

The HACQIT Idea¹

- *Continue delivering critical application services to selected users while under cyber-attack over the network*
 - Implement through adaptive (learning enhanced) specification-based control of security and fault tolerance mechanisms
- Focus on:
 - Building prototype but concentrate on new capabilities and minimize unnecessary duplication with other R&D or COTS
 - Increasing adversary work factor for developing successful attacks
 - Supporting broad classes of COTS HW & SW for near term military utility without footprint of Byzantine fault tolerance
 - Providing extensible (architecture-based) intrusion tolerance framework for longer term utility
- Project goals
 - Deliver critical user services for 4 hours while under active attacks with no more than 25% degradation in user performance
 - Understand the design space of intrusion tolerant systems designed for real world use with COTS/GOTS hardware and software

[1] This work was partially funded by Defense Advanced Research Project Agency under contract #N66001-00-C-8074. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Project Agency or the U.S. Government.

Phased Approach

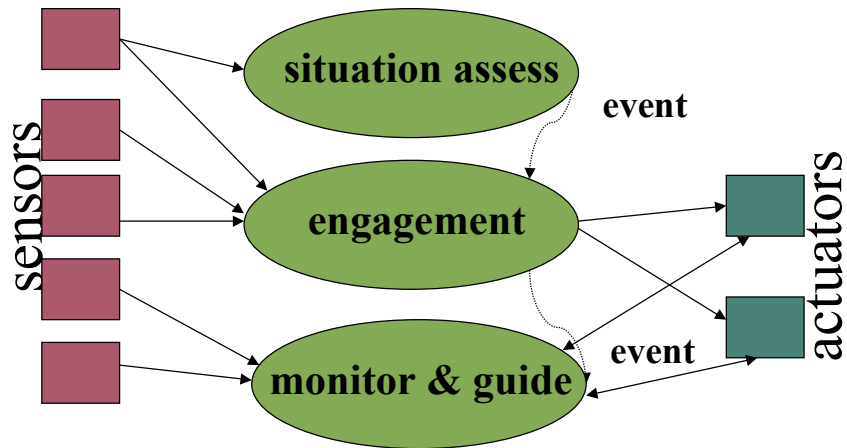
- Phase 1: Explore “ITS space” (9-00 thru 2-01)
 - Investigate capabilities and requirements
 - Leverage Quorum (e.g., Desiderata, other QoS components), fault tolerant and IA mechanisms, etc
 - Analyze more formal models
 - Refine HACQIT architecture and implementation plan
- Phase 2: Evolve Prototype (3-01 thru 9-02)
 - Implement innovative architecture in 2-3 increments
 - Explore several critical applications
 - Continue analytic efforts (e.g., more formal models)
 - Validate claims (Internet exposure, Red Teaming, testing new attacks, & analysis)
- Phase 3: Optional TBD.

Phase 1 Summary

- Phase 1: Explore space of “Intrusion Tolerant Systems”
 - Build demo prototype (throw-away)
 - Leverage Desiderata infrastructure
 - Incorporate fault tolerant and IA mechanisms, etc
 - Investigate capabilities and requirements
 - Analyze more formal models
 - Refine HACQIT architecture and implementation plan

DeSiDeRaTa Background: Real-time Path Paradigm

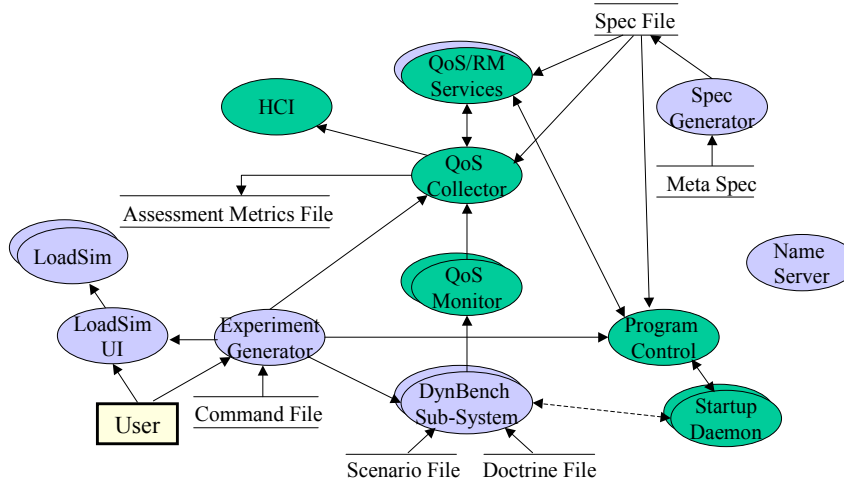
Goal: Meet requirements of real-time paths in “system”



Source: Lonnie Welch, path-based-eng-process.ppt
12/19/2001

Page 5

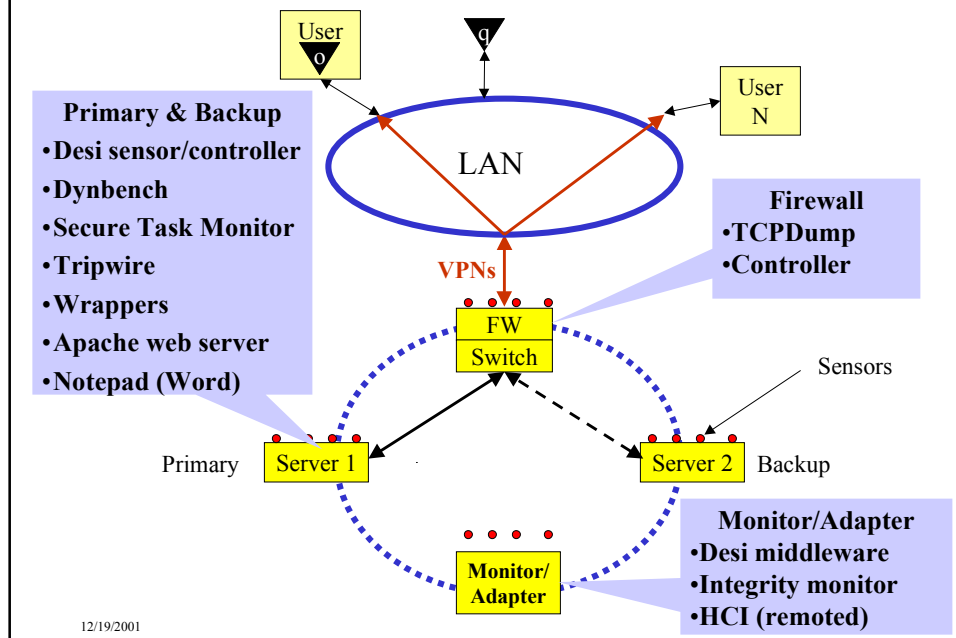
Desiderata Software Architecture



Source: Lonnie Welch, path-based-eng-process.ppt
12/19/2001

Page 6

Experimental Configuration (in Yellow/Blue)



Phase 1 Experiments & Demonstrations

- Desiderata
 - Console based migration (Apache startup and shutdown)
 - Migration behavior under graceful shutdown v. crash
- Prototype Demonstrations (Desi infrastructure with initial Monitor/Adapter & additional sensors)
 - Migration based on integrity violation (Tripwire) on primary with heightened monitoring & auditing
 - Migration based on rogue process detection with process kill and heightened monitoring & auditing
- Experiments (enhanced M/A with wrappers for protection & sensing)
 - Multiple applications (Apache & JAMES)
 - Cross platform migration
 - Simple state capture, save, and restore
 - Attacker address identification and blocking

12/19/2001

Page 8

Phase 1 -- New Capabilities

- New code developed for:
 - Secure Task Manager and heartbeat monitor
 - Sensor manager, e.g., Tripwire and wrappers
 - Response managers: e.g., firewall and auditing
 - Policy driven wrappers
 - Initial monitor/adapter

12/19/2001

Page 9

Lessons Learned -- Phase 1

- Original proposal
 - Strong separation boundaries are important
 - Intrusion resistance is critical
 - Innovative architecture can protect sensors and controls
 - Deal with attacks at most general level possible (e.g., common initiating events or common effects)
- Phase 1 experimentation
 - Attacks through weak clients are very serious but protecting weak clients seems to be beyond scope of project
 - Wrappers are powerful sensor & reaction mechanisms but
 - Intruders can be channeled to attack through applications and a minimal set of OS functions, e.g., the network stack
 - Continuing attacks mean failover is not enough without restoration
- Vulnerability framework analysis
 - Redundant sensors and control mechanisms are needed to deal with unknown vulnerabilities in other components
 - Integrity and confidentiality attacks through weak clients are devastating and weak client protection must be within scope
 - Random rejuvenation is potent weapon against process corruption

12/19/2001

Page 10

Lessons Learned -- Phase 1 (cont.)

- Phase 1 analysis
 - Attacks can be channeled and unknown attacks must be dealt with
 - Work factors to develop new attacks determine resource requirements for adversaries but time to launch new attack is more relevant to HACQIT
 - Diversity is powerful weapon against common mode failure
 - Process pair (hot spare) is best approach for rapid failover
 - Continual recovery is possible
 - If attackers can be prevented from downloading, storing, and/or executing “foreign” software, their attacks must focus on software resident on cluster
- “Worst case scenarios” – design points
 - Continuing unknown attacks that deny/compromise service – particularly those through a critical user that compromise an executing process but do not write out files or kill the process
 - Masquerading as critical user to corrupt or leak critical data

12/19/2001

Page 11

Lessons Learned -- Phase 1 (cont.)

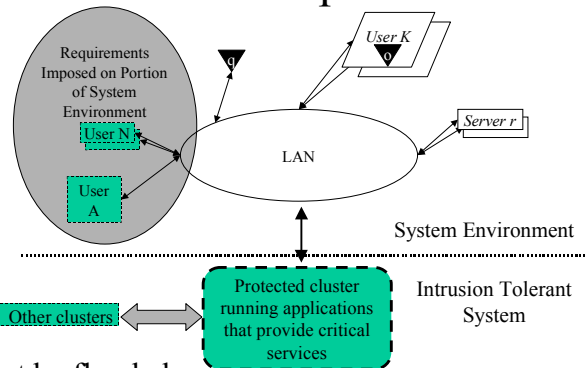
- Important similarities & differences of faults & intrusions
 - Both can be masked, have serious common-mode failure problems, and can propagate
 - Intrusions are malicious, non-random, and repeatable
 - Intrusions occur at SW interfaces & can be stopped if known
- Intrusion detection is hard: intrusion detectors (signature and anomaly based) are poor at detecting unknown attacks
- Specification based behavior approaches seem to work
 - Behavior deviation from specification is FT definition of failure
 - Fault tolerance field is well developed -- large body of knowledge, mechanisms, techniques, and active research
 - Specifying allowable SW behavior is also promising for detecting intrusions but incomplete specifications are problematic
- Learning is essential to survive unknown attacks
 - Adaptive filtering to block new attacks once seen
 - Basis for arguing asymptotic completeness of specifications

12/19/2001

Page 12

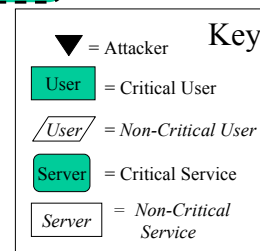
System Model & Assumptions

Goal: Provide critical services to selected users while under attack with <25% degradation in performance



Key Assumptions:

- LAN is reliable, cannot be flooded
- No direct DoS attacks against critical users
- Cluster HW & SW are pristine at startup & have up-to-date patches
- Critical users are trusted
- Unknown vulnerabilities exist in cluster
- Users interact with services via LAN & hosts



12/19/2001

Page 13

Intrusion Tolerant “Algorithm”

- Assume a backup (hot or cold)
- Detect an intrusion:
 - If intrusion does not constitute a threat to the critical application, then start a procedure to expunge the attack and block future occurrences, return;
 - If attack threatens the critical application, then switchover to backup, expunge the attack from the primary, block future occurrences of the attack, return;
- Detect a performance or integrity problem in critical application (including data files), operating system, or other critical process that indicates an undetected intrusion
 - Switchover to backup, expunge the attack from the primary, block future occurrences of the attack, return

12/19/2001

Page 14

Phase 2 – Efforts to Date

- Phase 2: Evolve Prototype
 - Increment 1 of innovative architecture
 - Explore IIS as critical application
 - Continue analytic efforts (e.g., more formal models)
 - Validate claims (Internet exposure, Red Teaming, testing new attacks, & analysis)

12/19/2001

Page 15

Phase 2: Design Goals & Requirements

- Maintain strong separation boundaries to isolate key components from attack and to minimize attack propagation, especially to sensor and control systems
 - Channel attacks in desired directions and defend in depth to cope with unknown attacks
 - Employ diversity if possible to avoid common mode failures
 - Use randomization to force attackers to deal with uncertainty, e.g., randomly rejuvenate servers and critical applications to limit effectiveness and extent of undetected attacks that corrupt executing processes
 - Deal with attacks at more abstract levels such as common initiating events like buffer overflows or common effects like file writes to enhance attack coverage
 - Utilize redundancy in sensors, protection mechanisms, and controllers for all important attack avenues

12/19/2001

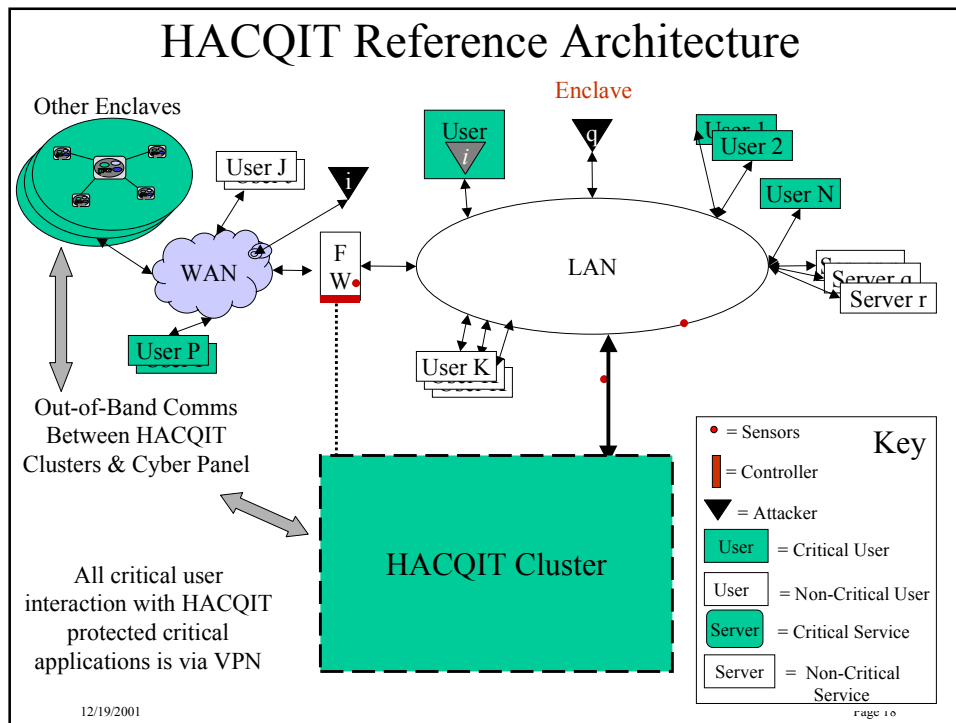
Page 16

Phase 2: Design Goals & Requirements (cont.)

- Employ rapid failover mechanisms
- Handle unknown attacks -- Learn after initial attack to prevent future successful use and communicate new settings to other clusters
- Use policy to define defensive posture & responses and to enable INFOCON-like changes between policy sets across multiple clusters
- Others
 - Maintain consistent replicas and enable rollback to deal with latent successful attacks
 - Reconstitute and reuse compromised servers
 - Provide or assume essential intrusion protections for critical users
 - Minimize false alarms and disruptive responses to them
 - Modularize the design for evolvability

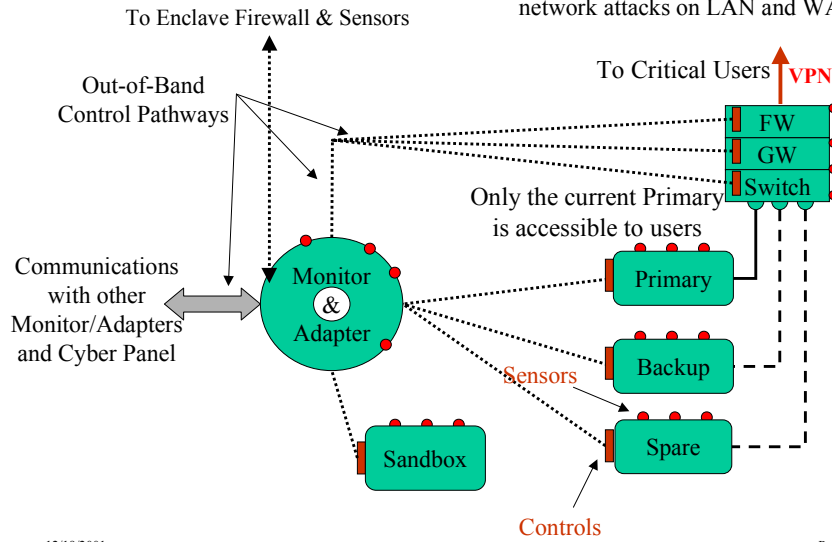
12/19/2001

Page 17



HACQIT Cluster HW Design

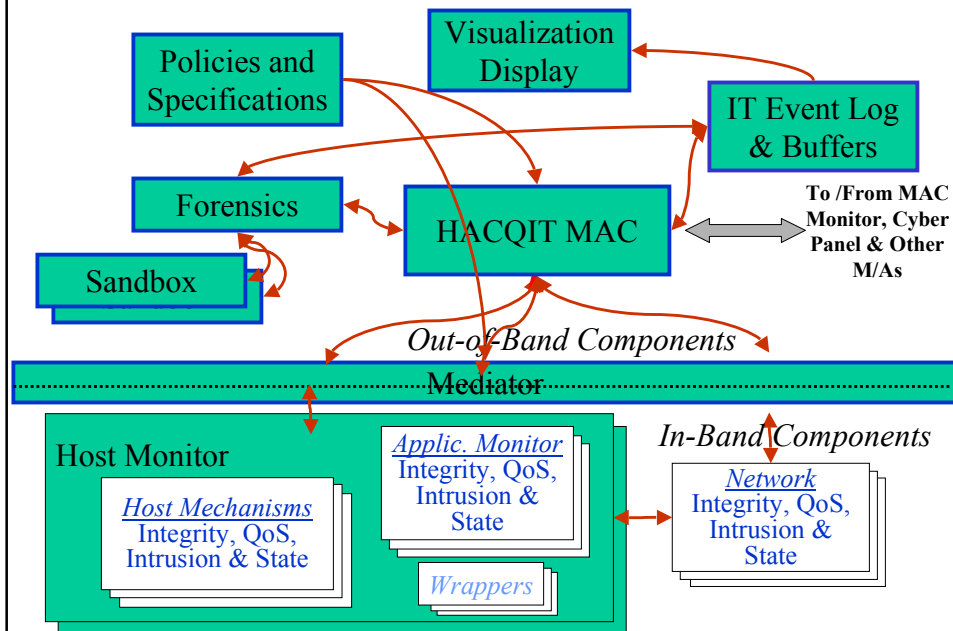
Monitor-adapter uses Out-of-Band signaling for complete separation from network attacks on LAN and WAN



12/19/2001

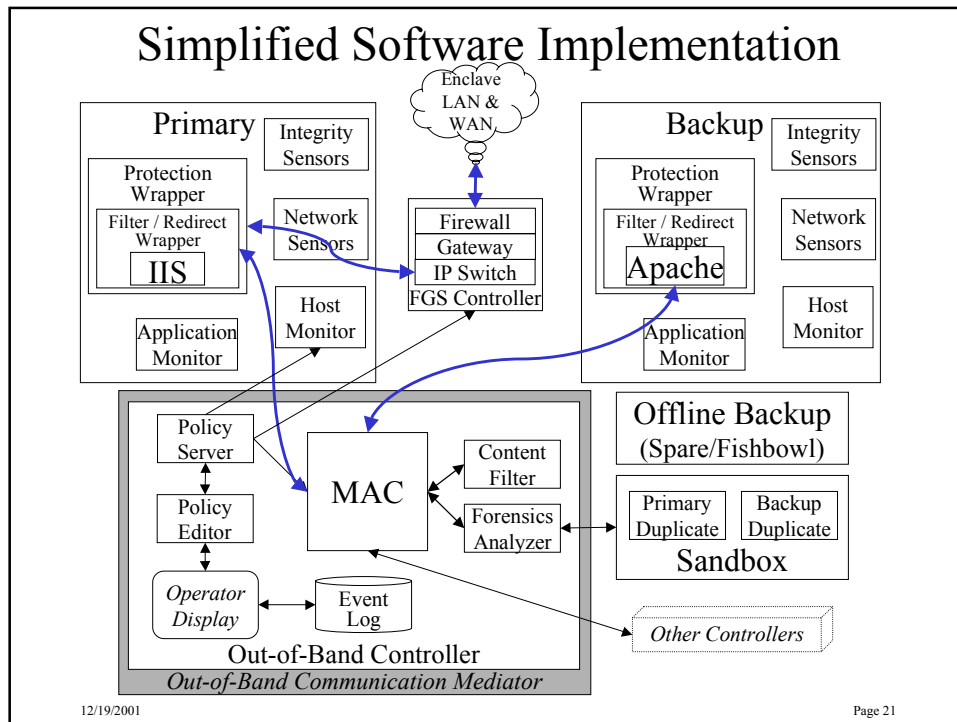
Page 19

HACQIT Monitor & Control Software Overview



12/19/2001

Page 20



- ## HACQIT Increment 1 Capabilities
- Current
 - Protection of IIS web-enabled message board
 - Multiple failovers in response to attacks
 - Restoring state on spare and resyncing a new process pair
 - Complete event logging (basis for future operator's display)
 - Learning and blocking unknown attacks through forensics on captured traffic and Sandbox testing to build filter - initial
 - Generalize filter to block future success of attack - initial
 - Randomized rejuvenation and reconstitution (simple)
 - Continuous recovery (initial)
 - Semi-automated mode to build allowed process list
 - Forthcoming in Increment 2
 - Group learning – distributed updating of attack filter settings
 - Operator GUI, fishbowl and proxy capabilities
 - Change security posture as result of receiving external alert
 - Send messages & responses (attack signature, health & status, etc) to other clusters, Cyber Panel, FW, others
- 12/19/2001 Page 22

HACQIT Statistics

Module	Language	Lines of Code	Comment
Connection Manager	Java	819	
MAC	Java	3589	
Policy	Java	2592	
Policy Editor	Java	3871	
Util	Java	569	
	Sub-total	11,440	
Application Protection	C++	1930	Wrapper
WCW	C++	2143	Wrapper / ISAPI Filter
Content Filter Bridge	C++	336	JNI
Monitor lib	C	653	
Native Bridge	C++	9081	JNI
String Utils	C	1576	
hacqithm	C/C++	6502	
	Sub-total	22,221	
12 modules	Total	33,661	

- Performance impact: less than 4% in simple tests

Testing & Validation

- Test against new attacks – old patch level
 - E.g., Code Red 1 and 2
- “Red Team” for selected prototypes
 - UC Davis Computer Security course student
 - Winter 2001-2
- Open Internet exposure (with advertising)
 - Project web site and current implementation at www.hacqit.net
- Analytics (design reviews, etc)

Code available shortly

- Contact reynolds@teknowledge.com
- 703-352-9300, x203

Research Questions

- How do you detect intrusions and how do you measure intrusion tolerance (fault injection)?
- What is the scope of HACQIT protection?
 - What classes of applications/services can be effectively protected by HACQIT architecture?
 - What categories of attacks can be prevented, detected and masked, or detected, learned and blocked?
 - Can integrity and confidentiality be protected?
- What is HACQIT design/implementation effectiveness?
 - Is intrusion tolerance useful without Byzantine FT?
 - To what extent can intrusion propagation be prevented? Is protection possible if mechanisms are only on cluster? Can process pairs provide effective masking of attacks? Is randomization a useful strategy?
 - Can random rejuvenation and continuous recovery provide effective protection from maliciously corrupted processes and files?
 - Is learning effective against unknown attacks?
 - How much effort is involved in protecting a new critical application?
- How does HACQIT compare to other designs/implementations

12/19/2001

Page 25

Plans and Schedule

- New Prototype Increments
 - Increment 2 –February 2002
 - Change in security posture as result of “Infocon” or similar alert received from Cyber Panel
 - Sending messages, blocking parameters, etc about attack to Cyber Panel, other HACQIT clusters and enclave firewall
 - Initial generalization of blocking sequence for buffer overflows
 - Initial operator GUI
 - Initial fishbowl capabilities
 - Increment 3 – September 2002
 - Second critical application – mail server
 - Selected new capabilities – MAC monitor
 - Enhancements to Increment 1 capabilities
- Validation efforts
 - New attack testing – Ongoing as new attacks arise
 - Red Team – January thru March 2002
 - Internet exposure – March thru September, 2002
 - Analysis – On-going

12/19/2001

Page 26

Major Innovations

- Intrusion tolerant architecture that stops many common attacks but still allows access to critical services
 - Isolate key components (e.g., an out-of-band control network, complete separation of primary and backup servers)
 - Minimize attack propagation (e.g., restrict ability to access files and execute programs, mediate all communications between in-band and out-of-band components)
 - Prevent critical user impersonation (e.g., via strong authentication, VPNs, and restricting connections to critical user machines)
- Specification based approach to defining proper behavior of the HACQIT components
 - Hierarchy of functional and performance specifications can be a priori, learned or a combination of both
 - With learning, we believe, specifications can be proven to be complete for integrity and availability
- Rapid failover of applications via process-pair architecture with time delay (to avert common mode failures)

12/19/2001

Page 27

Major Innovations (cont.)

- Random rejuvenation at various levels
 - Randomly fails over and restarts application, OS, host & install
 - Improves the reliability of a running application
 - Deals with a particularly difficult attack type -- stealthy attack that corrupts an executing process but does no damage initially.
- Forensics and learning to stop unknown attacks
 - Detect and respond to major classes of successful unknown attacks once they are used so as to prevent their future successful use
 - Use black box-like recorder and Sandbox for testing & verification
- Continual recovery
 - Mediates all potential changes to persistent data (files)
 - If the persistent data begins in a correct state, no change violates correctness.
- *Watching the watcher*
 - *Verifies the operation of the HACQIT out-of-band controller to ensure that it has not been compromised*

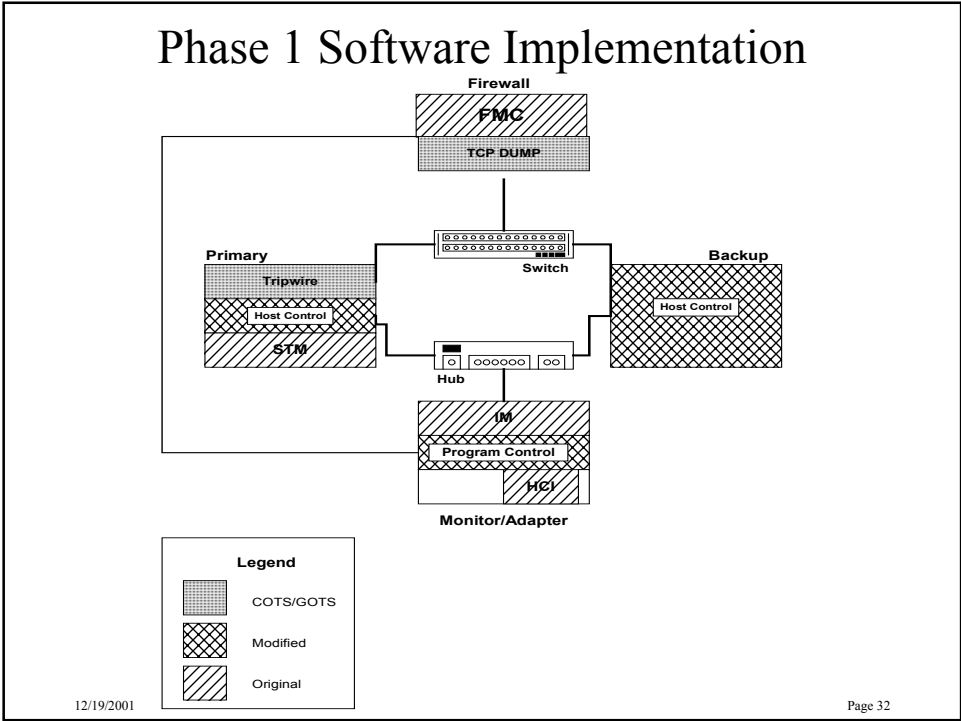
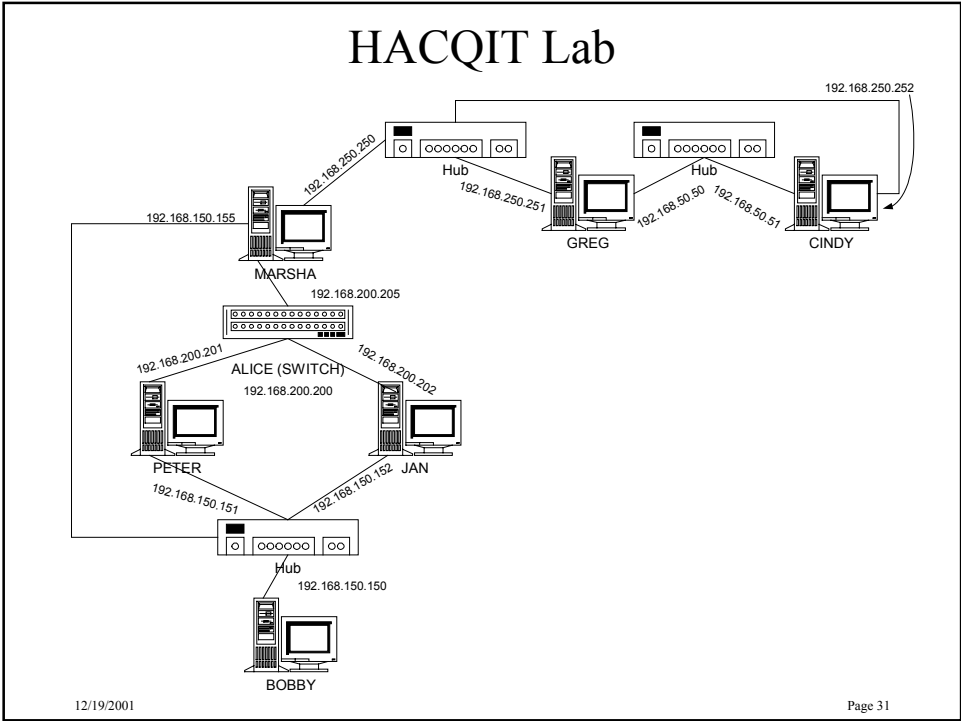
12/19/2001

Page 28

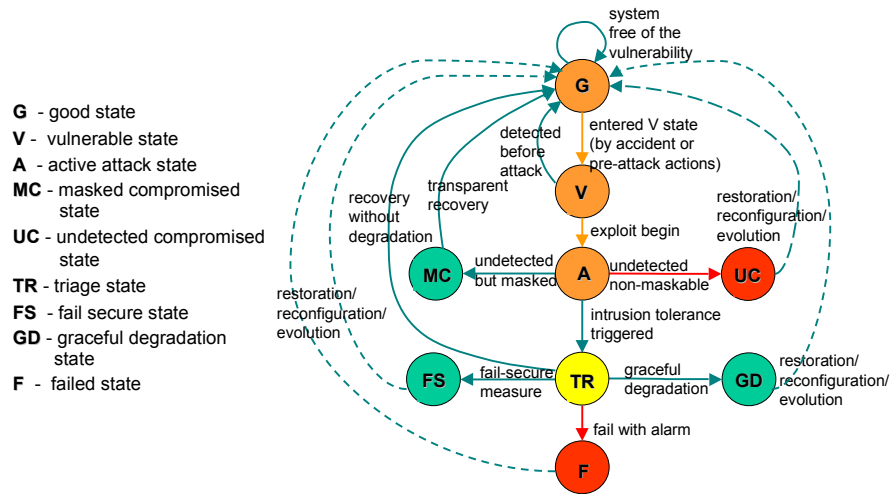
Thank you!

Questions?

Backup



State Transition Diagram for ITS



Source: Goseva – Popstojanova, SITAR: Characterizing Intrusion Tolerant Systems Using a State Transition Model”, OASIS PI Meeting, July 2001, Santa Fe, NM

12/19/2001

Page 33

Design & Rationale Module

How it meets requirements, what were options, why chosen?

- Intrusion resistant features & separation features
- Intrusion tolerance features
- Intrusion detection features & Sensor coverage
- Response options
- Policy drivers
- Outside communications

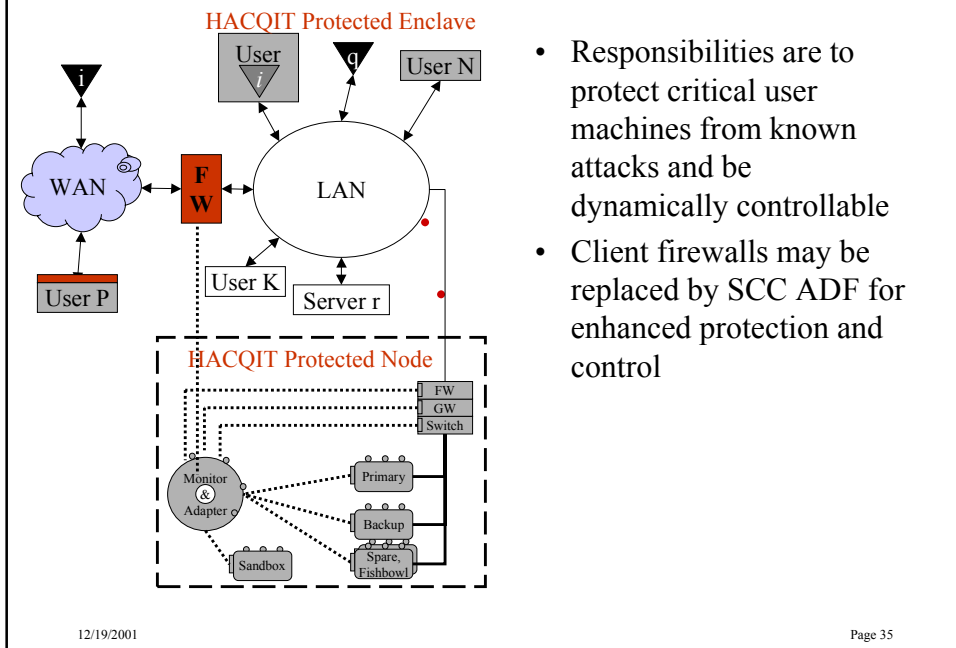
Tradeoffs

- How much protection is enough – failover to same machine, 2 machines, 3 machines, 4 or more machines
- Forensics and blocking future attacks vs. faster restore & reconstitution if needed

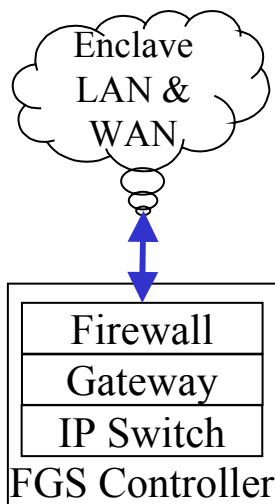
12/19/2001

Page 34

Enclave and Client Firewalls



VPN, Firewall, Gateway, Switch



- Ensures
 - No sniffing, no spoofing
 - Only authenticated users and properly formed packets and requests get to the critical applications
 - Connections only get to the application on the primary
 - No direct communication from the enclave/primary to the backup server
- Enables dynamic address blocking and content filtering
- Supports strong separation

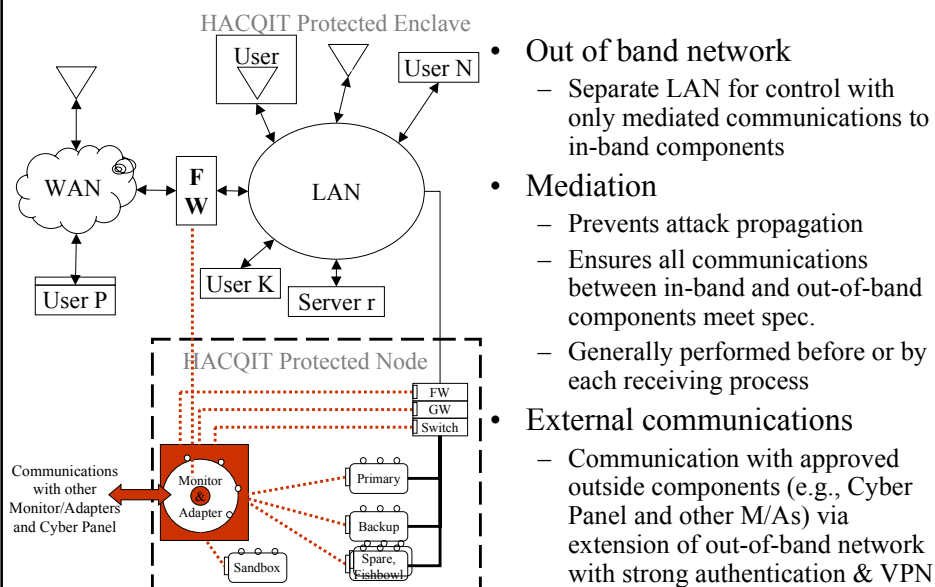
Continual Recovery

- Process monitoring and recovery
 - Monitor all process creation and file access events through security log
 - Terminate new processes not on the allowed list
 - Perform health measurements on protected processes
 - Restore and restart protected processes when monitored processes fail health checks
- File monitoring and recovery
 - Maintain a “shadow copy” of static files
 - Restore files which are deleted or damaged
 - Delete unauthorized files
 - Failover if critical application data is damaged

12/19/2001

Page 39

Out of Band Network & Mediation

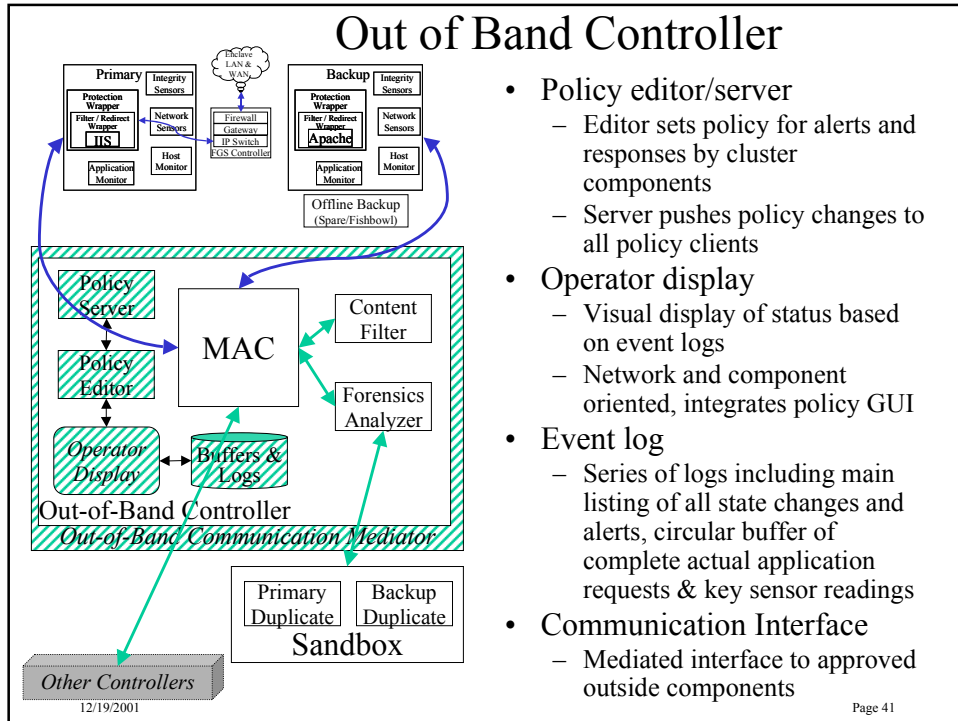


- Out of band network
 - Separate LAN for control with only mediated communications to in-band components
- Mediation
 - Prevents attack propagation
 - Ensures all communications between in-band and out-of-band components meet spec.
 - Generally performed before or by each receiving process
- External communications
 - Communication with approved outside components (e.g., Cyber Panel and other M/As) via extension of out-of-band network with strong authentication & VPN

12/19/2001

Page 40

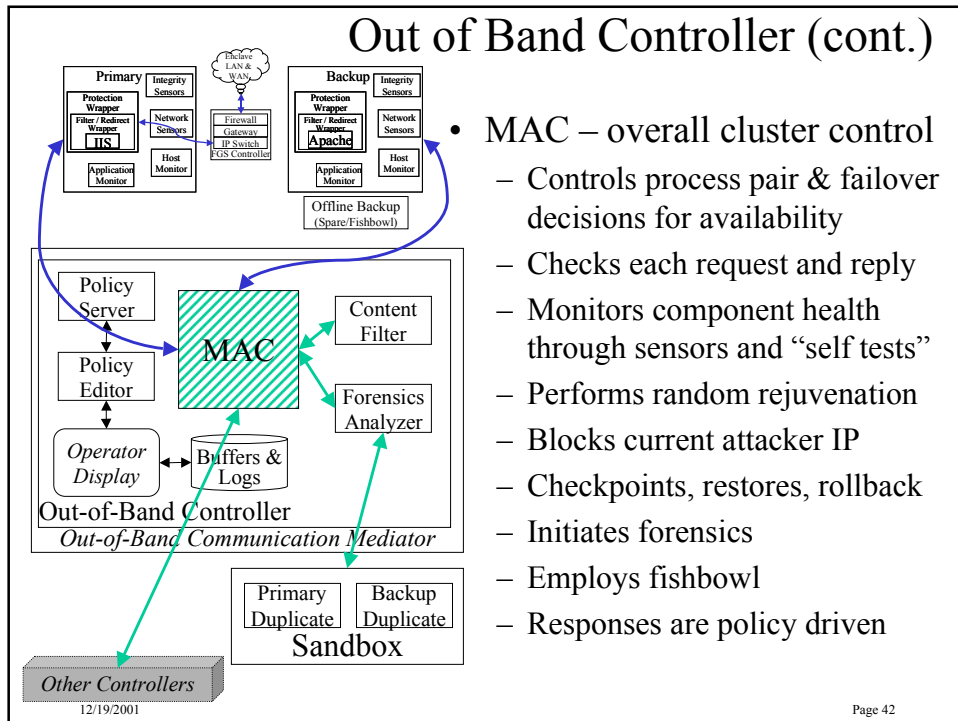
Out of Band Controller



- Policy editor/server
 - Editor sets policy for alerts and responses by cluster components
 - Server pushes policy changes to all policy clients
- Operator display
 - Visual display of status based on event logs
 - Network and component oriented, integrates policy GUI
- Event log
 - Series of logs including main listing of all state changes and alerts, circular buffer of complete actual application requests & key sensor readings
- Communication Interface
 - Mediated interface to approved outside components

Page 41

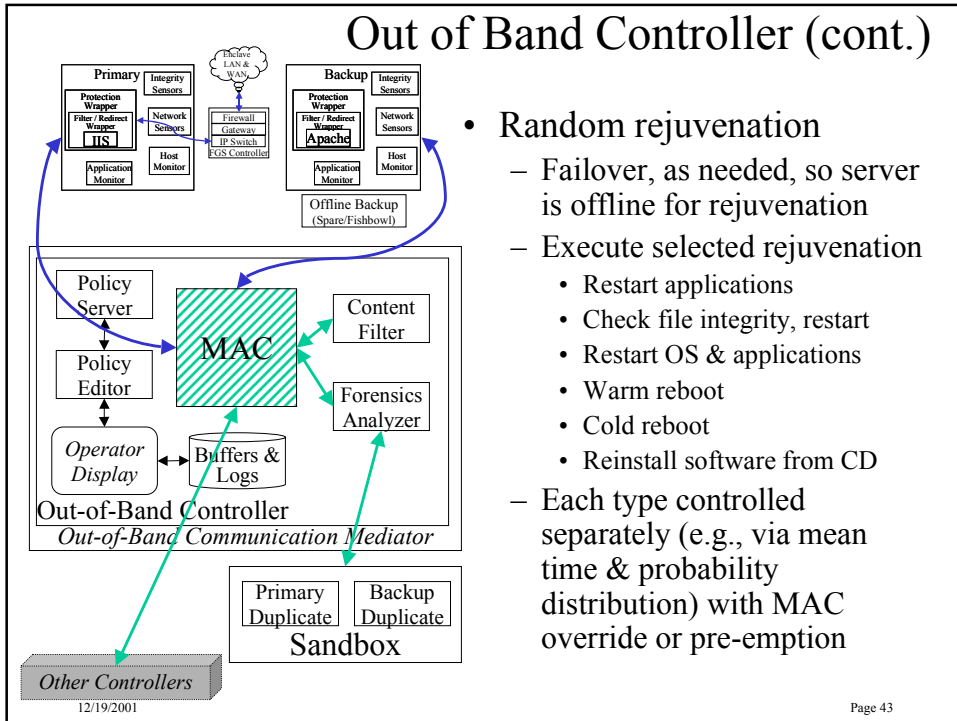
Out of Band Controller (cont.)



- MAC – overall cluster control
 - Controls process pair & failover decisions for availability
 - Checks each request and reply
 - Monitors component health through sensors and “self tests”
 - Performs random rejuvenation
 - Blocks current attacker IP
 - Checkpoints, restores, rollback
 - Initiates forensics
 - Employs fishbowl
 - Responses are policy driven

Page 42

Out of Band Controller (cont.)

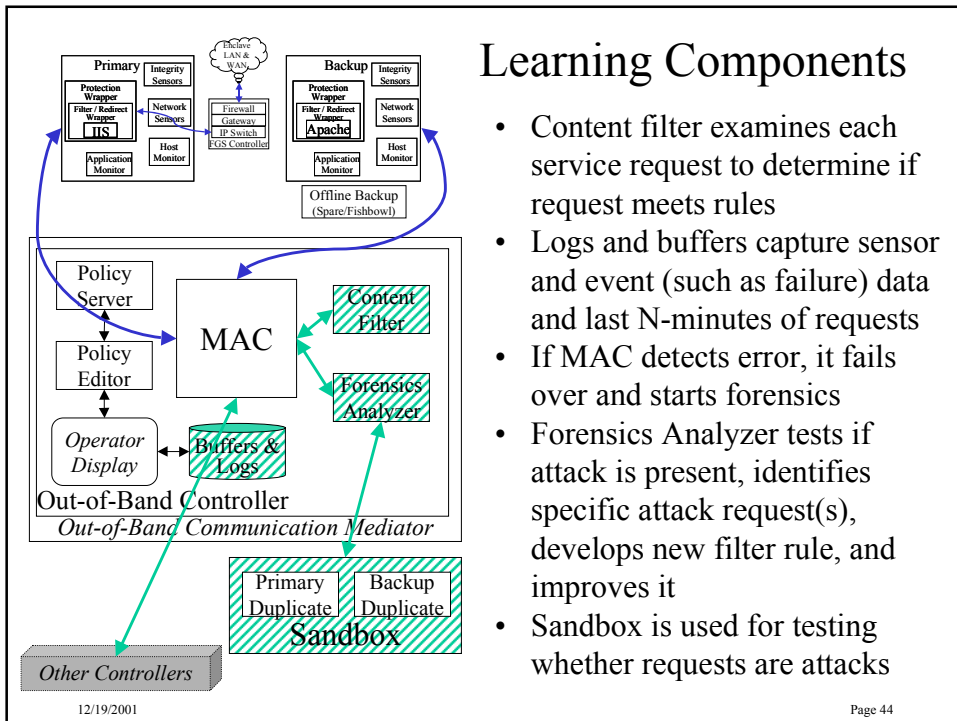


- Random rejuvenation
 - Failover, as needed, so server is offline for rejuvenation
 - Execute selected rejuvenation
 - Restart applications
 - Check file integrity, restart
 - Restart OS & applications
 - Warm reboot
 - Cold reboot
 - Reinstall software from CD
 - Each type controlled separately (e.g., via mean time & probability distribution) with MAC override or pre-emption

12/19/2001

Page 43

Learning Components

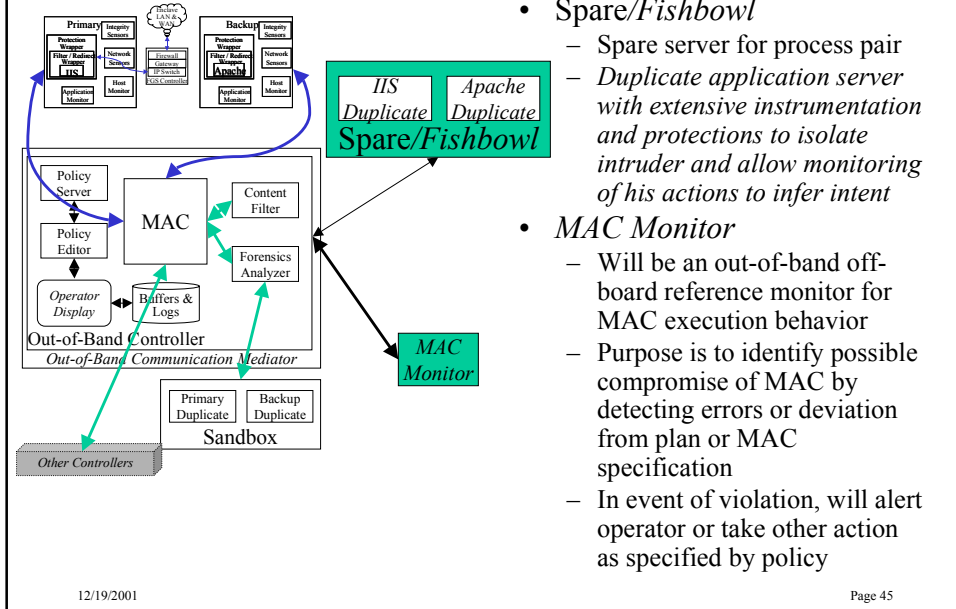


- Content filter examines each service request to determine if request meets rules
- Logs and buffers capture sensor and event (such as failure) data and last N-minutes of requests
- If MAC detects error, it fails over and starts forensics
- Forensics Analyzer tests if attack is present, identifies specific attack request(s), develops new filter rule, and improves it
- Sandbox is used for testing whether requests are attacks

12/19/2001

Page 44

Other Out-of-Band Components



- *Spare/Fishbowl*
 - Spare server for process pair
 - Duplicate application server with extensive instrumentation and protections to isolate intruder and allow monitoring of his actions to infer intent
- *MAC Monitor*
 - Will be an out-of-band off-board reference monitor for MAC execution behavior
 - Purpose is to identify possible compromise of MAC by detecting errors or deviation from plan or MAC specification
 - In event of violation, will alert operator or take other action as specified by policy

12/19/2001

Page 45

Response Module

12/19/2001

Page 46

Paradigm for Responding to Integrity Intrusion

- REPEAT UNTIL ATTACK SYMPTOMS
DISAPPEAR
- Detect integrity violation on a critical file
- Switchover to backup server; restore prior version
of critical file on primary
- Use Jigsaw model to determine possible causes
and sources of attack
- Deploy sensors and responders as determined by
model
- If attack persists block with responders

12/19/2001

Page 47

Paradigm for Responding to Internal DOS Attack

- REPEAT UNTIL ATTACKSYMPTOMS
DISAPPEAR
- Detect denial of service violation on primary
server
- Switchover to newly created process on server;
kill process causing denial of service
- Use Jigsaw model to determine possible causes
and sources of attack
- Deploy sensors and responders on server and on
firewall as determined by model
- If attack persists block with responders

12/19/2001

Page 48

HACQIT Actions in Responding to Connection Spoofing Attack

- Detect change to .rhosts file on primary
- Switchover to backup
- Restore previous version of primary, which is now the backup
- Use Jigsaw model of attacks to identify possible causes of integrity problem
 - Change is legitimate by “clean” process-- no integrity problem
 - Change is by an unauthorized process
 - Change is by a legitimate rcommand
 - Change is by an unauthorized rcommand

12/19/2001

Page 49

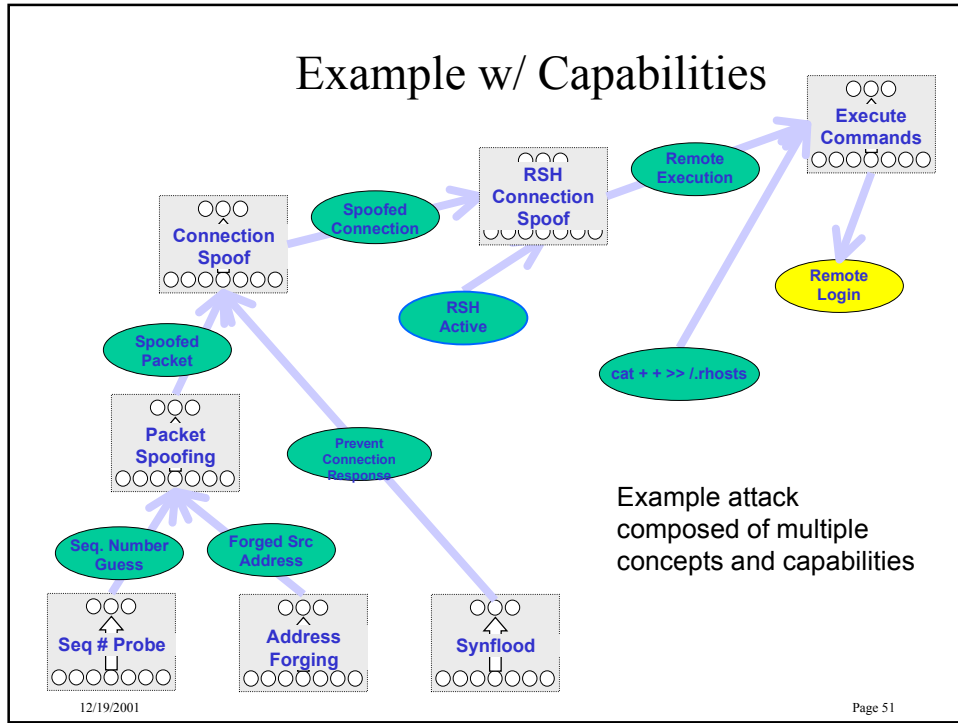
HACQIT response to Connection Spoofing (cont)

- HACQIT checks for erroneous processes -- finds none; so conclude change is legitimate or due to an rcommand
- HACQIT starts monitoring for rcommands
- Attack persists, but now on backup with arrival of rcommand
- HACQIT temporarily blocks rcommand until verification
- HACQIT monitoring detects symptoms of connection spoofing attack -- sequence number guessing, DOS on a host
- If traceback to true source s is possible, connections from s are blocked; otherwise, degraded mode (no rcommands)

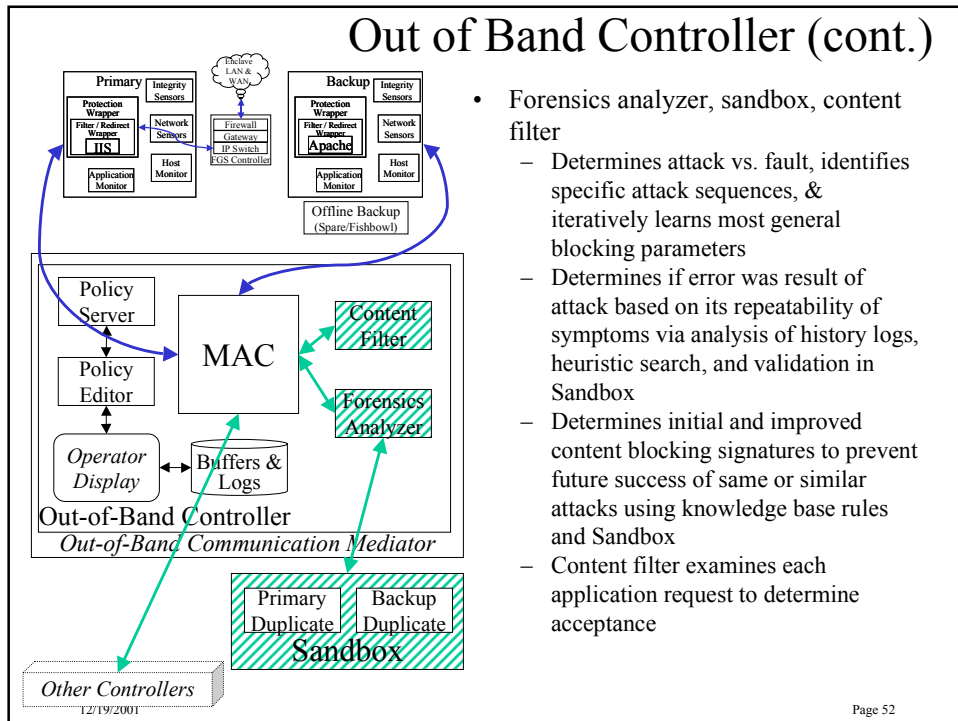
12/19/2001

Page 50

Example w/ Capabilities



Out of Band Controller (cont.)

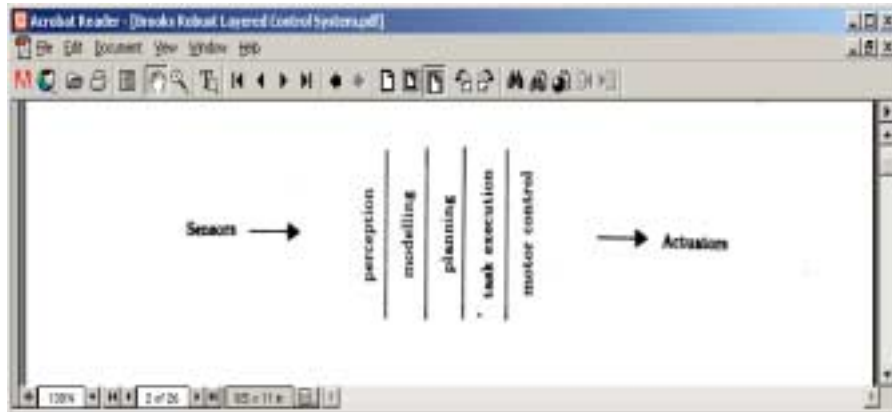


Architectural Exploration Module

Architectural Explorations

- Major focus of HACQIT is to develop Intrusion Tolerant – oops! – I mean Organically Assured and Survivable architecture
- Our levels of capability demonstrations suggest the Subsumption architecture (Brooks, 86)
- Brooks developed the architecture for his famous robot projects but many of our requirements are the same
 - Need certain amount of “stupid” reactive behavior
 - Need guaranteed fast response
- Brooks implemented each layer in his architecture as a deterministic finite state machine with simple I/O
- No world model is depended on
- Communication from higher to lower levels is done through suppression and injection

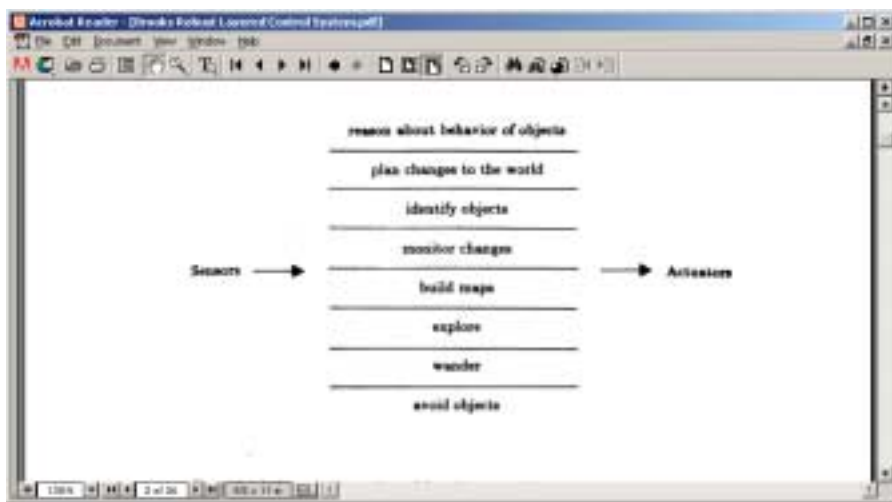
Traditional Architecture



12/19/2001

Page 55

Subsumption Architecture



12/19/2001

Page 56

HACQIT Mapping to Subsumption Architecture from Levels 0-2 Capabilities

- Pre-Level 0 capability features could be lowest layer like Brooks' "Avoid" module
 - Unauthorized process on primary boosts CPU utilization above threshold: kill process, move critical service to backup
 - Unauthorized modification of file: move critical service to backup
- Level 0 capabilities would be second layer
 - Wrapper intercepts suspicious call: move critical service to backup
 - Diversity advantage: Backup runs different OS than primary
 - TCPDump is turned on after suspicious call is intercepted (heightened awareness)
- Level 1 would be third layer
 - Migration is effected without interrupting critical users (change ARP table)
- Level 2 would be fourth layer
 - Source address of attack is identified
 - Address blocked by change to firewall policy

12/19/2001

Page 57

Higher Levels of Capability

- Multiple critical applications
- Failover which saves state
- Failover on the same machine
- Forensics
- All may be too complex, long-lived, or require global information in order to implement
- We're looking at DICAM as architecture for these functions

12/19/2001

Page 58

DICAM Control

