Privacy Preserving Data Dissemination and Adaptable Service Compositions in Trusted and Untrusted Cloud

25/26 April 2016

Bharat Bhargava
Purdue University

Technical Champion(s): Leon Li, Sunil Lingayat, Jason Kobes, Donald Steiner
Adaptable Service Compositions and Data Dissemination in Cloud

- Need systematic monitoring of service operations and data dissemination for:
  - **Resiliency** (withstand cyber-attacks, sustain and recover critical function)
  - **Antifragility** (increase in resilience and robustness as a result of failures)
  - **Adaptability** (swiftly adapt to changes in context, choose services in orchestration to comply with QoS requirements)

- Shortcomings of existing systems:
  - Coarse-grain monitoring capabilities (e.g. CloudWatch)
  - Splunk, GrayLog, Kibana etc. provide storage, search and analysis of big data, but require human intelligence for detection and action for resiliency
  - DataSafe and Encore (sticky policies) require special architectures or are prone to TTP related attacks
Core Technology

• A novel distributed monitoring tool to:
  – Audit and detect service behavior and performance changes*
  – Gather service trust data and share them securely in various domains
  – Dynamically reconfigure service orchestrations based on security context and QoS requirements**

• A secure and adaptable data dissemination technology to deal with:
  – Context changes (e.g. crypto capabilities of web browser, user location/device etc.),
  – Trust
  – Sharing policies of data owner


Benefits

• System modules for *service anomaly detection*, *service performance monitoring* and *trust management* can be easily integrated into NGC cybersecurity software.

• The **modular architecture** and use of **standard software** in the monitoring framework allows for **easy plugin** to any system.

• The different components of the adaptability framework can be extended and integrated with various IRADs and demonstrated at TechFest:
  
  – Cyber 2.0 IRAD (resiliency) (with Dr. Sunil Lingayat)
  – CURATE 2 (with Jason C. Kobes)
  – Citadel EBS (with Jonathan E. Fulkerson)
  – Adaptive Cloud Enterprise (ACE)
  – Cyber Observation of Latent Anomalies (COLA)
System Overview

*: service performance & security parameter values
+: summary service health data, cyber threat information
Service Monitoring / Anomaly Detection System Details

Request Interceptors / Performance Monitor

Service 1

Service 2

Service 3

Service 4

...  

Interaction/performance data stream

Local monitor DB

Anomaly Training/Detection Module

HMM

One-class SVM

K-means clustering

EM

Other learning algorithms

Analysis results

Central Monitor
Service Monitoring and Dynamic Service Composition Implementation Details

- **Local Service Monitor**: Implemented using Apache Axis2 valves for service request interception, MySQL database for logging
- **Central Monitor**: Implemented as Web service on Amazon EC2 with MySQL database for logging of service health, threat and trust data
- **Anomaly Detection Module**: Implemented in Java with pluggable unsupervised learning algorithms (from the Weka machine learning library: http://www.cs.waikato.ac.nz/ml/weka/)
- **Dynamic Service Reorchestration**: Implementation of business process composition using dynamic partner links in BPEL (business process execution language)
- **Aspect-Oriented Programming (AOP)** based actions in case of anomaly detection / violation of policy
Core Machine Learning Technique Used for Service Anomaly Detection in System

- No class labels needed for training.
- Training data: input vectors without any corresponding target values
- Unsupervised learning with security and performance parameters used to find clusters
- Values outside clusters will be detected as outliers and help detection of anomalies.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Cloud services</th>
<th>Cloud data services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of requests/sec</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Bytes downloaded/sec</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bytes uploaded/sec</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Total error rate</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CPU utilization</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Memory utilization</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of authentication failures</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of connections</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of connection failures</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of disk reads/writes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Network latency</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Service response time</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Disk space usage</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Throughput</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of database connections</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Service/cluster health status</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Implemented Unsupervised Learning Algorithms

- Two unsupervised learning algorithms implemented.
- Training performed offline, under normal system operation. Classification time negligible.
- Different learning algorithms pluggable into system.

**K-Means Clustering:**

- **Training:**
  
  - Input: Matrix $V_{d \times t}$ of service performance record
  - $d$: number of performance parameters
  - $t$: number of time points observed

  Cluster each set of performance parameter values using K-means algorithm

- **Testing (system operation):**
  
  - for each service interaction log
    - measure distance of performance parameter values to each cluster, assign time point to closest cluster
  - if latest interaction does not belong to any cluster
    - raise anomaly signal

**One-class SVM (Support Vector Machines):**

- Learns decision function for novelty detection
- Decision hyper-plane boundary based on normal runtime conditions
- Multiple features of services enter the model for classification

*Figure from http://stackoverflow.com/questions/9480605/*
Dynamic Service Composition Experiments

Experiment settings:

<table>
<thead>
<tr>
<th>Service instance</th>
<th>t2.small (1 vCPU, 2GB memory, and EBS storage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE and TM instances</td>
<td>t2.small (1 vCPU, 2GB memory, and EBS storage)</td>
</tr>
<tr>
<td>Client instance</td>
<td>t2.micro (1 vCPU, 1GB memory, and EBS storage)</td>
</tr>
<tr>
<td>Operating system</td>
<td>Amazon Linux 2015.03 64-bit OS</td>
</tr>
<tr>
<td>Geographical region</td>
<td>US-w2 (Oregon region)</td>
</tr>
</tbody>
</table>

- Overhead evalution for three cases:
  - Different number of service categories in composition
  - Different number of services to choose from for each category
  - Different number of QoS contraints

- Composition time not affected significantly by the number of QoS constraints. Number of services and service categories have more visible effect, with still reasonable overhead.
Overhead of Service Monitoring / Request Interception

- Performance evaluation of service domain in terms of throughput and service response time
- Negligible overhead incurred by monitoring

**Experiment settings:**

<table>
<thead>
<tr>
<th>Service instance</th>
<th>PE and TM instances</th>
<th>Client instance</th>
<th>Operating system</th>
<th>Geographical region</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2.small (1 vCPU, 2GB memory, and EBS storage)</td>
<td>t2.small (1 vCPU, 2GB memory, and EBS storage)</td>
<td>t2.micro (1 vCPU, 1GB memory, and EBS storage)</td>
<td>Amazon Linux 2015.03 64-bit OS</td>
<td>US-w2 (Oregon region)</td>
</tr>
</tbody>
</table>

![Bar chart and line graph showing performance metrics over different number of concurrent requests and percentiles.](image_url)
Focus: Secure Data Dissemination in Cloud

- Authorized service can only access data items for which it is authorized
- Unauthorized service denied
- Provide data dissemination based on cryptographic capabilities of client’s browser and authentication methods
- Support different authentication methods for client service
- Detect data leakage to unauthorized services
Data Leakage in Untrusted Cloud


* This work is used in PhD Thesis Proposal of Denis Ulybyshev, Purdue University
Core Design (in collaboration with MIT)

Authenticated Client

Browser’s Crypto Level: High
Authentication Method: Fingerprint
Client’s device: Desktop
Source network: Corporate Intranet

Authenticated Client

Browser’s Crypto Level: Low
Authentication Method: Password
Client’s device: Mobile
Source network: Unknown
Core Design (in collaboration with MIT)

- Redirect unauthenticated client’s request from Cloud Provider to Authentication Server (AS)
- Selective Data dissemination based on:
  - Role-based access control policies
  - Security level of client’s browser (crypto capabilities)
  - Authentication method (password-based, fingerprint etc)
  - Source network (secure intranet vs. unknown network)
  - Type of client’s device: desktop vs. mobile (detected by Authentication Server)
Key Generation during AB Creation

- AB Template is used to generate new ABs with data and policies (specified by data owner)
- Template includes implementation of invariant parts (monitor) and placeholders for customized parts (data and policies)
- Template is executed to simulate interaction between AB and service requesting access to each data item of AB
Key Derivation during AB Execution

• AB receives access request to data item from service
• AB authenticates the service and authorizes its request
• If any module fails (i.e. service is not authentic or the request is not authorized) or is tampered: derived decryption key $K_i$ is incorrect => data is not decrypted

Key Management in AB

- Key is not stored inside AB
- Separate symmetric key is used for each separate data field
- Ensure protection against tampering attacks (discussed with Jason Kobes)

Aggregation\{d_i\}
\((\text{Execution info}; \text{Digest(AB Modules)}; \text{Resources})\)

Aggregation\{d_i\} (Tampered (\text{Execution info}; \text{Digest(AB Modules)}; \text{Resources}))
Scenario for TechFest’16: Electronic Health Record Dissemination in Cloud

Cloud Server

Doctor

Insurance

Researcher

MIT:
- Web Crypto Authentication
- Authorized Info for Doctor
- Web Crypto Authentication
- Authorized Info for Insurance
- Web Crypto Authentication
- Authorized Info for Researcher

Scenario of EHR Dissemination in Cloud (by Dr. Leon Li, NGC)
Scenario for TechFest’16: Electronic Health Record Dissemination in Cloud

Active Bundle: Contact, Medical and Billing Information

- **Researcher**
  - E(Contact Info)
  - Medical Info
  - Billing Info

- **Insurance**
  - Contact Info
  - E(Medical Info)
  - Billing Info

- **Doctor**
  - Contact Info
  - Medical Info
  - Billing Info

**Hospital (NodeJS Server)**

**Client (Browser)**

1. HTTP GET Request
2. Hospital’s Web Page
3. HTTP POST with Data Request and Role
4. HTTP 302 with AB Request and Role
5. HTTP 302 with Ticket
6. AS Web Page
7. HTTP POST with Credentials
8. HTTP Get Request with Ticket
9. HTTP Get Request with Ticket
10. Data provided by AB
Authentication Ticket Creation and Validation

### Authentication Server:
- Knows shared secret K and Private Key PrivKey
- $\text{Ticket\_Info} = (\text{Auth\_Level}, \text{Expiration\_Time}, \text{Client\_ID}, \text{Client\_Role}, \text{Request\_Field})$
- $\text{Enc\_Ticket\_Info} = \text{EncAES256}_K(\text{Ticket\_Info})$
- $\text{Ticket\_Signature} = \text{Enc}\_\text{PrivKey}(\text{SHA512}(\text{Enc\_Ticket\_Info}))$
- Ticket = $\langle \text{Enc\_Ticket\_Info}, \text{Ticket\_Signature} \rangle$

### Doctor, Insurance or Researcher Service:
- Knows shared secret K and Public Key PubKey
- Receives Ticket = $\langle \text{Enc\_Ticket\_Info}, \text{Ticket\_Signature} \rangle$
- Checks: $\text{Dec}\_\text{PubKey}(\text{Ticket\_Signature}) = \text{SHA512}(\text{Enc\_Ticket\_Info})$
- Gets data: $\text{DecAES256}_K(\text{Enc\_Ticket\_Info})$

- Doctor is authorized for Contact, Medical and Billing Info of either “only her own patients” or “her own and other doctor’s patients”
- Clients are directed to the corresponding services according to their role (researcher, insurance or doctor)
- **Authorization level (included in ticket) and role define the data accessible by requester**
Search on Encrypted Database of ABs

Result of weekly meetings with Dr. Leon Li, Jason Kobes, NGC
(figure created by Dr. Leon Li)

* Integration with homomorphic encryption project (Prof. Patrick Eugster, Purdue Univ.)
Components of Encrypted AB Database

- Collection agent gathers intelligence feeds (ABs)
- CryptDB storing encrypted data (keywords, abstract of AB) and providing SQL query capability over encrypted data
- Subscription API to provide methods for authorized access to data
Demonstration

- Dynamic service composition and trust management:
  - Composite trust algorithms
    https://www.youtube.com/watch?v=6uHEfoxjEgs
  - Trust update mechanisms
    https://www.youtube.com/watch?v=xnm0-MzGBzw
  - Policy enforcement in service interactions
    https://www.youtube.com/watch?v=ePtAM0N7jdY
  - Service redirection
    https://www.youtube.com/watch?v=e8xkCgZcQ1s
  - Adaptive and secure service composition
    https://www.youtube.com/watch?v=VQDbPD2q9-8

- Secure data dissemination in untrusted cloud
  https://www.dropbox.com/s/30scw1srqsmyq6d/BhargavaTeam_DemoVideo_Spring16.wmv?dl=0
  https://www.youtube.com/watch?v=SIUupq5V6zk&feature=youtu.be
• **Prototype implementation of monitoring framework:**
  – Local (domain-level) Service Monitor (implemented using Apache Axis2 valves for interception, MySQL database for logging)
  – Central Monitor (implemented as Web service on Amazon EC2)
  – Anomaly Detection Module (with pluggable algorithms)
  – Dynamic Service Composition Module (algorithm)
  – Active Bundle Module
    • AB implementation as an executable JAR file
    • AB API implementation using Apache Thrift RPC framework
    • Policy specification in XACML/JSON and evaluation using WSO2 Balana
    • Healthcare scenario with services running on a remote Virtual Machine with functionality to communicate with remote Authentication Server

• **Documentation:**
  – Deployment and user manual
  – Report on performance and security testing of solution
Comprehensive security and privacy auditing and enforcement architecture for trusted and untrusted cloud

- Continuous monitoring of SLA and policy compliance
- Swift detection of failures and attacks in the system
- Efficient mechanism to dynamically reconfigure service composition based on the system context/state (failed, attacked, compromised) and resiliency requirements
- Resilient architecture to ensure continuous service availability under failures and attacks
- Privacy-preserving data sharing approach for client-to-service and service-to-service interactions
- Solution compatible with industry-standard SOA/cloud frameworks
Ongoing Tasks

- Refining monitoring and anomaly detection process with quantitative performance tradeoff model
- Investigation of different system architectures and operating systems for better dynamic configurability allowance (e.g. Linux vs. Windows? based on discussion with Dr. Sunil Lingayat)
- Implementation of data leakage based on watermarks
- Deploy CryptDB storing encrypted AB-related data and providing SQL query capability over encrypted data
- Running AB in isolated container (e.g. Linux Docker)
- Experiments:
  - Tampering attacks on services (code injection etc.)
  - Man in the middle attacks (using an intercepted active bundle)
  - Attacks against data privacy (trying to bypass active bundle protection mechanism)
  - Tampering attacks on active bundle’s policies and code
  - Service failures (evaluation of speed of detection by service monitor)
  - Cloud experiments: Framework scalability on industry standard cloud platforms (e.g. Amazon EC2)
- Integration with NGC projects:
  - Cyber 2.0 IRAD (resiliency) (with Dr. Sunil Lingayat)
  - CURATE 2 (with Jason C. Kobes)
  - Citadel EBS (with Jonathan E. Fulkerson)
Data Leakage Detection

- X is authorized to extract d₁ from AB
- X leaks d₁ to Y without AB

- Watermarks are embedded into data (e.g. with DCT for images)
- d₁ (pdf or jpeg) can only be decrypted and viewed by authorized party from our special software (similar to DRM)
- Software notifies our service monitor: d₁ arrived to Y from X
- SM checks policies: whether d₁ is supposed to be at Y. If NO then:
  - Blacklist X, Y
  - Mark d₁ as compromised and notify other services about it

Service X
AB, d₁

Service Y
Src ID (of X)
Dest ID (of Y)
Data d₁

d₁ leakage

Data d₁ is encrypted
CryptDB integration

- CryptDB stores keywords (in encrypted form) for locating ABs
- CryptDB is a proxy to a database server
  - Provides SQL query capability over encrypted data
  - Never releases decryption key to database
- When database is under attack, only ciphertext is revealed
  - Database has anonymized-schema with encrypted table and column names
  - Database has encrypted data
- Even when CryptDB is compromised, data leakage is limited to data for currently logged in users
- Status: setting up with node.js and MySQL is complete
Scenario: Electronic Health Record Dissemination in Cloud

Insurance Company Forwards AB to Researcher

Insurance Company

Cloud Server

Researcher

Web Crypto Authentication

Authorized Information for Insurance Company

Active Bundle

Scenario of EHR Dissemination in Cloud (by Dr. Leon Li, NGC)
Presentations and Publications

• “A Distributed Monitoring and Reconfiguration Approach for Adaptive Network Computing,” B. Bhargava, P. Angin, R. Ranchal, S. Lingayat. DNCMS in conjunction with SRDS 2015 (Best paper award)


• “Policy-based Distributed Data Dissemination,” R. Ranchal, D. Ulybyshev, P. Angin, and B. Bhargava. CERIAS Security Symposium, April 2015 (Best poster award)
THE VALUE OF PERFORMANCE.

NORTHROP GRUMMAN

CYBERSECURITY RESEARCH Consortium
<Reserve for back-up slides>
Active Bundle Solution Components

- **Sensitive data**
  - Encrypted data items, each field encrypted with its own key
- **Metadata**
  - Access control and operational policies
- **Virtual Machine (Policy Enforcement Engine)**
  - Protection mechanism (self-integrity check)
  - Policy evaluation, enforcement and data dissemination

**AB IMPLEMENTATION**
- Executable JAR file
- Apache-thrift based API
- JSON-based policies
- WSO2 Balana-based policy engine
- Node.js-based SOA architecture
- RESTful web-services

**AB ADVANTAGES**
- Supports different authentication methods
- Independent of source availability
- Ability to operate in untrusted environment
- Reduced host liability for data
Key Generation during AB Creation

- AB Template is used to generate new ABs with data and policies specified by a user
  - Template includes implementation of invariant parts (monitor) and placeholders for customized parts (data and policies)
- User specified data and policies are included in the AB Template
- AB Template is executed to simulate the interaction process between an AB and a service requesting access to each data item of AB
- Information generated during the execution of different AB modules and the digests of these modules and their resources (such as authentication (authentication code, CA certificate that it uses), authorization (authorization code, applicable policies, policy evaluation code)) are collected and aggregated into a single value for each data item
- Value for each data item is input into a Key Derivation module (such as SecretKeyFactory, PBEKeySpec, SecretKeySpec provided by javax.crypto library)
- Key Derivation module outputs the specific key relevant to the data item
- This key is used to encrypt the related data item
Key Derivation during AB Execution

• AB receives access request to a data item from a service
• AB authenticates the service and authorizes its request
• The information generated during the execution of different AB modules and the digests of these modules and their resources (such as authentication (authentication code, CA certificate that it uses), authorization (authorization code, applicable policies, policy evaluation code)) are collected and aggregated into a single value for each data item
• The value for each data item is input into the Key Derivation module (such as SecretKeyFactory, PBEKeySpec, SecretKeySpec provided by javax.crypto library)
• The Key Derivation module outputs the specific key relevant to the data item
• This key is used to decrypt the requested data item
• If any module fails (i.e. service is not authentic or the request is not authorized) or is tampered, the derived is incorrect and the data is not decrypted
Active Bundle Solution Components

- Service authentication can be based on
  - Password, Certificate, Biometric, PKI

- Service request authorization is based on policy evaluation
  - Flexible policy specification based on access control models such as Attribute/Role-based

- Key management for data disclosure
  - Key inclusion (prone to attacks)
  - Centralized key management service (use of trusted third party for key storage and distribution)
  - Distributed key management that splits the keys into shares using threshold secret sharing and uses a Distributed Hash Table (DHT) to store the shares and reconstruct the key by retrieving minimum threshold number of shares (unsuitable for real-time interactions in a service environment)
  - Dynamic key derivation based on the unique information generated in AB execution control flow steps only if the service is authenticated and authorized

- Tamper Resistance
  - Correct data dissemination depends on the correct execution of AB control flow steps
  - Verify the integrity of the execution steps to ensure there is no difference from the original code (using secure one-way hash function)
  - Derive keys based on digests of AB execution steps and their resources
  - Any modification of AB changes the digest resulting in incorrect key derivation
Context-Sensitive Data Disclosure

• Perfect data dissemination not always desirable
  – Example: Confidential business data shared within an office but not outside

• Context-sensitive AB evaporation
  – AB evaporates in proportion to their “distance” from their owner

• “Closer” subscribers trusted more than “distant” ones

• Illegitimate disclosures more probable at less trusted “distant” guardians

• Different distance metrics
  – Context-dependent
Isolated AB Execution

• Recipient may be reluctant to execute AB => we support the isolated execution of AB by means of Docker.
  – Docker is based on Linux container which is light-weight virtual machine
  – When AB arrives at recipient machine, one virtual machine is created and AB is copied into that virtual machine.
  – AB can be executed inside virtual machine. Only the result returns to host machine
Secure End-to-End Information Flow in Cloud

1. Client sends request to the service using browser and shares data by means of Active Bundle (AB)

2. Service checks the request source (secure or insecure browser)
   – Based on W3C Crypto standards

3. Service executes AB in Cloud if created by an insecure browser

4. Service interacts with AB and requests data

5. AB behaves differently under different contexts
   – Full data dissemination based on service authorization/trust level
   – Context-based partial data dissemination based on insufficient authorization level
   – No data dissemination for unauthorized access/attacks

6. Cross-domain information exchange with trustworthy/untrustworthy subscribers
   – Data dissemination is done on a “need to know” basis by limiting the disclosure of decryption keys
   – Incremental disclosure of keys based on increase in the “need”