INDIANA UNIVERSITY

Microsoft Research

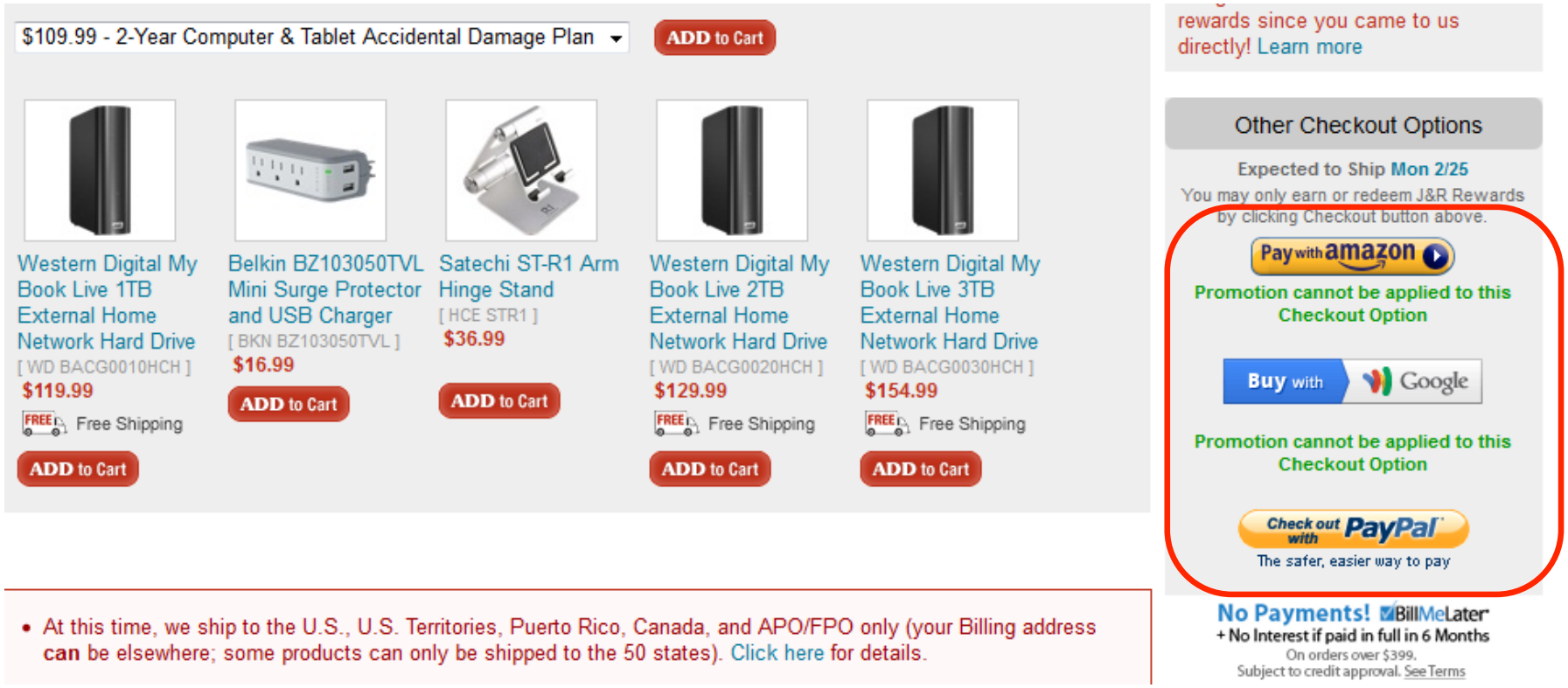
InteGuard: Toward Automatic Protection of Third-Party Web Service Integrations

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

INTRODUCTION

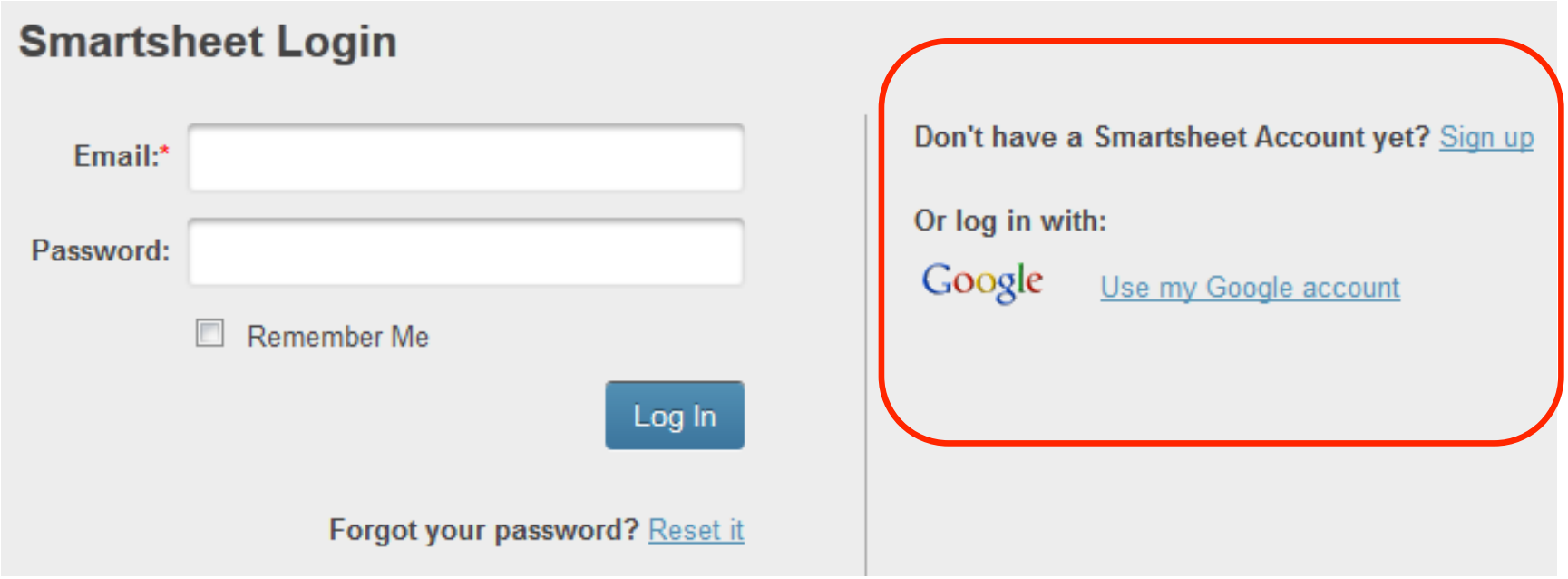
Introduction

- Web applications integrate third-party Web services.



Introduction

- Web applications integrate third-party Web services.



Introduction-cont.

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

Introduction-Cont.

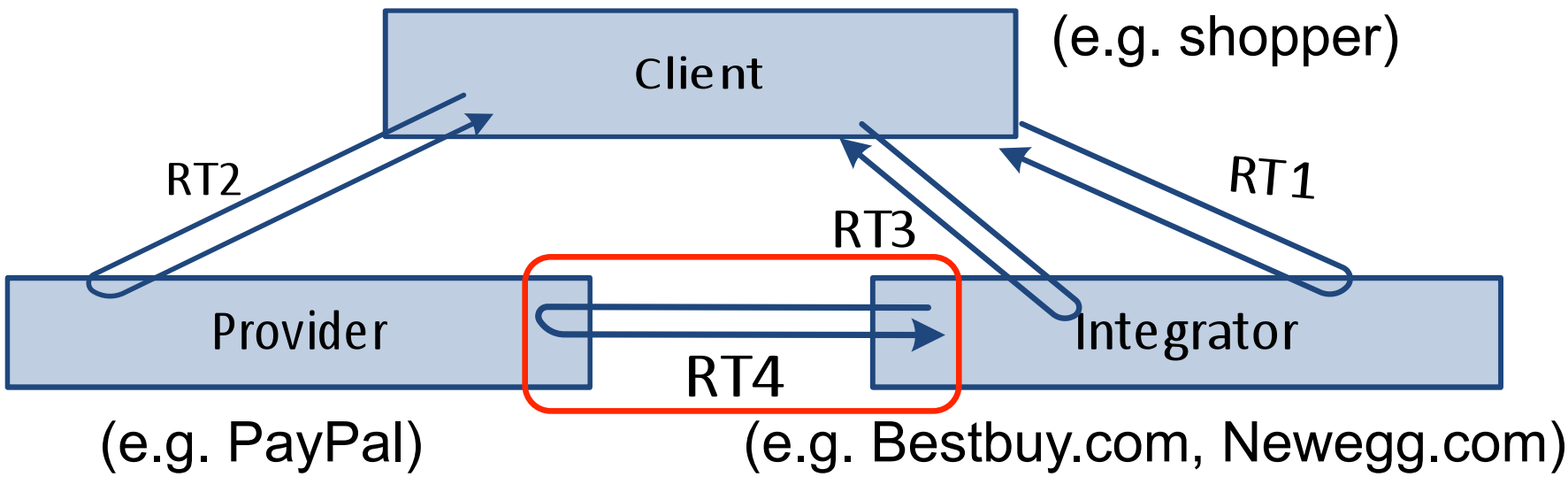
- Protection
  - Integrator side more error-prone.
  - Traffic among integrator, provider and clients is generally mechanic.

BACKGROUND

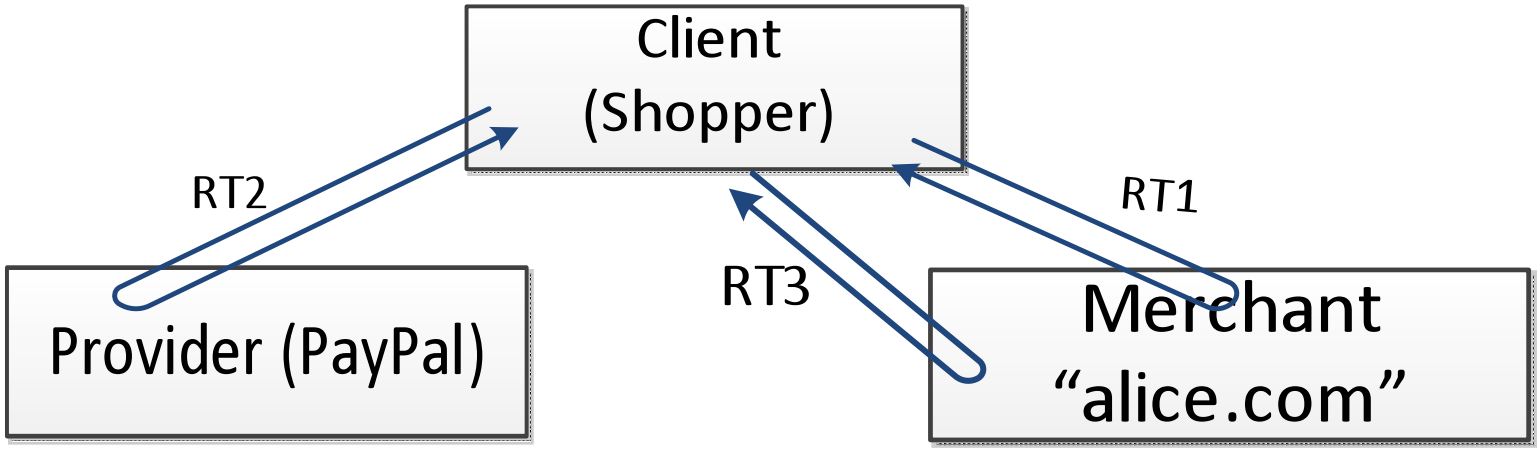
Background

- 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**
&**payeeEmail=alice@email.com**

AccountId = bob

payeeEmail = bob@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

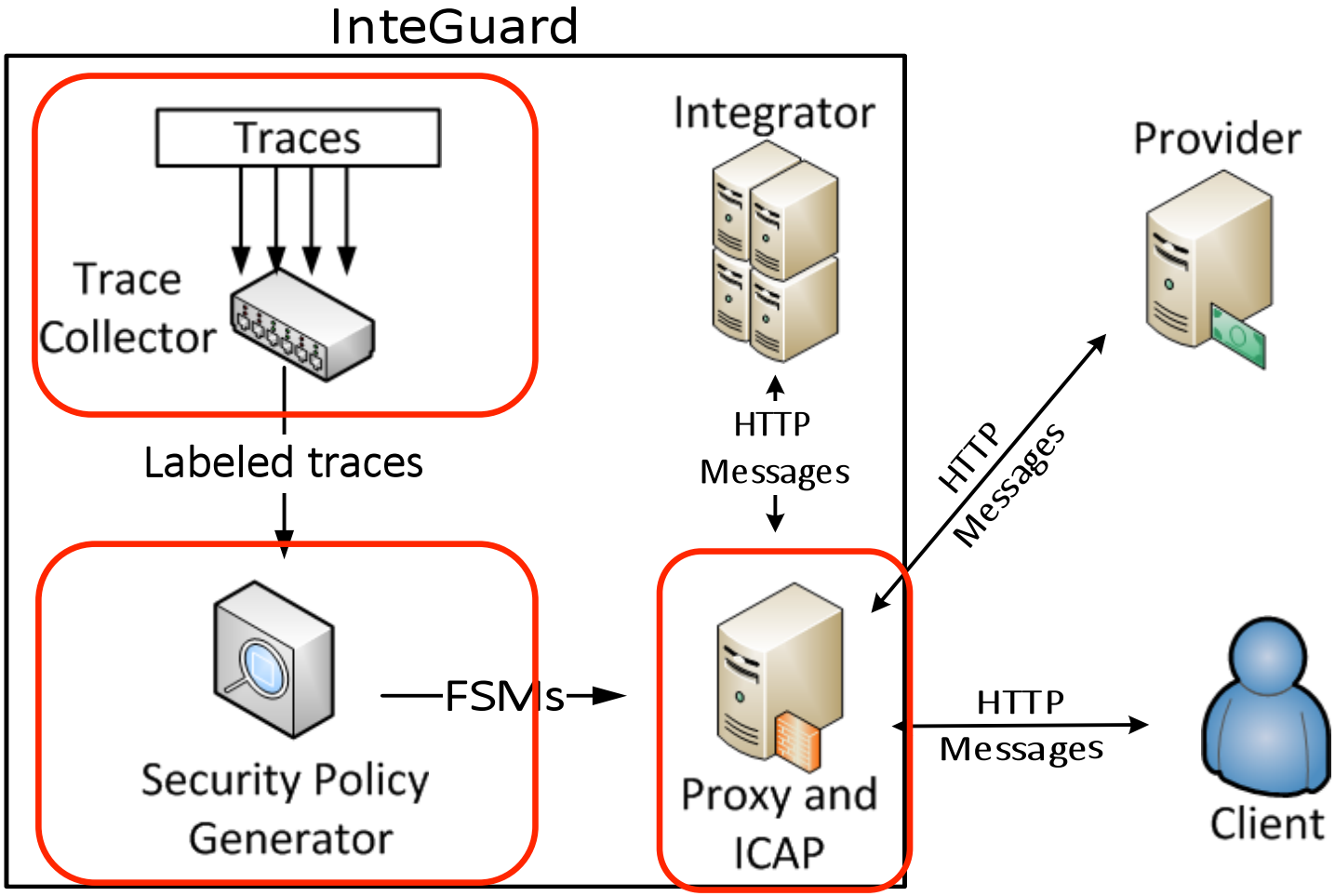
- Logic flaws in Integration
- Service provider is trusted
- Client is not trusted

Contribution

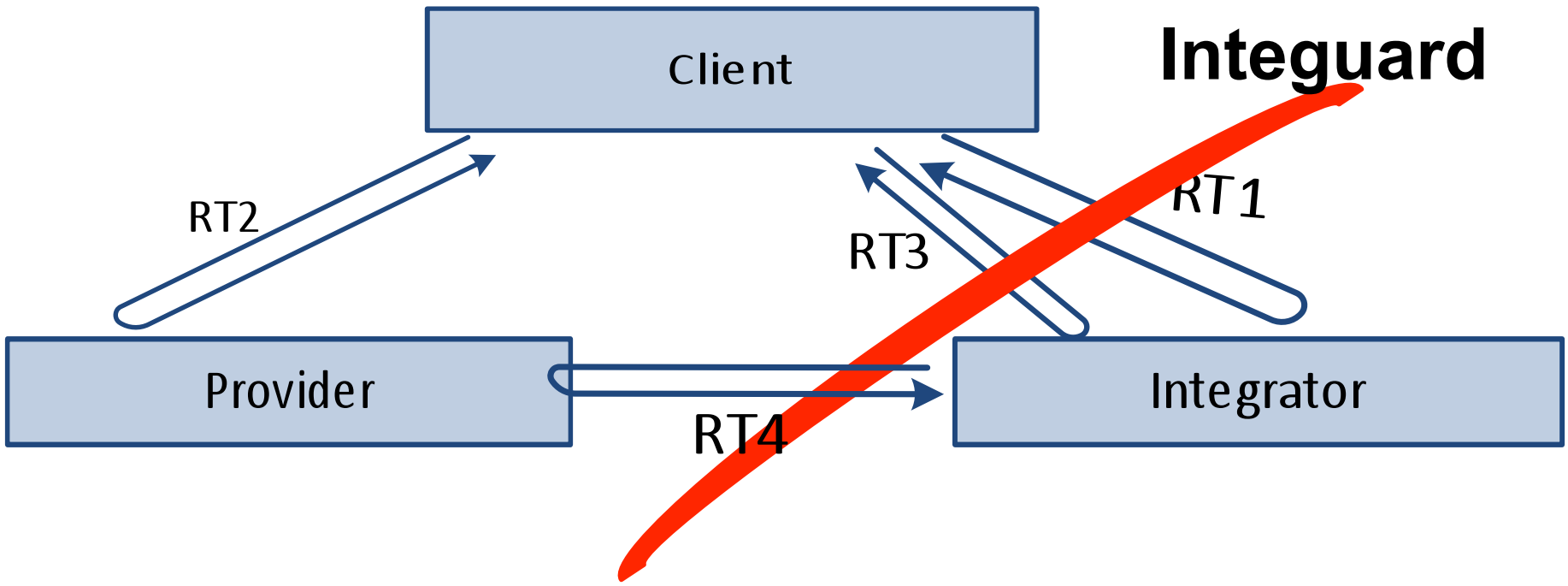
- Integuard
  - First step toward automatic and generic protection of Web service integrations.
  - New challenges in multiple-party settings.
  - Effective false positive control.
  - Evaluate with real exploits and performance test.
  - Practical protection.

DESIGN

Design – Architecture



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)

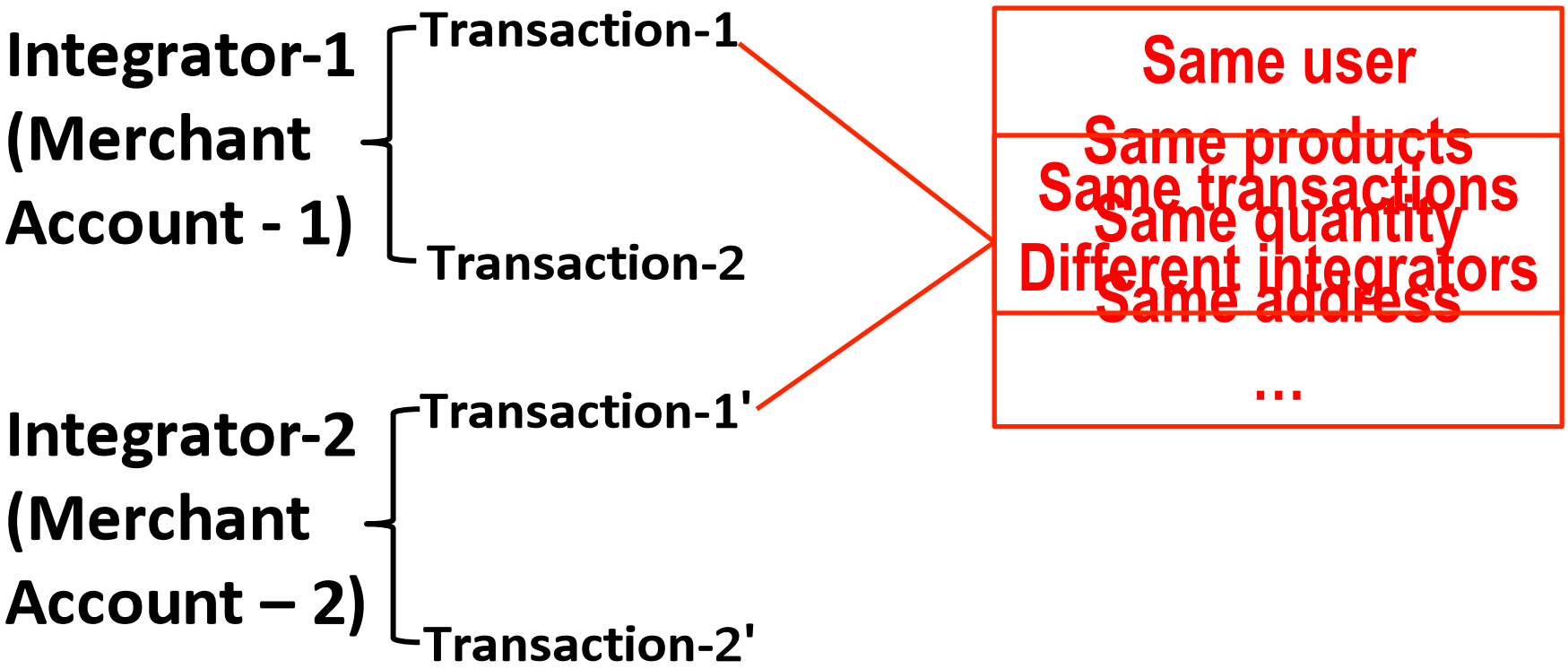
- **Transaction 1**

Integrator 1 and Integrator 2 are the same Web application, configured with different … merchant accounts

Different users
Different products
Different transactions
Different quantities
Same integrator
Different addresses

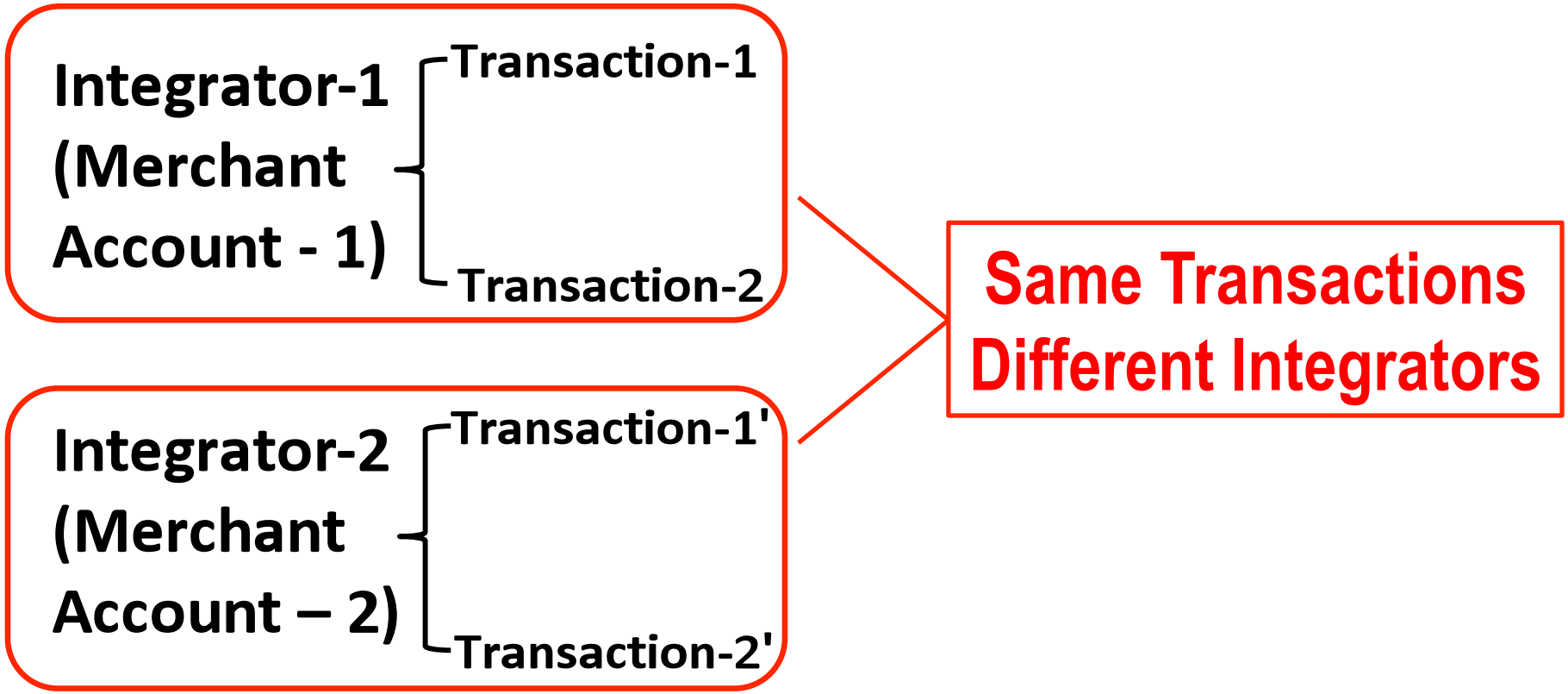
Design – Training Trace Collection

- Four traces



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'**

Differential Analysis

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

Integrator1

T_1 RT3: payeeEmail=**jeff@email.com**&...

T_2 RT3: payeeEmail=**jeff@email.com**&...

jeff@email.com
Specific to
Integrator 1

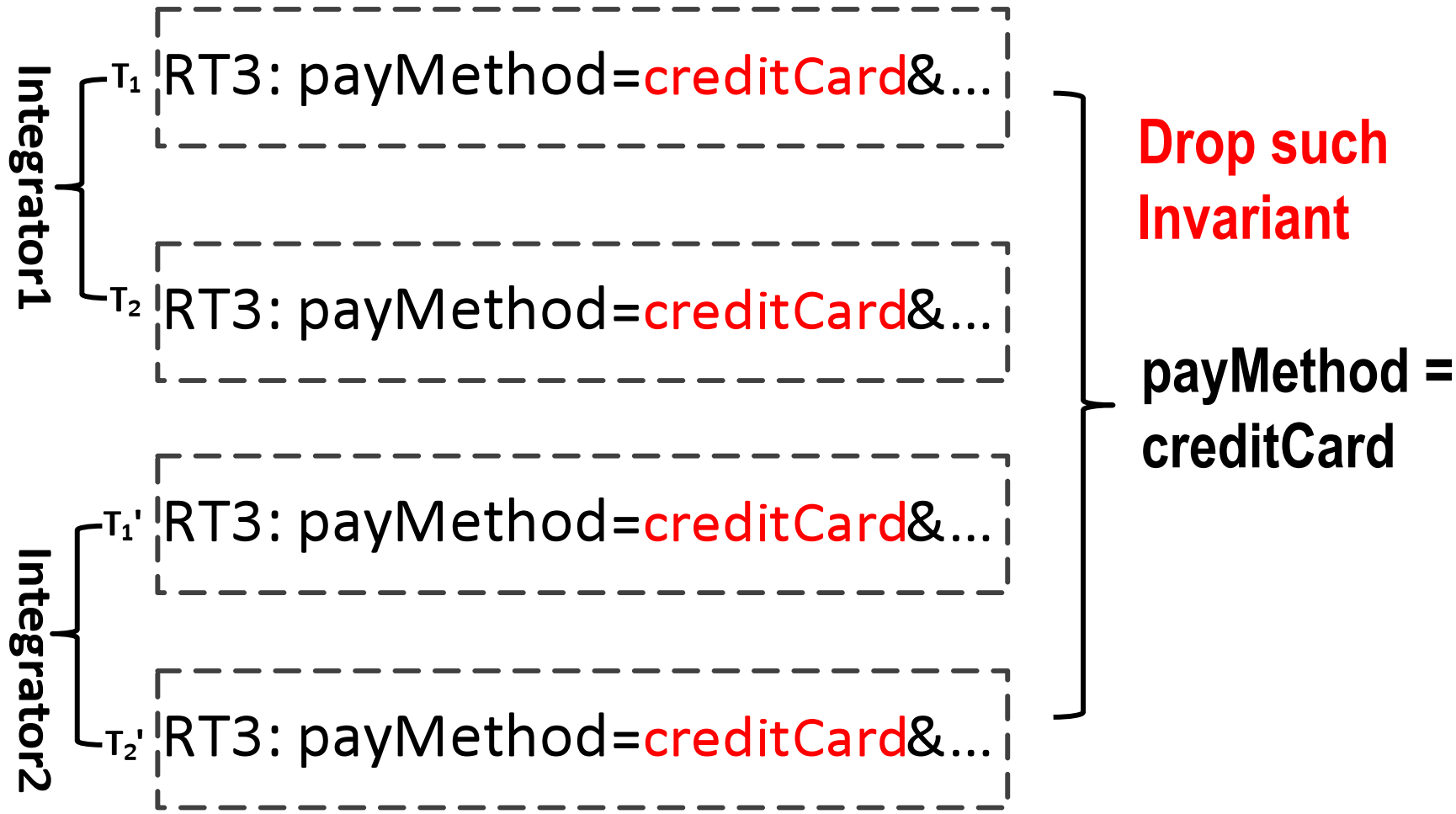
Integrator2

T_1' RT3: payeeEmail=**alice@email.com**&...

T_2' RT3: payeeEmail=**alice@email.com**&...

alice@email.com
Specific to
Integrator 2

Integrator-Specific Invariant: RT3.payeeEmail


Integrator1
T1
RT3: payMethod=creditCard&...
T2
RT3: payMethod=creditCard&...
Integrator2
T1'
RT3: payMethod=creditCard&...
T2'
RT3: payMethod=creditCard&...
Drop such Invariant
payMethod = creditCard

Design – Invariant Analysis

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

Integrator1

T_1

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

amount = gross

T_2

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

amount = gross

Local Invariant:
amount == gross

Integrator2

T_1'

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

amount = gross

T_2'

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

amount = gross

Integrator1

T_1

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

returnFlag = status

T_2

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

returnFlag = status

Drop Invariant With Length < 3

Integrator2

T_1'

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

returnFlag = status

T_2'

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

returnFlag = status

Design – Invariant Analysis

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

Integrator1

T_1
RT1: transactionID=0519&...
RT3: orderID=0519&...

T_2
RT1: transactionID=0520&...
RT3: orderID=0520&...

Integrator2

T_1'
RT1: transactionID=0521&...
RT3: orderID=0521&...

T_2'
RT1: transactionID=0522&...
RT3: orderID=0522&...

transactionID
==
orderID

Integrator1

T_1
RT1: transactionID=0519&...
RT3: orderID=0519&...

T_2
RT1: transactionID=0520&...
RT3: orderID=0520&...

Integrator2

T_1'
RT1: transactionID=0521&...
RT3: orderID=0521&...

T_2'
RT1: transactionID=0522&...
RT3: orderID=0522&...

0519

0520

Specific to each transaction

0521

0522

Integrator1

T_1
RT1: transactionID=0519&...
RT3: orderID=0519&...

T_2
RT1: transactionID=0520&...
RT3: orderID=0520&...

Integrator2

T_1'
RT1: transactionID=0521&...
RT3: orderID=0521&...

T_2'
RT1: transactionID=0522&...
RT3: orderID=0522&...

Local Invariant
→
Transaction specific invariant

Integrator1

T_1
RT1: amount=9.99&...
RT3: gross=9.99&...

T_2
RT1: amount=13.99&...
RT3: gross=13.99&...

Integrator2

T_1'
RT1: amount=9.99&...
RT3: gross=9.99&...

T_2'
RT1: amount=13.99&...
RT3: gross=13.99&...

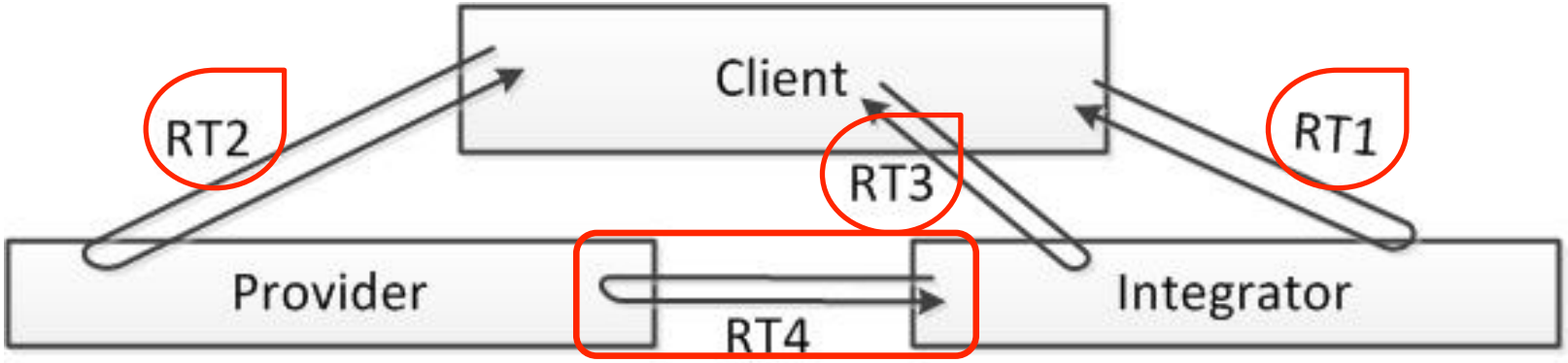
9.99

Same transactions
Different integrators

amount = gross

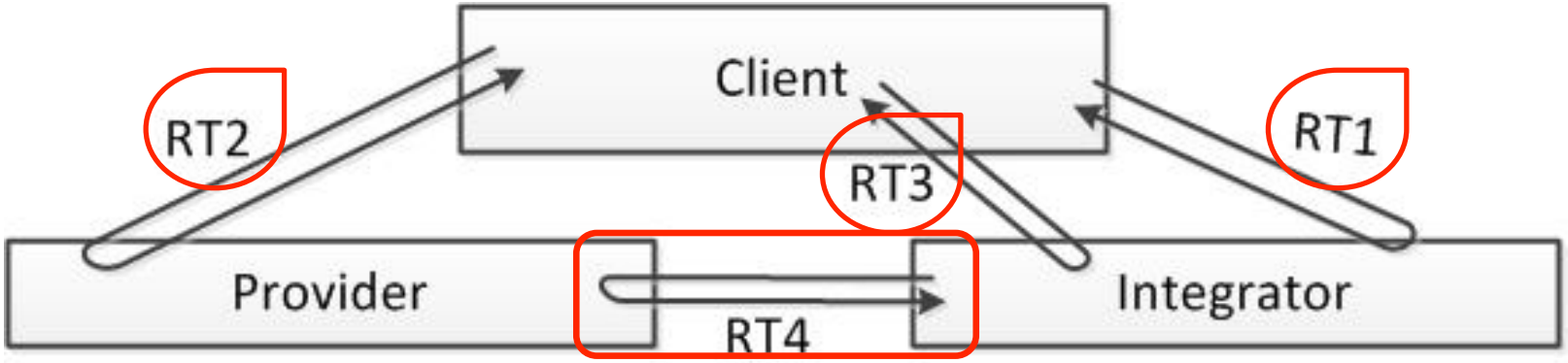
9.99

Transaction-specific Invariant



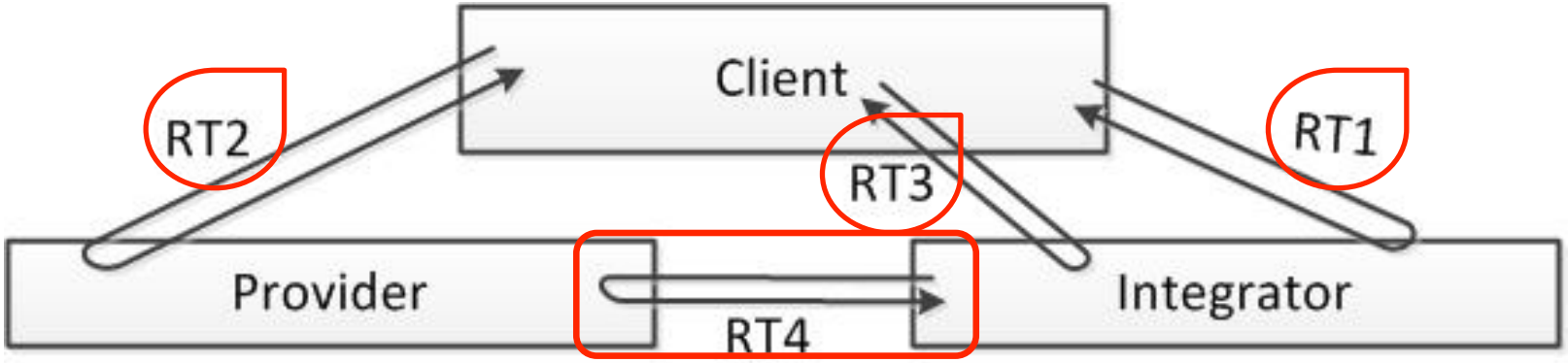
Is RT4 different?

Transaction-specific Invariant



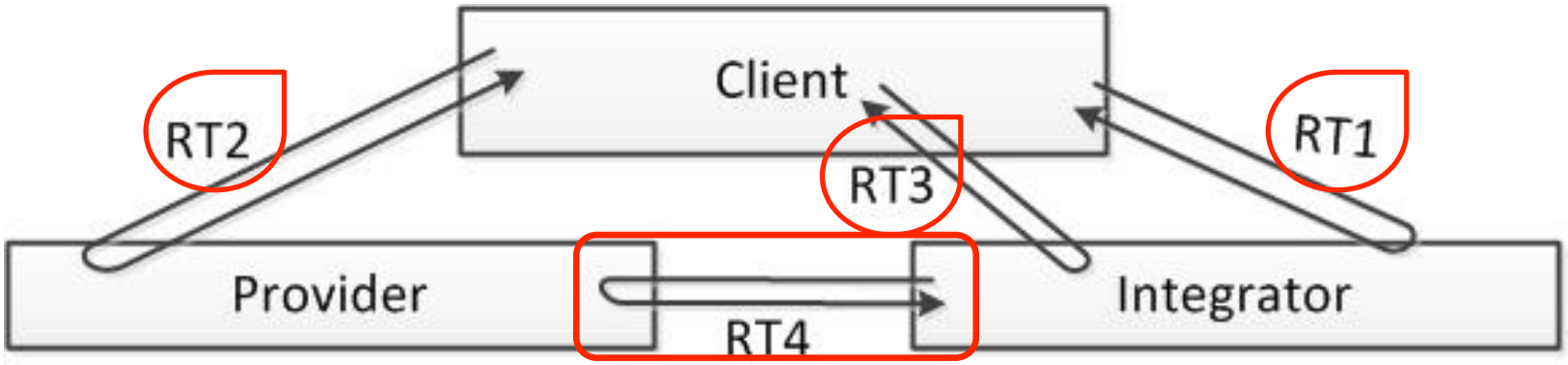
RT4 has no cookies

Transaction-specific Invariant



Which transaction does a RT4 belong to?

Transaction-specific Invariant



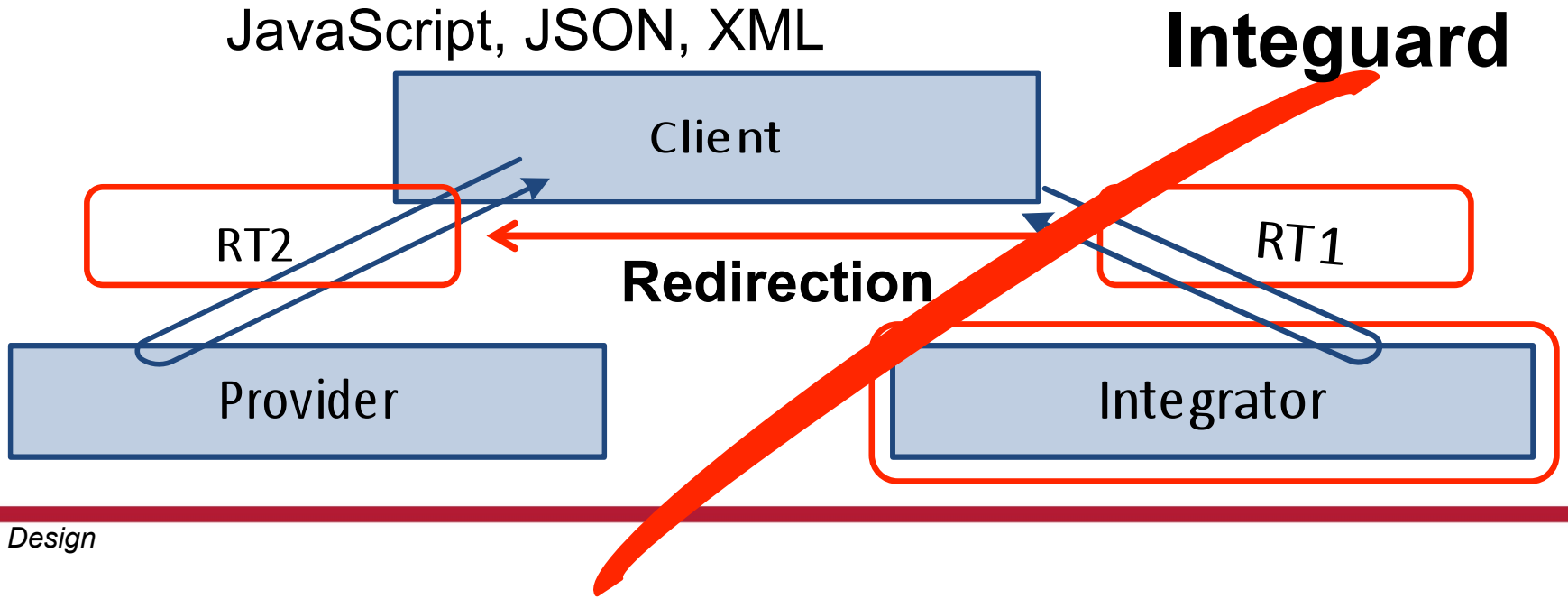
Transaction-specific Invariants
help
Grouping RT4 into its belonging transaction

Design – Invariant Analysis

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

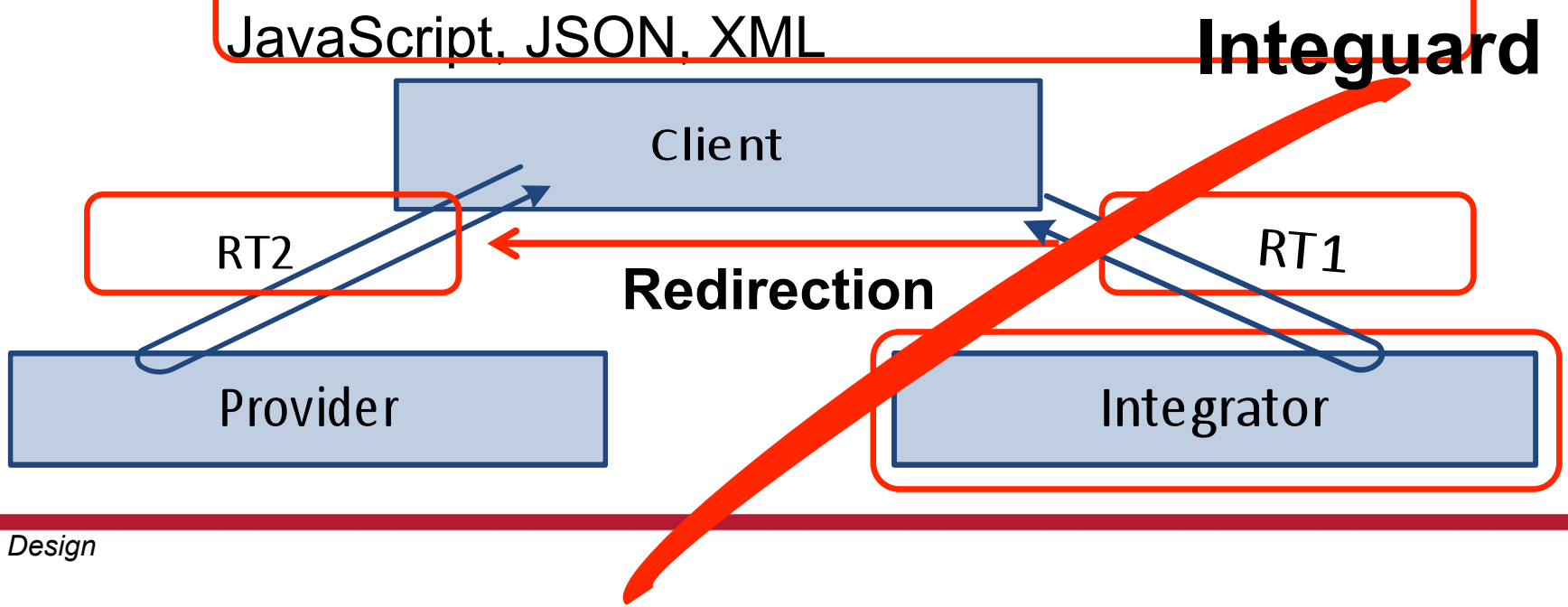
Design – Element Extraction

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



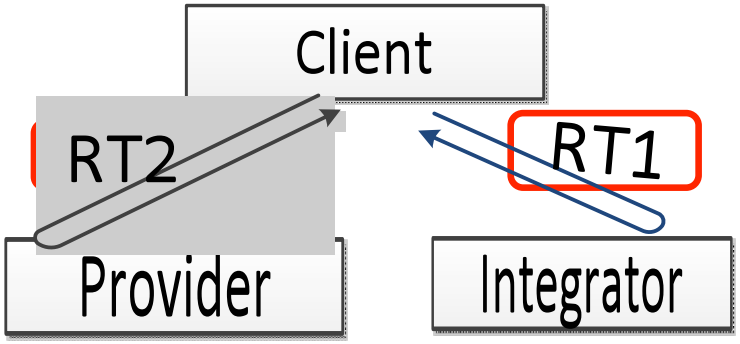
Design – Element Extraction

- 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

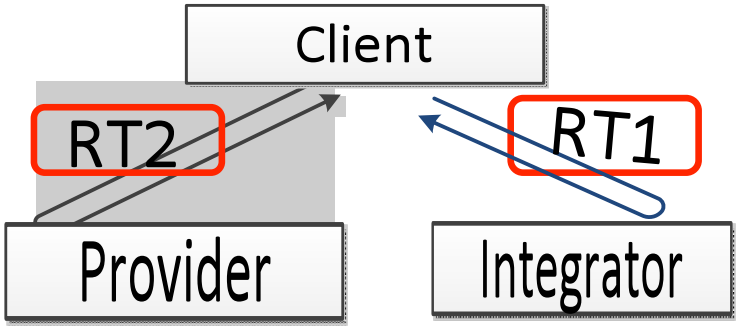
Extracted DOM path:

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

From RT1, extract RT2's parameters

Record the DOM locations for each parameter

Design – Element Extraction



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>
```

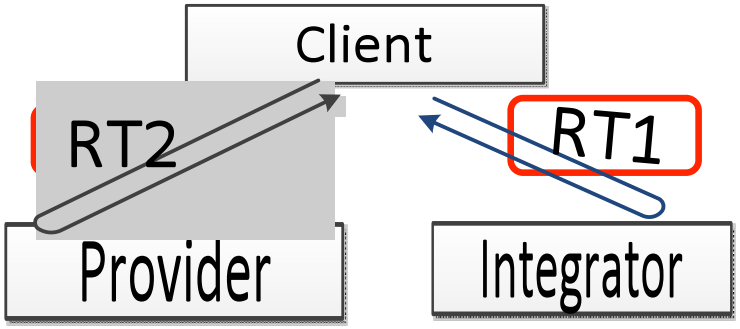
Extract a DOM path for each parameter

Extracted DOM path:

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

Don't parse all content of RT1

Design – Element Extraction



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

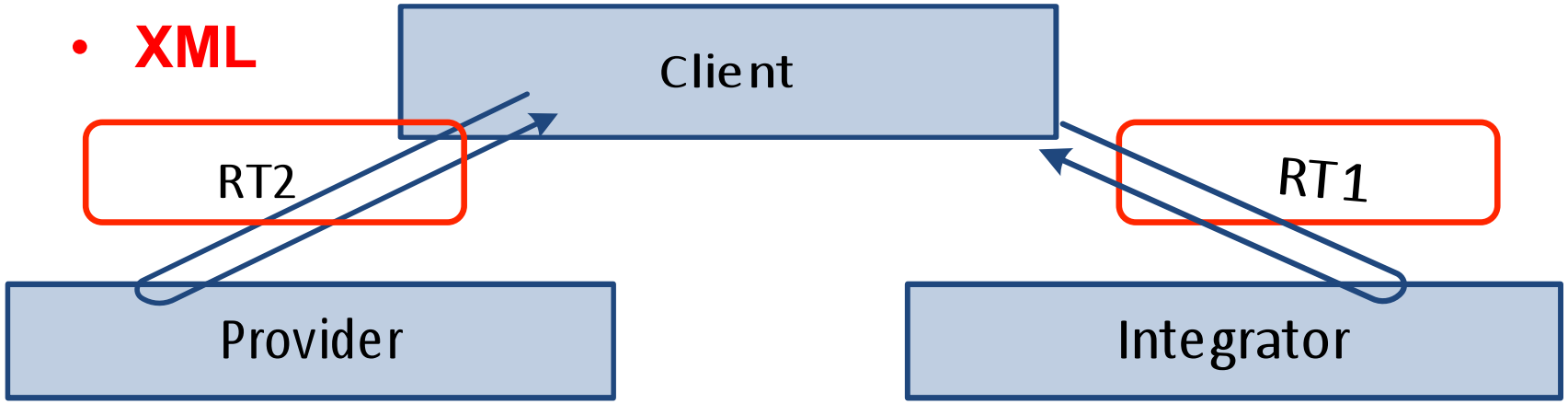
Extracted DOM path:

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

Just extract desired parameters from known locations

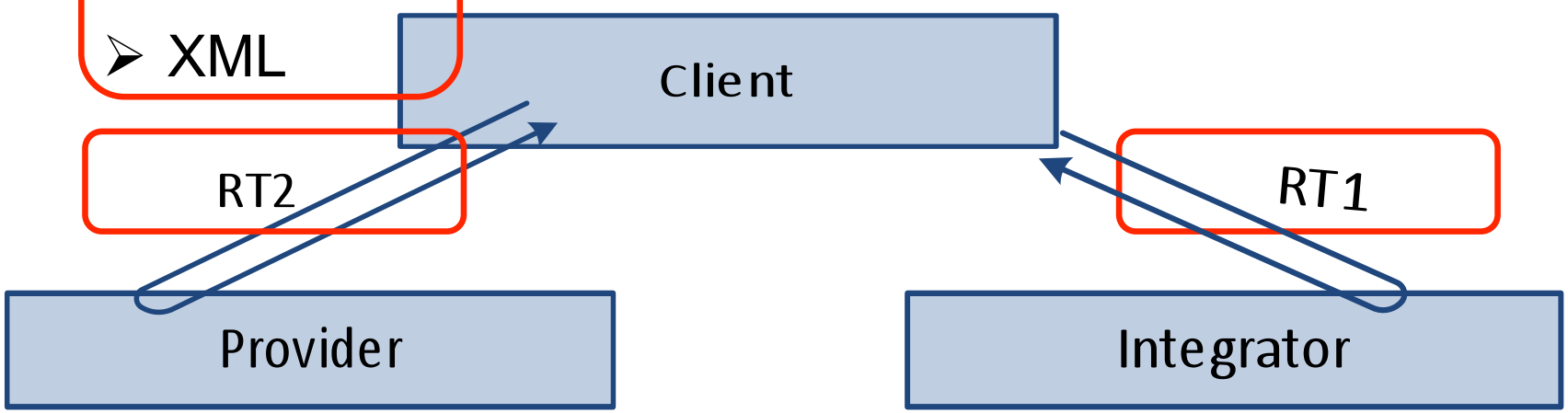
Design – Element Extraction

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



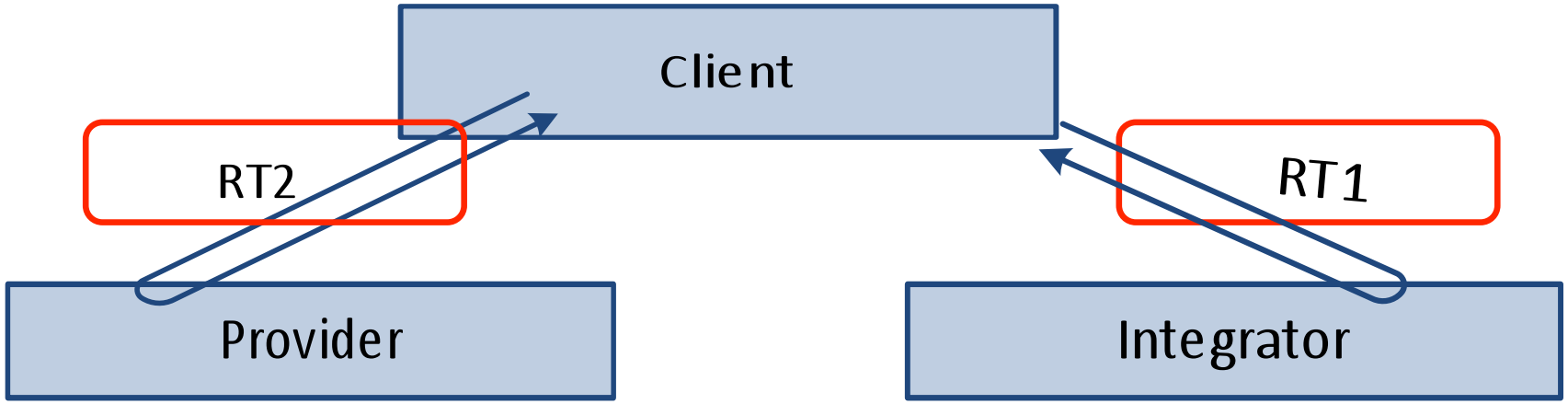
Design – Element Extraction

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



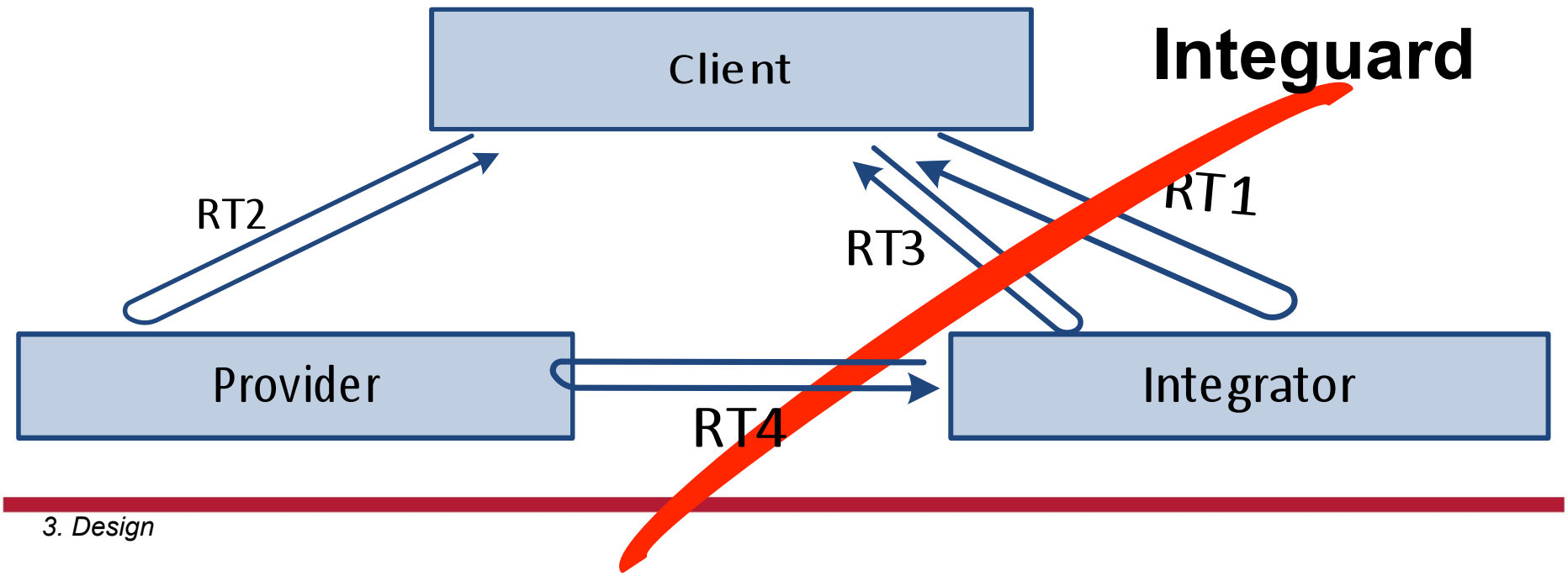
Design – Element Extraction

- 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.



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)

CONCLUSION

Conclusion

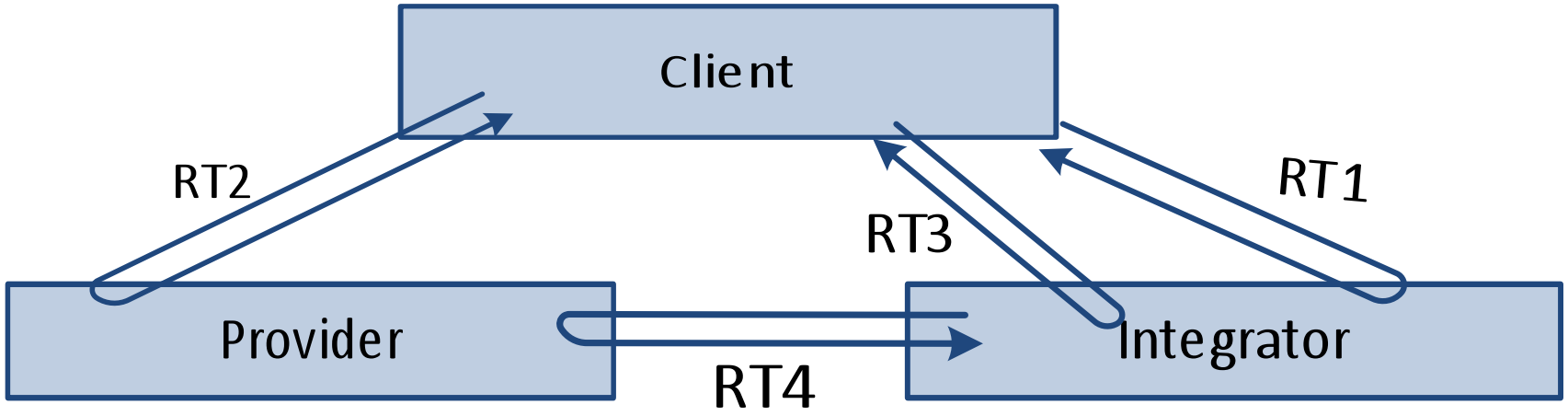
- First to protect vulnerable integrations of third-party Web services.
- New challenges in multi-party settings.
- Generate invariants through a suit of new techniques.
- Effective false positive control and low performance expense.

INDIANA UNIVERSITY

Microsoft Research

THANK YOU!

LUYI XING


Client
RT2
RT1
RT3
Provider
Integrator
RT4

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.