Authentication of User’s Device and Browser for Data Access in Untrusted Cloud

Denis Ulybyshev¹, Bharat Bhargava¹, Leon Li², Jason Kobes², Donald Steiner², Harry Halpin⁴, ByungChan An¹, Miguel Villarreal¹, Rohit Ranchal³

¹Computer Science and CERIAS, Purdue University; ²NGC Research Consortium; ³IBM Watson Health Cloud; ⁴MIT

MOTIVATION

- Data (D) = [d₁, ..., dₙ]
- Access Control Policies (P) = [p₁, ..., pₙ]

PROPOSED SOLUTION

- Notification Script (NS) is attached to each key-value pair when extracted from AB
- Watermarks are embedded into data
- NS notifies Service Monitor (SM) on: 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

SERVICE X

D, P

SERVICE Y

D, P
d₁

UNKnown
DOMAIN

SERVICE 1

D₁, d₁

SERVICE 2

D₂, d₂

SERVICE 3

D₃, d₃

ACTIVITY BUNDLE (AB)

Policy Enforcement
Engine

Sensitive Data

Policy

Policies

ACCESS CONTROL POLICIES

- Data (D) = [d₁, ..., dₙ]
- Access Control Policies (P) = [p₁, ..., pₙ]

PROBLEMS

- Opaque data sharing
- Undetected data leakage

OBJECTIVES

- Detect data leakage to unauthorized services
- Authorized service should only be able to access data items for which it is authorized
- Unauthorized service should not be able to access any data item
- Provide data dissemination based on cryptographic capabilities of client’s browser
- Support different authentication methods for client service

FEATURES

- Selective dissemination based on access control policies, browser crypto capabilities, authentication method, source network
- Supports different authentication methods
- Independent of source availability
- Independent of trusted third parties
- Ability to operate in untrusted environment
- Reduced host liability for data

IMPLEMENTATION

- AB implemented as an executable JAR file
- Apache-thrift based API
- JSON-based policies
- WSO2 Balana-based policy engine
- Node.js-based SOA architecture
- RESTful web-services

AUTHENTICATION TICKET CREATION AND VALIDATION

Authentication Server:

- Knows shared secret K and Private Key PrivKey
- Ticket_Info = (Auth_Level, Expiration_Time, Client_ID, Client_Role, Request_Field)
- Enc_Ticket_Info = EncPrivKey(Enc_SHA512(Enc_Ticket_Info))
- Ticket = <Enc_Ticket_Info, Ticket_Signature>

Doctor, Insurance or Researcher Service:

- Knows shared secret K and Public Key PubKey
- Receives Ticket = <Enc_Ticket_Info, Ticket_Signature>
- Checks: DecPubKey(Ticket_Signature) = SHA512(Enc_Ticket_Info)
- Gets data: DecAES256, (Enc_Ticket_Info)

FUTURE WORK

- Implement data leakage detection based on watermarks
- Support database of ABs in Hospital Information System
- Comprehensive performance and scalability evaluation
- Support isolated AB execution AB (Linux Docker Container)

ACKNOWLEDGEMENT: This research is supported by NGC Research Consortium