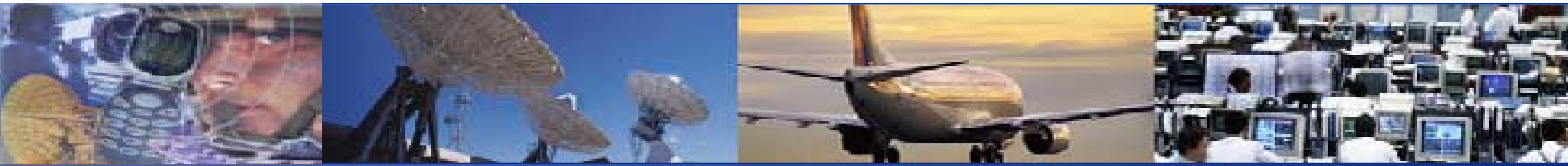
# Using CORBA Object References as Authorization Tokens — A Good Idea or Not ?



Sebastian.Staamann@PrismTech.com
Christoph.Becker@PrismTech.com

**PrismTech Security Solutions**

## Implicit Authorization: A Common Practice

- Often, CORBA based application systems and architectures (also in RTE systems) use object references as a means of implicit authorization.
- Implicit authorization (in this context) means handing over an object reference to a client object does not only provide the necessary addressing and context information but also authorizes the client entity to access the respective object.
- An implicit assumption in such schemes is that other entities which have not legally received an object reference for the object can not access the object. (object references are often provided by a CORBA Name Service instance only and access to the Name Services is restricted to legal client entities).
- The main advantage of the implicit authorization principle is that it avoids the need for the implementation of an additional logic consisting of explicit access rights basically just restate the interaction logic between the CORBA objects.

## The Problem with Implicit Authorization

- The overall system must, with the required assurance, guarantee that the implicit authorization mechanism cannot be circumvented.
- Possible threats for naïve implementations of implicit authorization are:
- External attackers or rogue software can illegally obtain object references (e.g., via unprotected access to Name Service, or by eavesdropping communication links)…
- External attackers or rogue software can create valid object references without having legally obtained them (fabrication, guessing)…
- …and access objects in non-intended ways, thus undermining implicit authorization.

## CORBA Object References – a Closer Look

- A CORBA object reference is a handle for a particular CORBA object implemented by a server.
- A CORBA object reference identifies the same CORBA object each time the reference is used to invoke a method on the object.
- A CORBA object may have multiple, distinct object references.
- **A CORBA object reference contains all information for the client side ORB to locate and access the respective object,**
- Has a well-defined structure
- Object key is vendor-specific and is not meant to be interpreted by the client side ORB

## How to Fabricate Object References?

- First, analyze the target ORB
  - Open Source: look how the ORB builds the References
  - Close Source: generate objects and analyze their references
- Analysis results
  - Object-Counter
  - POA-Counter, POA-Names, POA-Hierarchies
  - TypeIds
  - Timestamps
- What's the difference?
  - Object to Object: Object-Counter
  - POA to POA: „POA-ID“

## Object Key: One Example ORB

- Object Key (hex representation):

```
3a 3e 02 31 31 0c 5b 22 31 e4 d0 c1 e5 31 c3 f4 07 e3
08 00 00 00 00 00 00 00 01 (object no. 1)

3a 3e 02 31 31 0c 5b 22 31 e4 d0 c1 e5 31 c3 f4 07 e3
08 00 00 00 00 00 00 00 02 (object no. 2)
```

- **Guess, what object key #100 looks like:**
  `[…] 00 00 64`

- **Object #10.000:**
  `[…] 00 27 10`

## POA ID: One Example ORB

- Our first POA
  ```
  3a 3e 02 31 31 0c 5b 22 31 e4 d0 c1 e5 31 c3 f4 07 e3
  08 00 00 00 00 00 00 00 01
  ```
- Another POA
  ```
  3a 3e 02 31 31 0c 5e 33 4d b6 5c d4 9a c2 1e 30 da 60
  08 00 00 00 00 00 00 00 01
  ```

- 12Byte difference from POA to POA -> POA-ID?
- 12Byte = 256^12 = too much for brute force guessing

- Attacker’s conclusion:
  Only guess the for the same POA, but for further objects

## Complete Example Attack

- We need
  - the Interface of our target (TypeID, IDL)
  - one other IOR (or corbaloc URL), e.g.

```
corbaloc::1.2@10.10.1.10.:3285/%3a%3e%02...%00%00%00%01
```

- We get (a valid)
  - ORB-Type
  - Host:Port
  - POA-ID (or -name or ..)
- We want access OTHER objects
- We change the Object-Counter

- Testing all possible Object References with object._non_existent()

## Result and Practical Tests

- Result:
  - all active objects of this POA accessible
  - possibility to test for known interfaces
- Context information:
  - most POAs don't activate >100k objects
  - 100k objects --> 100k tries
  - at least 500 tries/s possible
  - attacker knows all objects after 2000 sec (33 min)

## Another Example

- Another example ORB… other rules
- Object Reference:
  ```
  /777/1139427943/0/_0  (obj. #1)
  /777/1139427943/0/_1  (obj. #2)
  ```
- **Other POA, same ORB:**
  ```
  /777/1139427943/1/_0  (obj. #1)
  /777/1139427943/1/_1  (obj. #2)
  ```
- Structure is:
  ```
  /<orb_specific1>/<orb_spec2>/<poa#>/_<obj#>
  ```
- 2 Counters (POA/Object)
- **We could scan all POAs and all Objects**

## All Tests Summarized

- easy to gain access to other objects if access was granted once
- possible to gain access to any object if network access is possible (depends on the ORB)
- possible to scan all POAs

- Proof-of-Concept implementation for scanner was easy to build...

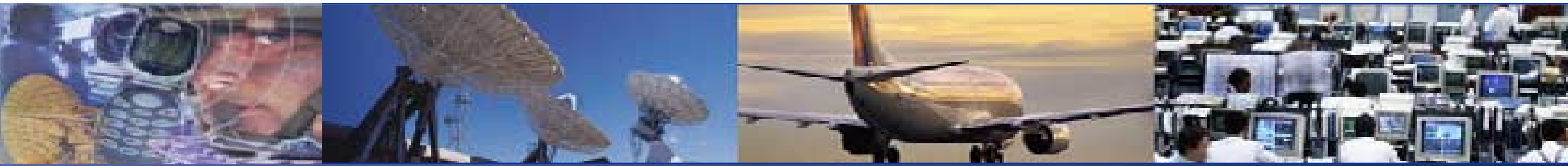
## Conclusion

- Implicit authorization principle can indeed avoid the need for the implementation of an additional logic consisting of explicit access rights basically just restate the interaction logic between the CORBA objects.

- However, before relying on implicit authorization (using CORBA object references as authorization tokens ), additional measures must be implemented as part of the CORBA infrastructure that rule out that illegally obtained objects references can be used to illegally access objects.

PRISMTECH
Productivity Tools & Middleware

# Using CORBA Object References as Authorization Tokens — A Good Idea or Not ?



Sebastian.Staamann@PrismTech.com
Christoph.Becker@PrismTech.com

**PrismTech Security Solutions**