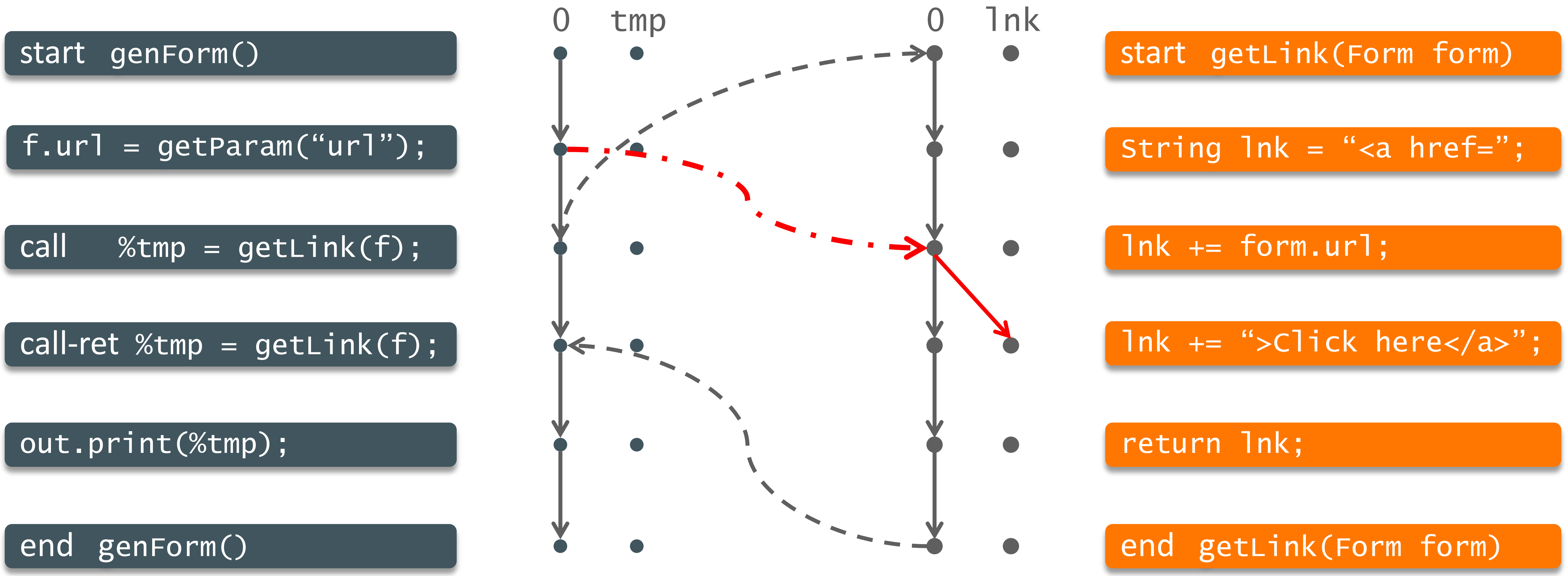
Eliminating heap abstraction in points-to analysis



IFDS + Points-to



IFDS + Points-to

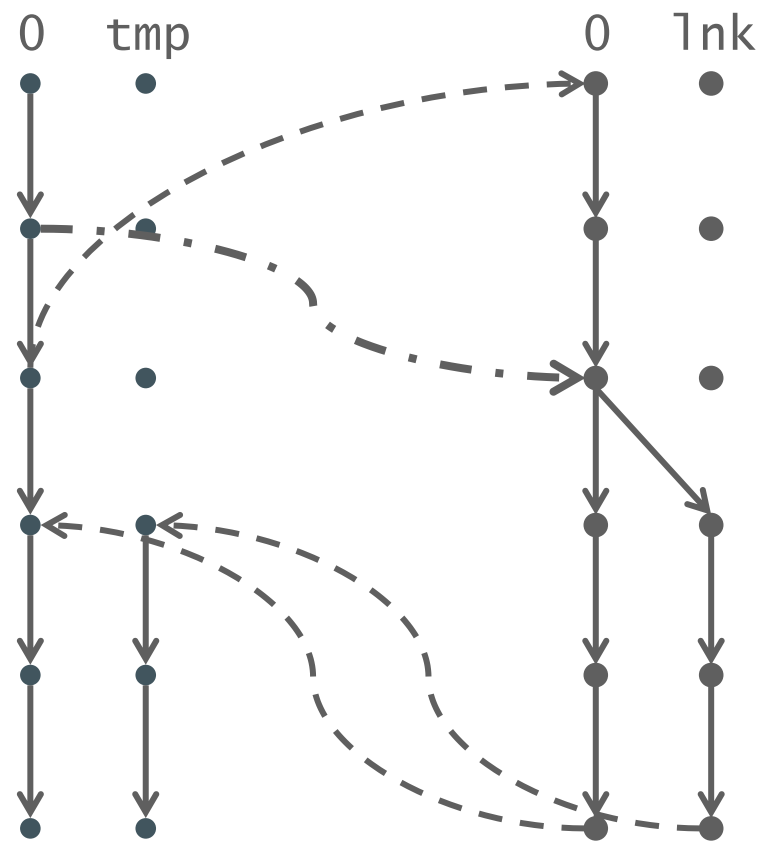


```
start  genForm()
f.url = getParam(“url”);
call    %tmp = getLink(f);
call-ret %tmp = getLink(f);
out.print(%tmp);
end  genForm()
```

```
start  getLink(Form form)
String lnk = “<a href=”;
lnk += form.url;
lnk += “>Click here</a>”;
return lnk;
end  getLink(Form form)
```

IFDS + Points-to

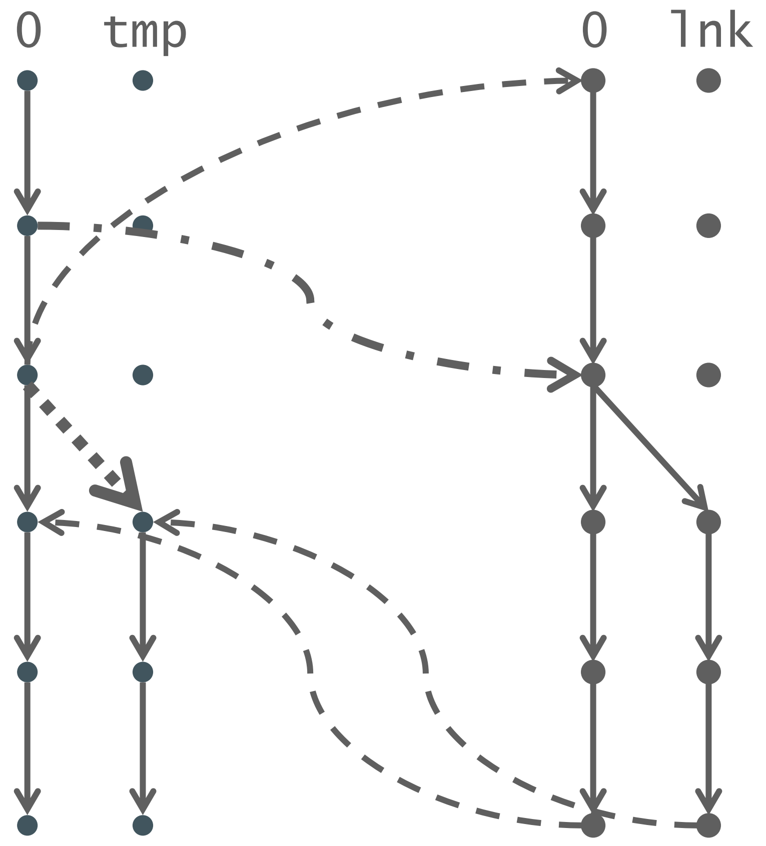
```
start  genForm()
f.url = getParam("url");
call    %tmp = getLink(f);
call-ret %tmp = getLink(f);
out.print(%tmp);
end  genForm()
```



```
start  getLink(Form form)
String lnk = "<a href=";
lnk += form.url;
lnk += ">Click here</a>";
return lnk;
end  getLink(Form form)
```

IFDS + Points-to

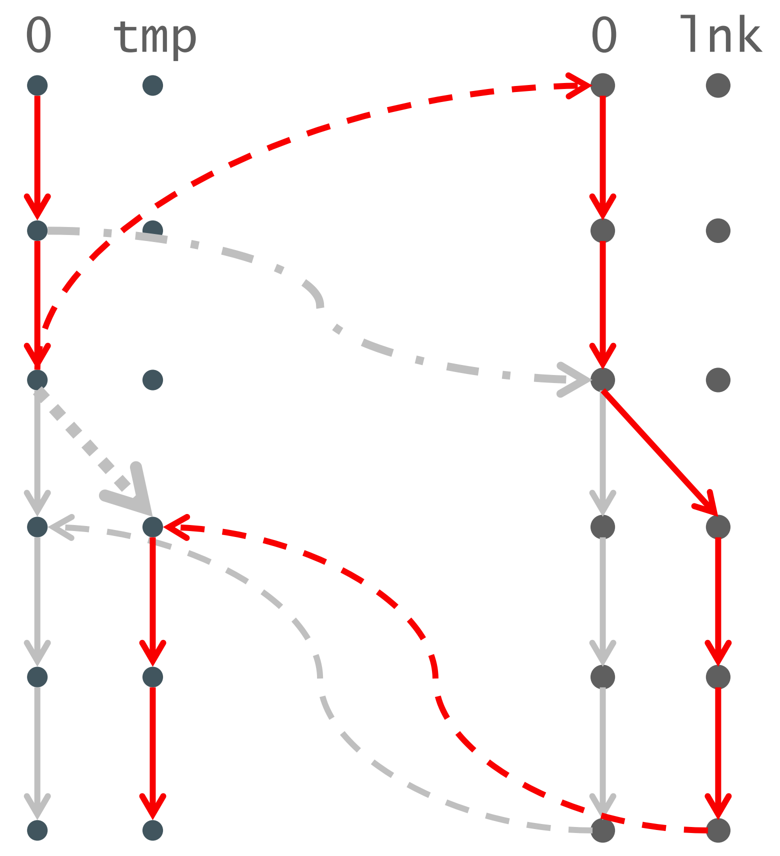
```
start  genForm()
f.url = getParam("url");
call    %tmp = getLink(f);
call-ret %tmp = getLink(f);
out.print(%tmp);
end  genForm()
```



```
start  getLink(Form form)
String lnk = "<a href=";
lnk += form.url;
lnk += ">Click here</a>";
return lnk;
end  getLink(Form form)
```

IFDS + Points-to

```
start  genForm()
f.url = getParam("url");
call    %tmp = getLink(f);
call-ret %tmp = getLink(f);
out.print(%tmp);
end  genForm()
```



```
start  getLink(Form form)
String lnk = "<a href=";
lnk += form.url;
lnk += ">Click here</a>";
return lnk;
end  getLink(Form form)
```

Implementation

- Implemented in Datalog, using the Soufflé engine.
- Pre-computed control-flow graph + CHA call graph.
- 2obj + 1heap points-to analysis.
- On-demand computation of flow edges.

Current Limitations

- IFDS is call-site sensitive
  - The analysis distinguishes between different calling contexts.
- Points-to analysis is object sensitive (2obj + 1heap).
  - **2 object:** On method calls, keep track of the allocation site of the receiver object and the allocation site of the object that allocated the receiver object.
  - **1 heap:** On allocation, keep track of the allocation site and the allocation site of the current object (this).
- Current implementation doesn't connect IFDS and points-to contexts.
  - Load-to-store aliases are **context-insensitive**.

Research Questions

RQ1: How does modelling of dynamic web application features impact precision and recall?

RQ2: How does points-to and context-sensitivity impact precision and recall?

RQ3: How does our approach compares to state-of-the-art?

Results — Don’t Panic, I’ll Walk You Through...

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	3	50%	3	50%	2	100%	3	50%	3	50%
blueblog	1	16%	3	30%	3	37%	1	16%	3	30%	3	37%
firingrange	4	100%	4	100%	4	100%	4	100%	4	100%	4	100%
gestcv	3	50%	3	50%	3	50%	3	50%	3	42%	3	50%
ginp	18	90%	19	57%	19	73%	18	90%	159	67%	159	69%
photov	42	100%	47	44%	48	100%	42	100%	53	44%	53	85%
securibench	86	91%	126	85%	125	86%	86	91%	126	85%	125	86%
webgoat	13	35%	239	55%	229	68%	13	35%	239	55%	230	68%

RQ1: Impact of Modelling

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	3	50%	3	50%	2	100%	3	50%	3	50%
blueblog	1	16%	3	30%	3	37%	1	16%	3	30%	3	37%
firingrange	4	100%	4	100%	4	100%	4	100%	4	100%	4	100%
gestcv	3	50%	3	50%	3	50%	3	50%	3	42%	3	50%
ginp	**18**	**90%**	19	57%	19	73%	**18**	**90%**	159	67%	159	69%
photov	**42**	**100%**	47	44%	48	100%	**42**	**100%**	53	44%	53	85%
securibench	86	91%	126	85%	125	86%	86	91%	126	85%	125	86%
webgoat	13	35%	239	55%	229	68%	13	35%	239	55%	230	68%

RQ1: Impact of Modelling

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	3	50%	3	50%	2	100%	3	50%	3	50%
blueblog	1	16%	3	30%	3	37%	1	16%	3	30%	3	37%
firingrange	4	100%	4	100%	4	100%	4	100%	4	100%	4	100%
gestcv	3	50%	3	50%	3	50%	3	50%	3	42%	3	50%
ginp	18	90%	**19**	**57%**	19	73%	18	90%	**159**	**67%**	159	69%
photov	42	100%	**47**	**44%**	48	100%	42	100%	**53**	**44%**	53	85%
securibench	86	91%	126	85%	125	86%	86	91%	126	85%	125	86%
webgoat	13	35%	239	55%	229	68%	13	35%	239	55%	230	68%

RQ1: Impact of Modelling

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	3	50%	3	50%	2	100%	3	50%	3	50%
blueblog	1	16%	3	30%	3	37%	1	16%	3	30%	3	37%
firingrange	4	100%	4	100%	4	100%	4	100%	4	100%	4	100%
gestcv	3	50%	3	50%	3	50%	3	50%	3	42%	3	50%
ginp	18	90%	19	57%	**19**	**73%**	18	90%	159	67%	**159**	**69%**
photov	42	100%	47	44%	**48**	**100%**	42	100%	53	44%	**53**	**85%**
securibench	86	91%	126	85%	125	86%	86	91%	126	85%	125	86%
webgoat	13	35%	239	55%	229	68%	13	35%	239	55%	230	68%

RQ1: Summary

RQ1: How does modelling of dynamic web application features impact precision and recall?

When used with points-to, our current modelling improves recall but might decrease precision.

RQ2: Impact of Points-To

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	3	50%	**3**	50%	2	100%	3	50%	3	50%
blueblog	1	16%	3	30%	**3**	37%	1	16%	3	30%	3	37%
firingrange	**4**	100%	4	100%	**4**	100%	4	100%	4	100%	4	100%
gestcv	**3**	50%	3	50%	**3**	50%	3	50%	3	42%	3	50%
ginp	18	90%	19	57%	**19**	73%	18	90%	159	67%	159	69%
photov	42	100%	47	44%	**48**	100%	42	100%	53	44%	53	85%
securibench	86	91%	126	85%	**125**	86%	86	91%	126	85%	125	86%
webgoat	13	35%	239	55%	**229**	68%	13	35%	239	55%	230	68%

RQ2: Impact of Points-To

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	**100%**	3	50%	3	**50%**	2	100%	3	50%	3	50%
blueblog	1	**16%**	3	30%	3	**37%**	1	16%	3	30%	3	37%
firingrange	4	**100%**	4	100%	4	**100%**	4	100%	4	100%	4	100%
gestcv	3	**50%**	3	50%	3	**50%**	3	50%	3	42%	3	50%
ginp	18	**90%**	19	57%	19	**73%**	18	90%	159	67%	159	69%
photov	42	**100%**	47	44%	48	**100%**	42	100%	53	44%	53	85%
securibench	86	**91%**	126	85%	125	**86%**	86	91%	126	85%	125	86%
webgoat	13	**35%**	239	55%	229	**68%**	13	35%	239	55%	230	68%

RQ2: Impact of Context Sensitivity

Applications	Without JEE support						With JEE support					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	**3**	50%	**3**	50%	2	100%	3	50%	3	50%
blueblog	1	16%	**3**	30%	**3**	37%	1	16%	3	30%	3	37%
firingrange	4	100%	**4**	100%	**4**	100%	4	100%	4	100%	4	100%
gestcv	3	50%	**3**	50%	**3**	50%	3	50%	3	42%	3	50%
ginp	18	90%	**19**	57%	**19**	73%	18	90%	159	67%	159	69%
photov	42	100%	**47**	44%	**48**	100%	42	100%	53	44%	53	85%
securibench	86	91%	**126**	85%	**125**	86%	86	91%	126	85%	125	86%
webgoat	13	35%	**239**	55%	**229**	68%	13	35%	239	55%	230	68%

RQ2: Impact of Context Sensitivity

Applications	**Without JEE support**						**With JEE support**					
	No points-to		Context-insensitive points-to		Context-sensitive points-to		No points-to		Context-insensitive points-to		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	2	100%	3	**50%**	3	**50%**	2	100%	3	50%	3	50%
blueblog	1	16%	3	**30%**	3	**37%**	1	16%	3	30%	3	37%
firingrange	4	100%	4	**100%**	4	**100%**	4	100%	4	100%	4	100%
gestcv	3	50%	3	**50%**	3	**50%**	3	50%	3	42%	3	50%
ginp	18	90%	19	**57%**	19	**73%**	18	90%	159	67%	159	69%
photov	42	100%	47	**44%**	48	**100%**	42	100%	53	44%	53	85%
securibench	86	91%	126	**85%**	125	**86%**	86	91%	126	85%	125	86%
webgoat	13	35%	239	**55%**	229	**68%**	13	35%	239	55%	230	68%

RQ2: Summary

RQ2: How does points-to and context-sensitivity impact precision and recall?

Adding points-to analysis definitely improves recall and usually improves precision.

Adding context-sensitivity definitely improves precision.

State-Of-The-Art

- **TAJ: Effective Taint Analysis of Web Applications (PLDI 2009)**
  - Hybrid thin slicing (IFDS on the value flow graph + context-insensitive points-to analysis).
- **Andromeda: Accurate and Scalable Security Analysis of Web Applications (FASE 2013)**
  - Data-flow taint analysis with on-demand access path computation.
- **FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps (PLDI 2014)**
  - IFDS based taint analysis with on-demand access path computation.

ORACLE

RQ3: Comparison with State-Of-The-Art

Applications	TAJ (2009)		Andromeda (2013)		Our approach	
	Context-insensitive points-to		On-demand access-path computation		Context-sensitive points-to	
	# TP	Precision	# TP	Precision	# TP	Precision
blojsom	—	—	**83**	**60%**	3	50%
blueblog	6	50%	**13**	**100%**	3	37%
gestcv	4	50%	**53**	**60%**	3	50%
ginp	—	—	49	40%	**159**	**69%**
photov	—	—	18	10%	**53**	**85%**
webgoat	35	90%	41	60%	**230**	**68%**

Summary

- Taint analysis can detect several types of flaws in web applications.
- Modelling of dynamic features can dramatically improve recall.
- Context-sensitive points-to + modelling yields best results.

Integrated Cloud

Applications & Platform Services