



INDIANA UNIVERSITY

Microsoft®
Research

InteGuard: Toward Automatic Protection of Third-Party Web Service Integrations

**Luyi Xing (Indiana University)
Yangyi Chen (Indiana University)
XiaoFeng Wang (Indiana University)
Shuo Chen (Microsoft Research)**



INDIANA UNIVERSITY






Microsoft®
Research

INTRODUCTION

Introduction

- Web applications integrate third-party Web services.

\$109.99 - 2-Year Computer & Tablet Accidental Damage Plan ADD to Cart

 <p>Western Digital My Book Live 1TB External Home Network Hard Drive [WD BACG0010HCH] \$119.99 FREE Free Shipping ADD to Cart</p>	 <p>Belkin BZ103050TVL Mini Surge Protector and USB Charger [BKN BZ103050TVL] \$16.99 ADD to Cart</p>	 <p>Satechi ST-R1 Arm Hinge Stand [HCE STR1] \$36.99 ADD to Cart</p>	 <p>Western Digital My Book Live 2TB External Home Network Hard Drive [WD BACG0020HCH] \$129.99 FREE Free Shipping ADD to Cart</p>	 <p>Western Digital My Book Live 3TB External Home Network Hard Drive [WD BACG0030HCH] \$154.99 FREE Free Shipping ADD to Cart</p>
---	--	---	--	---

rewards since you came to us directly! [Learn more](#)

Other Checkout Options

Expected to Ship **Mon 2/25**
You may only earn or redeem J&R Rewards by clicking Checkout button above.

Pay with amazon
Promotion cannot be applied to this Checkout Option

Buy with Google

Promotion cannot be applied to this Checkout Option

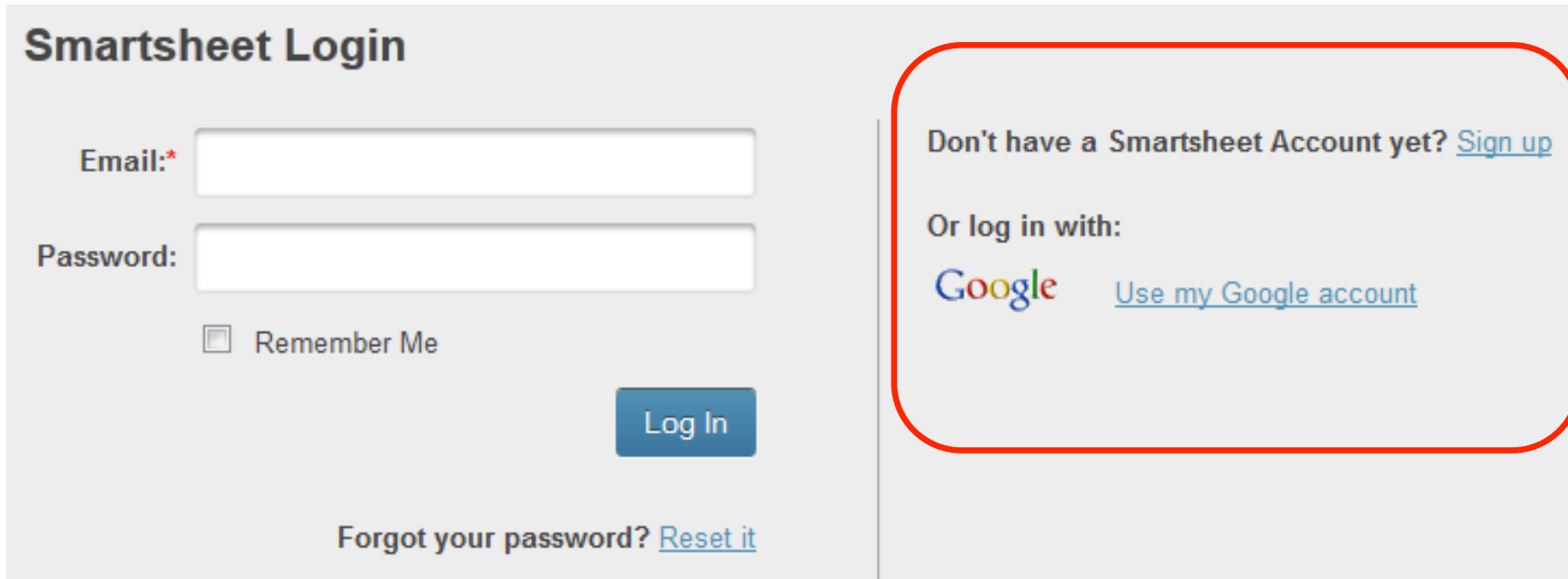
Check out with PayPal
The safer, easier way to pay

• At this time, we ship to the U.S., U.S. Territories, Puerto Rico, Canada, and APO/FPO only (your Billing address can be elsewhere; some products can only be shipped to the 50 states). [Click here](#) for details.

No Payments! **BillMeLater**
+ No Interest if paid in full in 6 Months
On orders over \$399.
Subject to credit approval. [See Terms](#)

Introduction

- Web applications integrate third-party Web services.



Smartsheet Login

Email:*


Password:

Remember Me

Forgot your password? [Reset it](#)

Don't have a Smartsheet Account yet? [Sign up](#)

Or log in with:

 [Use my Google account](#)

Introduction-cont.

- Security challenge: coordinate **Website (Integrator)**, Service Provider and Web Client.
- Integrator error-prone, difficult to be secure ([Oakland'11, Oakland'12]).

The image displays a collection of logos for various e-commerce and payment services. On the left, there are buttons for 'Buy with Google', 'Check out with PayPal', and 'Connect with facebook.'. Below these is the 'Rakuten.com Shopping Formerly Buy.com' logo. In the center is a 'Sign in using your account with' dialog box showing options for Facebook, Google, PayPal Access, Windows Live ID, Aol., and OpenID, with a 'Powered by Janrain' note at the bottom. On the right, there is a 'J&R' logo with 'FREE SHIPPING On Orders Over \$99!' and a truck icon, followed by the 'nopCommerce' logo (open source e-commerce solution), and finally the 'janrain' and 'interspire' logos (Web software inspired by you).



Introduction-Cont.

- Protection
 - Integrator side more error-prone.
 - Traffic among integrator, provider and clients is generally mechanic.



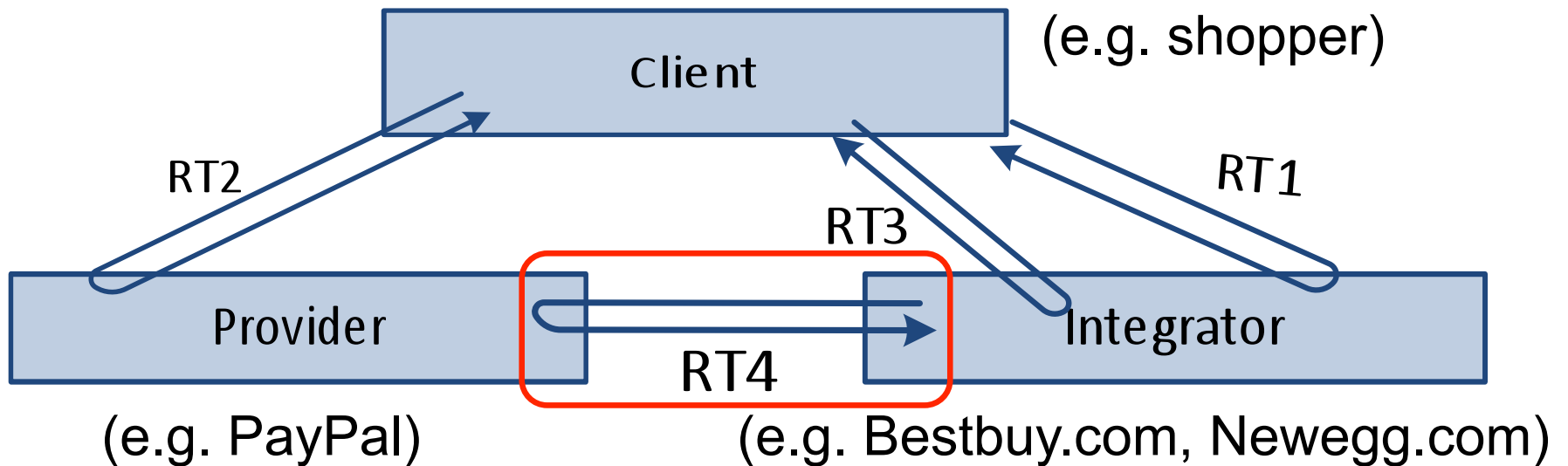
INDIANA UNIVERSITY

Microsoft®
Research

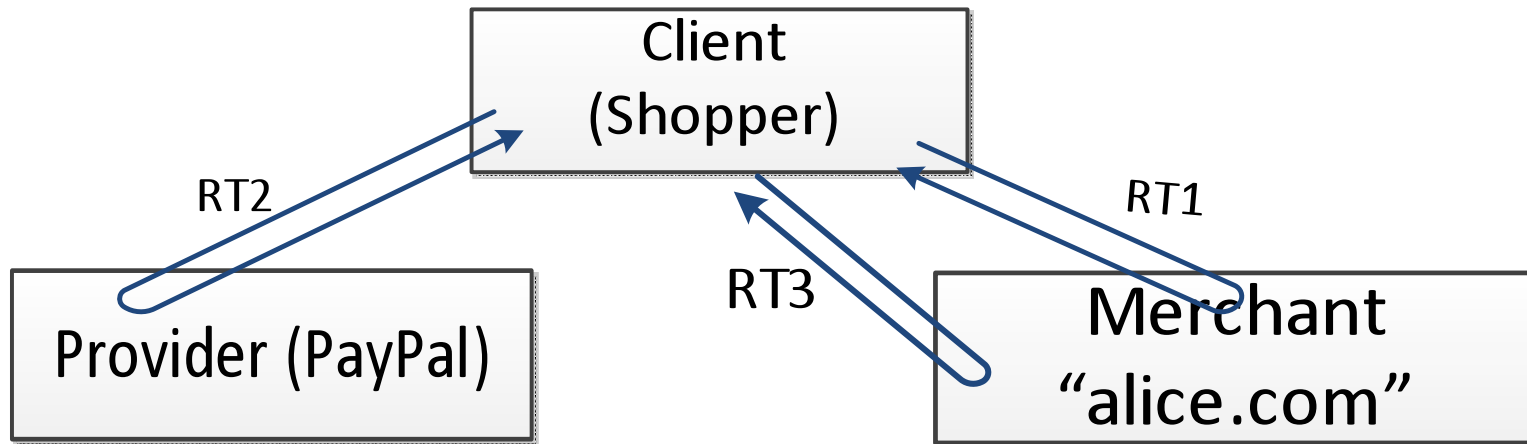
BACKGROUND

Background

- Third-Party Web Service Integration
 - RT
(HTTP request/response pair, or HTTP Round Trip)



Logic Flaws



RT1.request = https://Jeff.com/placeOrder

RT2.request = https://PayPal/pay?amount=9.99&&orderID=123
AccountId=alice&sig=cde

RT3.request = https://alice.com/finishOrder?gross=9.99&orderID=123

AccountId = bob

**payeeEmail =
bob@email.com**

&payeeEmail=alice@email.com

Previous Research

- **Web Logic Flaws.**
 - Swift, Ripley, Swaddler, MiMoSA, Waler, Rolecast, Execution After Redirect, SAFERPHP, WAPTEC, APP_LogGIC, Fix_Me_Up, NoTamper, Block
- **Conventional two-party settings** (websites, clients).



Adversary Model

- Logic flaws in Integration
- Service provider is trusted
- Client is not trusted



Contribution

- Integuard
 - First step toward automatic and generic protection of Web service integrations.
 - New challenges in multiple-party settings.
 - Effective false positive control.
 - Evaluate with real exploits and performance test.
 - Practical protection.



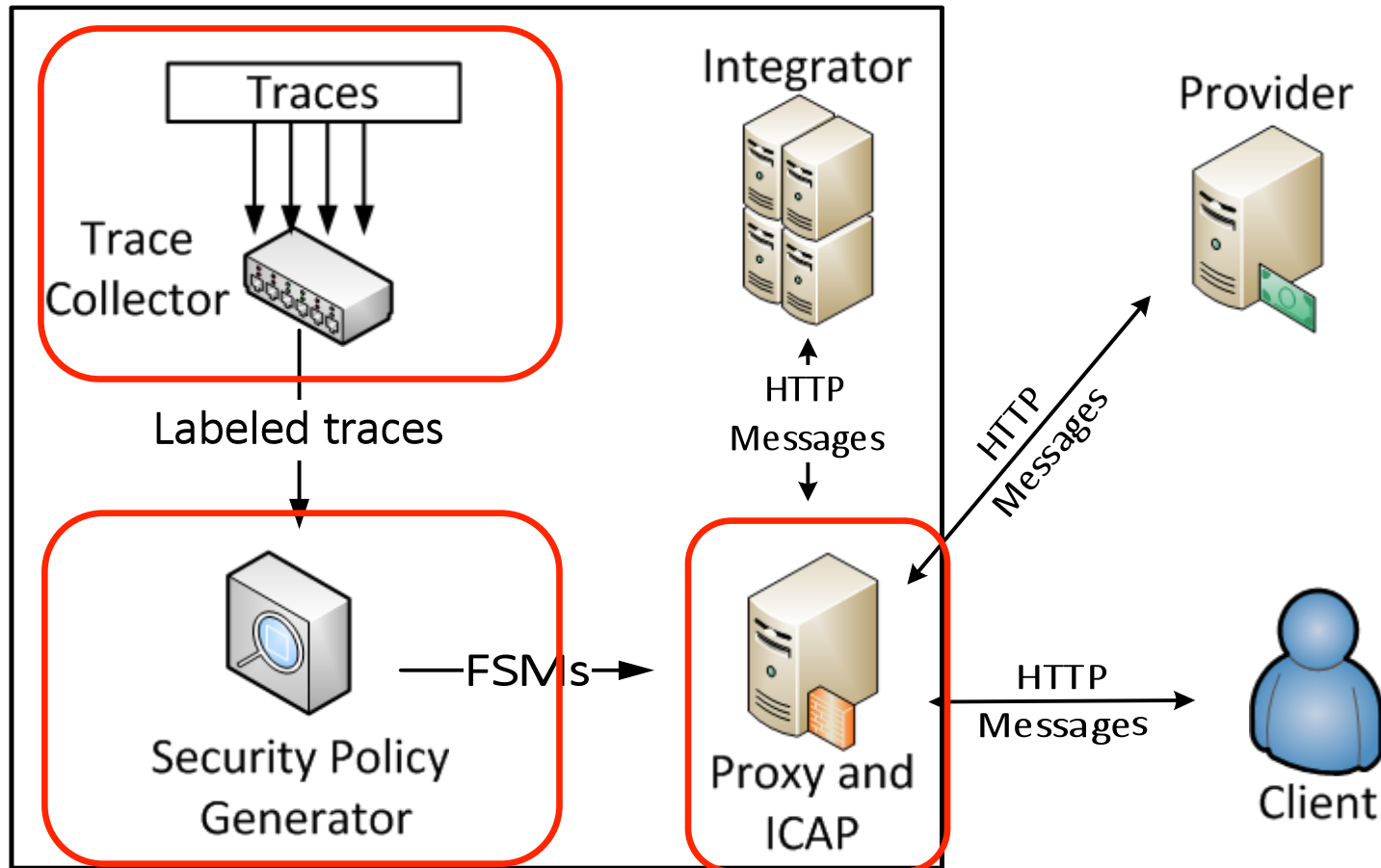
INDIANA UNIVERSITY

Microsoft®
Research

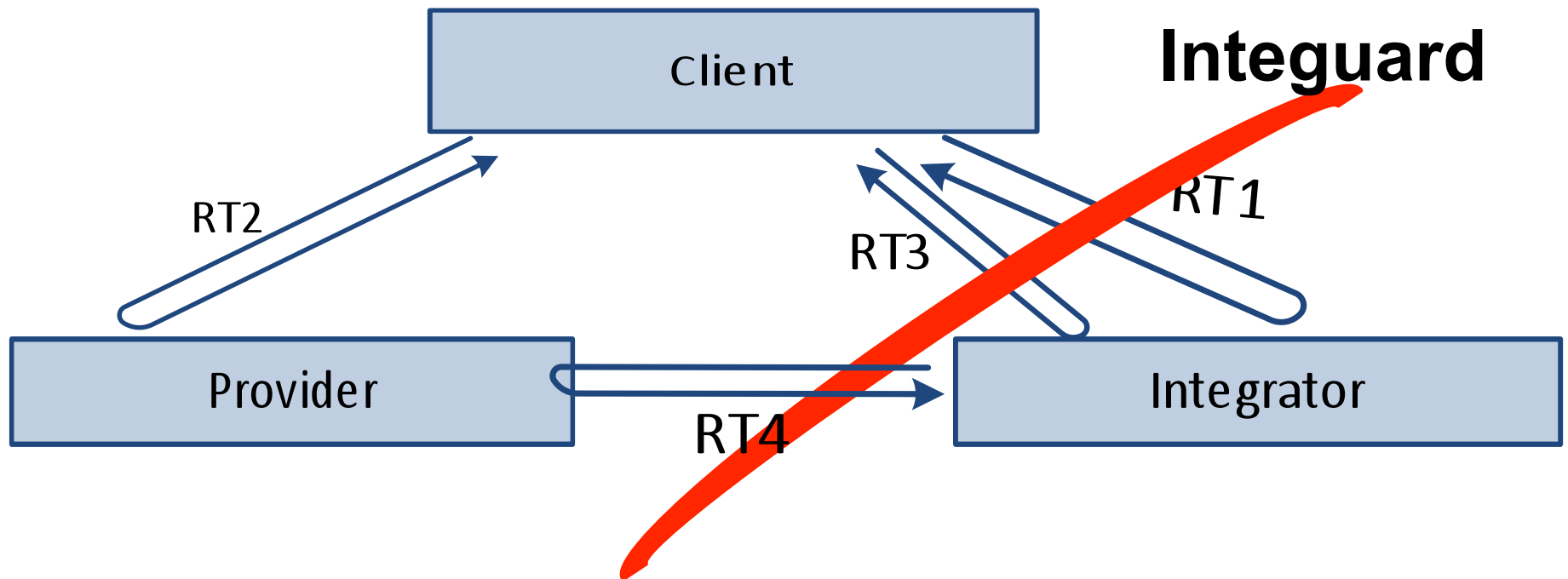
DESIGN

Design – Architecture

InteGuard



Design – Architecture





Design – Training Trace Collection

- Traces → Security Invariants
- Challenge
 - Random transactions for invariant extraction
→ **False Positive**

Design – Training Trace Collection

- Four traces
- Integrator opens two accounts at service provider,
e.g. open two PayPal Merchant accounts

**Integrator-1
(Merchant
Account - 1)**

Transaction 1
Transaction 2

**Integrator-2
(Merchant
Account – 2)**

Transaction 1

Different users

Different products

Different transactions

Different quantities

Same integrator

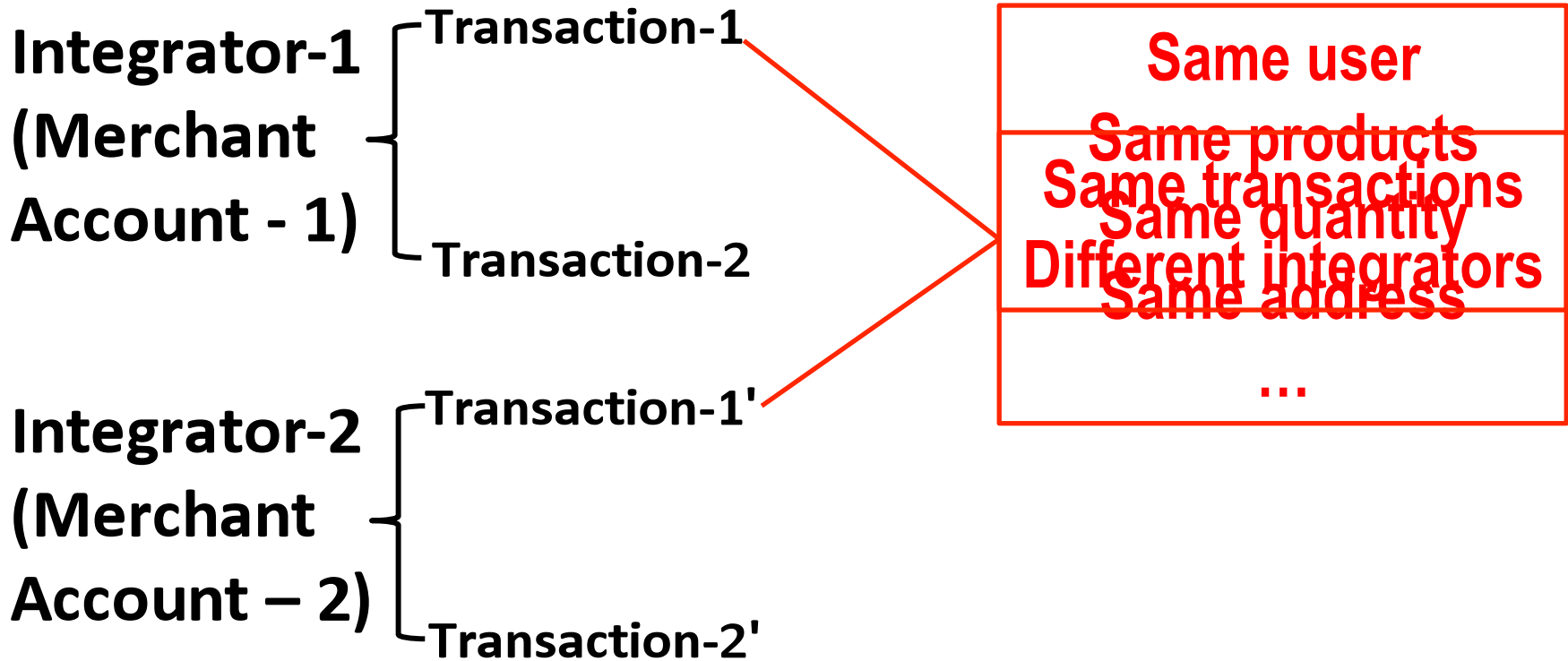
Integrator 1 and Integrator 2 use the same Web application, addresses configured with different ...

different merchant accounts



Design – Training Trace Collection

- Four traces



Design – Training Trace Collection

- Four traces

Integrator-1
(Merchant Account - 1)

Transaction-1
Transaction-2

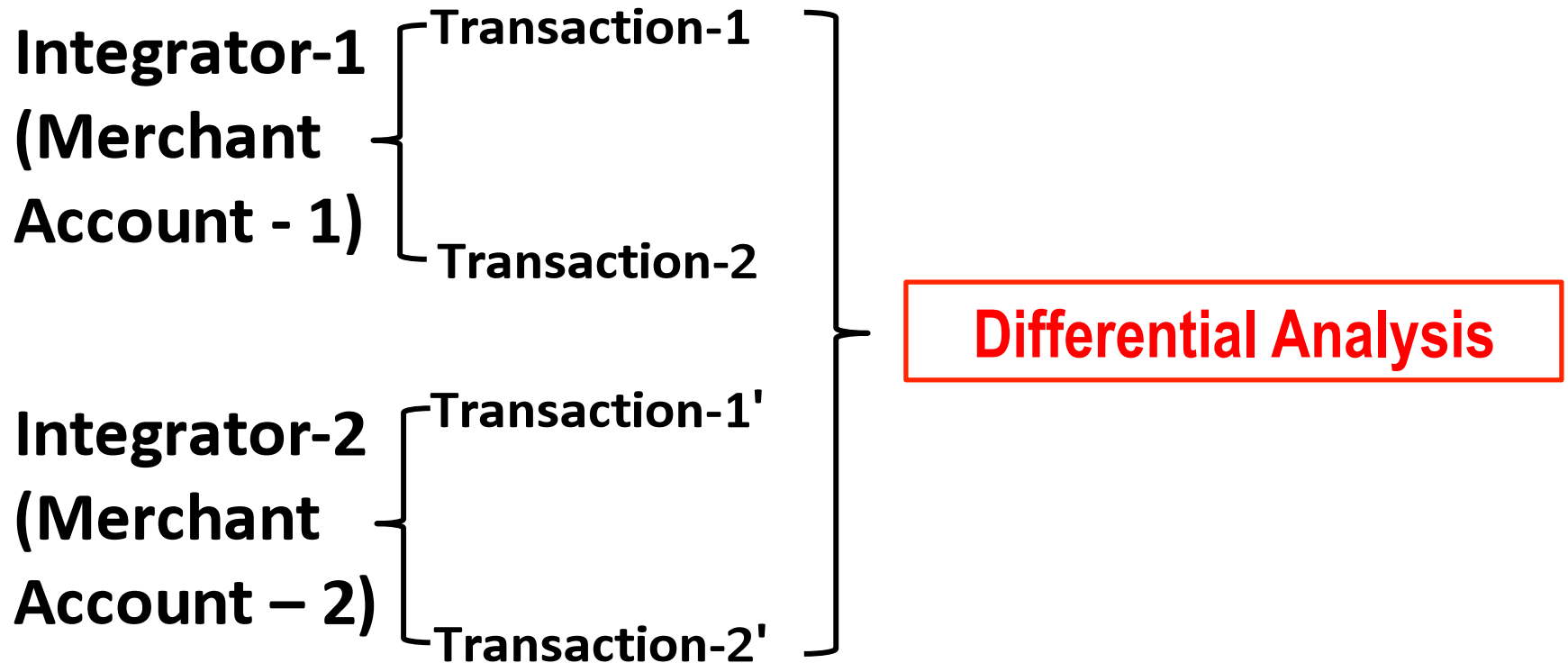
Integrator-2
(Merchant Account - 2)

Transaction-1'
Transaction-2'

Same Transactions
Different Integrators

Design – Training Trace Collection

- Four traces





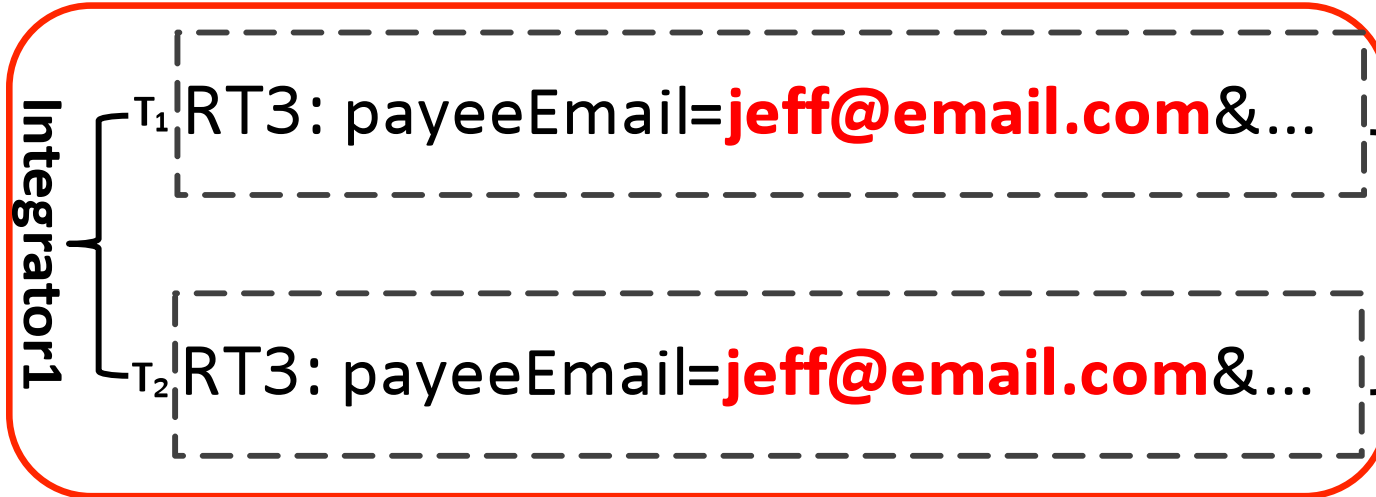
Design – Invariant Analysis

- Integrator-specific invariant
- Local Invariant
 - Transaction-specific invariant
- Other invariant
 - Start of transaction
 - End of transaction
 - API sequence



Design – Invariant Analysis

- **Integrator-specific invariant**
- Local Invariant
 - Transaction-specific invariant
- Other invariant
 - Start of transaction
 - End of transaction
 - API sequence

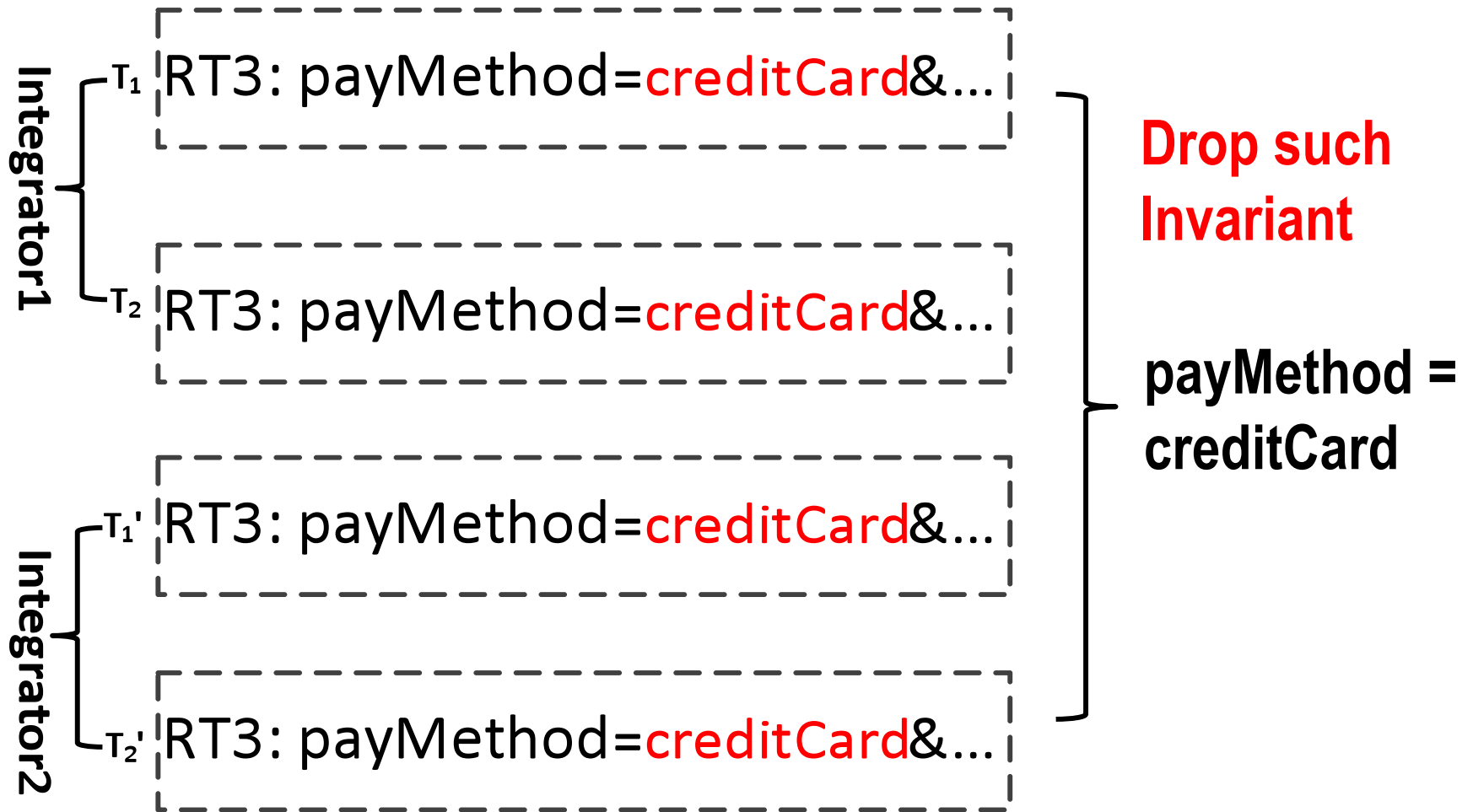


jeff@email.com
 Specific to
 Integrator 1
Integrator-

Specific
Invariant:
RT3.payeeEmail
 alice@email.com



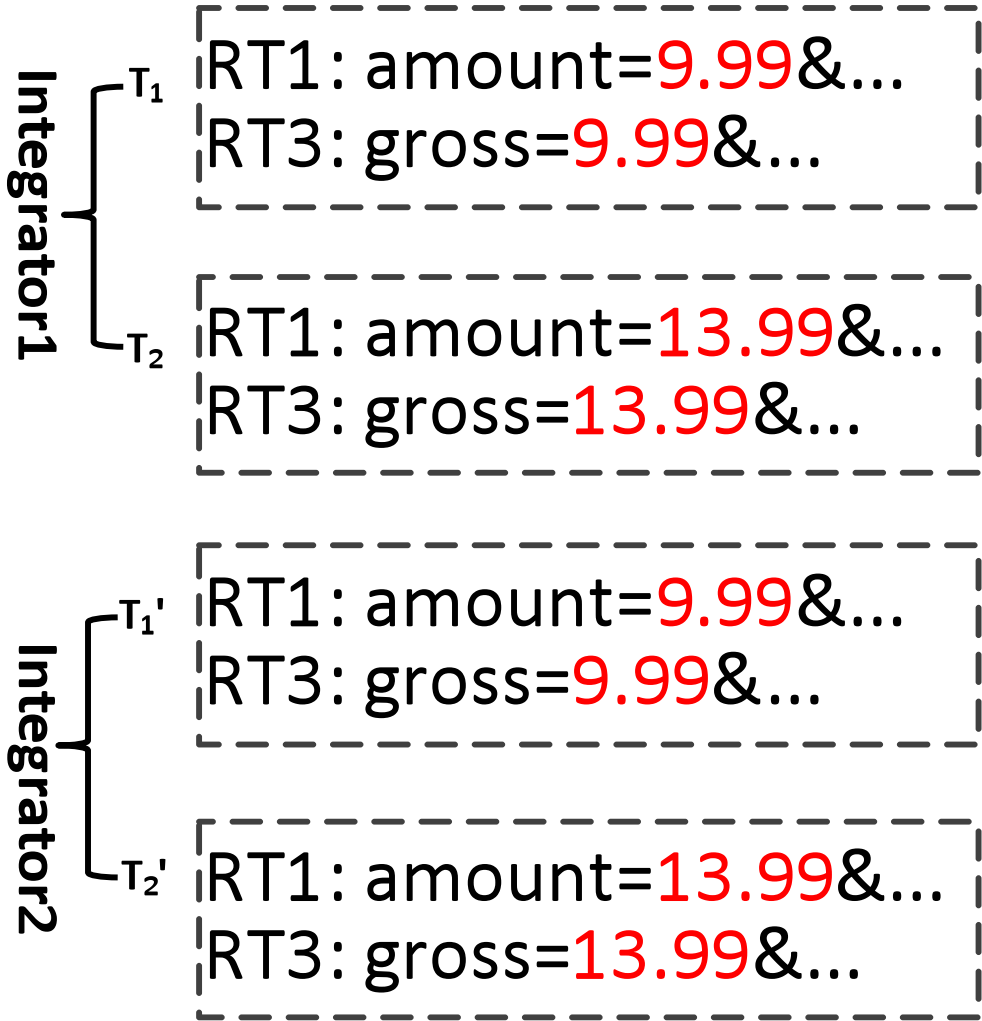
Specific to
 Integrator 2





Design – Invariant Analysis

- Integrator-specific invariant
- **Local Invariant**
 - Transaction-specific invariant
- Other invariant
 - Start of transaction
 - End of transaction
 - API sequence



amount = gross

amount = gross
Local Invariant:
amount == gross

amount = gross

amount = gross



Integrator1

T₁

RT1: returnFlag=1&...
RT3: status=1&...

T₂

RT1: returnFlag=1&...
RT3: status=1&...

Integrator2

T₁'

RT1: returnFlag=1&...
RT3: status=1&...

T₂'

RT1: returnFlag=1&...
RT3: status=1&...

returnFlag = status

returnFlag = status

Drop Invariant

With

Length < 3

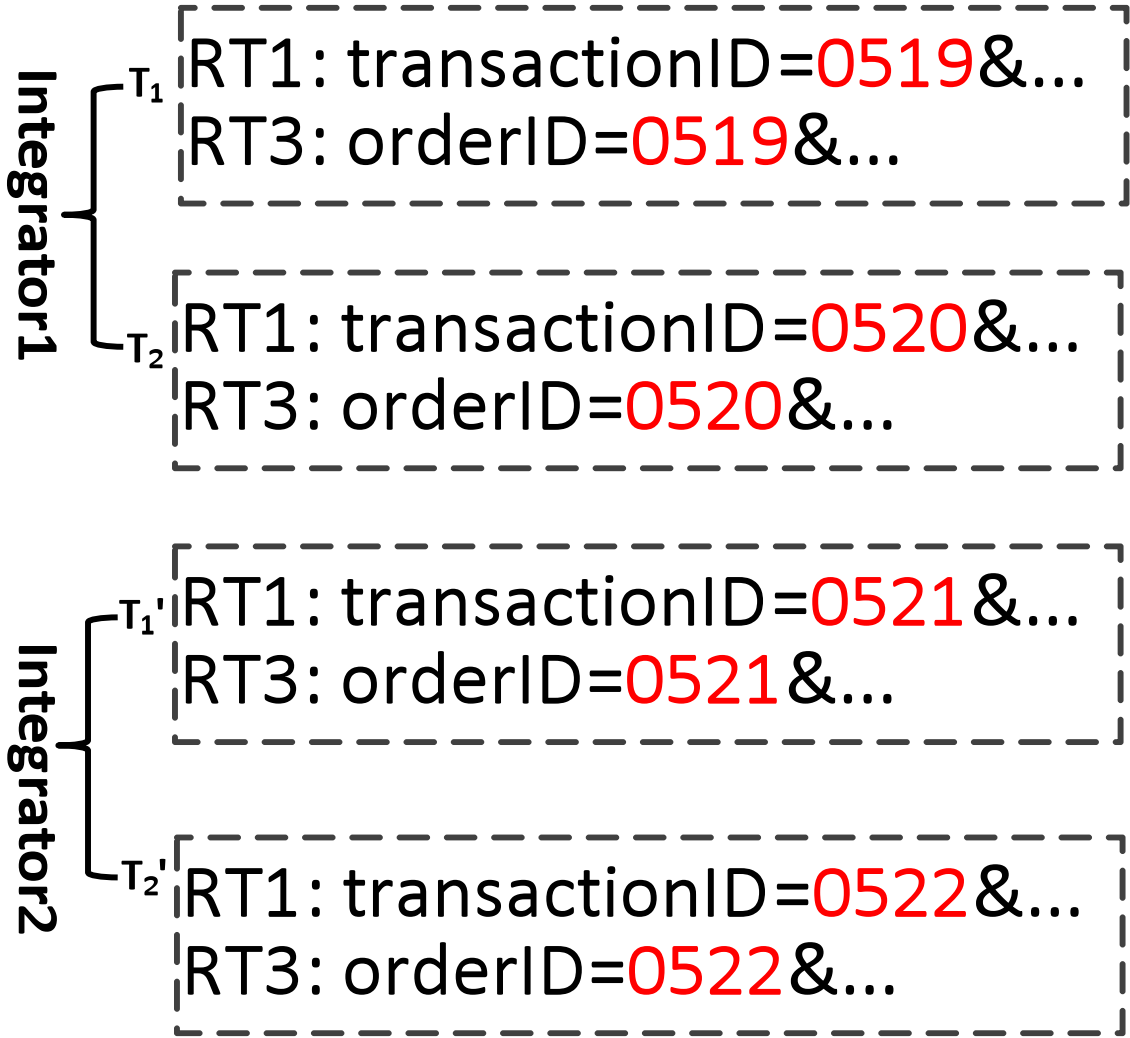
returnFlag = status

returnFlag = status

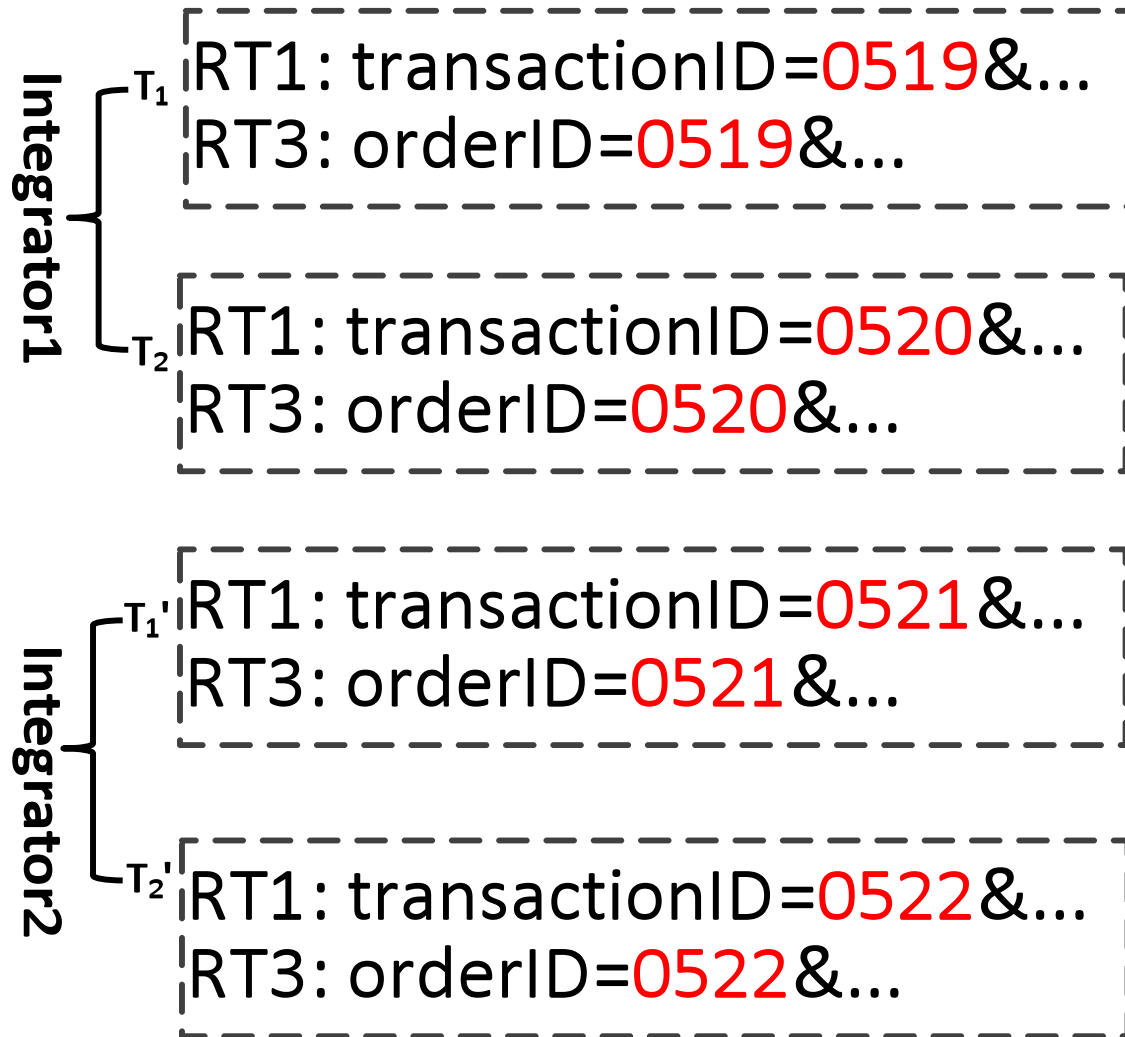


Design – Invariant Analysis

- Integrator-specific invariant
- Local Invariant
 - **Transaction-specific invariant**
- Other invariant
 - Start of transaction
 - End of transaction
 - API sequence



transactionID
==
orderID

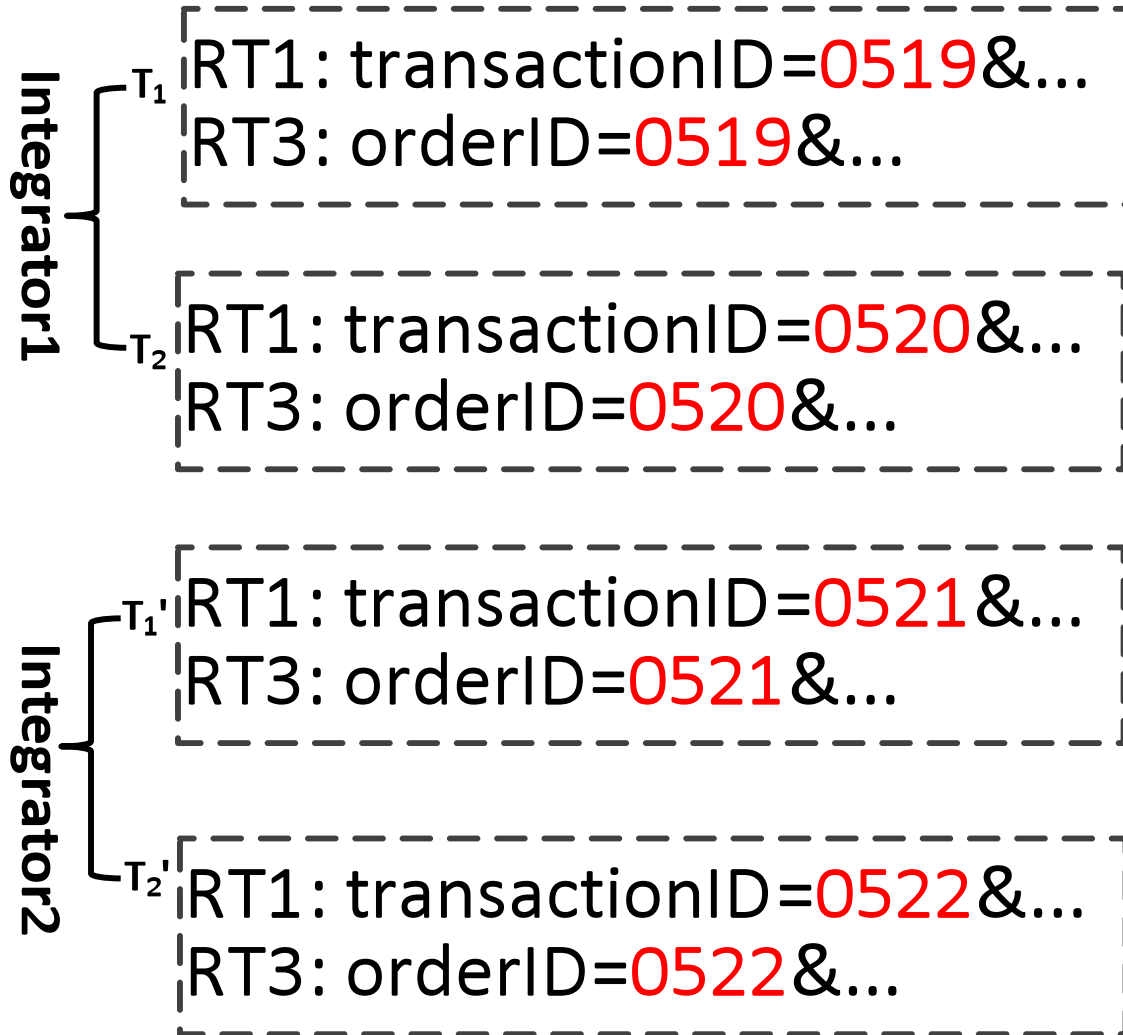


0519

0520
Specific to
each
transaction

0521

0522



Local Invariant



**Transaction
specific
invariant**



Integrator1

T₁

RT1: amount=9.99&...
RT3: gross=9.99&...

T₂

RT1: amount=13.99&...
RT3: gross=13.99&...

Integrator2

T₁'

RT1: amount=9.99&...
RT3: gross=9.99&...

T₂'

RT1: amount=13.99&...
RT3: gross=13.99&...

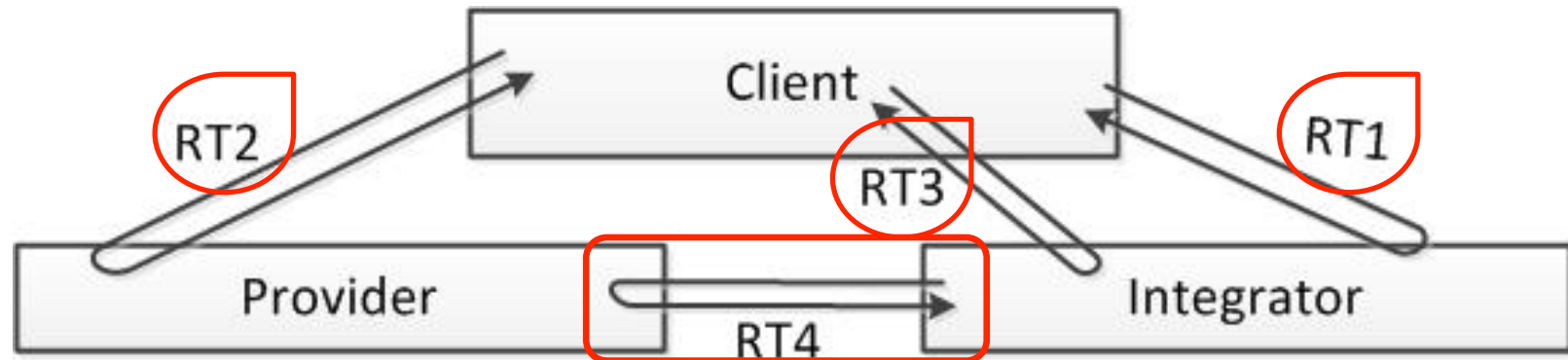
9.99

**Same transactions
Different integrators**

amount = gross

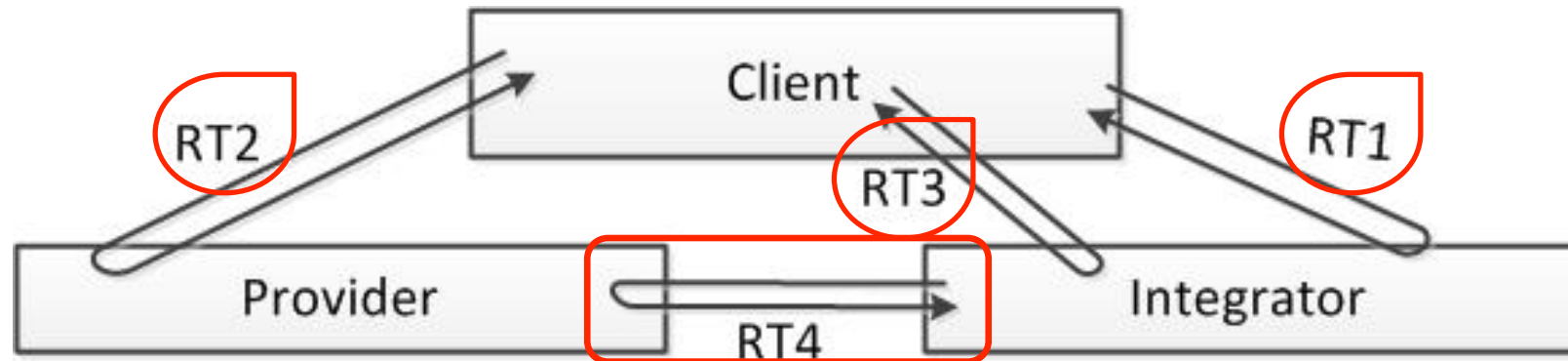
9.99

Transaction-specific Invariant



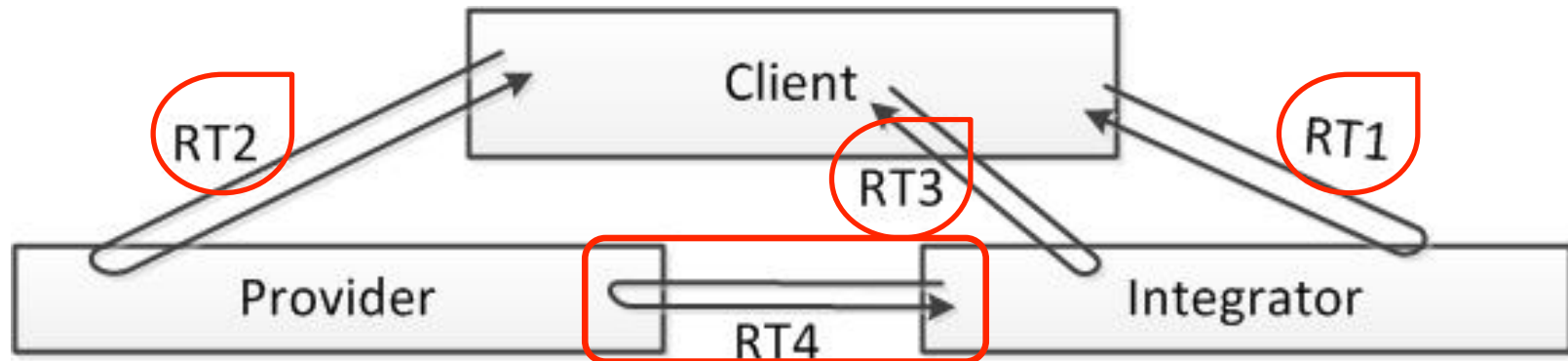
Is RT4 different?

Transaction-specific Invariant



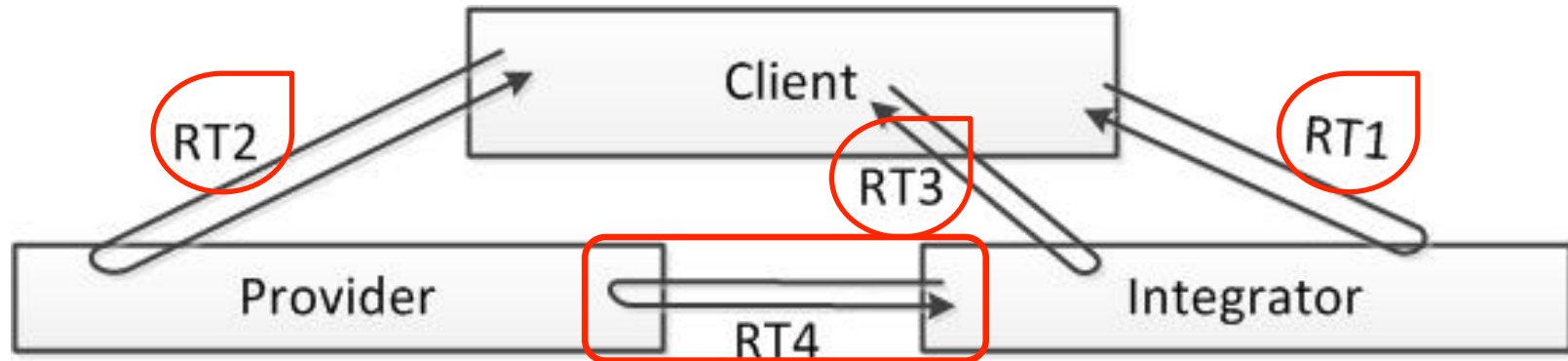
RT4 has no cookies

Transaction-specific Invariant



Which transaction does a RT4 belong to?

Transaction-specific Invariant



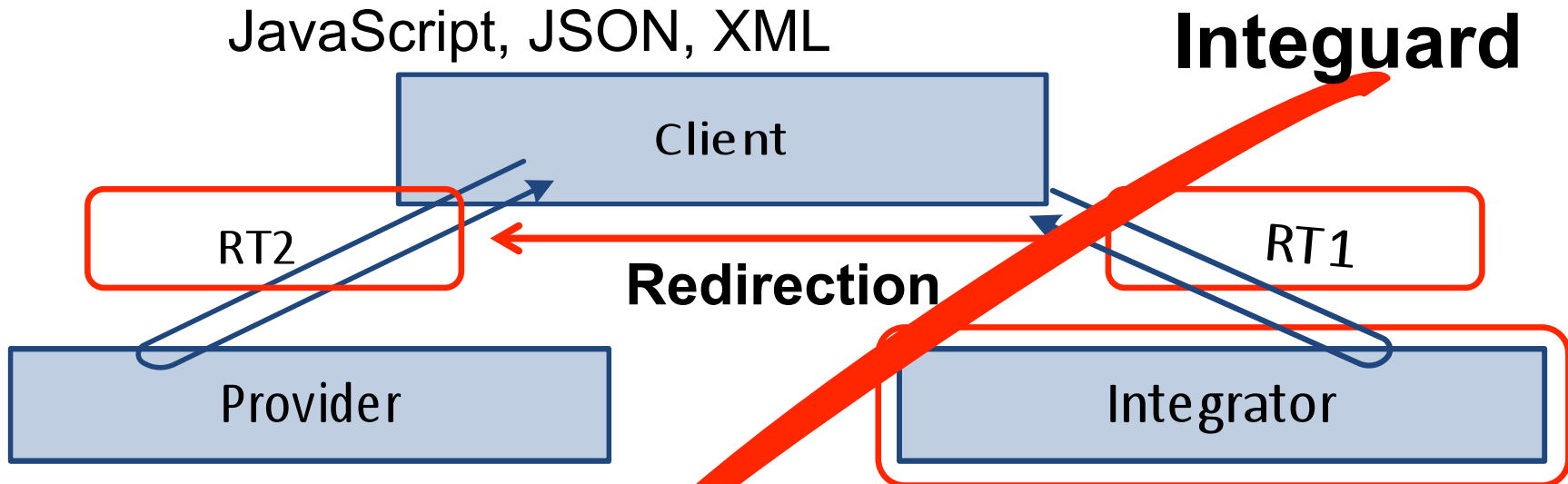
**Transaction-specific Invariants
help
Grouping RT4 into its belonging transaction**

Design – Invariant Analysis

- Local Invariant
- Integrator-specific invariant
 - Transaction-specific invariant
- **Other invariant**
 - **Start of transaction**
 - **End of transaction**
 - **API sequence**

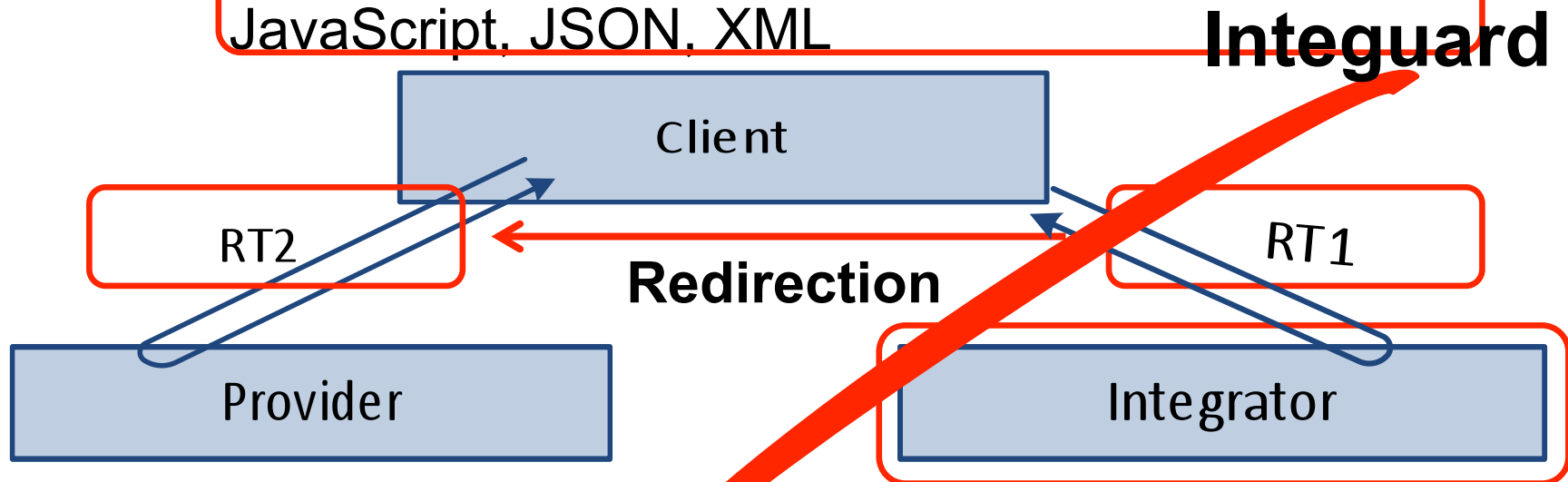
Design – Element Extraction

- Challenges
 - RT2 not observable
 - RT2 parameters in RT1's response
 - **Channels:** HTTP 3xx, meta refresh, HTML Form, JavaScript, JSON, XML



Design – Element Extraction

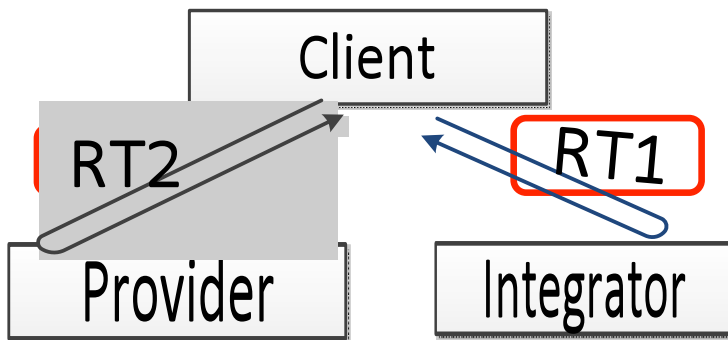
- Challenges
 - RT2 not observable
 - RT2 parameters in RT1's response
 - **Channels:** HTTP 3xx, meta refresh, HTML Form, JavaScript, JSON, XML





Design – Element Extraction

- RT2 parameters in



Request of RT2:

```
POST https://PayPal/pay?
accountId=MULW&amount=9.99&orderId=0519 &...
```

Trace parameters in RT1's response

HTML form in RT1's response:

```
<form id= "simplepay" name="SimplePay" method="POST"
action="https://PayPal/pay">
<input name=" AccountId" value="MULW">
<input name=" amount" value="9.99">
<input name=" orderId" value="0519">...</form>
```

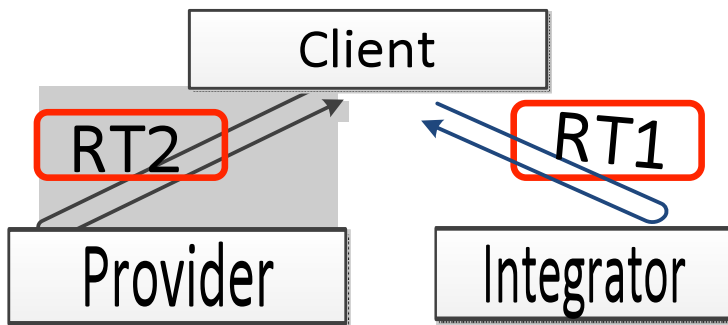
Extract a DOM path for each parameter

Extracted DOM path:

```
AccountId: form[id,name,action]->inputTag[AccountId]
Amount: form[id,name,action]->inputTag[amount]
orderId: form[id,name,action]->inputTag[orderId]
```

Record the DOM paths for each RT2's parameters

Design – Element Extraction



Don't parse all content of RT1

Request of RT2:
POST https://PayPal/pay?
accountId=MULW&**amount**=9.99&**orderId**=0519 &...

Trace parameters in RT1's response

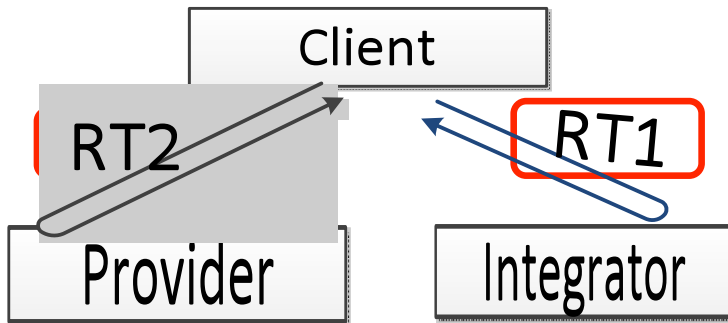
HTML form in RT1's response:
<form id="simplepay" name="SimplePay" method="POST" action="https://PayPal/pay">
<input name="**AccountId**" value="MULW">
<input name="**amount**" value="9.99">
<input name="**orderId**" value="0519">...</form>

Extract a DOM path for each parameter

Extracted DOM path:
AccountId: form[id,name,action]->inputTag[AccountId]
Amount: form[id,name,action]->inputTag[amount]
orderId: form[id,name,action]->inputTag[orderId]



Design – Element Extraction



Just extract desired parameters from known locations

Request of RT2:
 POST https://PayPal/pay?
accountId=MULW&**amount**=9.99&**orderId**=0519 &...

Trace parameters in RT1's response

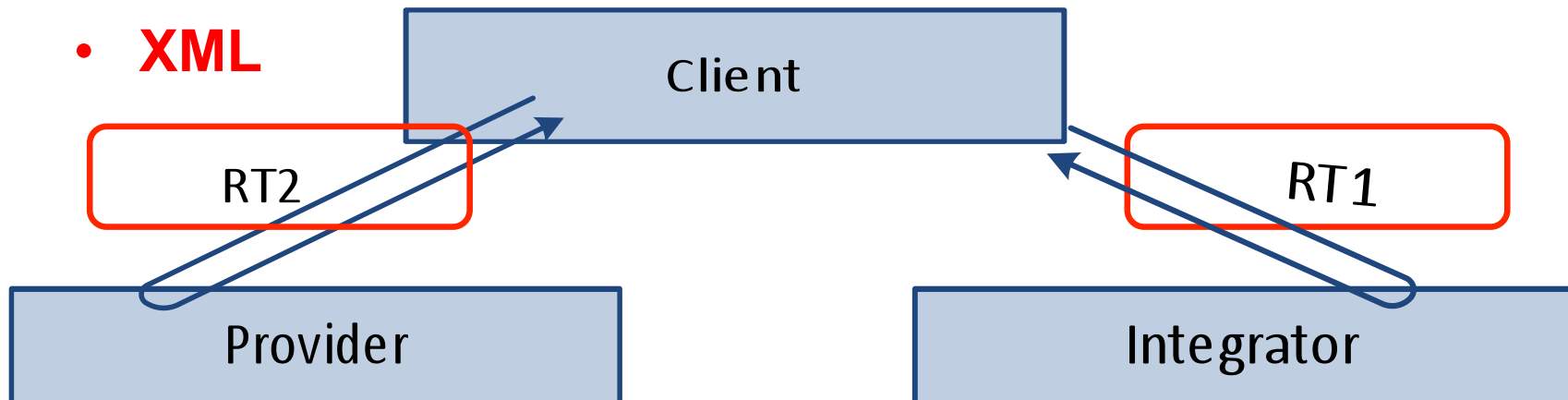
HTML form in RT1's response:
 <form id= "simplepay" name="SimplePay" method="POST"
 action="https://PayPal/pay">
 <input name="AccountId" value="MULW">
 <input name="amount" value="9.99">
 <input name="orderId" value="0519">...</form>

Extract a DOM path for each parameter

Extracted DOM path:
AccountId: form[id,name,action]->inputTag[AccountId]
Amount: form[id,name,action]->inputTag[amount]
orderId: form[id,name,action]->inputTag[orderId]

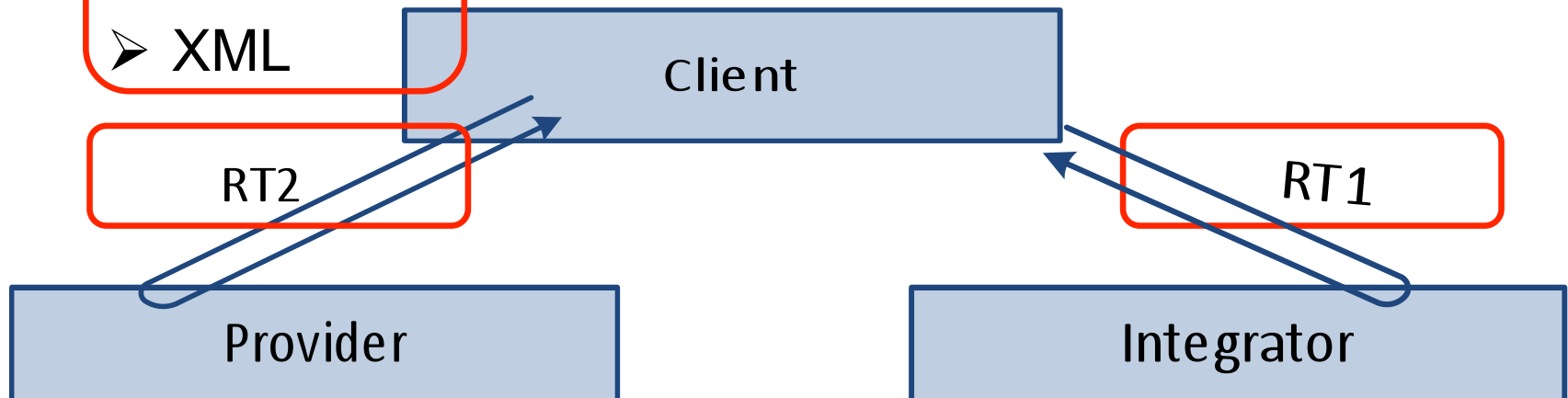
Design – Element Extraction

- HTTP 3xx
- Meta refresh
- **JavaScript**
 - **Abstract Syntax Tree (AST)**
- **JSON**
- **XML**



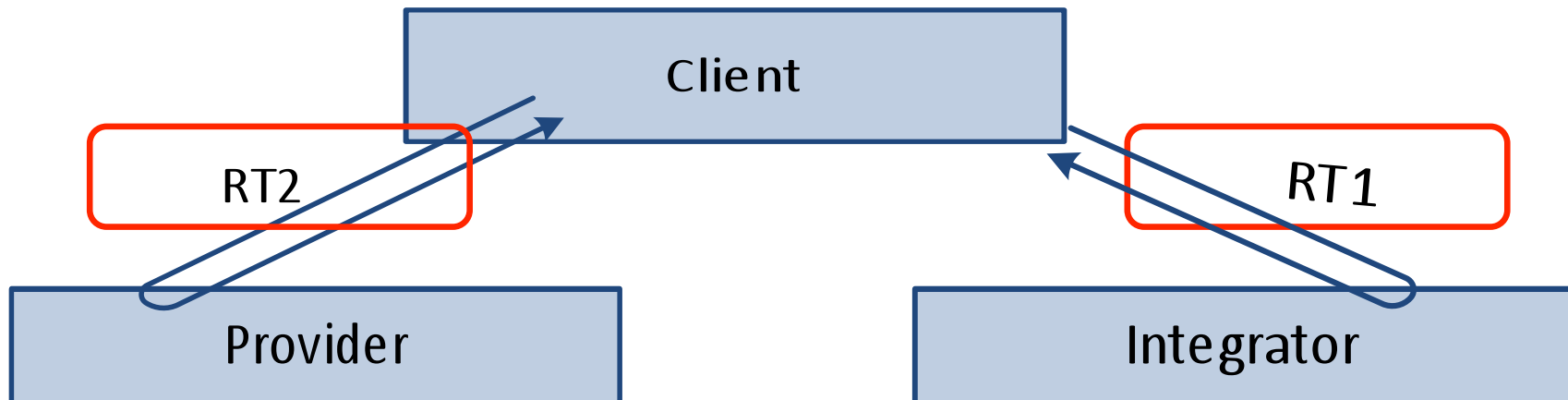
Design – Element Extraction

- RT2's parameters in RT1's response
 - HTTP 3xx
 - Meta refresh
 - JavaScript
 - JSON
 - XML



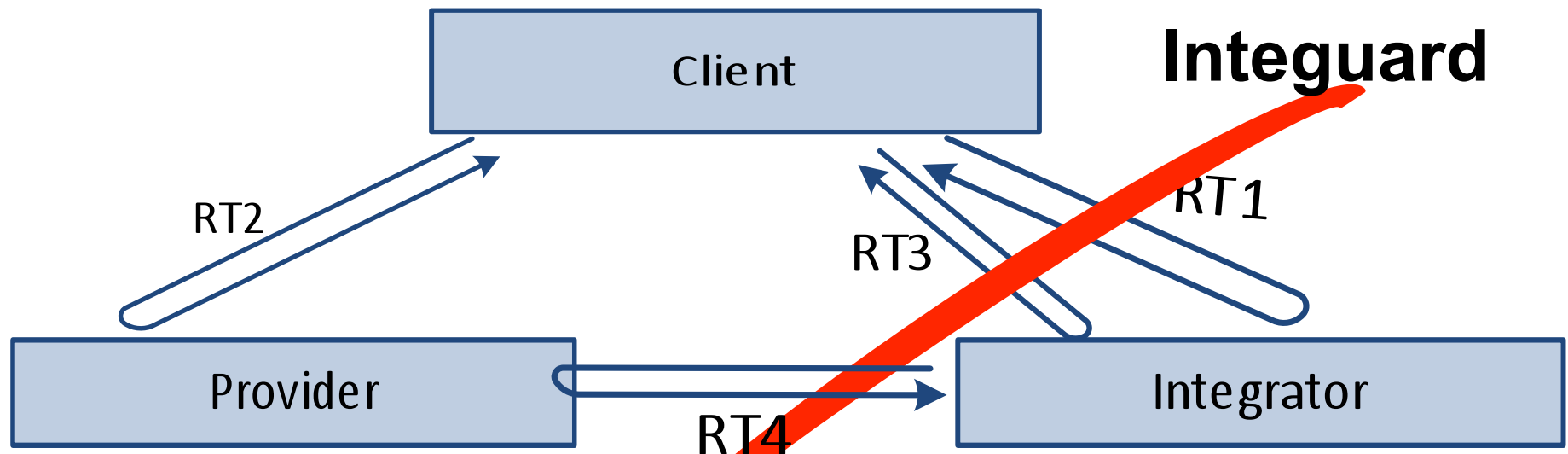
Design – Element Extraction

- **JavaScript**
 - **Abstract Syntax Tree (AST)**
 - **Mark parameters' locations**
- **JSON, XML**
 - **Tree structure, mark locations**



Design – Security Policy Enforcement

- Security invariants.
- **Intercept** HTTP traffic on integrator.
- **Runtime** detection of invariant.





INDIANA UNIVERSITY

Microsoft®
Research

EVALUATION

Evaluation

- Integrations
 - Web Shopping Cart applications with known vulnerabilities.
 - Intersipre starter edition 5.5.4
 - Nopcommerce v1.60
 - 5 faulty SSO integrations.
 - involving sears.com, janrain.com, Google, Facebook, PayPal



Effectiveness

Application	Service Integrated	Invariant type	Detected
Nopcommerce	PayPal Std	Local	Yes
Nopcommerce	Amazon Simple Pay	Integrator-specific	Yes
Nopcommerce	Amazon Simple Pay	Integrator-specific	Yes
Interspire	PayPal Std	Transaction-specific	Yes
Interspire	PayPal Exp	Local	Yes
Interspire	Google Checkout	API Sequence	Yes
Smartsheet.com	Google ID	Local	Yes
Janrain	Google ID	Local	Yes
Sears.com	Facebook SSO	Integrator-specific	Yes
Shopgecko.com	PayPal Access	Local	Yes
Farmville	Facebook SSO	N/A	No

False Positives

- Each CaaS integration, 100 to 300 checkouts.
- Each SSO integration, 20 checkouts.
- Altogether 1,000 real transactions.
 - Random user behaviors, clicking back button, returning through old URLs, etc.
 - Randomly crawl URLs.

→ No false alarms



Performance

- 32 to 256 (default MaxClients of Apache Web server) concurrent transactions.
- Negligible overhead (3.32%).
- Memory:
 - Almost constant 1,250 MB .
(32 to 256 concurrency)
 - 150MB difference.
(256 concurrency, with and without security check)



INDIANA UNIVERSITY

Microsoft®
Research

CONCLUSION



Conclusion

- First to protect vulnerable integrations of third-party Web services.
- New challenges in multi-party settings.
- Generate invariants through a suit of new techniques.
- Effective false positive control and low performance expense.

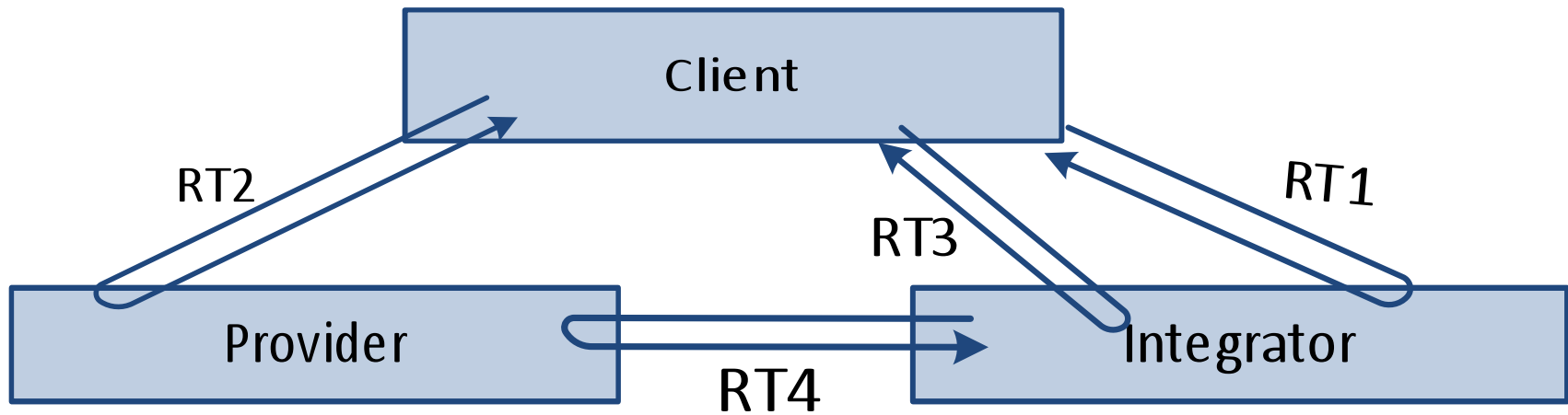


INDIANA UNIVERSITY

Microsoft®
Research

THANK YOU!

LUYI XING





ICAP

- The **Internet Content Adaptation Protocol** (ICAP) is a lightweight [HTTP](#)-like protocol which is used to extend transparent [proxy servers](#). ICAP is generally used to implement [virus scanning](#) and [content filters](#) (including [censorware](#)) in transparent HTTP proxy caches.
-