

# Network Security

## Rethinking the Network to Support:

Security  
Mobility  
Management  
Experimental Evaluation

Karl Levitt  
NSF/CISE/NSF  
and  
UC Davis

# Thanks to our PIs and NSF Colleagues

- Dave Clark
- John Doyle
- Vern Paxson
- Wenke Lee
- R. Sekar
- Scott Shenker
- David Anderson
- Fred Schneider
- Nick Feamster
- John Mitchell
- Ty Znati
- Ralph Wachter
- Darleen Fisher
- Allison Mankin
- Kevin Thompson
- Jie Wu
- David Du

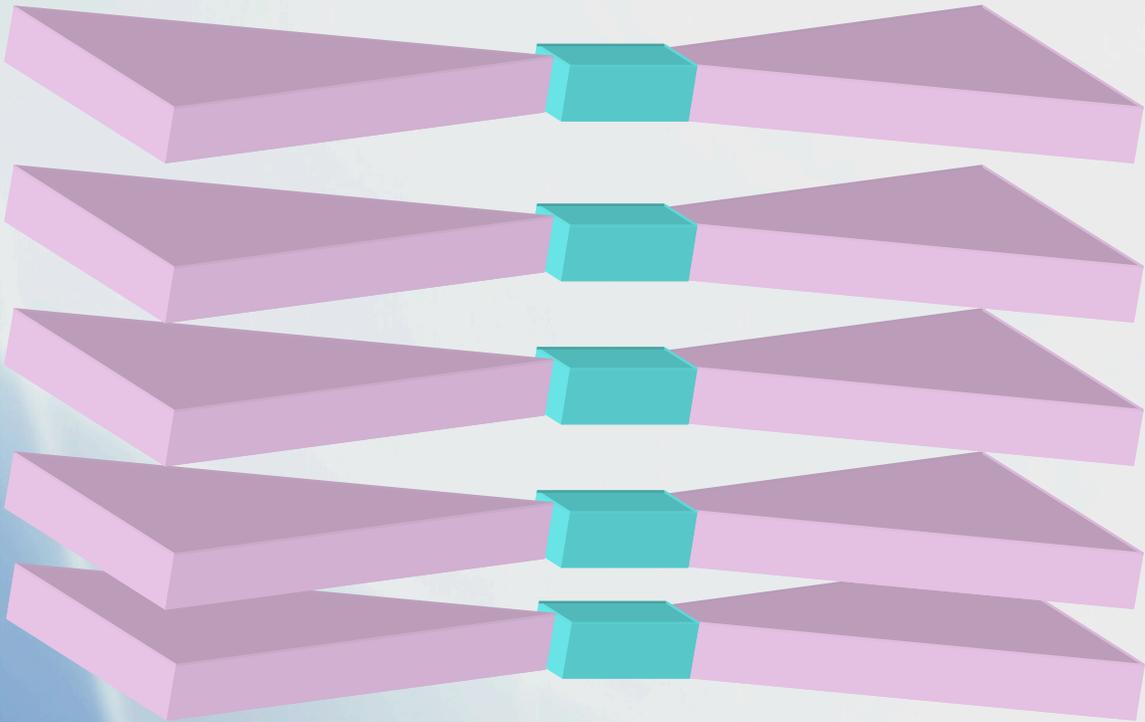
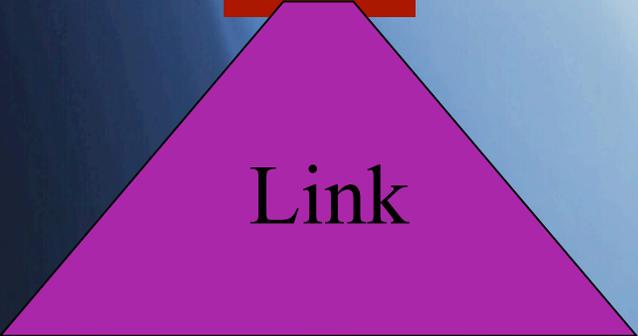
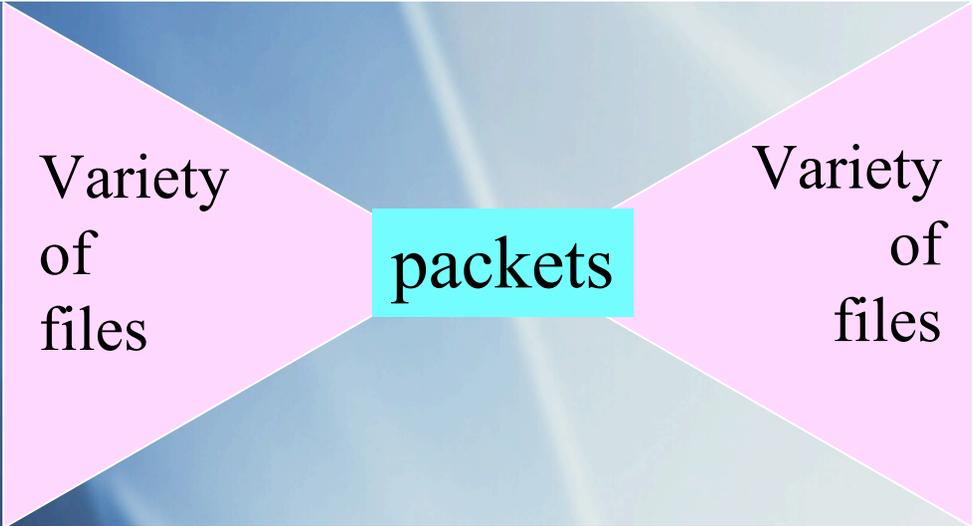
# Outline

- Security issues in the network
  - Current Internet
  - A future Internet
- Host vs. Network
  - Vulnerabilities
  - Attacks
  - Defenses
- Other issues
  - Mobility
  - Economics
- Towards a science of security so we can reason about the security of real systems *analytically* and *experimentally*
- Priorities of CISE's (with OCI) new Trustworthy Computing (TC) program

## ***Punch Lines:***

- *Hosts and the network must cooperate to defend against attacks, especially those sure to come*
- *An overall security architecture is needed to integrate the (very good) existing point solutions*

# Many bowties in Internet



# Consequences of a Simple **Routing** Core

## Benefits

- Universal connectivity
- Data forwarding permits packets to be sent from anywhere to anywhere
- Routers perform a very simple function and can be realized at any scale: central office to consumer devices
- Internet is *open*: supports creation of many applications and link technology
- Many faults are handled easily by the core

## Problems

- Little support for management
- Diagnosis can be a nightmare
- Bad guys can launch attacks across Internet to any vulnerable node
- Impossible to trace attackers to their source
- Quality of service (especially RT) not easily achieved

# The Core is more than Routers

- Different kinds of routers
- Domain Name Service (DNS)
- Firewalls
- ISPs
- NICs
- Others?

All of these

- Contain vulnerabilities
- Are subject to attack
- But help mitigate attacks
- Are difficult to manage
- Have economic consequences

# The Many Topics of Security

- **Cryptography:** provable security, key management, lightweight cryptographic systems, conditional and revocable anonymity, improved hash functions
- **Formal methods:** access control rule analysis, analysis of policy, verification of composable systems, lightweight analysis, on-line program disassembly
- **Formal models:** access control, artificial diversity and obfuscation, deception
- **Defense against large scale attacks:** worms, distributed denial of service, phishing, spam, adware, spyware, stepping stone and botnets
- **Applications:** critical infrastructures, health records, voice over IP, geospatial databases, sensor networks, digital media, e-voting, federated systems
- **Privacy: models, privacy-preserving data-mining, location privacy, RFID networks**
- **Hardware enhancements for security:** virtualization, encryption of data in memory, high performance IDS, TPM
- **Network defense:** trace-back, forensics, intrusion detection and response, honeynets
- **Wireless & Sensor networks:** security, privacy, pervasive computing
- **New challenges:** spam in VoIP, “Google-like” everywhere, virtualization, quantum computing, service oriented architecture
- **Metrics:** Comparing systems wrt security, risk-based measurement
- **Testbeds and Testing Methodology:** DETER and GENI, scalable experiments, anonymized background data

# What is Attackable?

## Where should Defenses be situated?

	Attacks on Network	Attacks on End Points
Defenses on Network	In scope	In scope
Defense on End-Points	Marginally in scope	Not in scope
Cooperative Defenses	In scope	In scope

# Traditional (CIA) Security Objectives Apply to Network Core

**Confidentiality:** E.g., Router passwords can be compromised

**Integrity:** E.g.,

- Router tables can be erroneously modified
- DNS caches can be poisoned

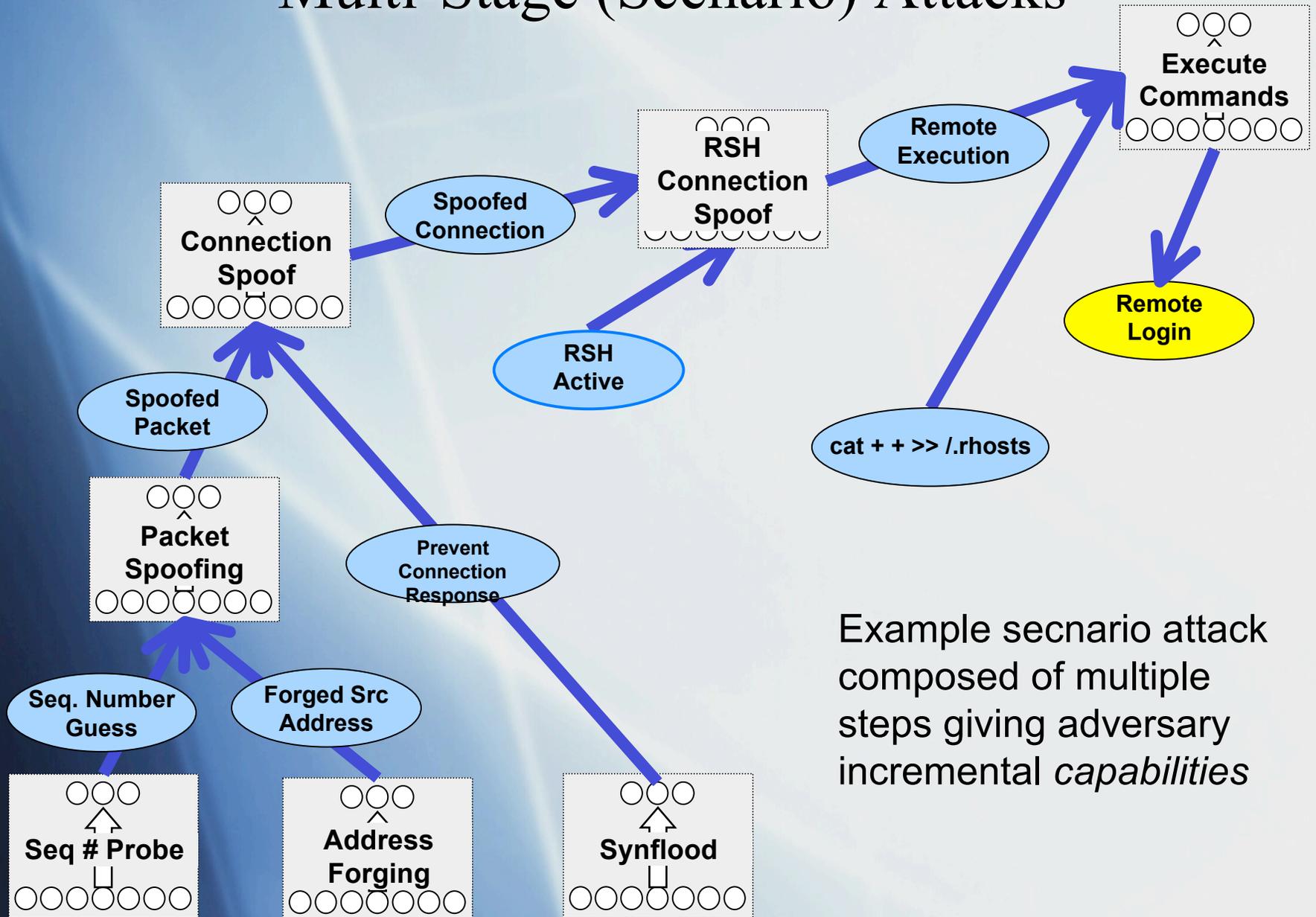
**Availability:** E.g.,

- Routers can be flooded; is this true for core routers?
- ISPs can be spammed, causing *denial of information (DOI)*

Attacking the core can be an adversary's end objective in itself

Or, a means to attacking a host, e.g., routing traffic to a enterprise under the control of an attacker

# Multi-Stage (Scenario) Attacks



Example scenario attack composed of multiple steps giving adversary incremental *capabilities*

# The Many Kinds of Vulnerabilities Enable Many Kinds of Attacks

- Man in the middle (MITM)
- Spoofing
- Spam
- Phishing
- Targeted
- Botnets
- Stealing identity
- Insider
- Installation of Malware, Trojan Horses
- Worms (many kinds), viruses

*Most apply to end-points and network core*

*A taxonomy of network vulnerabilities and attacks is needed*

# A Taxonomy of Memory Error Exploits

## Memory Corruption Attacks

### Corrupt target of existing pointer

*Compromise security critical data*

- File names opened for write or `execute`
- Security credentials -- has the user authenticated himself?

Includes common buffer overflows, `strncpy()`, off-by-one, cast screw-up, format strings, double-free, return to libc, other heap structure exploits

## Corrupt a pointer value

### Corrupt data pointer

- Frame pointer
- Local variables, parameters
- Pointer used to copy input

### Corrupt code pointer

- Return address
- Function pointer
- Dynamic linkage tables (GOT, IAT)

Point to injected data

### Point to existing data

Example: corrupt string arguments to functions, pointing to attacker desired data already in memory, e.g., `"/bin/sh"`, `"/etc/passwd"`

Point to existing code

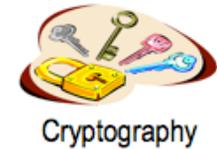
Point to injected code

# The Meaning of Network Defense has Changed

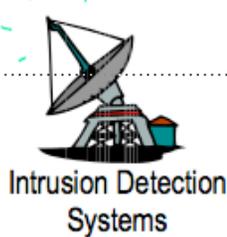
## 1st Generation

(Prevent Intrusions)

'80s



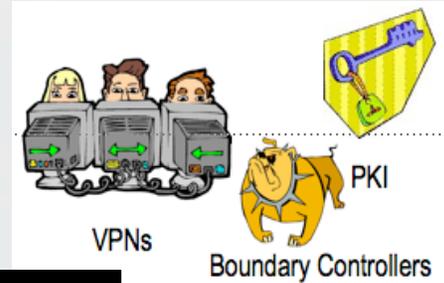
**Intrusions will Occur**



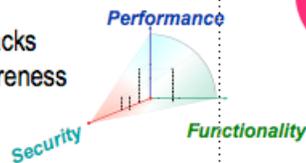
## 2nd Generation

(Detect Intrusions, Limit Damage)

'90s



**Some Attacks will Succeed**



## 3rd Generation

(Operate Through Attacks)

'00s

## 4<sup>th</sup> Generation in '10s

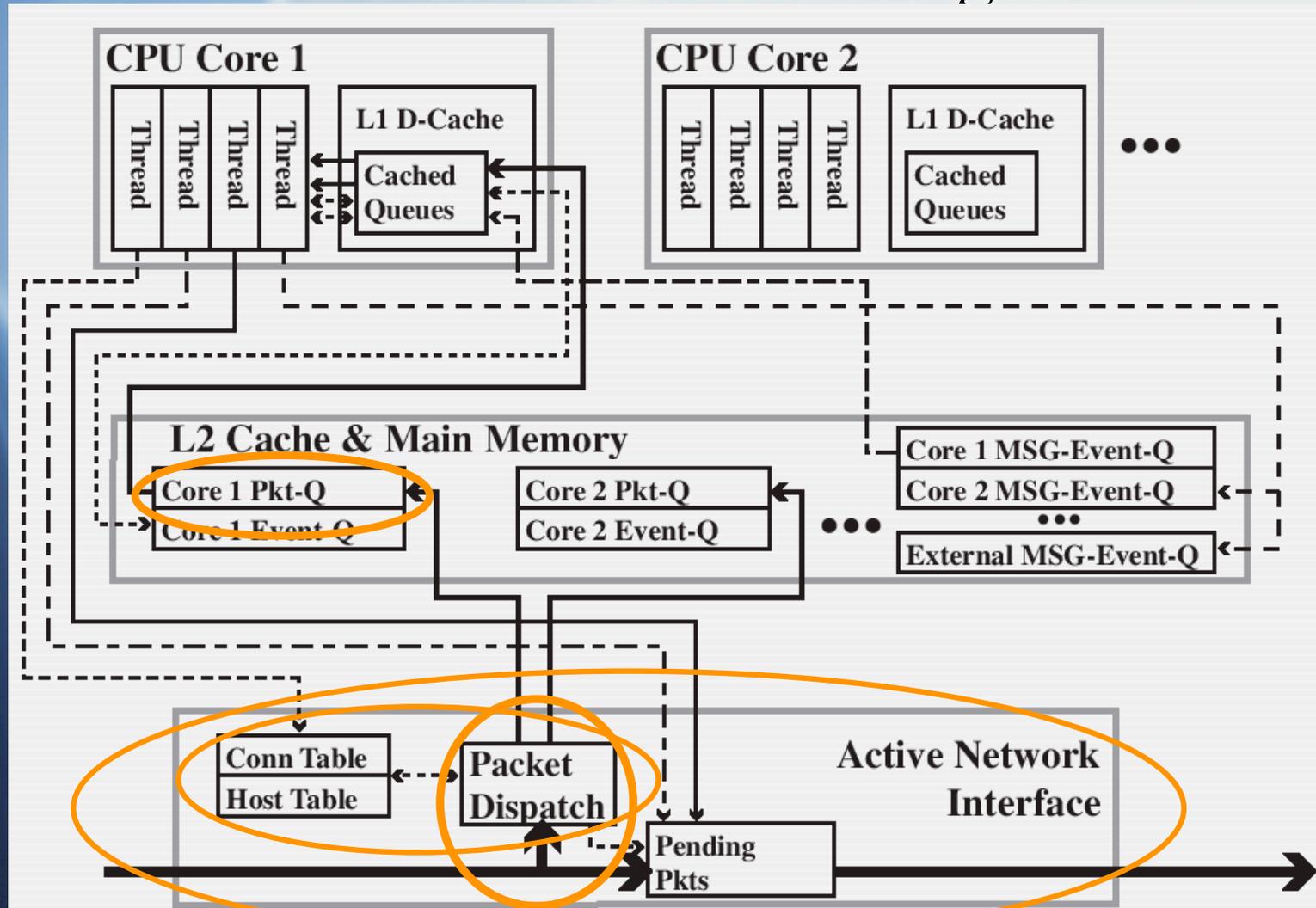
(E.g., prediction of vulnerabilities, cross-enterprise negotiation before attacks, real-time reverse engineering of attacks and malware, planning methods to deal with expected attacks, automatic patch synthesis and distribution)

**"Intel" Will Direct Defenses**

# Some Sobering Growth Trends that make Network Monitoring Difficult

- Network traffic **rates** inexorably grow
- Network traffic **volumes** inexorably grow
- We need to do **more analysis** on **larger amounts** of data at **higher speeds** ...
- But CPU performance is **NOT** inexorably growing any more.

# Multi-Core Architecture for Parallelized Network Monitoring



If ~~the~~ ~~network~~ ~~has~~ ~~limited~~ ~~bandwidth~~ ~~for~~ ~~each~~ ~~core~~ ~~or~~ ~~block~~ ~~or~~ ~~process~~

# Identity Management is Central to Security

## The current situation with source addresses

- They are often used to identify end users
- But, they can be forged
- And, it is impossible to extract information from the network to permit traceback

## Some thoughts on how a future Internet could improve the situation

- Network could require a binding between a packet's source address and the identity of the sender
- But, this permits the network to violate end-users' privacy
- There is a middle-of-the road possibility: The linking of a user to a source address is held by a trusted third party that can (partially) revoke anonymity

*In any event, new protocols and network services are needed*

# Towards an Accountable Internet Protocol (AIP)

- Key idea: New addressing scheme for networks and hosts
- Addresses are self-certifying
- Simple protocols that use properties of addressing scheme as foundation
  - Anti-spoofing, secure routing, DDoS shut-off, etc.

# AIP Addressing

Autonomous domains,  
each with unique ID

An AD.  
Would fail together  
Single administrative  
domain

Key Idea:

AD and EID are *self-certifying flat names*

- AD = hash( public\_key\_of\_AD )
- Self-certification binds name to named entity

a glo

# Botnets Are a Long-Term Problem

Individual Machines Used to Be  
Targets ---

Now They Are Resources

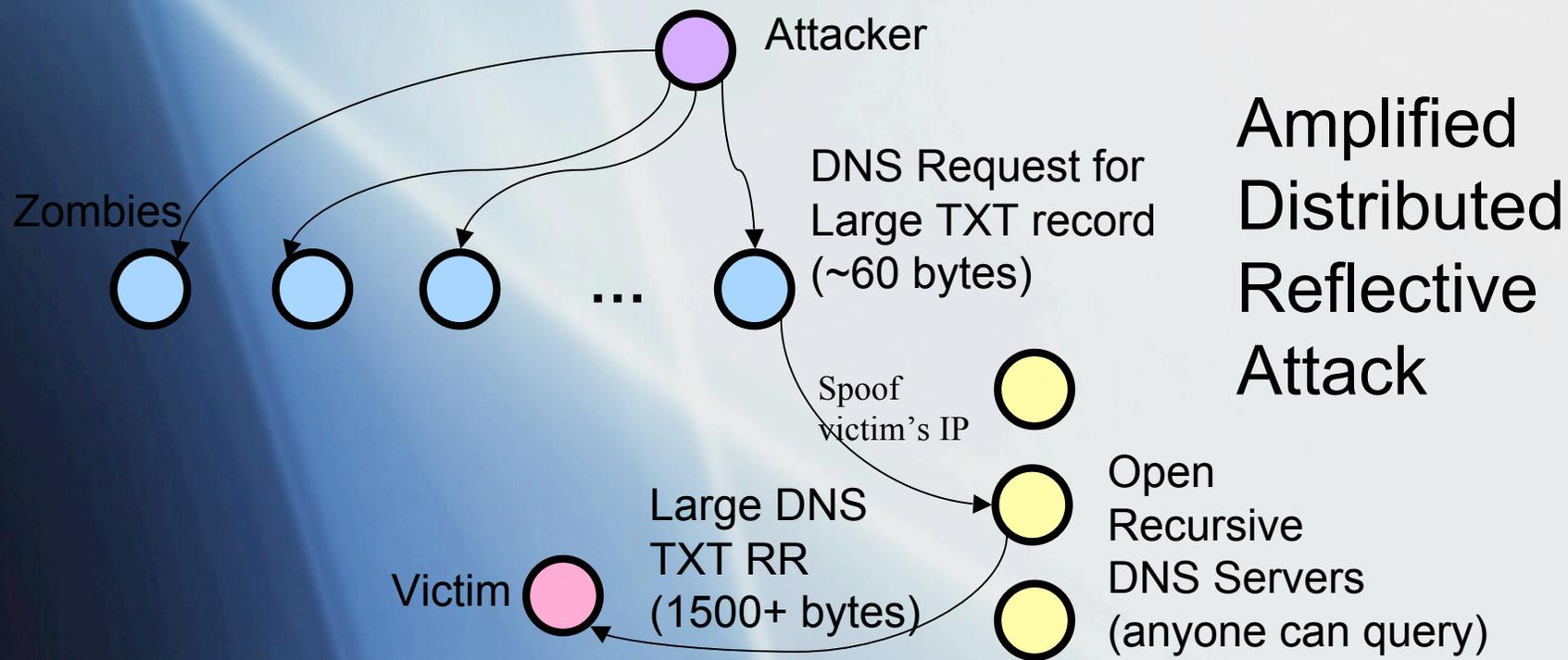
- Bot (Zombie)
  - Software Controlling a Computer Without Owner Consent
  - Professionally Written; Self-propagating; 7% of Internet
- Bot Armies (Botnets)
  - Networks of Bots Controlled by Criminals
  - Key Platform for Fraud and other For-Profit Exploits

# Botnet Epidemic

- More Than 90% of All Spam
- All Denial of Service (DDOS) Attacks
- Clickfraud
- Phishing & Pharming Attacks
- Key Logging & Data/Identity Theft
- Key/Password Cracking
- Anonymized Terrorist & Criminal Communication

# Attack Example

- Botnets increasingly used for amplified distributed reflective attacks



# Thinking About the Botnet Problem

Botnets will continue to be an issue

- Any vulnerable host can become a bot
- There will always be vulnerable hosts

The source of a Botnet will be difficult to determine

- Without accountability it is impossible to identify the commander of a Botnet

So, it is essential to stop or delay the growth or damage associated with Botnets; *only the network can do this*

- An ISP or an enterprise router can detect Bot-like traffic
- And, perhaps block or delay such traffic

But, there are consequences to blocking

- Blocking consumes precious human and device resources
- False positives will lead to many calls to a help desk

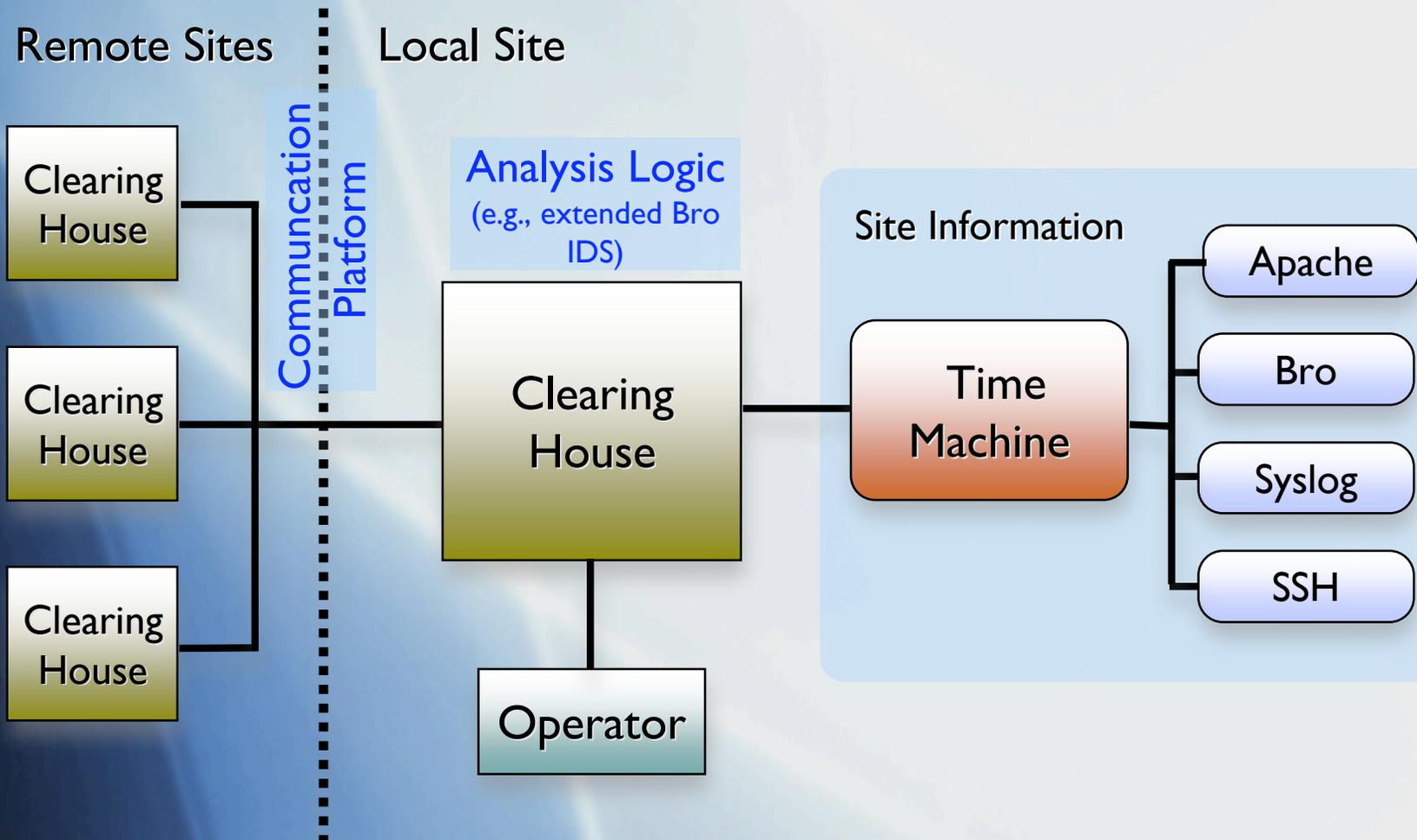
# Denial of Service Attacks

- DDoS attacks are a consequence of Botnets
- Mitigation of DDoS attacks: Host (especially service solution)
  - Distribute services over many machines; packets will be routinely routed to closest machine which might not be DoSed (yet)
- Mitigation of DDoS attacks: Network solution
  - *Pushback* to block or delay traffic from Bots, but there are consequences due to false positives
  - *Diffusion* in routing: choose a route that avoids DDoSed hosts and machines instead of the optimal route

# Envisioning a Rich Inter-site Analysis for Cooperative Attack Mitigation

- Sites deploy *activity repositories* using common data format
- Site A can send request for analysis against activity seen by Site B
  - E.g. “have you seen the following access sequence?”
  - Done by sending an *analysis program*
  - Note: due to co-aligned threat models, it’s often in B’s interest to investigate
- B runs query against their repository ...
  - ... can also install **same** query against **future activity**
- B decides what (sanitized) results to return to A
  - If request was unreasonable, B can **smack** requestor

# Clearing House Architecture



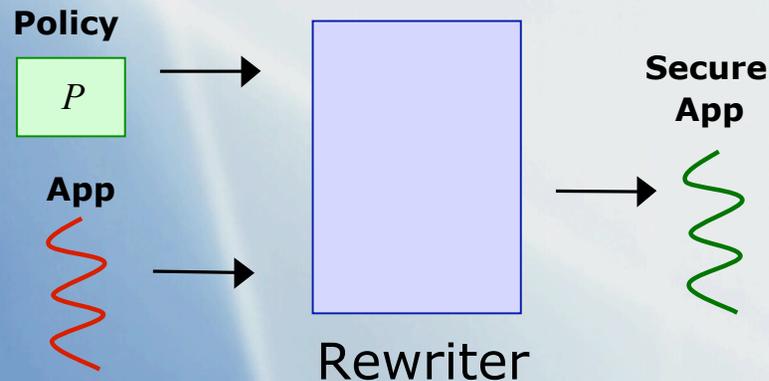
# Is There a Science of Security?

- Are there *impossibility* results?
- Are there powerful *models* (like Shannon's binary symmetric channel) so that realistic security and privacy properties can be computed?  
Possibilities include:
  - Control Theory for security
  - Kirchoff-like laws to capture normal behavior for routers
- Is there a theory that enables:
  - Secure systems to be *composed* from insecure components, or even
  - Secure systems to be composed from secure components
- *Metrics*: Is there a theory such that systems can be ordered (or even partially ordered) with respect to their security or privacy?
- Can entire systems (hosts, networks) and their "defenses" be *formally verified* with respect to realistic security objectives and threats?
- Are there security-related hypotheses that can be validated *experimentally*?
- What kind of an instrument (*testbed*) is needed to validate such hypotheses?

# Enforcement by Program Rewriting?

*Fred Schneider*

- Fundamental issues:
  - Does the application behave the same?
  - Can the application subvert enforcement code?
- Pragmatic issues:
  - What policies can be enforced?
  - What is the overhead of enforcement?



# Towards a Science of Security: Possible Experiments

- What properties can be evaluated by experiment?
  - Usability?
    - By designers of system?
    - By additional users?
  - Performance?
    - Lab environment?
    - Under realistic conditions?
  - Security?
    - Resilience to known attacks?
    - Challenge community to explore new attacks?
    - Security against all attacks within given threat model?

# Security Experiments

- What properties can be evaluated by experiment?
  - Usability?
    - By designers of system? **Yes**
    - By additional users? **Yes, if open user community**
  - Performance?
    - Lab environment? **Yes**
    - Under realistic conditions **Yes, if realistic user community**
  - Security?
    - Resilience to known attacks? **Yes**
    - Challenge us to explore new attacks? **Yes, if realistic user community**
    - Security wrt all attacks threat model? **No, not an experimental property**
- **A Possible Position**
  - Experimental evaluation is important for security mechanisms, applications
  - Open experiments, allowing users other than designers, are essential

# Requirements for Security Facility

- *Ability to determine performance effectively*
  - Facility must allow accurate measurement of a system under stress
- *Resource allocation and accounting*
  - Example: resistance to DoS from an attacker with local but not global control of network.
  - Need to allocate specific resources to agents running in virtualized environment
- *Open access to experimental systems*
  - Usability studies informative only if the test user community is diverse and unlimited
- *Isolation*
  - Experimental systems will subject to attack by designated and unknown attackers
  - Facility must provide isolation between independent slices allocated to diff experiments
- *Privacy*
  - Experimental systems that offer privacy or anonymity to experimental users must not have these guarantees compromised arbitrarily by the facility itself

# Sample experiments

- Spam-resistant email
- Electronic voting systems
- Distributed decentralized access control
- Worm propagation and mitigation
- Reputation systems
- Improved network infrastructure protocols
- Selective traceability and privacy
- SCADA simulation
- Botnet and overlay network security and detectability
- Economic incentives in network infrastructure and applications
- Anonymity in routing and applications
- Experimental combinations of security mechanisms for enterprise security
- Others?

# Main points about Security Experimentation

- Security experiments are important
  - Only way to test usability, performance, some security properties
  - Adoption by test user community is best indicator of usability
- Security experiments do not provide security guarantees
  - Experimental systems should also be subjected to security analysis
- Facility must meet needs of security experimenters
  - *Performance measurements*
  - *Resource allocation and accounting*
  - *Open access to experimental systems*
  - *Isolation*
  - *Privacy*
- More ideas?
  - Please send experiment descriptions

# Trustworthy Computing (TC)

- \$45M/year
- Deeper and broader than CT
- Five areas:
  - **Fundamentals:** new models that are analyzable, cryptography, composability (even though security is not a composable property), new ways to analyze systems
  - **Privacy:** threats to privacy, surely metrics, privacy needs security, privacy might need regulation, database inferencing, tradeoffs between privacy and  $x$

## Trustworthy Computing (TC) (cont'd)

- **Usability:** for home user (parent wanting to keep files from child), security administrator (who is overloaded), forensics
- **Overall Security Architecture:** much of what CT has funded; currently we have point solutions, so we need to combine them, one size might not fill all. For example, should there be a security layer in the protocol stack?
- **Evaluation:** especially experimental, testbed design, looking for research needed for better testbeds but also to use testbeds, data (sanitized) to support experiments

# A Problem to Motivate Security Research

Suppose an adversary inserts malicious logic into a program that controls a critical process. Can the presence of the malicious logic be reliably detected?

*Jim Gossler, Sandia Corp.*

Possible solutions:

- Determine by proof that the program does **more** than intended; requires a specification
- Monitor the behavior of the program with respect to a specification.
  - What if the adversary knows the specification?
  - What if the adversary knows details of the monitoring system?