A Network Architecture for Security Management

Martin Casado
Justin Pettit
Jianying Luo
Michael Freedman
Tal Garfinkle
Dan Boneh
Nick McKeown
Scott Shenker

Presented By: Martin Casado
PhD Student in Computer Science
Stanford University

casado@cs.stanford.edu
http://www.stanford.edu/~casado
What we’d like

Principle 1: Manage network using policy over real names
“Nancy can access Payroll”
“Laptops can’t accept incoming connections”
“VoIP phones mustn’t move”

Principle 2: Policy should dictate the path packets follow
“CEO traffic should not pass through engineering”
“Guest flows must pass through http proxy”
“Laptop flows must pass through IDS”

Principle 3: The origin of packets should be known

Principle 4: Network should log all connectivity
For diagnostics and auditing
Enforcement Hurdles (Today)

- Bindings between names and addresses keep changing, are not authenticated.
- Route is generally unknown to the manager (and security system). And changes.
- No standard for source routing
- How to keep security policy consistent in dynamic network?
Security and Policy Today

1. Network Access Control/Policy/Management
2. Proxies (Web, Email, …)
3. Monitoring/Mapping
Security and Policy Today

- Throw silicon at the problem
  - Special purposes processors
  - TCAMs (the universal hardware hammer)
- Layer security on top of networking
  - Often at the cost of redundancy
  - Often at odds with routing
  - Often at the cost of diagnostics
- **Bottom Line:** Desired service model requirements not provided by yesterday’s architecture
Ethane

- Manage network from centralized controller
  - Standard Desktop PC
  - Routing
  - Name-address bindings
  - Policy declaration
  - Permission Checks

- Perform security check per flow

- Enforce security decisions using very simple switches

- Carefully manage all name to address bindings
Nancy can access Payroll
“Laptops can’t accept incoming connections”
“VoIP phones mustn’t move”
“CEO traffic should not pass through engineering”
“Guest flows must pass through http proxy”
“Laptop flows must pass through IDS”
Ethane in six steps

1. Register
2. Authenticate
3. Send
4. Check
5. Setup flow
6. Communicate

Payroll

Host: a
IP: i
MAC: m

OK: Nancy \( \rightarrow \) payroll. Port 2

OK: Nancy \( \rightarrow \) payroll. Port 4

OK: Nancy \( \rightarrow \) payroll. Port 3

OK: Nancy \( \rightarrow \) payroll. Port 4

Payroll \( \rightarrow \) credentials
Nancy \( \rightarrow \) credentials

“Nancy can access Payroll”
“Laptops can’t accept incoming connections”
“VoIP phones mustn’t move”
“CEO traffic should not pass through engineering”
“Guest flows must pass through http proxy”
“Laptop flows must pass through IDS”

Login?

Nancy
Host: b
IP: j
MAC: n
Forwarding

- End-to-end L2 isolation
- NAT by overwriting IP header
- MAC hiding by overriding the MAC header
Properties

- Backwards compatible
  - No modifications to end-hosts
  - Cooperates with existing switches

- Extremely simple switches
  - no need for TCAMS
  - Line speeds are easy
  - More is good!

- Sophisticated service model
  - Isolation, NAT, MAC filtering, waypoints

- Fine-grain control of each flow
  - Can require different forms of authentication for different access points (e.g. stronger for wireless than wired)
Supporting Policy Namespace

• Namespace covers (users, hosts, access points, protocols)

• Controller manages all name bindings
  – Require authentication for each binding
  – Do not update bindings without re-authentication
  – Revoke bindings on user movement

User Name
Host Name

IP
Mac
Switch Port
Namespace Properties

- DNS-like interface to all bindings
- Namespace binding can match packet to ..
  - Sending user, host
  - Sending location
    (regardless of when it was sent)
- Journalling of global policy allows
  - Full policy roll-back
  - ‘What-if’ testing
Design Summary

• Rather than rely on custom hardware for per-packet computation, centralize and use commodity processors for per-flow computation
• Reduce switches to cheap, simple flow tables
• Policy runtime supports secure namespace
  – Policy allows complex service model
  – Interface to the namespace, rich and with “memory” to aid in diagnostics

• So … does it work?
Many Questions

• Central controller
  – Performance & Scalability
  – Robustness to failure
• Appropriate policy language
• Simple/intuitive management interface…
• …and transparent to user.
Our Deployment

- 300+ hosts at Stanford: Servers, laptops, desktops, phones.
- 19 switches
  - Hardware 4x1GE switch (FPGA)
  - Wireless access point (openWRT)
  - Software 4x100MbE switch (Linux)
- Controller cheap-o Fry’s PC
- Policy: 132 rules to replicate policy
Lessons so far…

• Controller handles >10,000 flows/second
• Enough for 22k active IPs?
• Multiple ways to handle redundancy
  – Cold-standby, hot-standby, stateless, stateful
• Transparent to users
  (even remotely at home!)
Lessons so far...

- Service discovery is a pain
  - Makes up the majority of flows
  - Inherently sends to broadcast
- Undocumented protocols exist (and are used!)
Future Plans

1. Grow deployment in department
   - 1GE switches and access points
   - Use traditional switch with VLANs as a mux.

2. Increase deployment at Stanford?

3. Other schools…?
Questions?
Principle 1: Manage network using policy over real names

“Nancy can access Payroll”
“Laptops can’t accept incoming connections”
“VoIP phones mustn’t move”

Today
“Everyone who is not Nancy cannot access payroll”
Q: How to identify them?
Q: Where do their packets flow?
Waypoints

Controller

Nancy can access Payroll
Laptops can’t accept incoming connections
VoIP phones mustn’t move
CEO traffic should not pass through engineering
Guest flows must pass through http proxy
Flows to Payroll must pass through IDS

Payroll

March, 2007 Stanford Security Forum