

---

# APOD-2

## *An Experiment combining APOD and SE-RSVP*

Doug Reeves (NCSU):

Bill Nelson (BBN):

John Clem (Sandia):

SE-RSVP Overview

Experimental Results

Attacks and Red Team Observations

Fault Tolerant Networks PI Meeting, Newport RI

25 July 2002

---

# Security-Enhanced RSVP

Douglas S. Reeves, NC State University  
reeves@eos.ncsu.edu

Fault Tolerant Networks PI Meeting, Newport RI  
25 July 2002

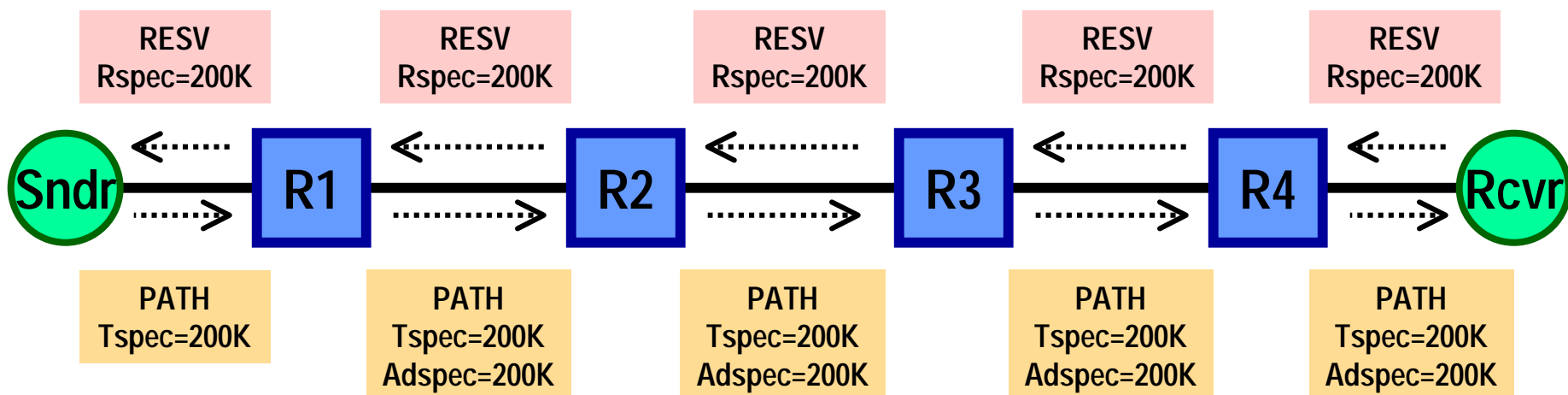
# RSVP Purpose

---

- Resource Reservation Protocol (RFC2205) allows bandwidth between a sender and a receiver to be reserved
  - ◆ Guaranteed quality of service (QoS)
- Steps
  1. The Sender sends a PATH message to the receiver
  2. The Receiver responds with a RESV message to the sender, and bandwidth is reserved at routers along the path
  3. Reservations are periodically refreshed by regenerating the PATH / RESV messages
  4. Reservations may be deleted implicitly (failure to refresh), or by explicitly tearing them down (PATHTEAR and RESVTEAR messages)

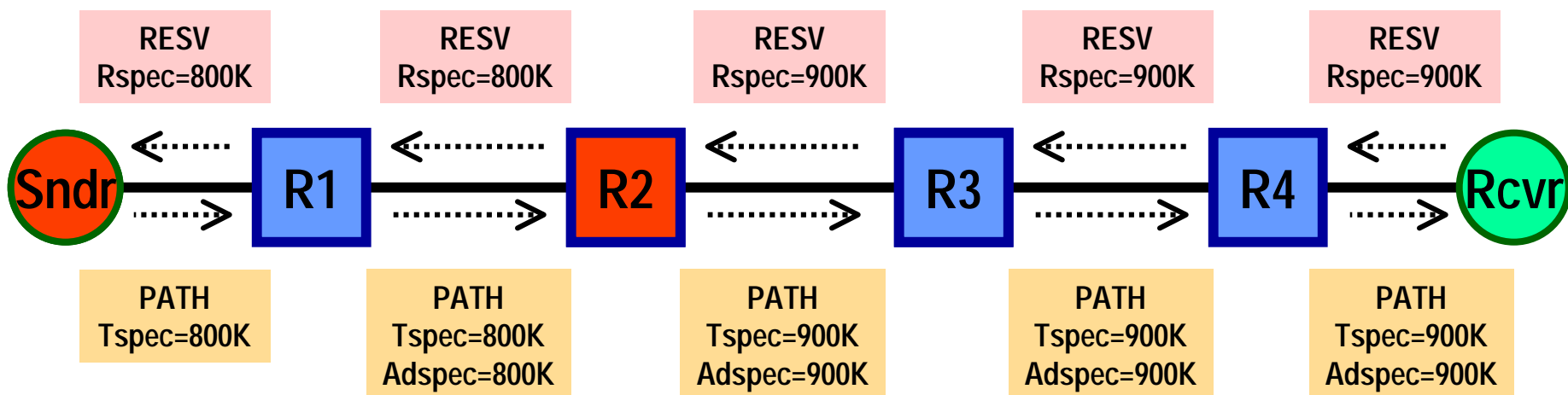
# RSVP Messages

- The PATH message TSpec object describes the bandwidth requirements of the sender's flow
- The PATH message Adspec object describes the minimum available bandwidth of routers on the path
- The RESV message describes the bandwidth that should be reserved for this flow



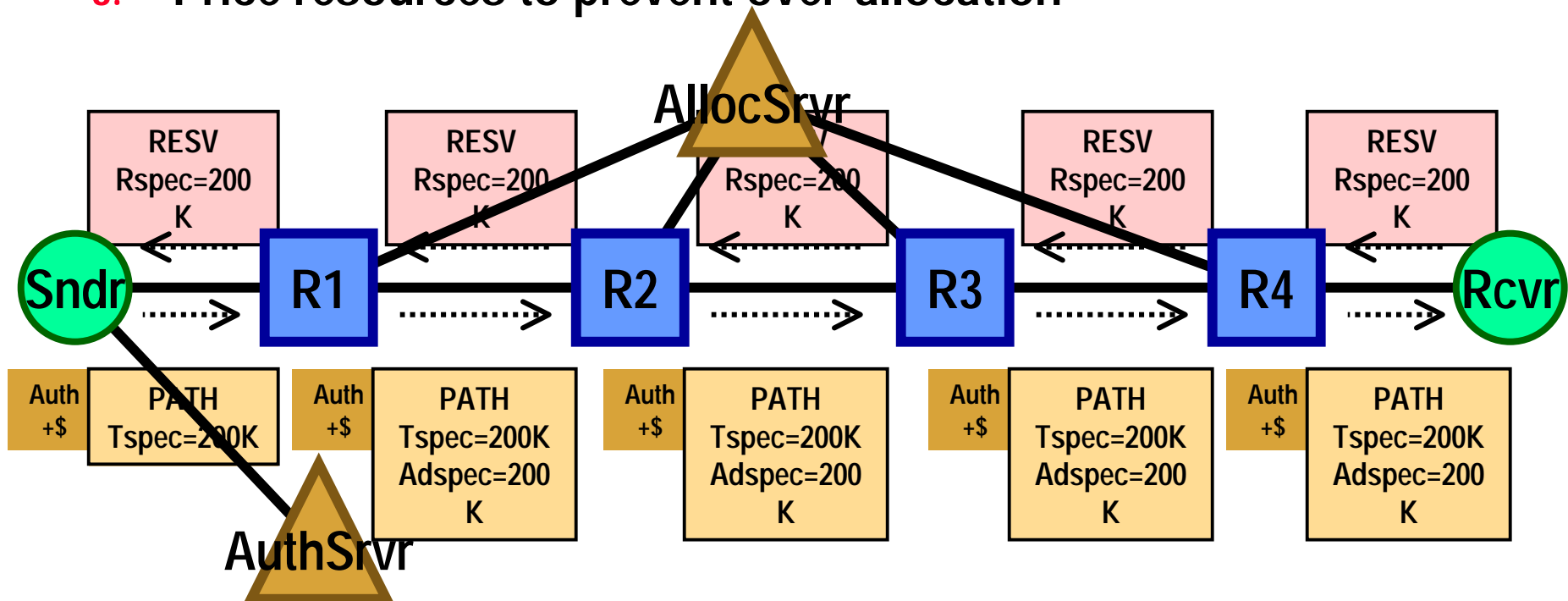
# RSVP Vulnerabilities

1. *Anyone* may request reservations, for any amount of bandwidth
2. Corrupted routers may invalidly modify the contents of the RSVP messages
3. No standard method for deciding how to allocate bandwidth to competing users



# Our Enhancements to RSVP

1. Authorize users making requests, and verify their "credit" level
2. Authenticate RSVP message contents hop-by-hop, detect illegal modifications on reverse path
3. Price resources to prevent over-allocation



# Protect QoS Mechanisms from Attack

---

- Goal 1: authenticate RSVP messages and user request authorization
- Progress: integrated with the APOD project, tested by BBN / Sandia
- Goal 2: Integrate service initiation (e.g. phone call) with resource authorization / reservation
- Progress: draft specifying solution (modifications required to COPS)
- Goal 3: protect reliable multicast against malicious receivers
- Progress: (i) demonstrated the problem, (ii) proposed auction-based solution, (iii) to be presented at ICQT 2002

# Resource Pricing

---

- Goal 1: allocate capacity to network links, and route traffic, to ensure different service classes all receive required QoS
- Progress: (i) formulated as constrained optimization problem, (ii) extensive testing, close to optimal, "reasonable" running time
- Goal 2: minimize number of price changes, despite changing user demands
- Progress: (i) demonstrated that a few price changes almost as good as unlimited number of price changes, (ii) to be presented at ICQT 2002
- Goal 3: find bidding strategies for "bundles" (combinations) of resources
- Progress: showed that evolutionary approach effective at discovering optimal bidding strategies

# Attack Detection and Tracing

---

- Goal 1: detect attacks on DiffServ
- Progress: (i) developed two methods of statistical anomaly intrusion detection, (ii) extensive experimentation, (iii) effective detection, low false positive rate
- Goal 2: trace attack traffic across "stepping stones" (intermediate hosts)
- Progress: (i) demonstrated (across Internet) inter-packet delays uniquely correlate the flows in an attack chain, (ii) accepted for ESORICS 2002

# Automatic VPN Configuration

---

- Goal: ensure that security policies are correctly enforced
- Progress: (i) method for synthesizing a correct-by-construction VPN set, (ii) Best Paper award at Policy 2001

---

# APOD-2

## Experimental Results

William H. Nelson, BBN Technologies  
wnelson@bbn.com

Fault Tolerant Networks PI Meeting, Newport RI  
25 July 2002

# Overview

---

- **Experiment design** ( 3 slides )
  - ◆ Experiment methodology
  - ◆ Hypothesis, flags, & metrics
  - ◆ Topology & operational components
- **Experimental results** ( 4 slides )
  - ◆ Scope - summary of runs (table)
  - ◆ Sample baseline plot
  - ◆ Sample attack plot
  - ◆ Time to deny per attack (bar graph)
- **Summary** ( 2 slides )
  - ◆ Conclusions
  - ◆ Lessons learned

# Schedule

- Apr. 2002
  - ◆ Define experiment, draft plan
  - ◆ Hold White Board (11,12<sup>th</sup>)
- May 2002
  - ◆ Integration & testing
- June 2002
  - ◆ Final plan published
  - ◆ Execution begins
- July 2002
  - ◆ Execution concluded
  - ◆ Hot Wash (8<sup>th</sup>)
  - ◆ Data reduction & analysis
  - ◆ PI meeting presentation (25<sup>th</sup>)
- Aug. 2002
  - ◆ Data reduction & analysis
  - ◆ Report prepared
- Sept. 2002
  - ◆ Final report published

# People

- Blue Team
  - ◆ APOD (BBN Technology)
    - Franklin Webber
    - Partha Pal
    - Michael Atighetchi
    - Chris Jones
    - Paul Rubel
  - ◆ SE-RSVP (North Carolina State University)
    - Doug Reeves
    - Kehang Wu
    - Khurram Matin Khan
- White team (BBN Technology)
  - ◆ Ken Theriault
  - ◆ Bill Nelson
  - ◆ Wilson Farrell
  - ◆ Lyle Sudin
  - ◆ Marla Shepard
- Red Team (Sandia National Labs)
  - ◆ David Duggan
  - ◆ John Clem
  - ◆ Dino Dai Zovi
  - ◆ Michael Berg

# Experiment methodology

- Utilize a laboratory test network
  - ◆ Networked image serving system (brokers, image servers, and clients)
  - ◆ Mission traffic (images)
  - ◆ Defense (APOD replacement of compromised brokers, APOD LAN containment on routers, and static SE-RSVP bandwidth reservation)
- Controlled introduction of LAN flooding and APOD / SE-RSVP specific attacks
- Scripted clients repeatedly request images from a data base and accumulate latency and delivery statistics

APOD Present	Attacks Present	Purpose
No	No	Image system baseline
No	Yes*	Attack baseline
Yes	No	Performance cost of APOD
Yes	Yes	Defense effectiveness of APOD

\* Attacks do not target APOD; difficult to compare with results of APOD-specific attacks

# Hypothesis, flags, & metrics

---

- Hypothesis

- ◆ Top-level: APOD improves immunity to cyber attacks relative to systems that do not use APOD

- APOD improves immunity to availability attacks relative to systems that do not use APOD
- APOD improves immunity to availability attacks at an acceptable cost to the defender

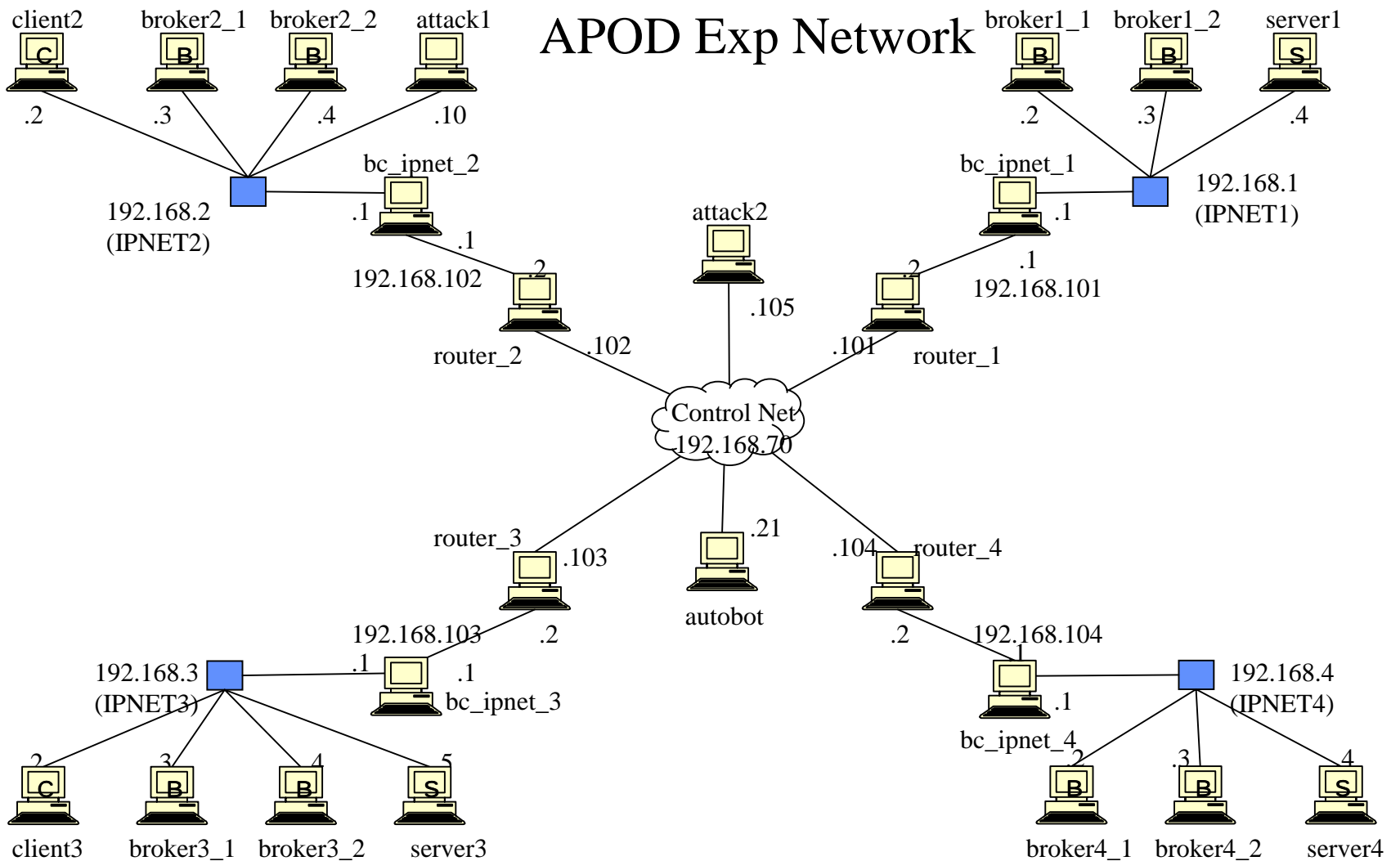
- Opponent flags

- ◆ Degrade image service (e.g. increase latency)
- ◆ Deny availability of images to users

- Principal Metrics

- ◆ Image serving latency across a suite of image clients (degrade flag)
- ◆ Fraction of image requests that do not succeed (deny flag)
- ◆ Time to denial (how long it takes attacker to capture deny flag)

# Topology



# Experiment scope

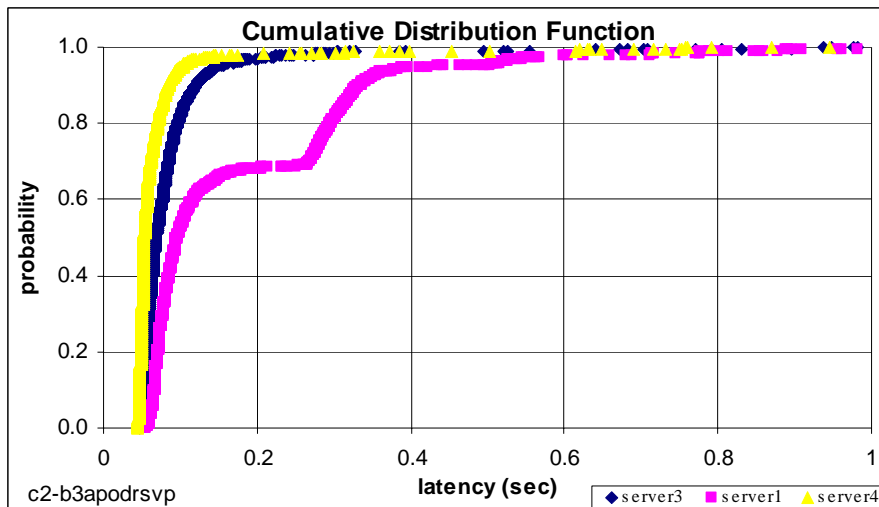
Number	Name	Attack Targeted	Flag Acquired
1	apodrsvp	n/a (baseline)	n/a
2	b2-apodrsvp	n/a (baseline)	n/a
3	noapod_norsvp	n/a (baseline)	n/a
4	apod	n/a (baseline)	n/a
5	rsvp	n/a (baseline)	n/a
6	b3-apodrsvp	n/a (baseline)	n/a
7	client2_server1_flood	APOD	deny
8	spooof_scan_brokers	APOD	none
9	net2_spoofscan_bcbr	APOD	deny
10	spooof_scan_bc	APOD	deny
11	localhost_single_packet	APOD	deny
12	broadping_spoofscanbc3_n2ping	APOD	none
13	echo_replReq_ovr_spoof_mirror	APOD	none
14	xflash_bcspoof_brscan	APOD	deny
15	spooof_block_arp	APOD	deny
16	rsvp_replay	RSVP	degrade
17	fuzz_bogus_blockarp	RSVP/APOD	deny
18	spooof_block_arp2	APOD	deny
19	rsvp_fuzz_craft	RSVP	none
20	spooof_block_arp3	APOD	deny
21	fuzz_flood_takedown	RSVP/APOD	deny, degrade
22	block_bc_outside_int	RSVP/APOD	deny
23	spooof_takedown	APOD	degrade (c2 down 13 min)
24	spooof_block_arp4	APOD	deny
25	spooof_block_compressed	APOD	deny
26	tcpjunk_tcpkill	APOD	½ deny (client 2)

\*Note: The systems under attack had both APOD and RSVP running.

# Baseline system behavior (no attacks)

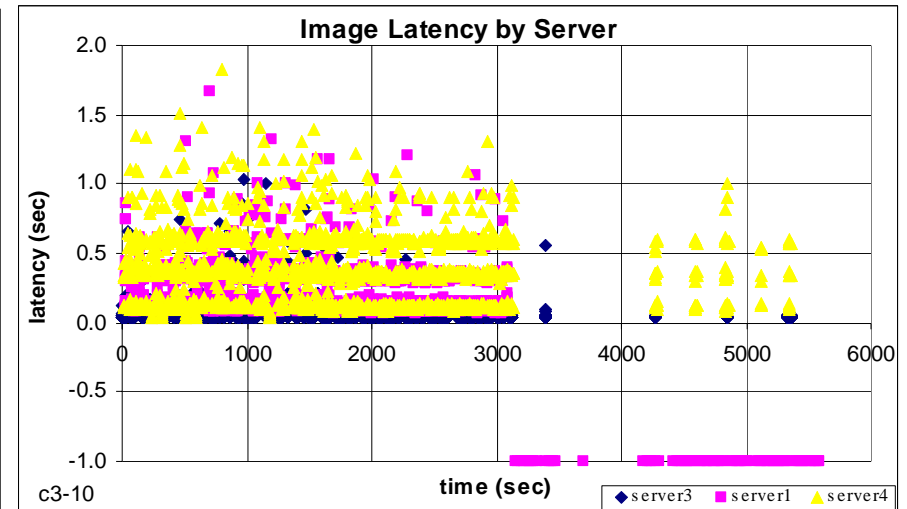
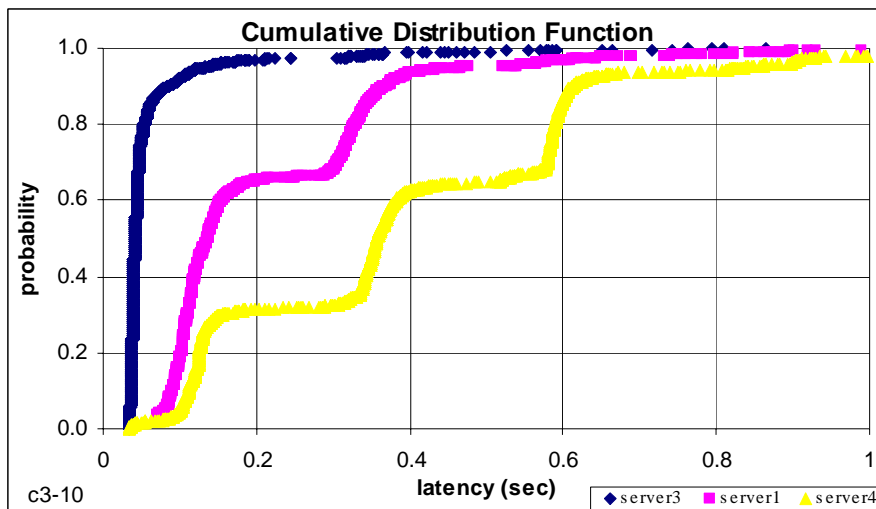
System Description	Average Image Latency (seconds)		
	Server 3 +/- SEM	Server 1 +/- SEM	Server 4 +/- SEM
Image system (IS)	0.060 +/- 0.000	0.127 +/- 0.002	0.051 +/- 0.000
IS + APOD	0.089 +/- 0.002	0.167 +/- 0.004	0.070 +/- 0.002
IS + RSVP	0.060 +/- 0.000	0.126 +/- 0.002	0.051 +/- 0.000
IS + APOD + RSVP	0.088 +/- 0.002	0.175 +/- 0.004	0.068 +/- 0.002

\*Note: SEM = Standard Error of the Mean



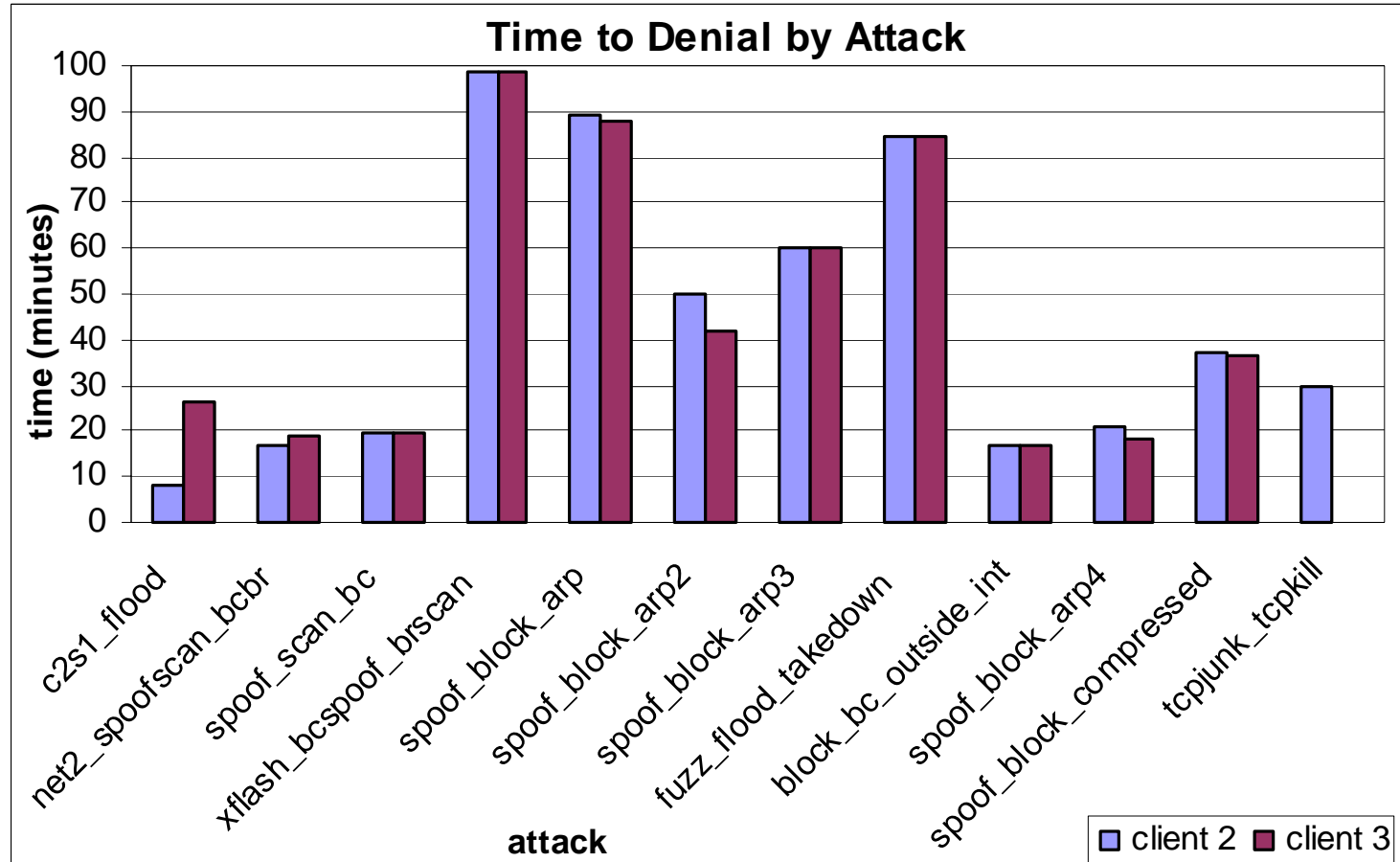
# Sample result, Attack - Client2\_Server1\_flood

- Client 2 stopped receiving images at about 2000 sec. (not shown)
- Client 3 started getting periodic images at about 3100 sec, and stopped receiving images at about 5300 sec. (shown below)
- Client 3 had 72 timeouts accessing Server 1. (e.g. server 1 stopped sending images, indicated by values at -1 on right plot)



# Time to denial (TTD)

- Average TTD: Client 2 = 44.4 minutes, Client 3 = 46.3 minutes
- Spoof\_Block\_Arp attacks, runs 1-4: TTD decreased (88.6 to 19.5 min.)



# Conclusions

---

- Red Team consistently achieved the deny flag ~ 75% attacks
  - ◆ Also large isolated point latencies – due to temporary broker outages
- Performance
  - ◆ APOD-2 always stood its ground till the last broker died
  - ◆ APOD-2 delayed the deny flag in most cases, <TTD> ~ 45 min.
  - ◆ Red Team was unable to overcome SE-RSVP's cryptography
- Cost
  - ◆ APOD-2 mechanisms had larger operational overhead (no attacks)
    - APOD-1 (broker replication), latency ~ 5 %
    - APOD-2 (broker replication, LAN containment, SE-RSVP), latency ~ 40 %
  - ◆ SE-RSVP introduced no additional overhead, latency ~ 0%
    - RSVP daemons crashed (e.g. by themselves, or when Red Team did something)
  - ◆ Added complexity and implementation effort (extra machines)
    - APOD-2 prone to configuration errors, and start up not reliable
    - This complexity introduced vulnerabilities that Red Team took advantage of

---

# APOD-2

## Attacks and Red Team Observations

John F. Clem, Sandia National Labs  
jfclem@sandia.gov

Fault Tolerant Networks PI Meeting, Newport RI  
25 July 2002

# Attack Strategy: Using APOD-2 Against Itself

Goal	Trigger Snort Violation	Sever APOD Bus	Disrupt APOD Comm.
<b>Methods</b>	<u>SpooF scans</u> <ul style="list-style-type: none"><li>● Xmas tree</li><li>● Null</li></ul> <u>Bad Traffic</u> <ul style="list-style-type: none"><li>● Localhost</li></ul>	<u>Arp-spoof</u> (Arp Cache Poison)	<u>TCP connection flood</u>  <u>TCP traffic additions</u>  <u>TCP connection resets</u>
<b>Effects</b>	<ul style="list-style-type: none"><li>● Mark spoofed APOD host "dirty"</li><li>● Propagates IPTables rules to block traffic</li><li>● Keep APOD from re-stabilizing</li></ul>	Effectively isolates APOD hosts	<ul style="list-style-type: none"><li>● Floods introduce latency to system communications</li><li>● Traffic additions to de-sync legitimate system communications</li><li>● Resets to halt legitimate communications</li></ul>
<b>Notes</b>		Limited to local LAN segment of attacking host.	

# Attack Strategy: Attacking SE-RSVP

Goal	Crash RSVP daemons	Interfere with legit RSVP traffic	Confuse RSVP nodes
<b>Methods</b>	<u>Inject bogus RSVP traffic</u> <ul style="list-style-type: none"> <li>● Some with legitimate RSVP header info</li> <li>● Some with illegitimate info</li> </ul>	<u>Inject bogus RSVP traffic</u>  <u>Flood bogus RSVP traffic</u>  <u>Replay RSVP traffic</u>	<u>Fake RSVP teardowns</u>
<b>Effects</b>	Intermittent* <ul style="list-style-type: none"> <li>● Static reservations unaffected by RSVP daemon crashes</li> <li>● RSVP crashed by itself</li> </ul>	<ul style="list-style-type: none"> <li>● Mere injection of traffic had no observable consequence</li> <li>● Flooded traffic introduced latency to system</li> <li>● Replayed RSVP traffic only had effect when flooded</li> </ul>	Intermittent <ul style="list-style-type: none"> <li>● RSVP teardowns not achieved</li> </ul>
<b>Notes</b>			

# APOD-2 Specific Attacks

<b>Name</b>	localhost_single_packet	localhost_scan_bc	localhost_scan_any	spooof_bh_block_bc_spoof_lan	spooof_bh_block_bc_spoof_lan_fast
<b>Attack Strategy Goal</b>	Trigger Snort	Trigger Snort	Trigger Snort	Trigger Snort Sever APOD Bus	Trigger Snort Sever APOD Bus
<b>How it Works (Method)</b>	Sends single, benign packet with a source addr. of 127.0.0.1  Propagates IPTables rule	Spoofs scan from outside interface of boundary controller on the inside interface of the same bc.  Propagates IPTables rule	Spoofs scan with source addr. of 127.0.0.1  Propagates IPTables rule whether or not an exception rule for localhost exists	Multi-stage, sequential attack.	Same as previous. Script contains reduced wait times between stages.
<b>Flag Captured?</b>	Yes(Deny)	Yes(Deny)	Yes(Deny)	Yes(Deny)	Yes(Deny)

# APOD-2 Specific Attacks

Name	rmiflood.c	tcpjunk	tcpkill
<b>Attack Strategy Goal</b>	Disrupt/Deny APOD communications	Disrupt/Deny APOD communications	Disrupt/Deny APOD communications
<b>How it Works (Method)</b>	Consumes all available ports on hosts, interfering with legitimate TCP connection requests.	Attempts to insert traffic in the TCP stream in order to force the TCP connection to re-sync or fail.	Sends TCP packets with reset flag set to terminate connections.
<b>Flag Captured?</b>	Yes(Deny)*	No	½ Deny (when client was on attacker's subnet)

\* Attack was later deemed out-of-bounds

# SE-RSVP Attacks

Name	tcpreplay	rsvp_fuzzer
<b>Attack Strategy Goal</b>	Interfere with legitimate RSVP traffic  Confuse RSVP nodes	Crash RSVP daemons  Interfere with legitimate RSVP traffic  Confuse RSVP nodes
<b>How it Works (Method)</b>	Using captured RSVP traffic from the current session or previous sessions, traffic is replayed on the network from the attacker's host.	Inject crafted bogus RSVP packets, including teardown packets.
<b>Flag Captured?</b>	Yes(degrade)*	No

\*Degrade achieved with flooded RSVP traffic.

# Red Team APOD-2 Observations

---

- APOD-2 software bus and relocation mechanisms performed well
- Defensive strategies
  - ◆ Rate limiting on BC's effective at containing floods
  - ◆ Brokers repelled connection reset attempts
  - ◆ Not "easy" to shutdown all brokers ... live
- Vulnerabilities
  - ◆ Boundary Controllers were the "Achilles heel."
  - ◆ Occasionally clients would not give up on stalled brokers
  - ◆ APOD-2 relocated both active brokers to one IPnet...one instance
  - ◆ Traffic flood on local IPnets
  - ◆ "Easy" to shutdown some brokers...live

# Red Team SE-RSVP observations

---

- Unable to overcome cryptographic challenge of SE-RSVP
- RSVP crashed...
  - ◆ When the Red Team did “something”
  - ◆ By itself (per the White Team)
- RSVP untested during high bandwidth demand

# Experiment lessons learned

---

- **Experimental software code must be frozen**
  - ◆ Avoid experimenting with a moving target – inefficient for Red Team
  - ◆ Avoid changes in code base - nullify previous results
- **Extraneous logging slowed experimentation and filled discs**
  - ◆ Most data logged was not relevant to metrics
  - ◆ One hour baseline run is 508 Mbytes and took 15 minutes to download
- **Start-up and shut-down need to be streamlined**
  - ◆ One-hour baseline run took 2 hours to execute
- **Experimental infrastructure must remain intact after execution to answer questions raised in data analysis**
- **Attacker logging is difficult**
  - ◆ Hard to instrument a live Red Teamer with a computer

# More Information

---

- **APOD Experiment Web Site (on Docushare)**

**<https://archive.ia.isotic.org/>**

- **To download copies of:**
  - ◆ **APOD Experimental Plans**
  - ◆ **APOD Hot Wash's**
  - ◆ **APOD FTN PI Meeting Presentations**
  - ◆ **APOD Final Reports**
- **Go to:**
  - ◆ **DARPA IA&S**
  - ◆ **FTN/DC Experiments**
  - ◆ **APOD Experiments**