A Risk-Evaluation Assisted System for Service Selection

Ennan Zhai and Liang Gu

Yale University

{firstname.lastname}@yale.edu
Service Oriented Architecture

Custom Built App

Low-level Systems

COTS Packages

External Services

Datasets

Developer

Service Bus

Service1

Service2
Unexpected Risks

Custom Built App
COTS Packages
Datasets
Low-level Systems
External Services

Service Bus
Developer
Service1
Service2
Unexpected Risks

Custom Built App

COTS Packages

Datasets

Low-level Systems

External Services

Service Bus

Service1

Service2

Developer
Example

Video App
Example Video App

- Encoding
  - S3
- Encoding
  - Azure
- Crypt Lib
Example

Video App

Query

Encoding

S3

Query

Encoding

Azure

Crypt Lib
Example

Video App

- Query
  - Encoding
    - S3
- Query
  - Encoding
    - Azure
- Crypt Lib
What leads to the problem?
What leads to the problem?

• Lack of systematic approach to avoid these bugs.
What leads to the problem?

- Lack of systematic approach to avoid these bugs.
- No service provider is willing to share the information.
Target

- Can we reduce such risk before the service selection of application developers?
Solution:
Risk-Based Service Selection
Solution: Risk-Based Service Selection

- Select services based on requirements.
Solution:
Risk-Based Service Selection

- Select services based on requirements.
- At-best effort to avoid potential bugs within services.
Solution:
Risk-Based Service Selection

- Select services based on requirements.
- At-best effort to avoid potential bugs within services.
- Do not leak information of service providers.
Road-Map

- Motivating Example
- REaaS Design
- Evaluation
Road-Map

- Motivating Example

- REaaS Design

- Evaluation
Motivating Example
Motivating Example

App Developer

Service A

Service B

Service C
Motivating Example

App Developer

Select a service without overflow bugs

Service A

Service B

Service C
Motivating Example

App Developer

Service A

Service B

Service C

ESaaS
Motivating Example

<table>
<thead>
<tr>
<th>Service</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service B</td>
<td>0.1</td>
</tr>
<tr>
<td>Service C</td>
<td>0.8</td>
</tr>
<tr>
<td>Service A</td>
<td>1.3</td>
</tr>
</tbody>
</table>

App Developer

ESaaS
Road-Map

• Motivating Example
• REaaS Design
• Evaluation
Road-Map

- Motivating Example
- REaaS Design
- Evaluation
REaaS Workflow

App Developer

Service A

Service B

Service C

ESaaS
Step 1: Service Registration

App Developer

ESaaS

Service A

Service B

Service C
Step 1: Service Registration

App Developer

ESaaS

Service A

Service B

Service C
Step 2: Requirement Submission

App Developer → ESaaS

Service A → Service B → Service C
Step 3: Risk Evaluation

<table>
<thead>
<tr>
<th>Service</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service B</td>
<td>0.1</td>
</tr>
<tr>
<td>Service C</td>
<td>0.8</td>
</tr>
<tr>
<td>Service A</td>
<td>1.3</td>
</tr>
</tbody>
</table>
Step1: Service Registration

• We developed a tool automatically getting dependency.

• Running on service side and very fast.
Step1: Service Registration

```plaintext
mysql-server
  Depends: mysql-server-5.5
mysql-server-5.5
  Depends: libc6 (>= 2.14)
  Depends: debconf (>= 0.5)
debconf
  Depends: perl-base (>= 5.6.1-4)
libc6
  Depends: libc-bin (= 2.15-0ubuntu10)
  Depends: libgcc1
  Depends: tzdata
```
Step1: Service Registration
Step 2: Requirement Submission
Step2: Requirement Submission

- Requirements include:
  - availability
  - integrity
  - confidentiality

- The options follow CVSS (an open bug DB)
Step3: Risk Evaluation
Step 3: Risk Evaluation

\[
TD_i = \frac{1}{n} \sum_{j=1}^{n} BS_{j(i)}
\]

\[
BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176
\]

\[
Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)}
\]
Step 3: Risk Evaluation

\[ TD_i = \frac{1}{n} \sum_{j=1}^{n} BS_{j(i)} \]

\[ BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \]

\[ Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)} \]

Risk score of service i
Step 3: Risk Evaluation

\[
TD_i = \sum_{j=1}^{n} \frac{BS_{j(i)}}{n} \\
BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \\
Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)}
\]

Different bugs’ impact under different objects
Step 3: Risk Evaluation

$$TD_i = \frac{1}{n} \sum_{j=1}^{n} BS_{j(i)}$$

$$BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176$$

$$Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(\text{SecObj})}$$

Different bugs’ impact under different objects

Gotten from CVSS + CVE
Step 3: Risk Evaluation

\[ TD_i = \sum_{j=1}^{n} \frac{BS_{j(i)}}{n} \]

\[ BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \]

\[ Imp_{j(i)} = 10.41 \cdot \text{ImpactLevel}_{j}(\text{SecObj}) \]
Trust?
Trust?

App Developer

Service A

Service B

Service C

ESaaS
TPM-Based REaaS
TPM-Based REaaS

App Developer

Service A

Service B

Service C

ESaaS
TPM-Based REaaS

App Developer

Service A

§=hashcode(actions)

Service B

Service C

ESaaS
TPM-Based REaaS

App Developer

Service A

Service B

Service C

ESaaS
TPM-Based REaaS

App Developer

Service A

Service B

Service C

ESaaS
Road-Map

• Motivating Example

• REaaS Design

• Evaluation
Road-Map

• Motivating Example

• REaaS Design

• Evaluation
# Case Study

<table>
<thead>
<tr>
<th></th>
<th>MySQL</th>
<th>PostgreSQL</th>
<th>Risk</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td># of packages</td>
<td>588</td>
<td>736</td>
<td>103</td>
<td>108</td>
</tr>
<tr>
<td>Risk score</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Performance Evaluation

• One Dell XPS14 laptop
  - 2.8GHz 4-Core Intel Xeon CPU
  - 16GB memory

• Public dataset with N packages
  - N = 10, 100, 1000, and 10000
Time for Dependency Collection

![Graph showing running time vs. number of packages within a service with and without TPM.](image)
Time for Risk Evaluation

![Graph showing the running time in seconds for different numbers of packages within a service with and without TPM. The x-axis represents the number of packages, ranging from 10 to 100,000, and the y-axis represents running time, ranging from 1 to 1e+06 seconds.]
Conclusion

• The first-step towards practical risk-based service selection approach.

• TPM-based approach to prevent privacy leakage.

• A realistic case study and performance evaluation.
Thanks!