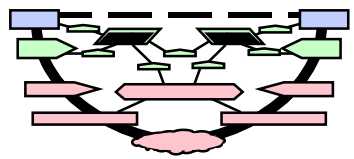
Defense Enabling Using QuO: Experience in Building Survivable CORBA Applications

Chris Jones, Partha Pal, Franklin Webber

BBN Technologies



QuO & APOD

APOD Overview

- APOD is toolkit of mechanism wrappers and adaptation strategies that allows an application to use security mechanisms to dynamically adapt to a changing environment
- We believe that adaptation increases an application's resiliency to certain kinds of attacks
- APOD uses QuO, which provides an application with adaptation.

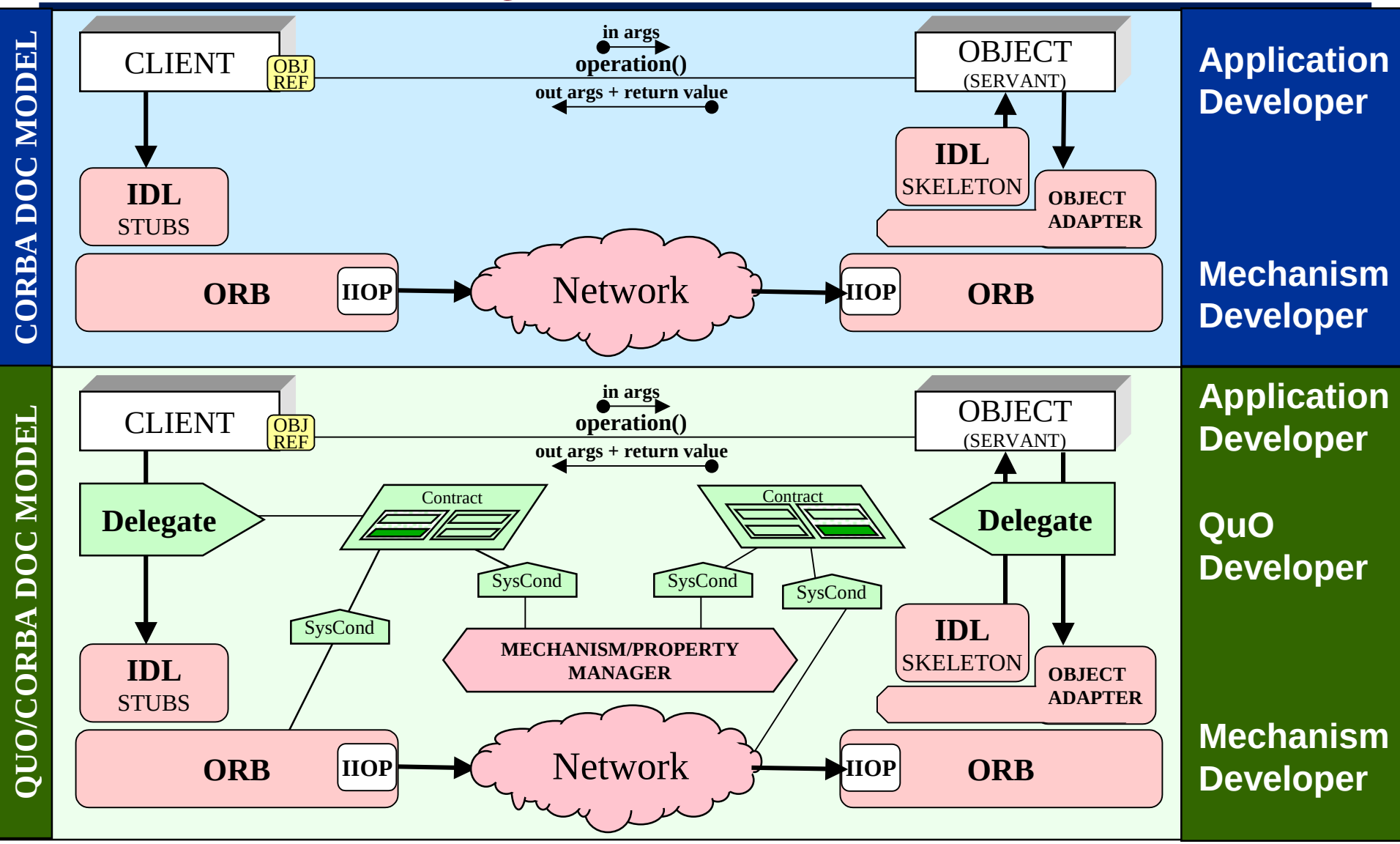
Quo Overview

- QuO is a middleware system that offers an application the ability to adapt to the changing environment in which it is running.
  - Host resources (CPU and memory) usage
  - Network resource availability
  - Intrusion status
- The adaptive code is separated into a QuO “qosket” for reuse.
  - A qosket is a set of specifications and implementations that define a quality of service module.
- It can be added into a distributed object application with minimum impact on the application

QuO Overview (cont.)

- Quality Description Languages (QDL)
  - Contract description language, adaptive behavior description language
  - Code generators that generate Java and C++ code for contracts, delegates, creation, and initialization
- System Condition Objects
  - Provide interfaces to resources, managers, and mechanisms
- QuO Runtime Kernel
  - Contract evaluator
  - Factory object which instantiates contract and system condition objects

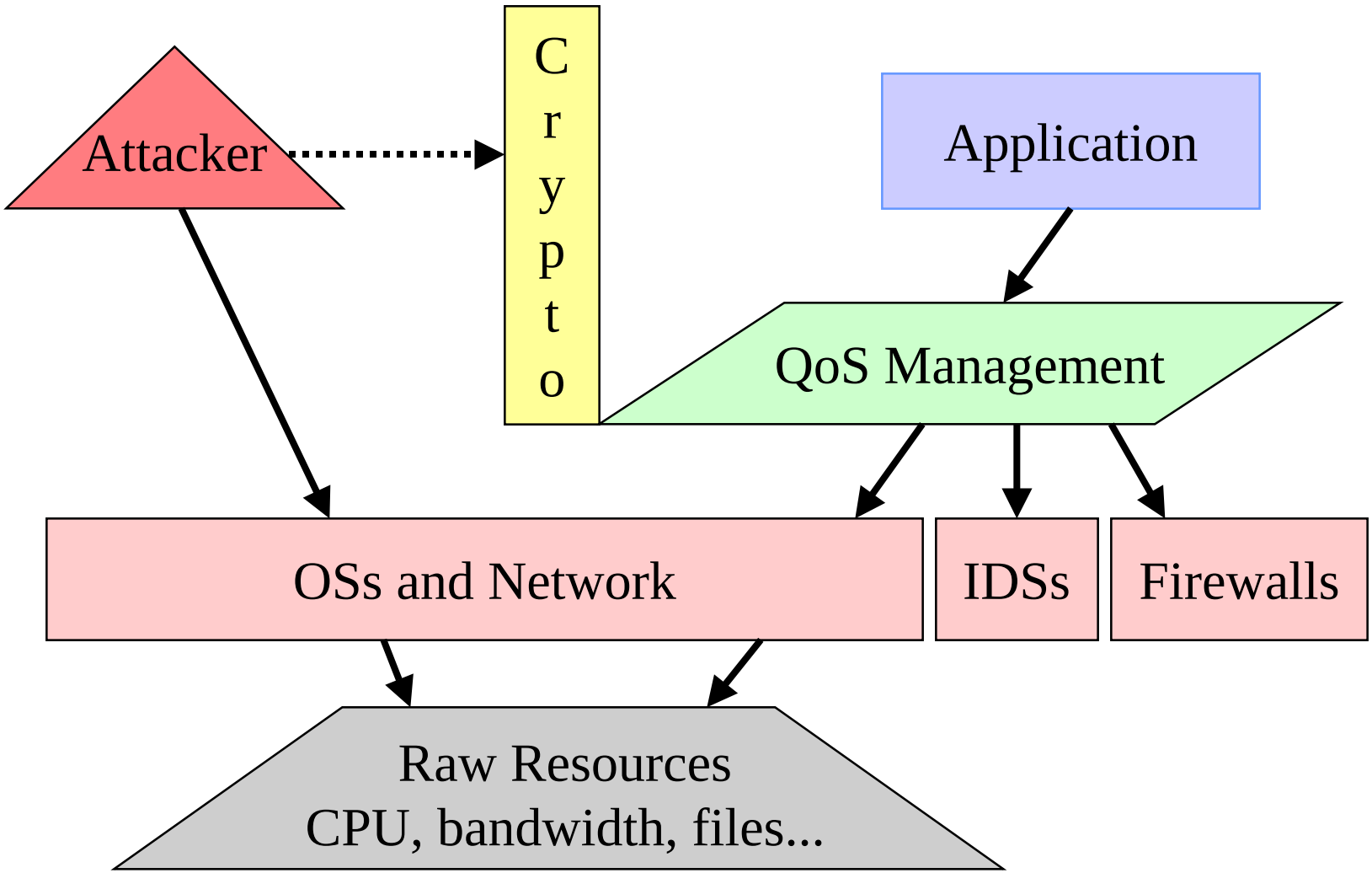
QuO Architecture



APOD Description

- We believe that by adapting to and trying to control its environment, an application can increase its chances of survival under attack
  - Use QuO to integrate multiple security mechanisms into a coherent strategy for adaptation
  - Use mechanisms where they exist to harden or protect an application, a resource, or a service
- Tie security information to the adaptation of an application through the QuO system condition objects
- QuO's contract language is used to define defensive strategies that a
  - Example strategies include
    - » containment, which uses snort and iptables to detect and limits the extent of attacks
    - » outrun, uses dependability with replication to move away from attacks

Defense-Enabled Application Competes With Attacker for Control of Resources



APOD Mechanisms

- Network and Host Sensors
  - Snort is a lightweight network intrusion detection system
  - Tripwire for detecting file systems integrity violations
- Actuators
  - Network traffic filters - Iptables
  - File systems recovery – secure backup
- Dependability management using replication
  - Aqua – replication system from University of Illinois, Urbana-Champaign
  - APOD Bus – mechanism for publishing data about application's status and for maintaining replicas of application processes
- Bandwidth Management
  - Intserv (RSVP, SecureRSVP) and Diffserv
- Access Control
  - NAI's OO-DTE at the interceptor layer

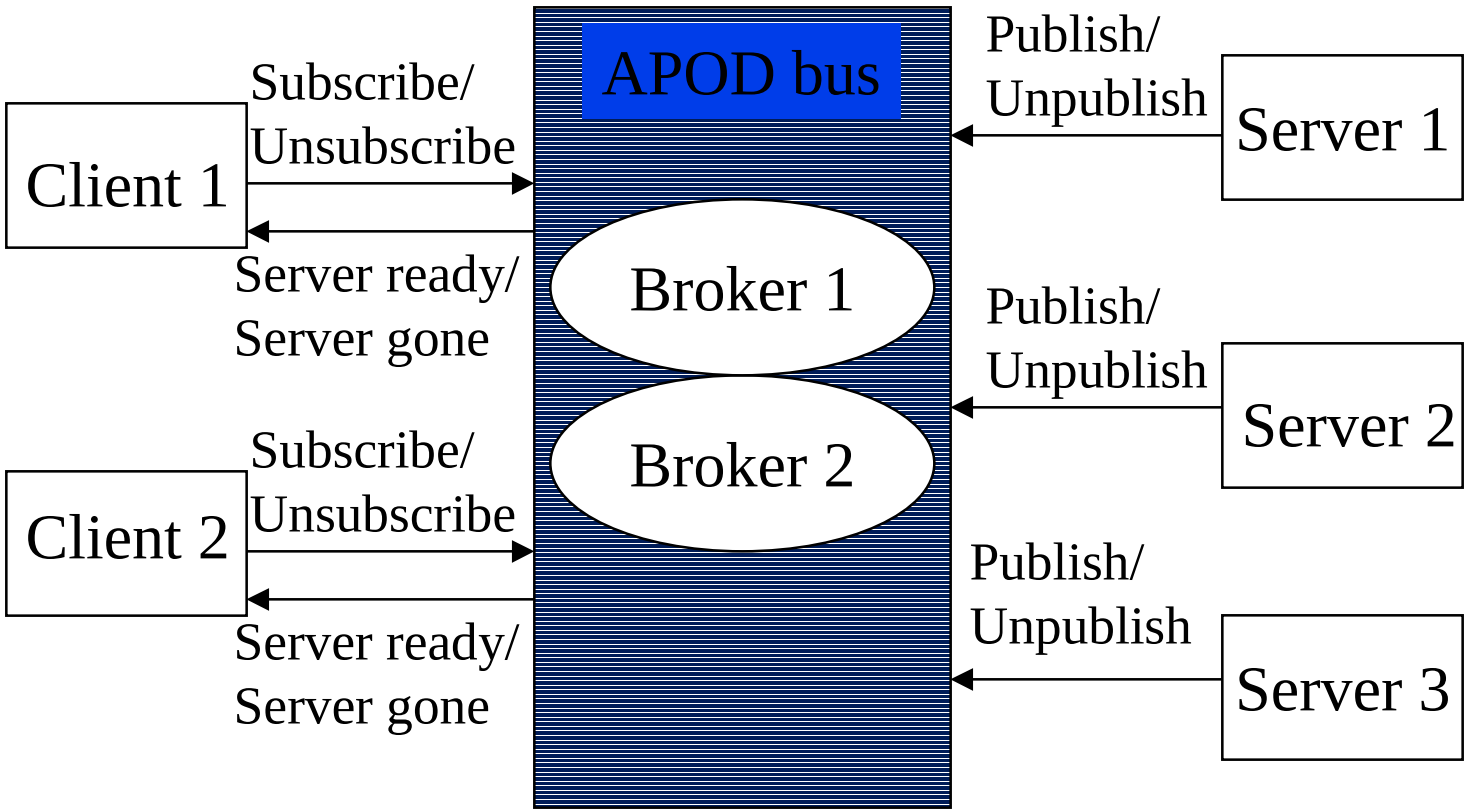
APOD Strategies

- QuO's contract language is used to define defensive strategies
- Example strategies include
  - containment, which uses snort and iptables to detect and limits the extent of attacks
  - outrun, uses dependability with replication to move away from attacks.

APOD Red-teaming Experimentation

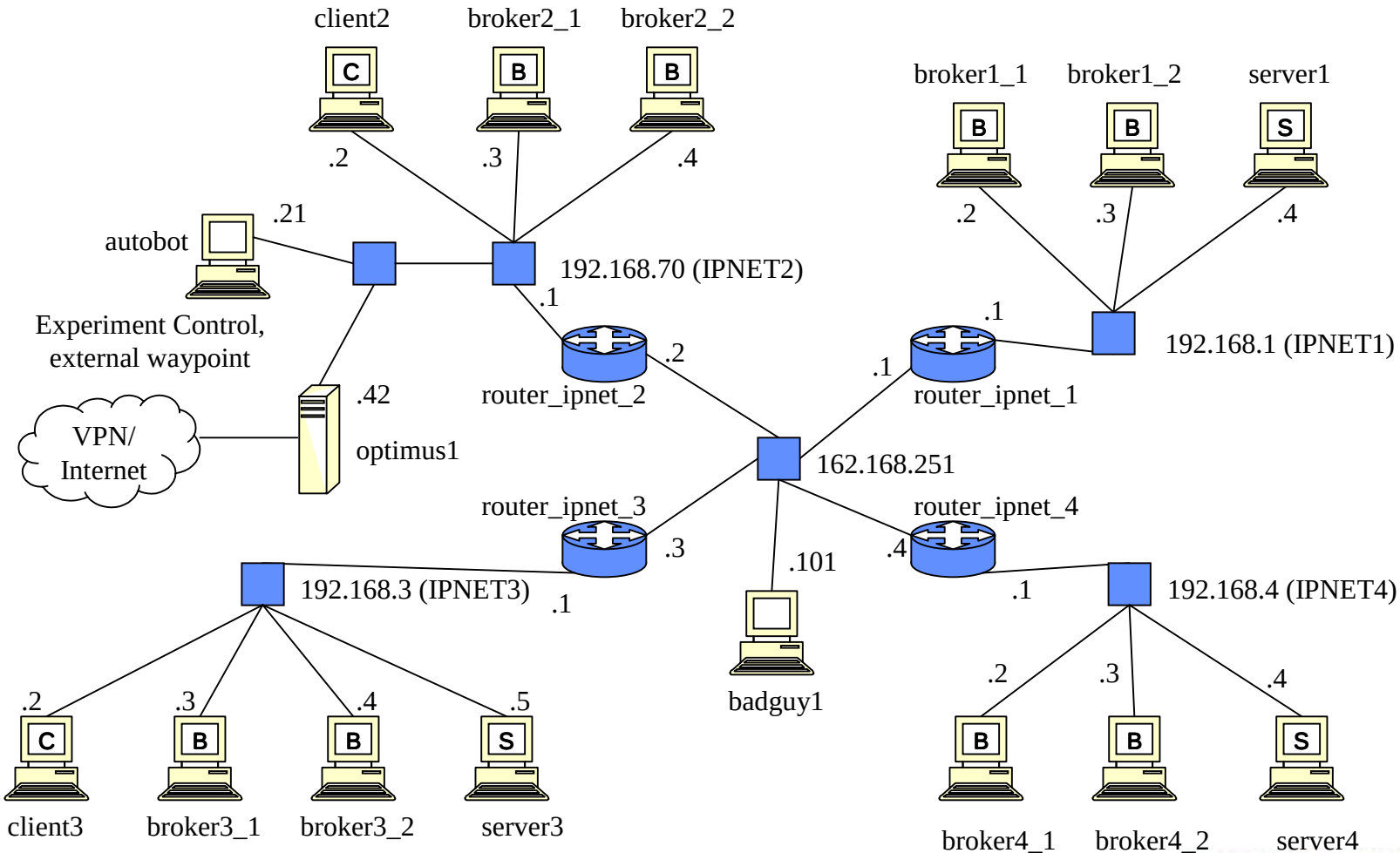
- Sandia labs redteam attacks an APOD enabled application
- Test the added value of APOD to an application
  - Also, analyzing the overhead of APOD
- Application has three pieces; client, image server, and broker. The broker is responsible for hooking clients requesting images to the server that can provide those images
- Two broker components are maintained by the APOD bus on a set hosts. Each of these hosts is running a qosket implementing a containment strategy
  - Using snort and tripwire as sensors and iptables and the bus to react to attack detections.

APOD Redteam Example



Experimentation Configuration

- Testing environment of 14 hosts on 4 subnets.



Red-teaming Attack and Preliminary Results

- With APOD and the configuration, the redteam was forced to combine three different attacks to cause a denial of service of the broker
  - The three attack are: ARP cache poisoning, Spoofing, connection flooding
  - The combined attacks took 5 minutes and we have implemented counter defenses
- APOD has added value and the method call latency overhead of QuO and APOD to application is very small (~ .3 msec.)
  - note: encryption at interceptor layer not included, but measured by APOD team as roughly 500 msec per method call using symmetric encryption can improve this number.
  - Ipsec overhead compared to interceptor layer is very small

What APOD would like from DOC security

- Better elimination of software flaws in ORB implementations
- Standardized support across CORBA implementations for
  - pluggable transports
  - interceptors
- “Knobs and Dials” at ORB and service level to adjust security
  - Easy configuration of ORB’s port selection dynamically under middleware control.
    - » Specific ports or range of ports
    - » How to select the next port to use
- Configuration of timeouts and how to deal with them.

Conclusion

- The red teaming has been extremely useful for APOD
  - Validated our work and “stress test” our defenses
- CORBA support needs to as secure and uniform as possible
  - Found lack of uniformity in implementations
  - Trusting the ORB is secure
- Websites:
  - QuO: www.dist-systems.bbn.com/tech/QuO
  - APOD: www.dist-systems.bbn.com/projects/APOD