Designing a Research Firewall:

Policy, Practice, and Experience with SURF

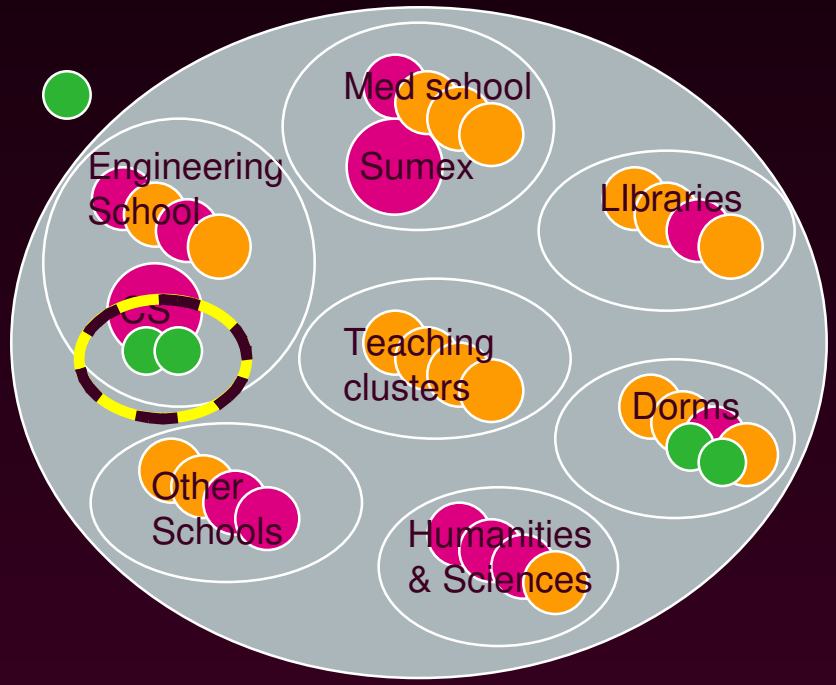
Jonathan R. Stone
Michael B. Greenwald
Sandeep K. Singhal
David R. Cheriton

Department of Computer Science
Stanford University

Stanford University Research Firewall

- One-way firewall
  - Cumbersome firewalls will be subverted
  - Beginning to be supported by vendors
- Many small firewalls
  - Limit scope of increased risk
  - Devolve responsibility to appropriate level
- We deployed the firewall, but ultimately, protocols need to be redesigned
  - Ad hoc techniques only go so far
  - Security concerns justify redesign

Stanford Campus Network



- T3 link to BARRNET
- 28,000 IP addresses in use on campus
- High-profile, open, **insecure** environment

- Us
- Bad guys
- Potential bad guys

Network Security on Campus

Security Problems:

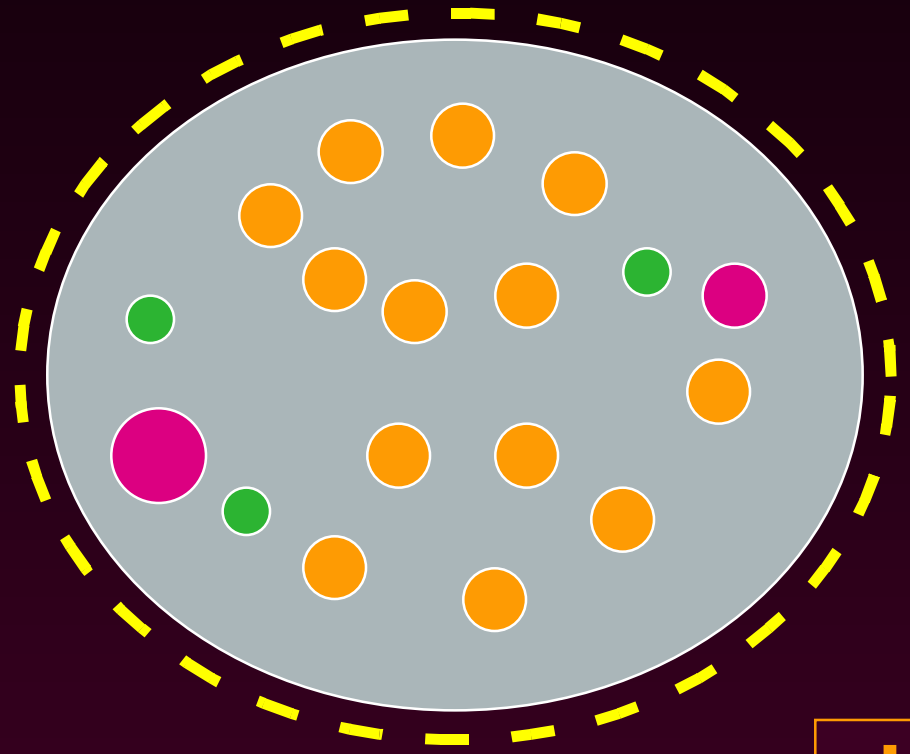
- Password-sniffing endemic on campus
- Known target for crackers
- Plugging security holes is a never-ending battle

Perceived Risks:

- Vandalism: rm -rf *
- Changing grades, stealing solutions
- Plagiarisim of work in progress
- 2-3 days to restore a workstation cluster

How can we reduce our exposure to network breakins?

Put a Firewall Around all of Stanford?

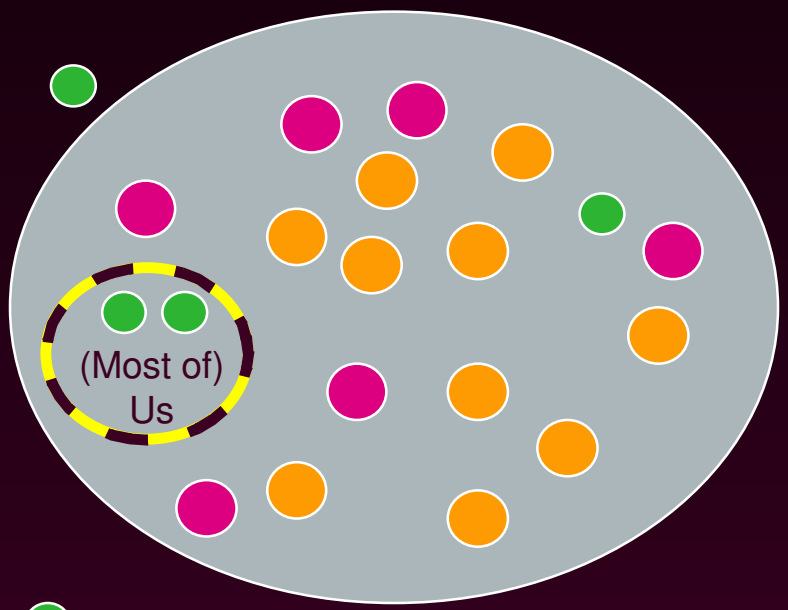


- Us
- Bad guys
- Potential bad guys

- Insiders with password sniffers
- Machines already compromised
- Strong opposition to changing “openness”

Just doesn’t solve the problem

Put a Two-Way Firewall Around Our Research Group?



- Us
- Bad guys
- Potential bad guys

- Would interfere with research
- Can't share resources
- Can't publish information
- Unhappy researchers will side-step security
- We can do better...

Perceived as too painful

The SURF Architecture

- One-way firewall around our workgroup
  - » Inside, don't notice firewall is there
  - » Outside, see strong firewall
- Publish selected data
  - » Users mark specific data as published
  - » SURF pushes that data to a host outside the firewall, where it can be accessed

Provide security without anyone noticing that it is no longer completely open

Publishing Selected Data

- Users partition all their data as either protected or published
  - » Protected data remains inside the firewall
  - » Published data is kept under a designated directory behind the firewall, mirrored outside
- Easy for users
- Easy to automate

Forcing users to make the private/public decision is a win

Where Are Packets Dropped?

- Top is bad: Keep packets away from potentially buggy applications
- Bottom is bad: Performance bottleneck
- Tradeoffs: trust, precision, performance

Realizing a One-Way Firewall: The Request-Response Model

- Anything going out is allowed out [if destined to an outside network]
- Packets allowed in if they are responses to an outgoing request
- Everything else is dropped

Request-Response is our framework for an "open" security architecture

…but, what is a request? a response?

Request-Response Example: TCP

- Need to distinguish requests from responses
- SYN filtering: Drop connection requests from outside
- Depend on correct TCP implementations to do the rest

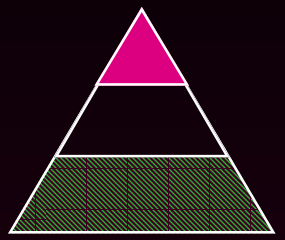
TCP is a lucky fit for request-response

Request-Response at the Application Level

- Sometimes, distinction between request and response is made by application, rather than transport protocol itself
- Example: TCP back channels
  - FTP, rsh, X
- For now, we handle it with proxies

Long-term fix: Change application-level protocols to eliminate back channels

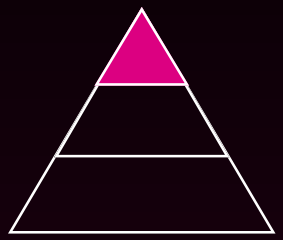
Filtering UDP "Connections"



- Client vs server not explicit at transport layer
  - » At best, filters can only guess based on payload
  - » Port numbers are ambiguous
- Request-vs-response resides only in apps
  - » Must admit liberally and rely on endpoints
  - » App state transient, relying on it is unsafe
- Fix: Application mark request-response to transport level

Think very carefully before deploying UDP "connection" packet filters

Multicast Traffic: All or Nothing



- If IGMP makes a hole in the firewall, it's an unbounded hole
  - » IP cannot distinguish packets for distinct apps using same multicast address, must deliver all packets to all subscribing apps on all hosts
  - » Applications need to filter packets that are directed to another conversation
- Some multicast application protocols (e.g. RTP) do not even use request-response
- Fix: Open research problem

Protocol Design Recommendations

Ad hoc techniques (proxies…) work, but:

- Eliminate back-channels
  - From transport layer up
  - Allow push-only or pull-only modes
- UDP: Expose request/response
- Multicast: No good solution yet
  - Cannot predict all applications and hosts sharing group address

Security is important and needs to be considered in protocol design.

Experience with SURF

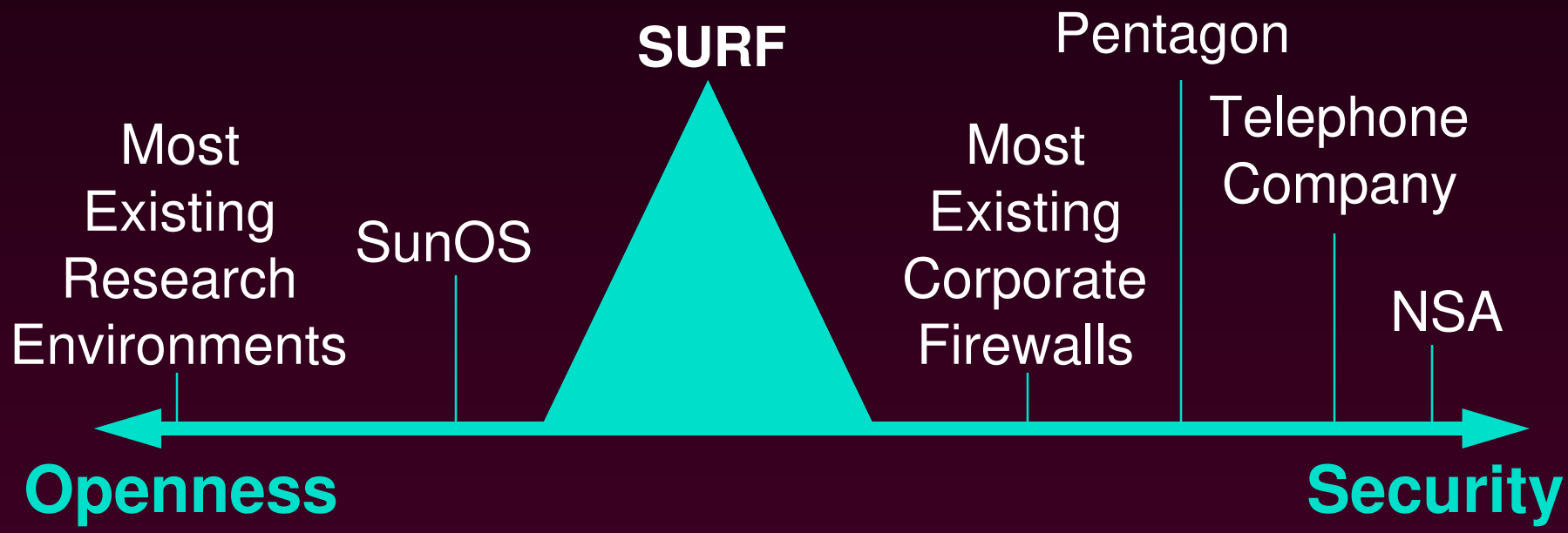
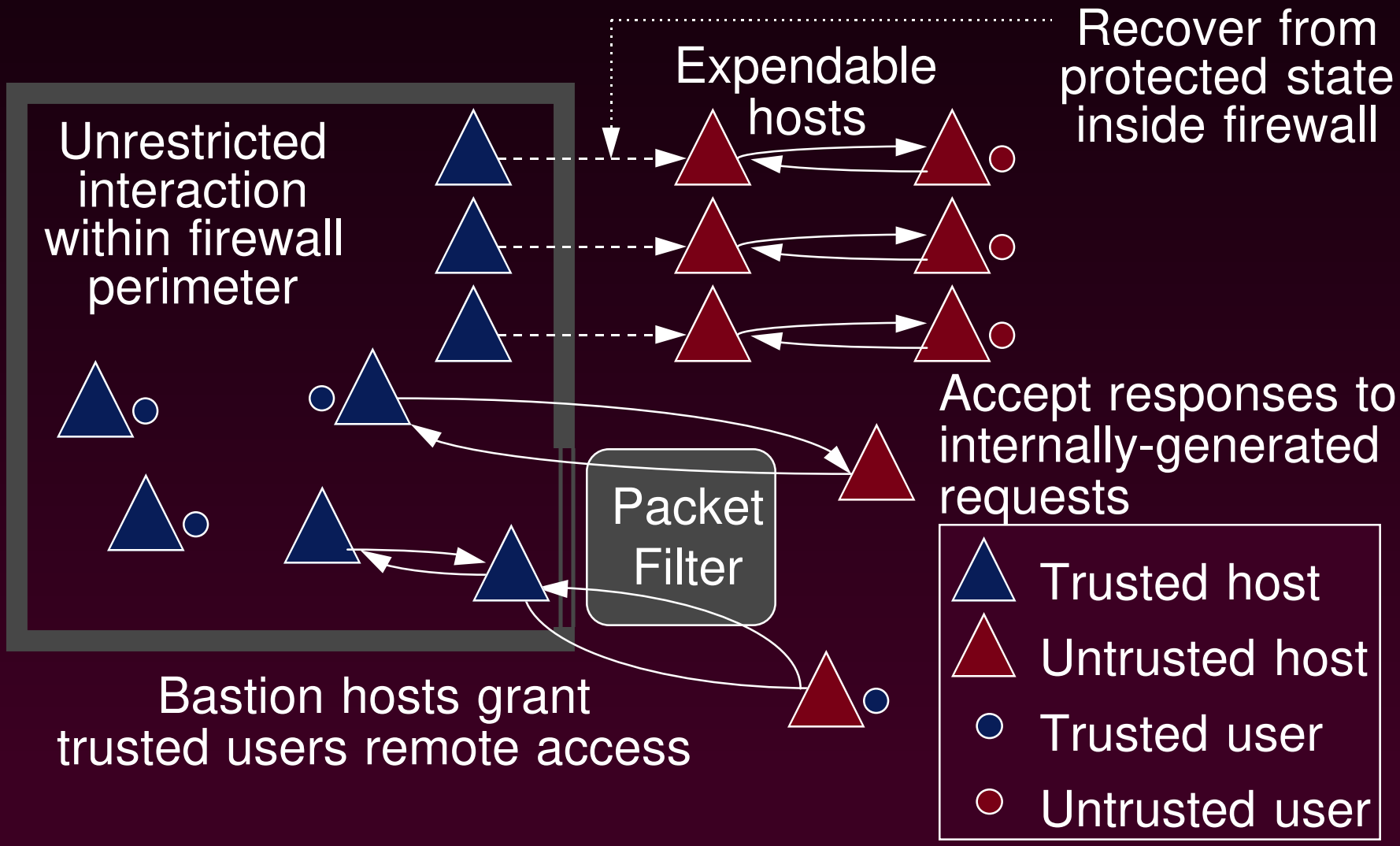
- Firewall transparency liked by users
- Easy to publish personal Web pages
- Ease increases acceptance of security
- Right security model for our environment
- No breakins detected (yet!)

Conclusions

- One-way firewalls or automatic publishing may be useful in other environments
- Designers of protocols and packet filtering hardware should consider request-response
- Lots of little firewalls is a win
- DO try SURFing at home!

SURF in the Security Spectrum

- SURF fills a void between existing corporate firewalls and the openness in existing research environments



SURF Implementation