



# **Leveraging IPsec for Mandatory Access Control of Linux Network Communications**

Trent Jaeger

*Department of Computer Science and Engineering*

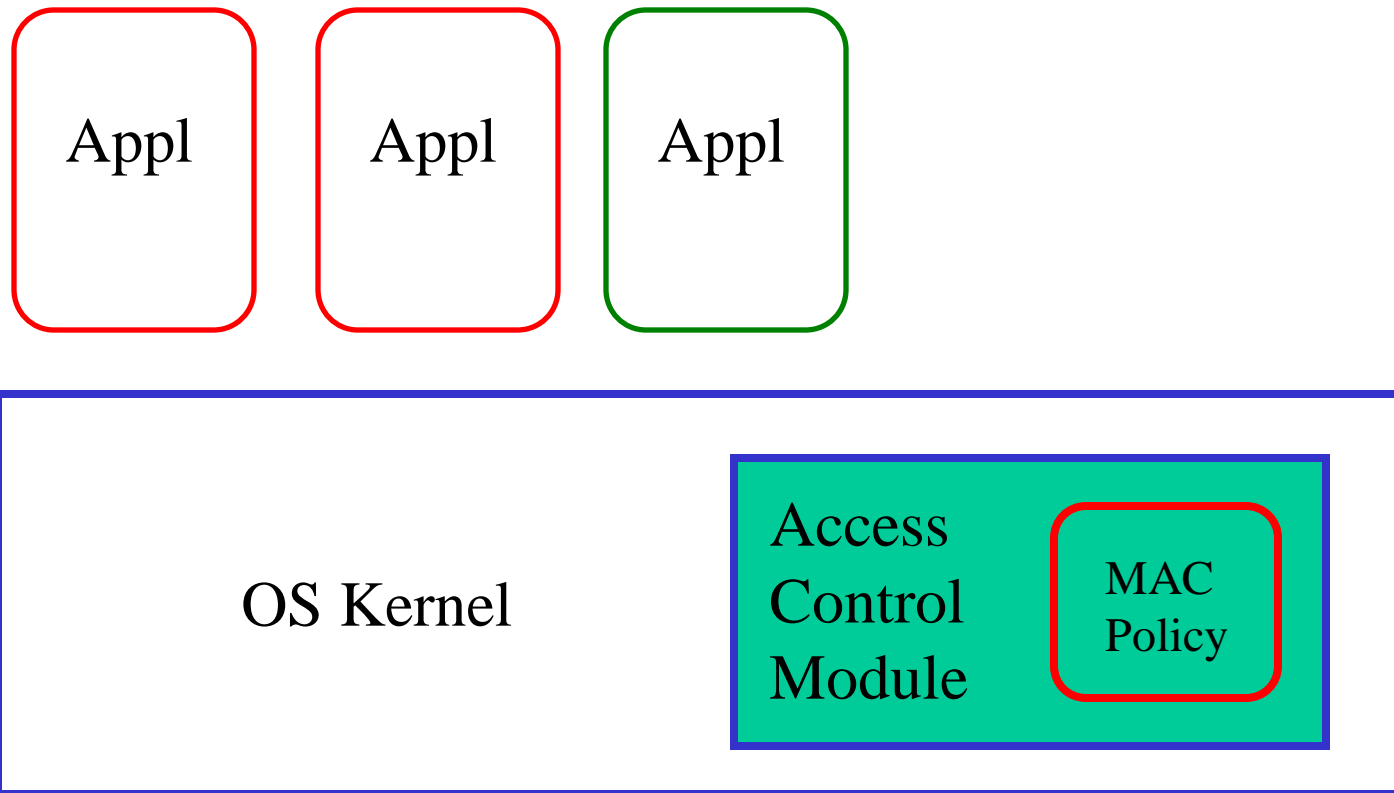
*Pennsylvania State University*

December 6, 2005



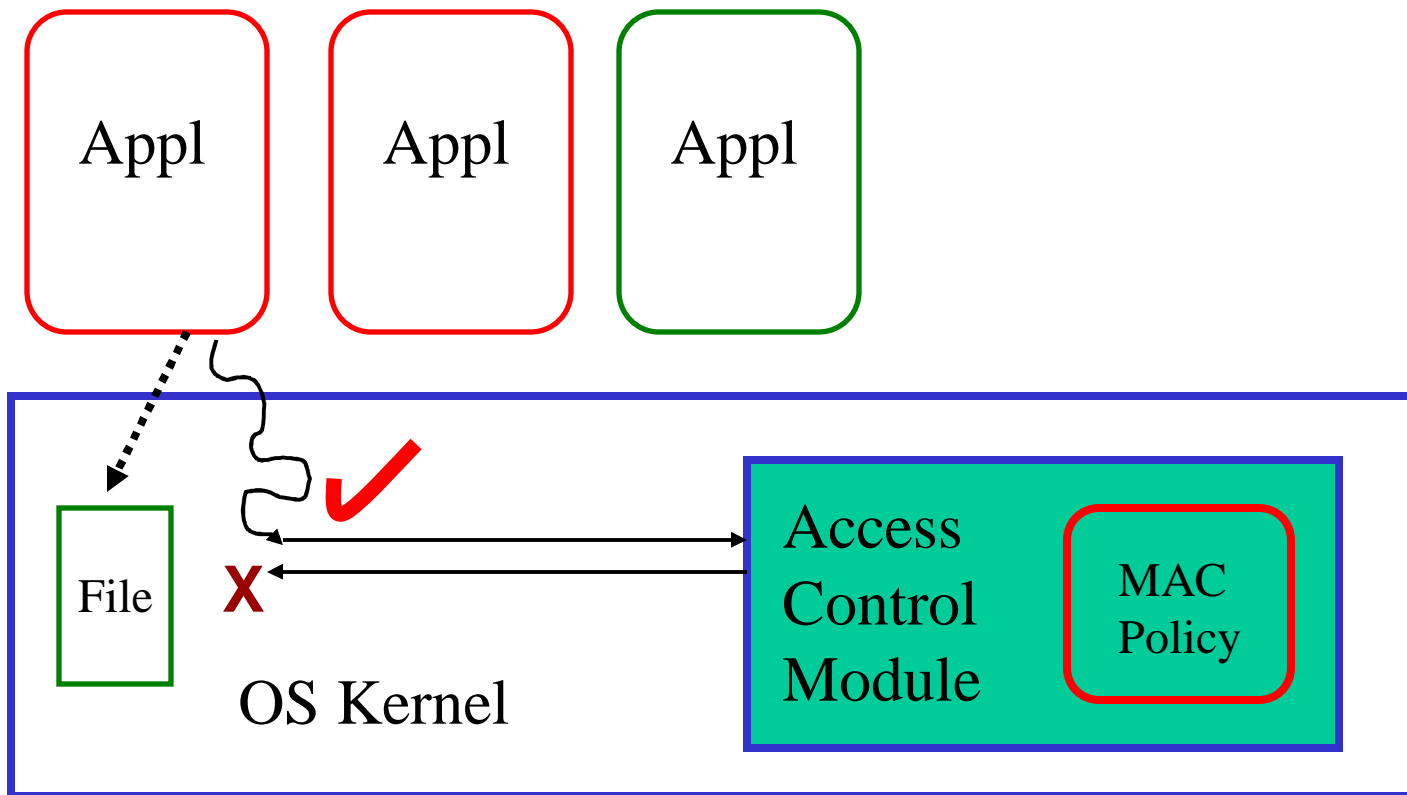


# Mandatory Access Control



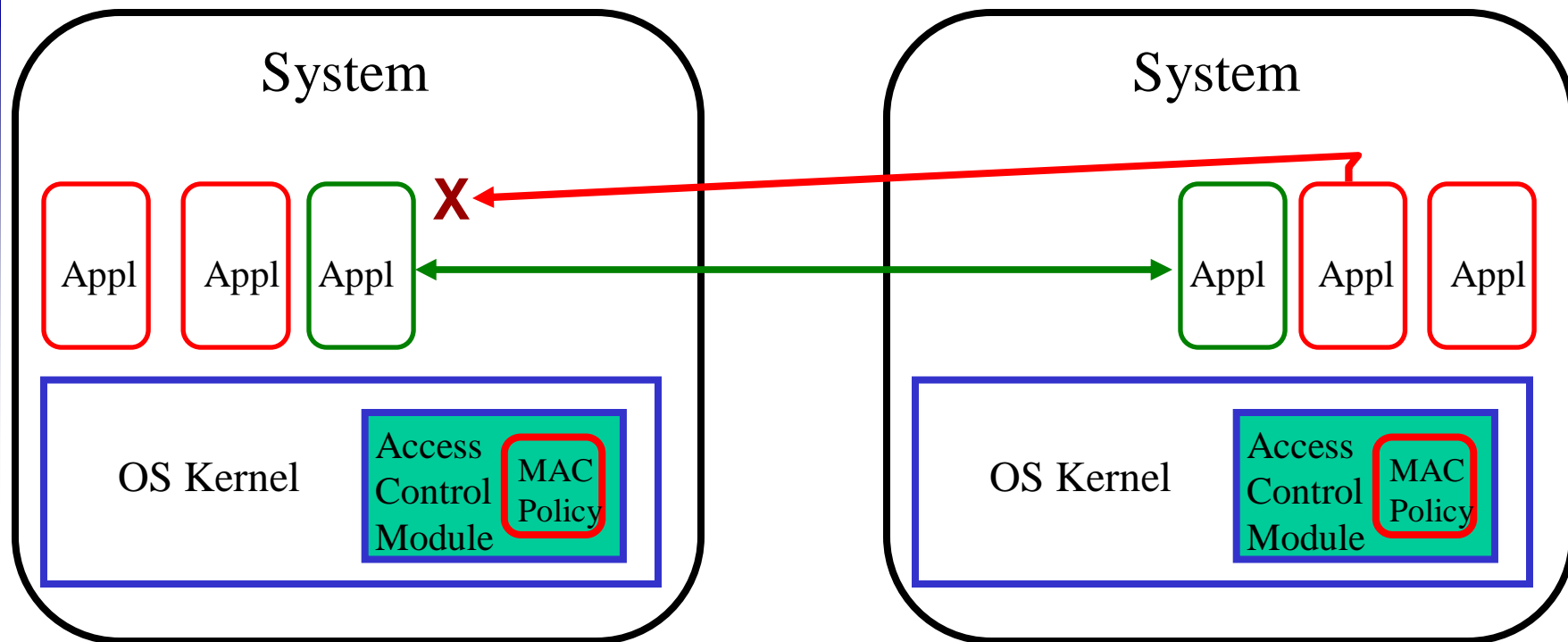


# Mandatory Access Control



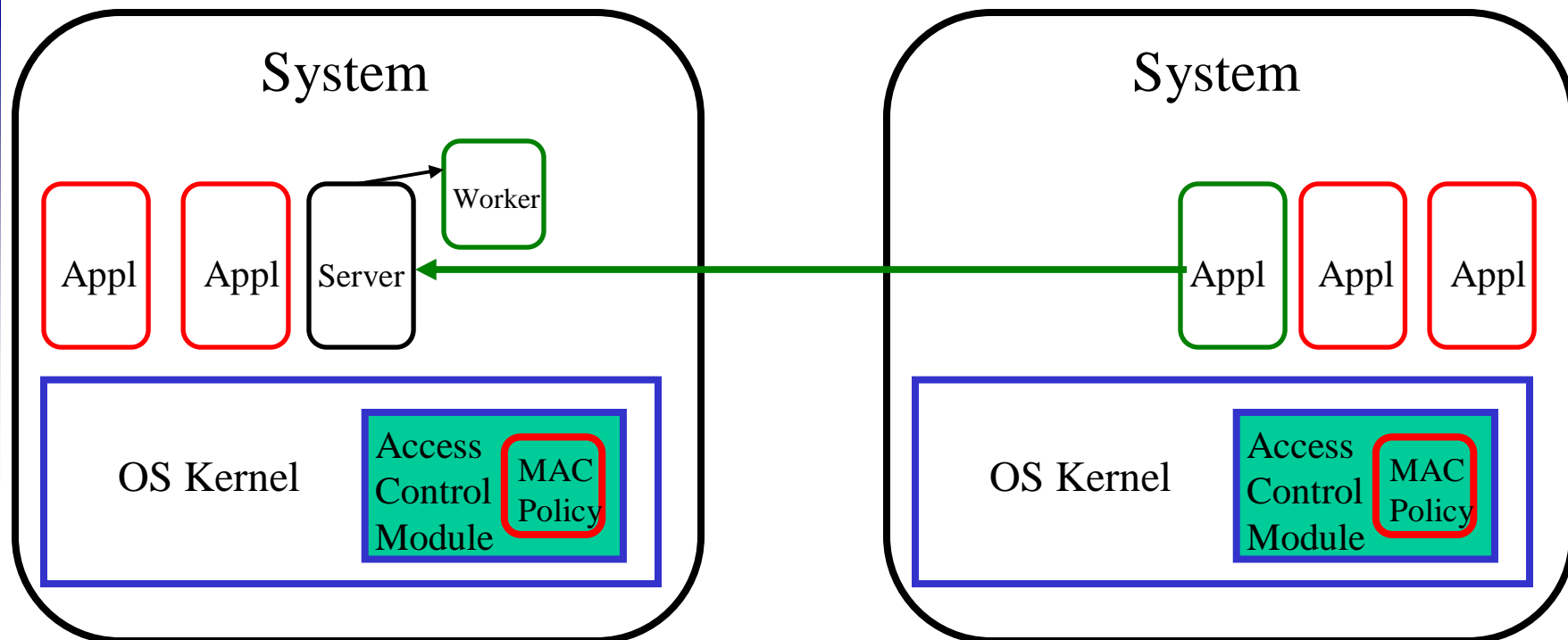


# Network MAC



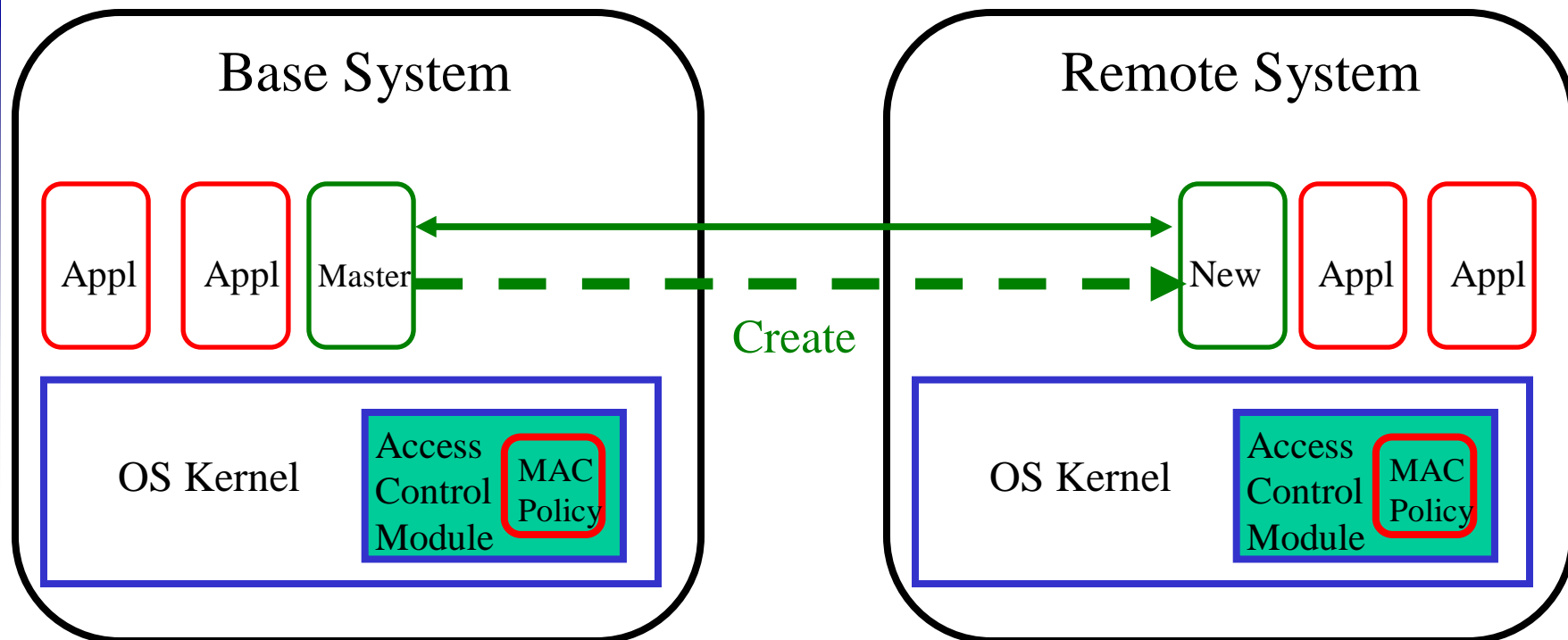


# Client-Server MAC





# Location-independent MAC





# Assumptions

- ❑ Mutual Trust in Labeling and Enforcement
  - Within administrative domain
  - Cross-domain trust is more challenging
  - Must authenticate, verify enforcement abilities, etc.
- ❑ Compatible Policies
  - Labels need to have consistent meaning
  - Negotiation of labels is possible
- ❑ Integrity-Preserving Communication
  - Strong crypto
- ❑ Here, we discuss the basic mechanism





# Alternatives

- ❑ SSL/TLS
  - Secure communication between applications
  - PKI identification (know user); no labels (don't know access)
  - Difficult to integrate into a kernel-enforced MAC framework
- ❑ IPsec
  - Secure communication between hosts/ports
  - Coarse granularity of identification, typically hosts
  - Need labels at application granularity
- ❑ IP Security Options
  - IP header labels
  - Parser IP headers on each packet -- performance/complexity death
- ❑ OpenBSD KeyNote
  - Authorization statements with keys
  - Integrated with IPsec -- But, discretionary in nature





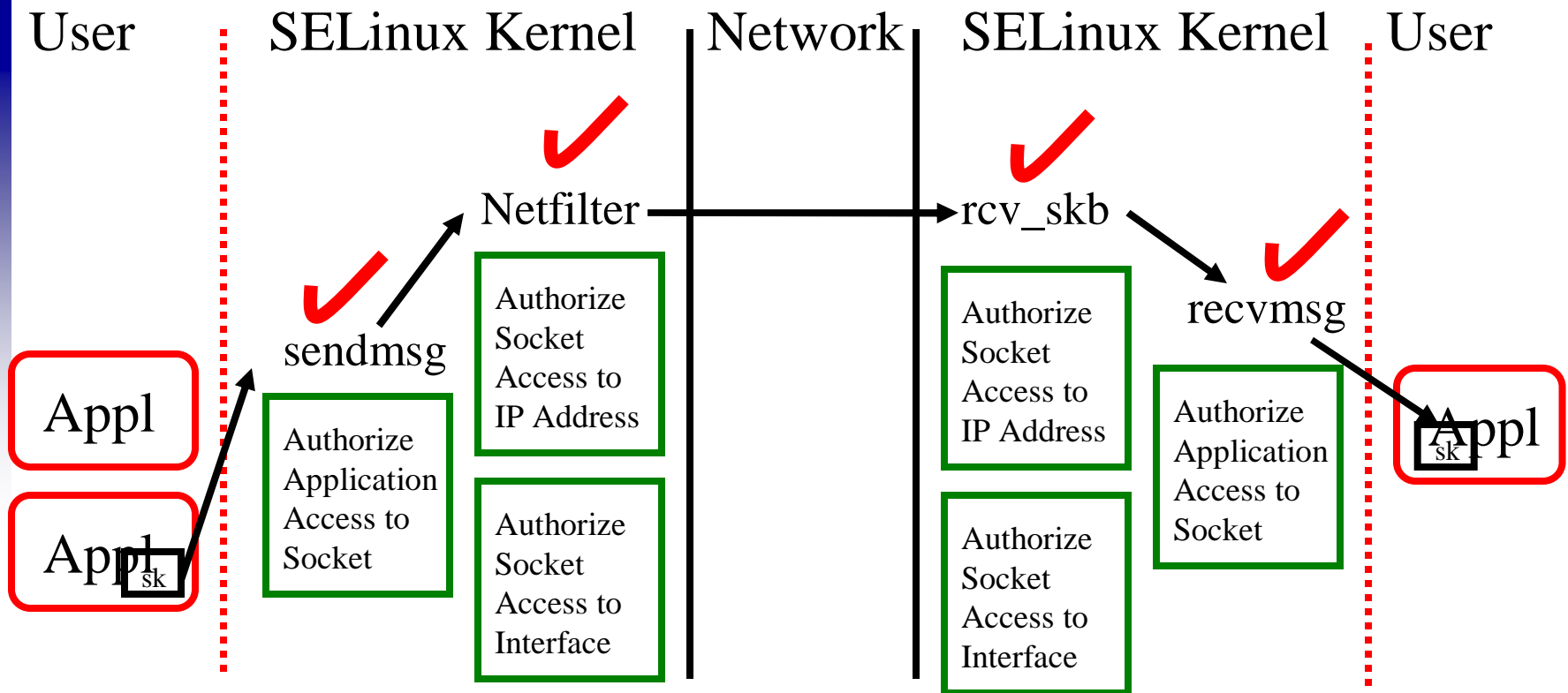
# Labeled IPsec

- ❑ Leverage IPsec Advantages
  - Secure communication
  - Easy to integrate to kernel MAC
- ❑ Add MAC Labeling to IPsec
  - Control application access to IPsec “channels”
  - Can only send/receive with MAC permission
- ❑ Results
  - Application to application control is possible
  - BLP controls between applications on different machines
  - Applications can use labeling information
    - Label child processes
- ❑ Part of Linux 2.6.15-rc3-mm1 kernel patch
  - Will be in 2.6.16 kernel





# Current MAC Network Controls





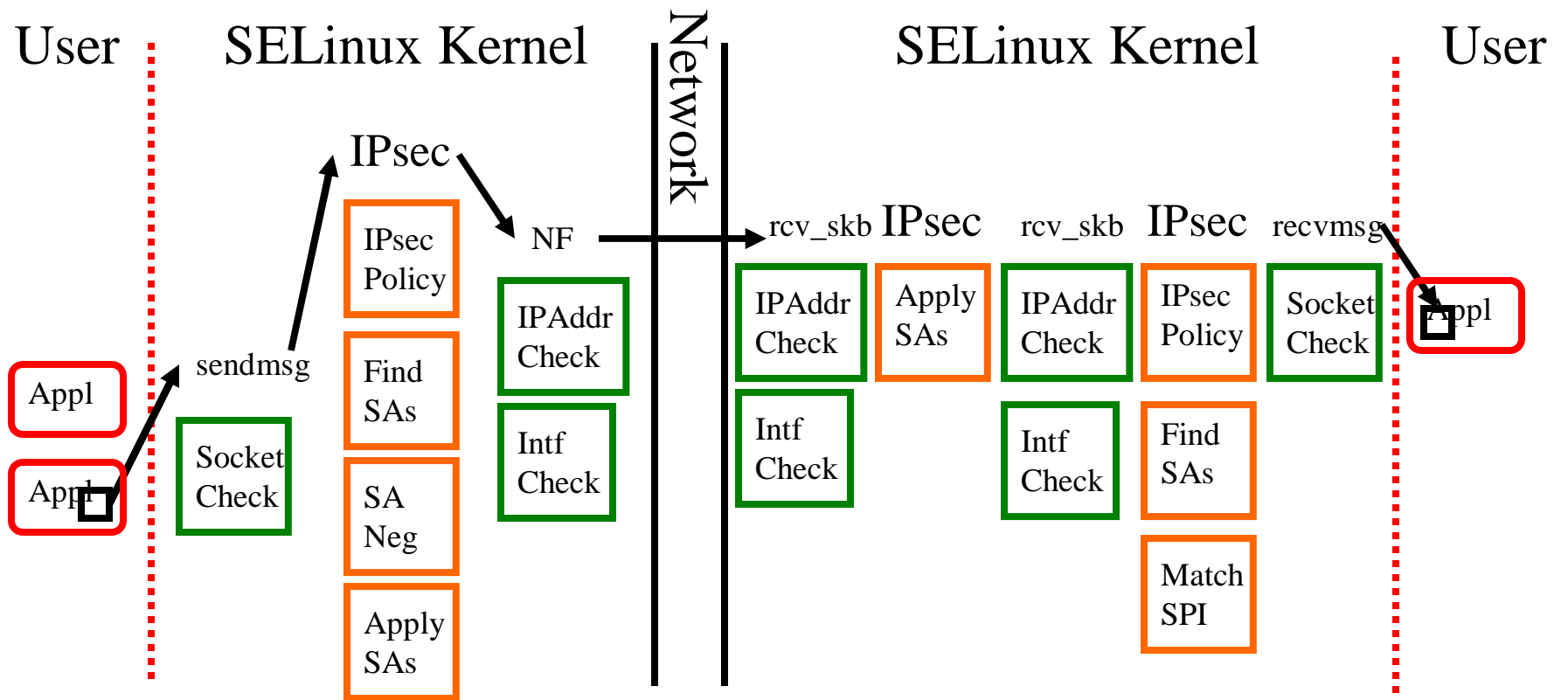
# IPsec

- ❑ Privacy and authentication services at the IP layer
  - IPv4 and IPv6
  - Protocols: ESP and AH
  - Paths: host-host, gateway-gateway, host-gateway
  - Transport or tunnel: single or multiple layers of security protocols
- ❑ Security Policy
  - Defines security protocols, mode for source-destination (port)
  - Input to negotiation
- ❑ Security Associations
  - Simplex representation of IPsec connection
  - Per protocol (AH or ESP)
  - One mode (transport or tunnel)



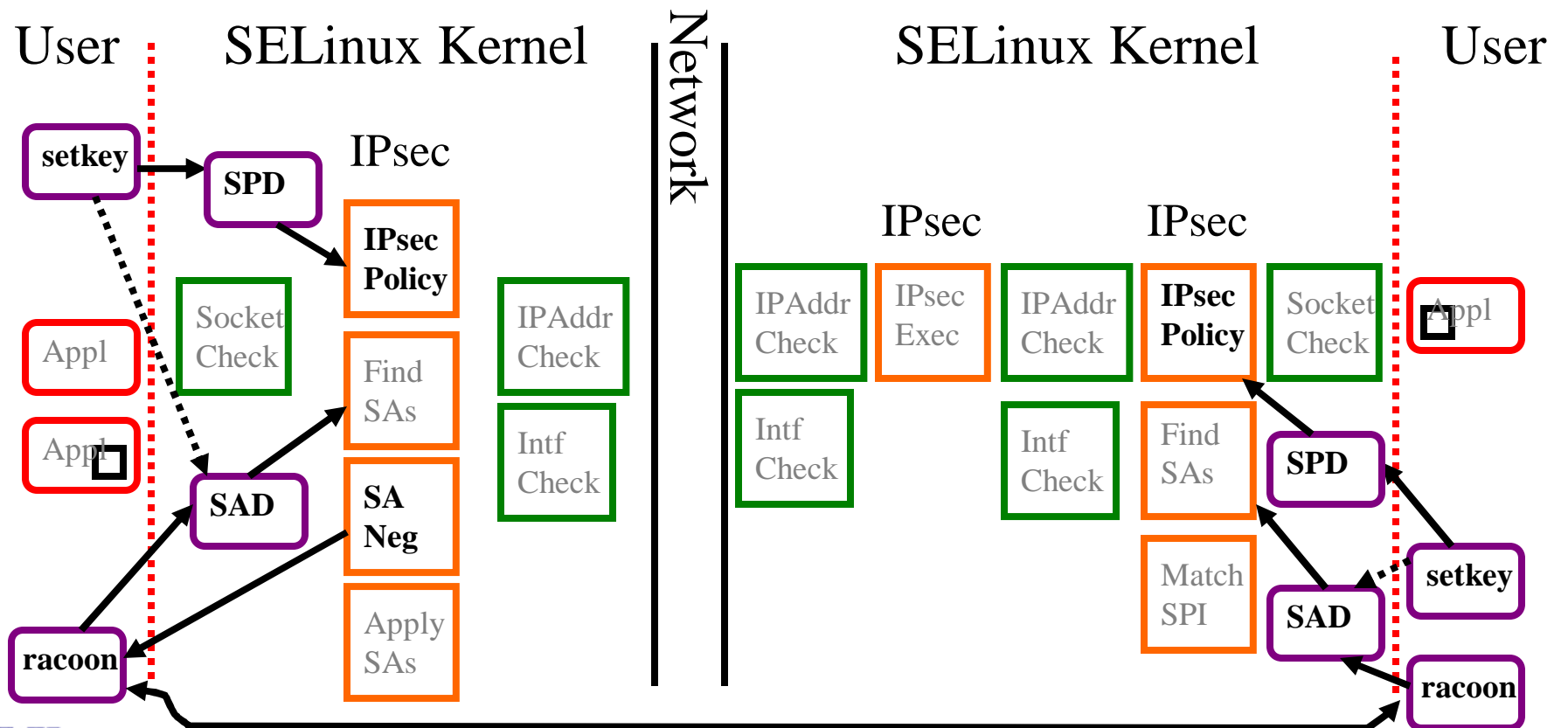


# IPsec and MAC Processing





# IPsec Tools





# Setkey Policy Changes

## ❑ Setkey SPD entries

```
spdadd 9.2.9.15 9.2.9.17 any -ctx 1 1 "system_u:object_r:zzyzx_t"  
    -P in ipsec esp/transport//require ;  
spdadd 9.2.9.17 9.2.9.15 any -ctx 1 1 "system_u:object_r:zzyzx_t"  
    -P out ipsec esp/transport//require ;
```

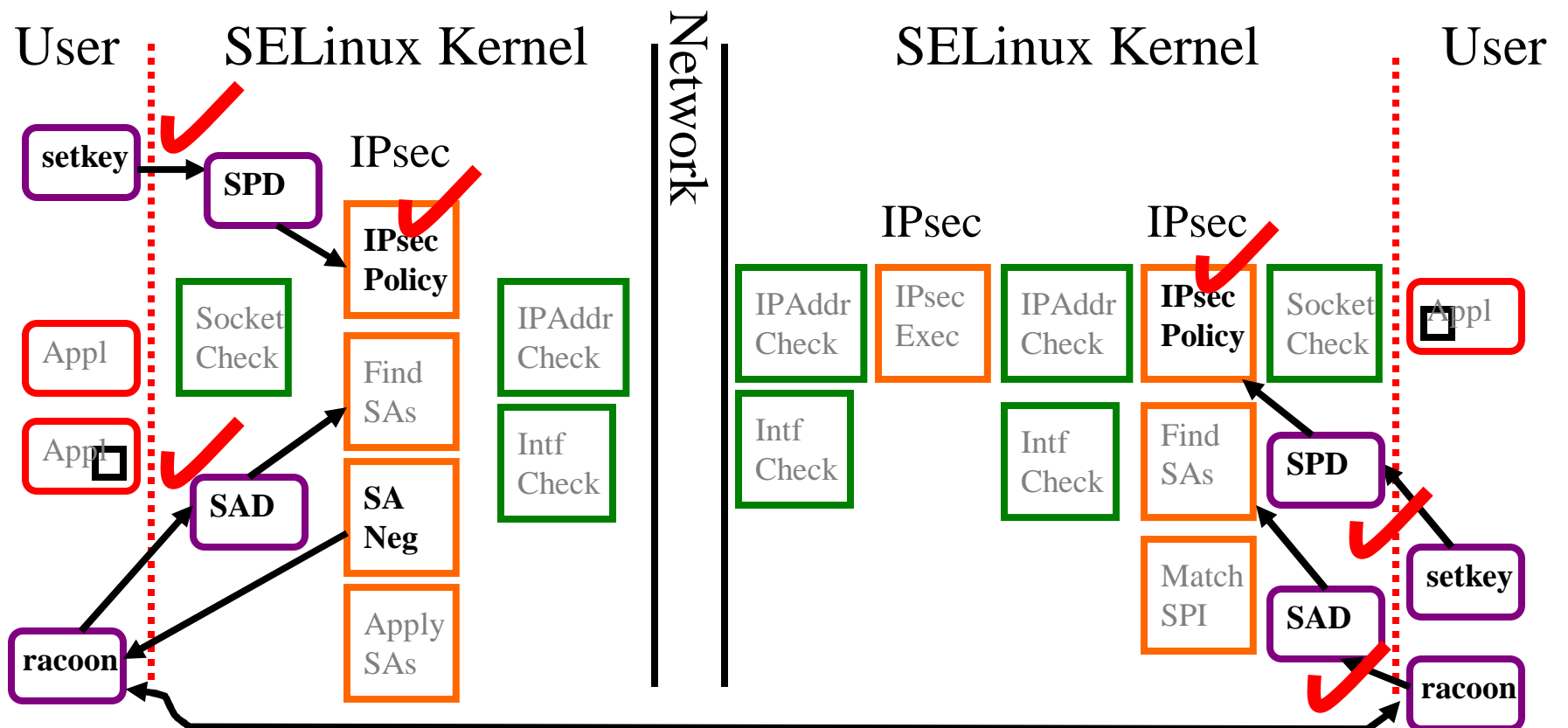
## ❑ Setkey SAD entries (optional as racoon can negotiate)

```
add 9.2.9.15 9.2.9.17 esp 0x123456  
    -ctx 1 1 "system_u:object_r:zzyzx_t"  
    -E des-cbc 0x0000000000000000;  
add 9.2.9.17 9.2.9.15 esp 0x123457  
    -ctx 1 1 "system_u:object_r:zzyzx_t"  
    -E des-cbc 0x0000000000000000;
```





# New LSM Hooks





# New LSM Hooks and SELinux Implementations

- ❑ **xfrm\_policy\_alloc**
  - Done when policy is added to the SPD (under xfrm\_selector)
  - Authorize subject that is updating SPD
  - Allocate security data structure in new xfrm\_policy
  - xfrm\_sec\_ctx
    - Domain of interpretation
    - Algorithm
    - Context length (string length)
    - Security ID
    - Context String
  
- ❑ **xfrm\_policy\_lookup**
  - Authorize socket's use of policy with security context
  - Only retrieve/build SA's with the security context of the policy
  
- ❑ **xfrm\_state\_alloc**
  - Done when SA is added to SAD
  - Authorize subject that is updating SPD
  - Allocate security data structure in new xfrm\_state



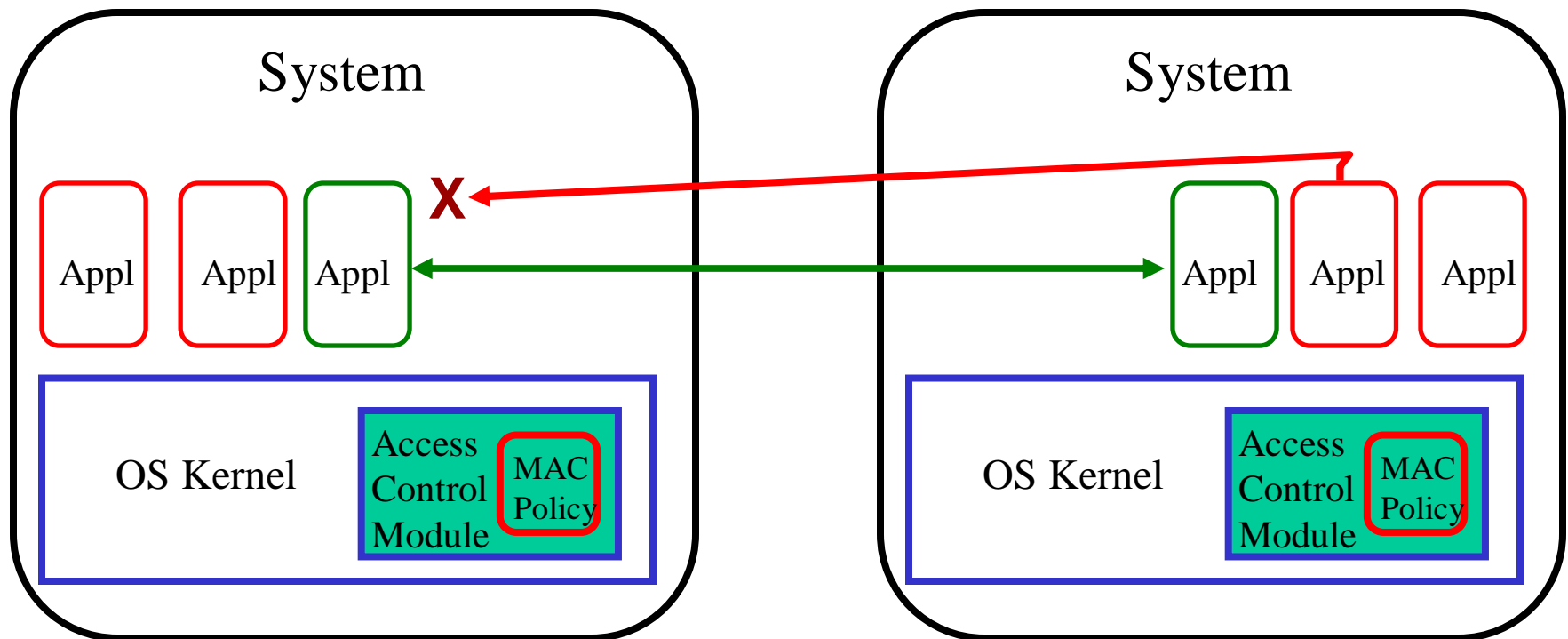


# Overall MAC Control

- ❑ (1) When labeled IPsec packet
  - Authorization of policy enforces access
    - Output: SAs must match policy selected
    - Input: SAs must have SPI for corresponding policy
- ❑ (2) When IPsec packet with no label
  - Must have access to unlabeled associations
- ❑ (3) When not IPsec packet
  - Must have access to unlabeled associations
- ❑ Extend existing input (rcv\_skb) and output (Netfilter) hooks
  - Output: if no labeled SA, then authorize for 'unlabeled'
  - Input: if no labeled SA, then authorize for 'unlabeled'

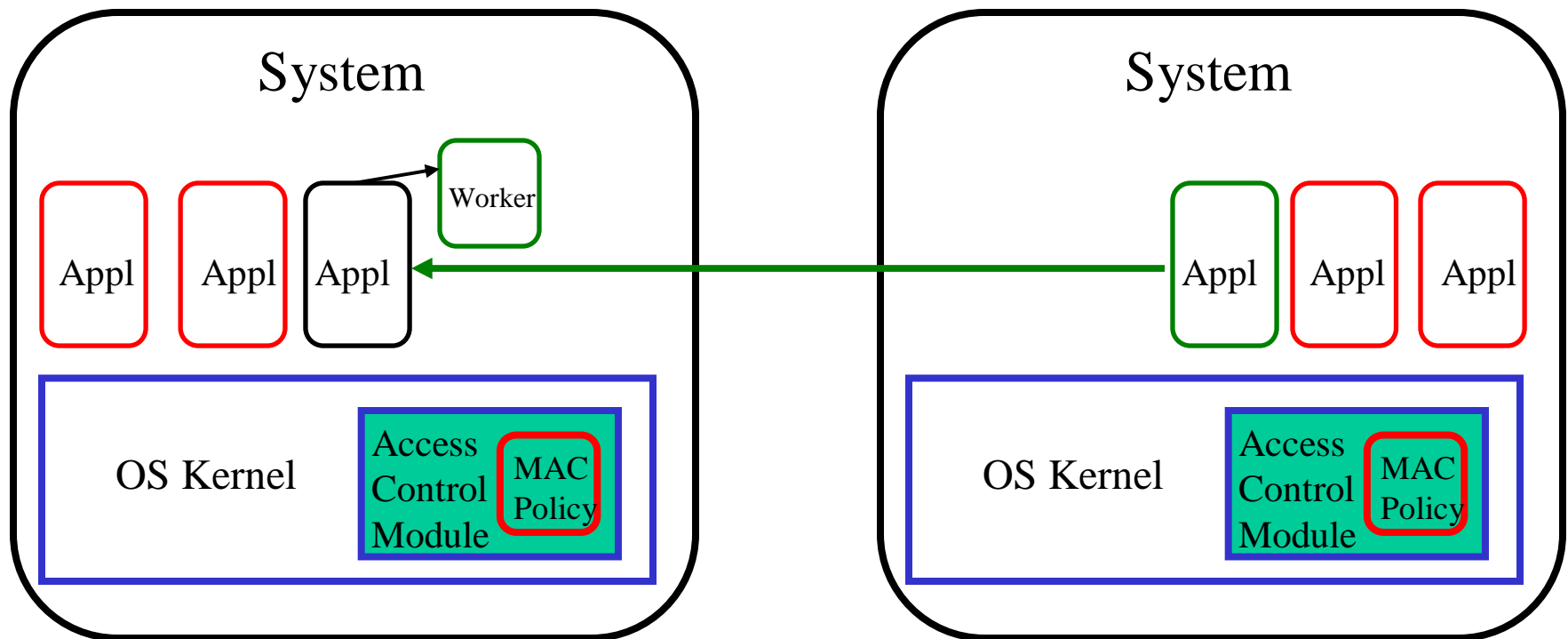


# IPsec-MAC Usage



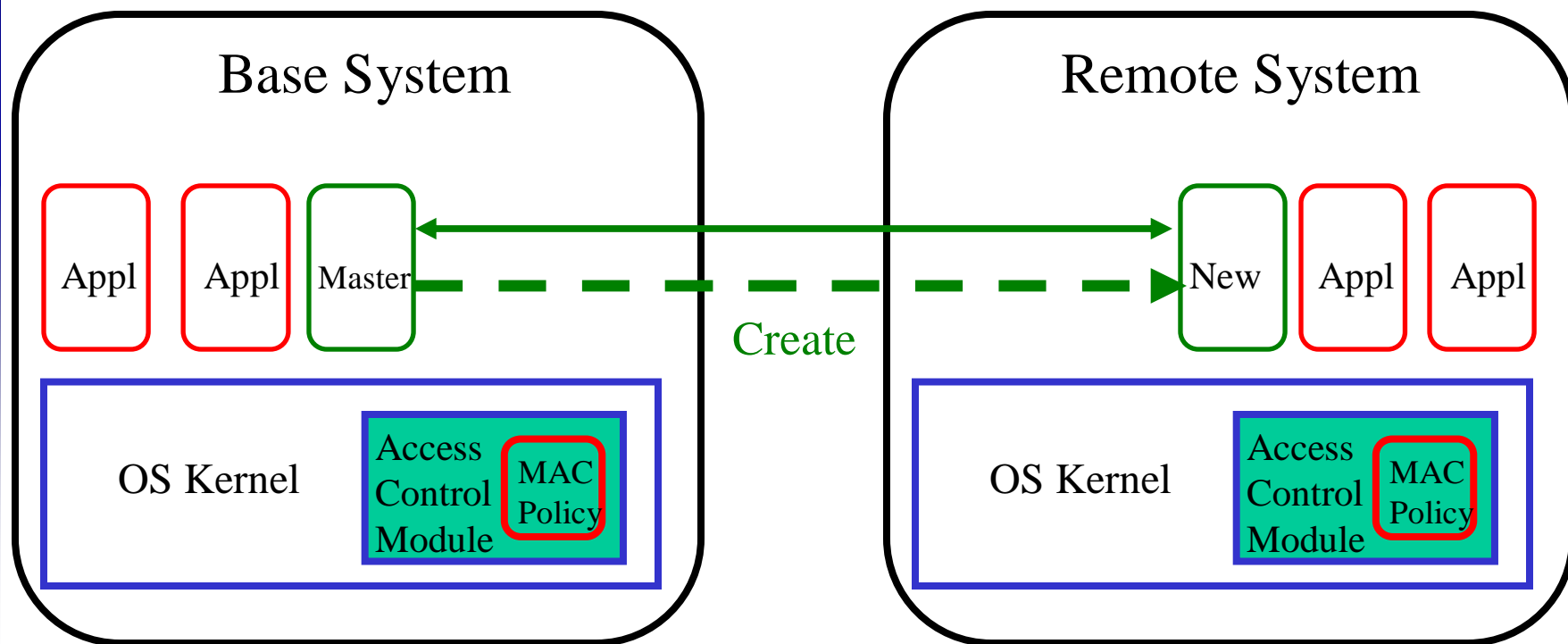
- (1) Green application can only use green IPsec policy
- (2) Resultant negotiated SA is labeled green
- (3) Red cannot send to green because red is limited to red policy

# Client-Server Usage



- (1) Black must be able to access **green** policy (among others)
- (2) Black can extract label of SA for socket
- (3) Prototyped using `getsockopt(..., SO_PEERSEC)`

# Location-independent Usage

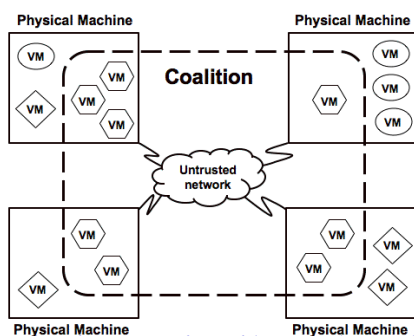


- (1) Master downloads code to remote system
- (2) Remote enforces **new green** access to **green SA** only
- (3) Enforcement -- Xen Prototype



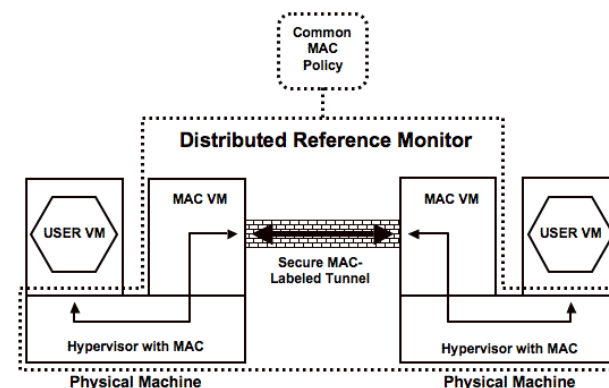
# Secure Distributed Platforms

- ❑ Joint work with IBM Research -- IBM Tech Report RC23778
- ❑ Location-independent computing
  - Distributed computation -- e.g., SETI@HOME
  - Mobile identity -- e.g., ATM
  - Geographically-distributed services -- e.g., search engine



- ❑ Solution: Distributed Reference Monitor

- **Tamperproof:** Attestation; Virtual Machine; Secure Communication; Integrity Protection
- **Mediation:** MAC enforced by VM system; MAC policy distribution
- **Simplicity:** "Smaller code base"; Simpler policy





# Issues

- ❑ Caching
  - Mapping of flows to IPsec policy (authorized)
  - May be multiple authorized policies per flow -- finer-grained
- ❑ Another hook
  - Get socket sid from module to check cache
- ❑ Label Extraction
  - More general solution needed for UDP
  - `setsockopt(..., SO_PASSSEC)` -- tell kernel to provide label in control message
- ❑ Supports transport
  - Tunnel -- keep interface updated throughout forward





# Summary

- ❑ Aim: Network MAC based on strong authentication on each packet
- ❑ IPsec is the kernel service that supports network control
  - XFRM IPsec implementation in Linux 2.6
- ❑ Integrate IPsec with LSM and SELinux
  - Control selection of policy for a socket
    - Propagated throughout SA retrieval/construction
  - IPsec-Tools modified to support the policy and SA contexts
    - Manual (setkey) and dynamic (racoon)
- ❑ Intrusiveness to critical path is minimal
  - 2 new LSM hooks on IPsec per packet processing – 2 offline
  - 1 more SELinux authorization for SA in rcv\_skb and Netfilter
  - Accepted in Linux mainline kernel





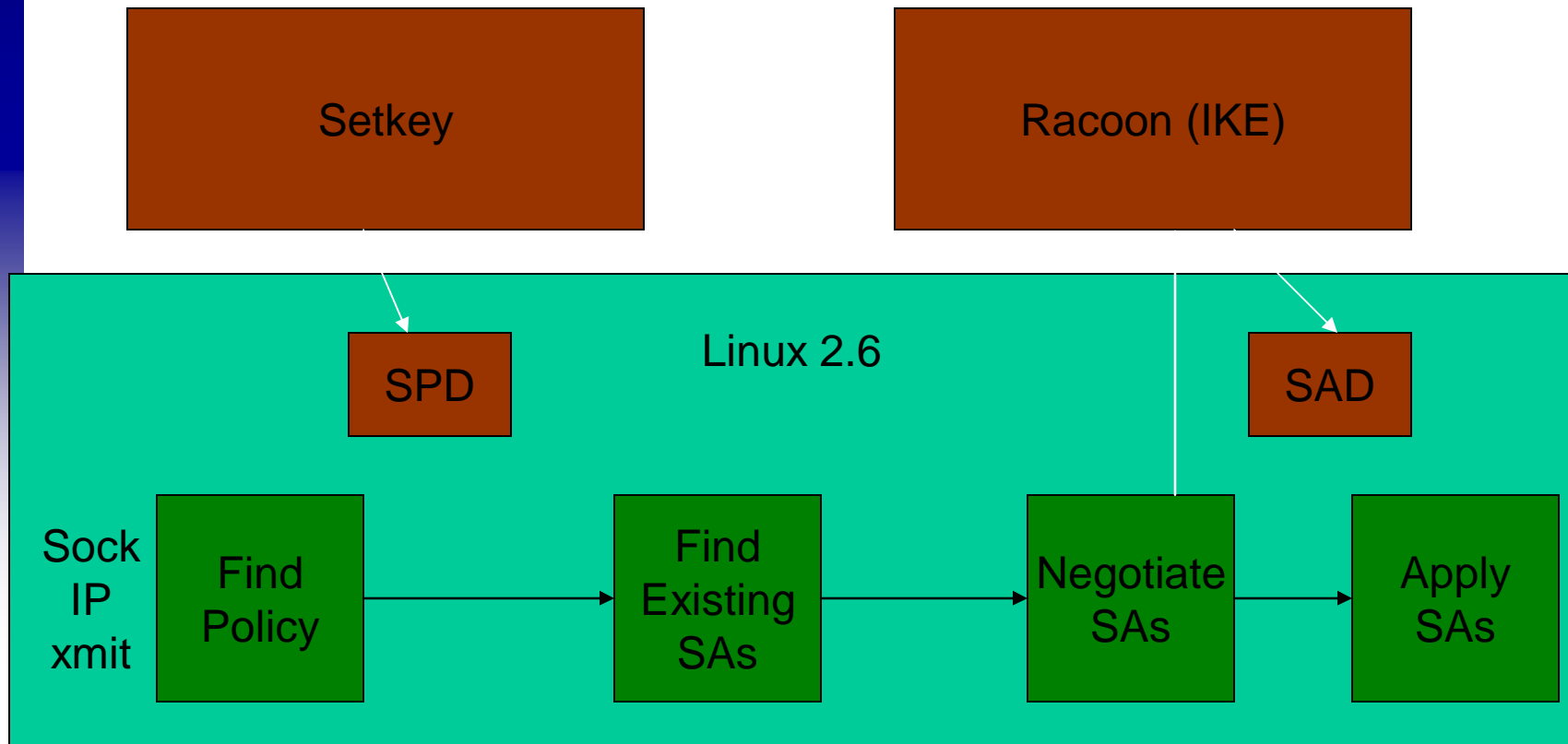
# Questions?

- ❑ Contact
  - Trent Jaeger, [tjaeger@cse.psu.edu](mailto:tjaeger@cse.psu.edu)
  - [www.cse.psu.edu/~tjaeger](http://www.cse.psu.edu/~tjaeger)
- ❑ IPsec system prototype report
  - IBM Tech Report
  - RC23642 -- With Serge Hallyn and Joy Latten
- ❑ Linux kernel
  - [www.kernel.org](http://www.kernel.org)
- ❑ SELinux
  - [www.nsa.gov/selinux](http://www.nsa.gov/selinux)



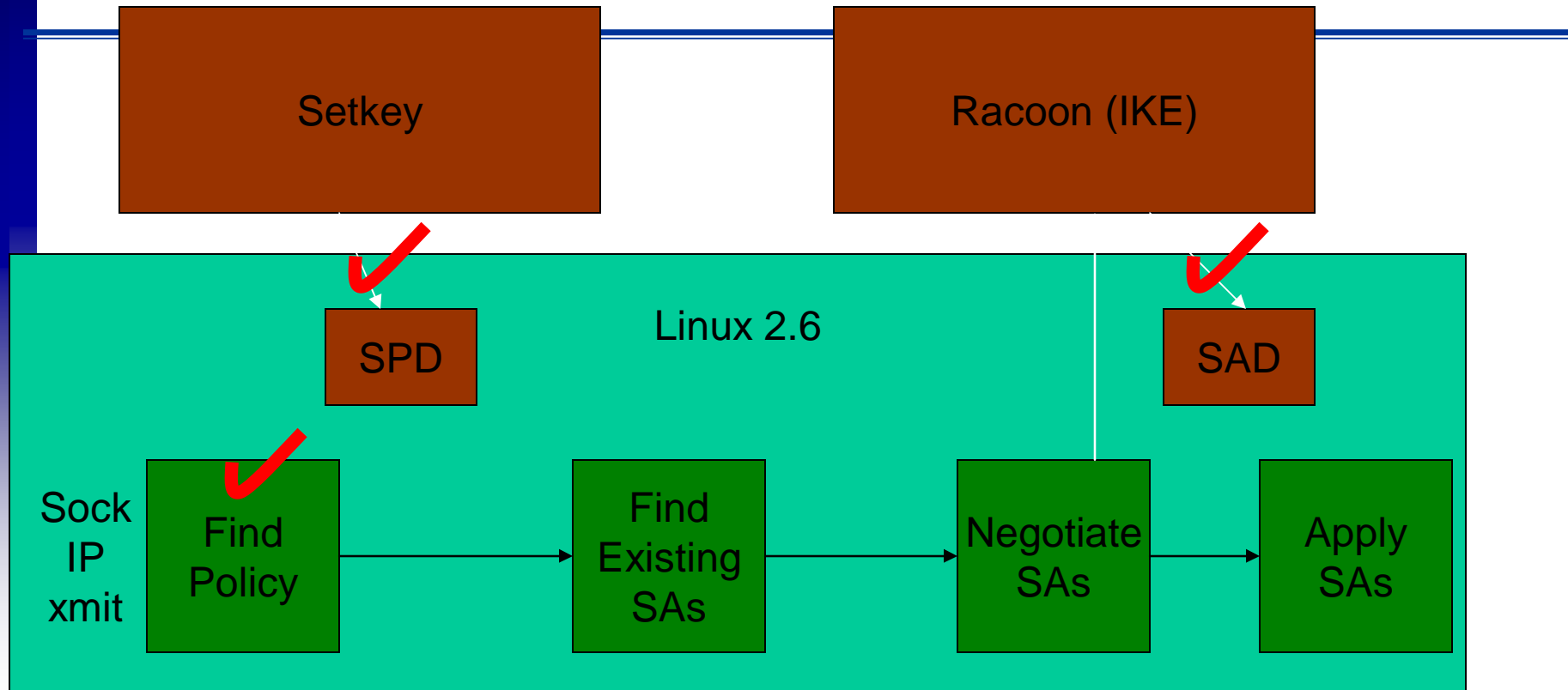


# IPSec protocol – IPSec Tools/Linux XFRM (output)



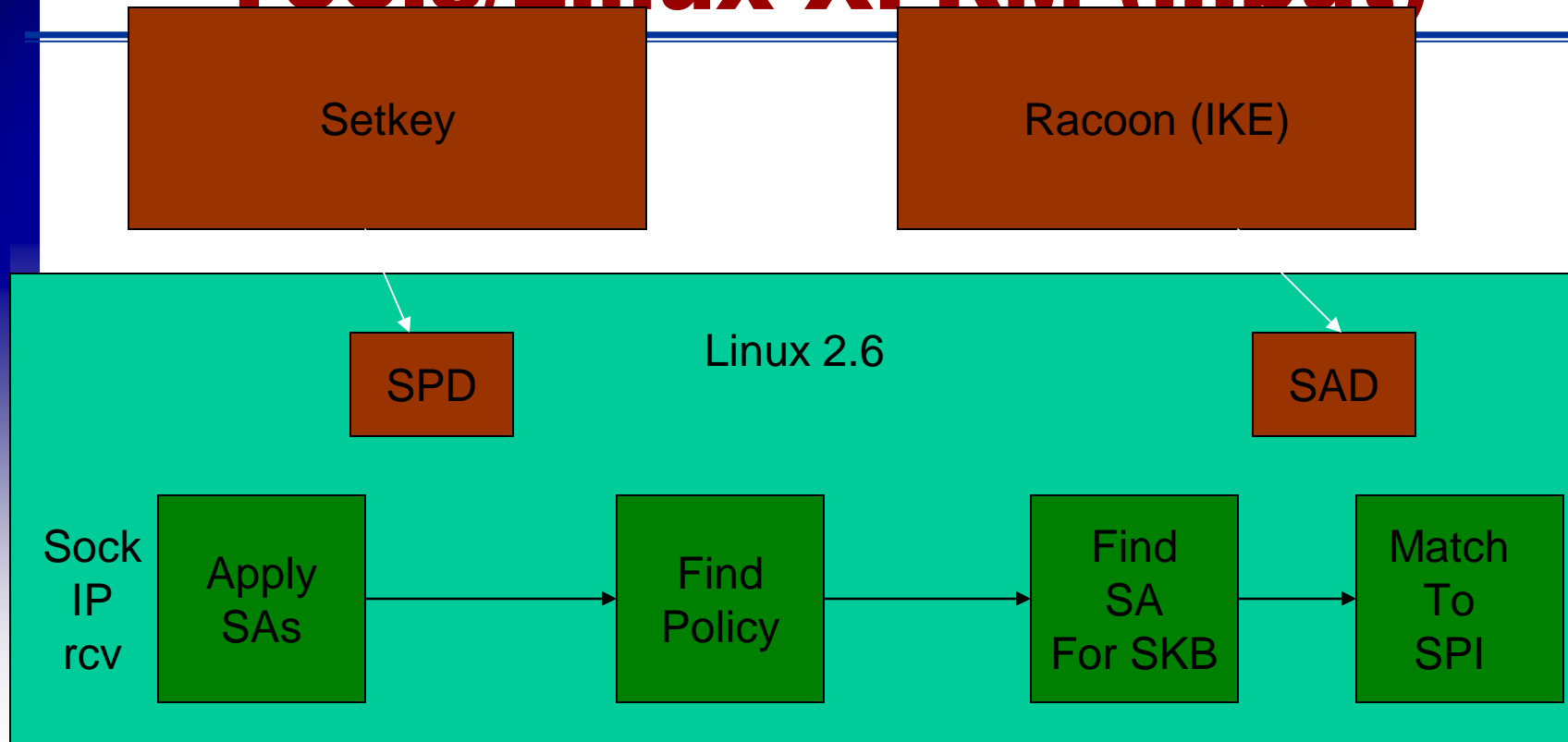


# New LSM Hooks (output)

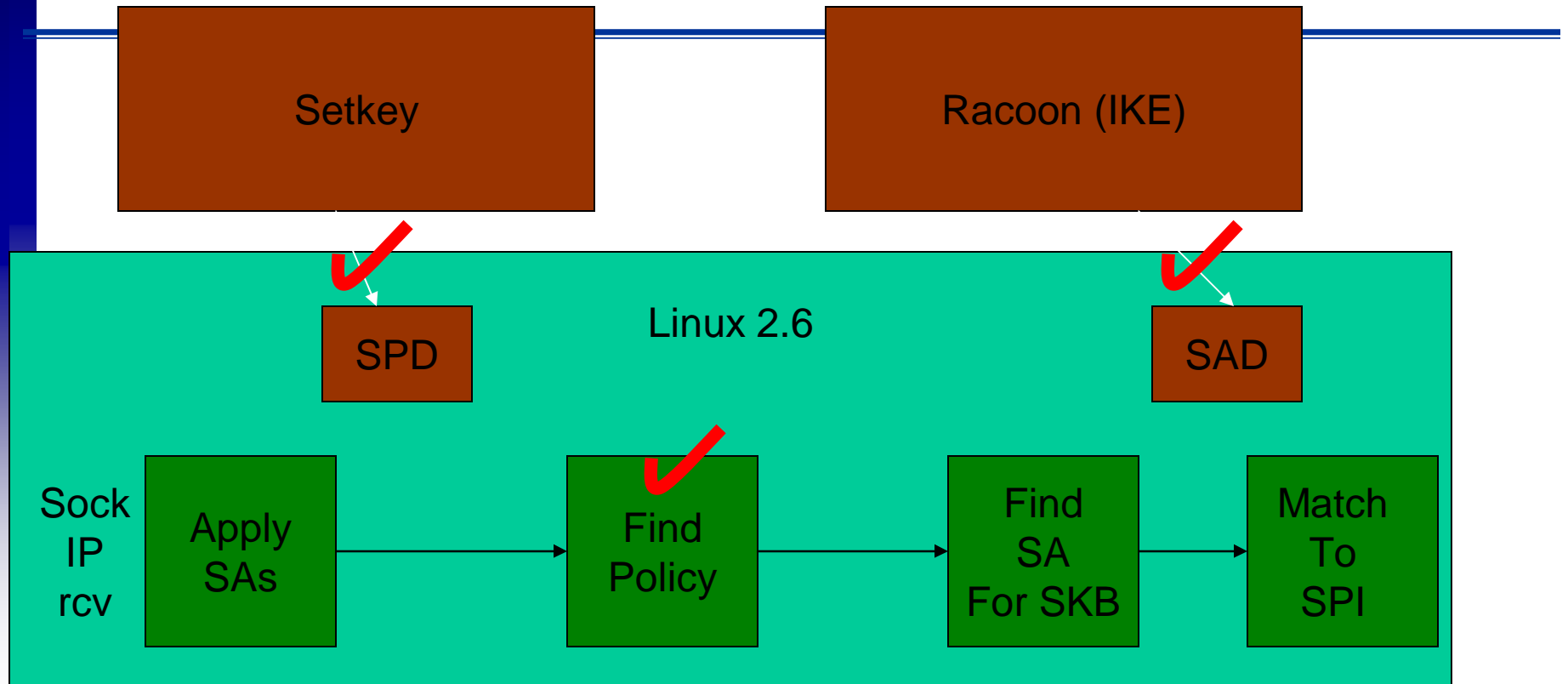




# IPSec Protocol: IPSec-Tools/Linux XFRM (input)



# New LSM Hooks (input)





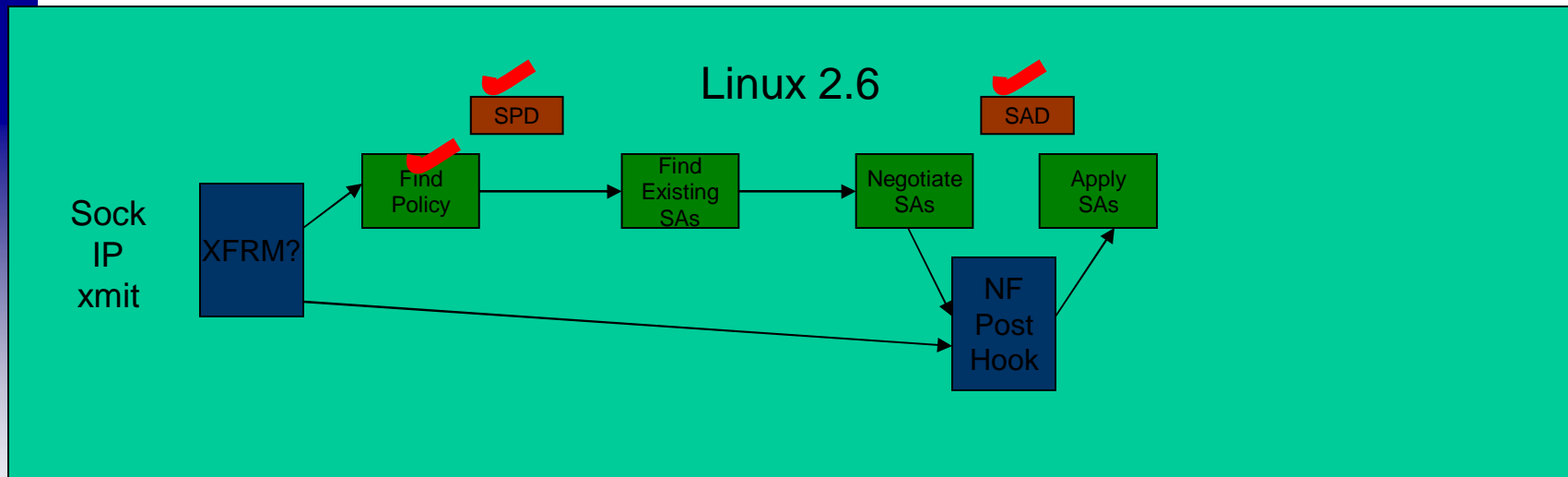
# Negotiation model

- ❑ Initiator is authorized to only one SA per source-destination-port
  - Granularity of selectors
  - Socket options might distinguish further by per socket policy
    - Not currently supported
  
- ❑ Initiator's racoon receives request with policy
  - Authorized to send unlabelled packets only
  
- ❑ Negotiation is a simple context match
  - Types should be same on both sides to indicate same semantics
  - Polyinstantiation
  
- ❑ Each side builds an SA with context
  - Control over sockets that can sendto/rcvfrom SA context
  
- ❑ Same context in each direction for racoon
  - Racoon negotiates and builds SAs for both directions based on initiator's outbound
  - Verified for encryption algorithms
  - Does not apply to setkey (manual SA creation)



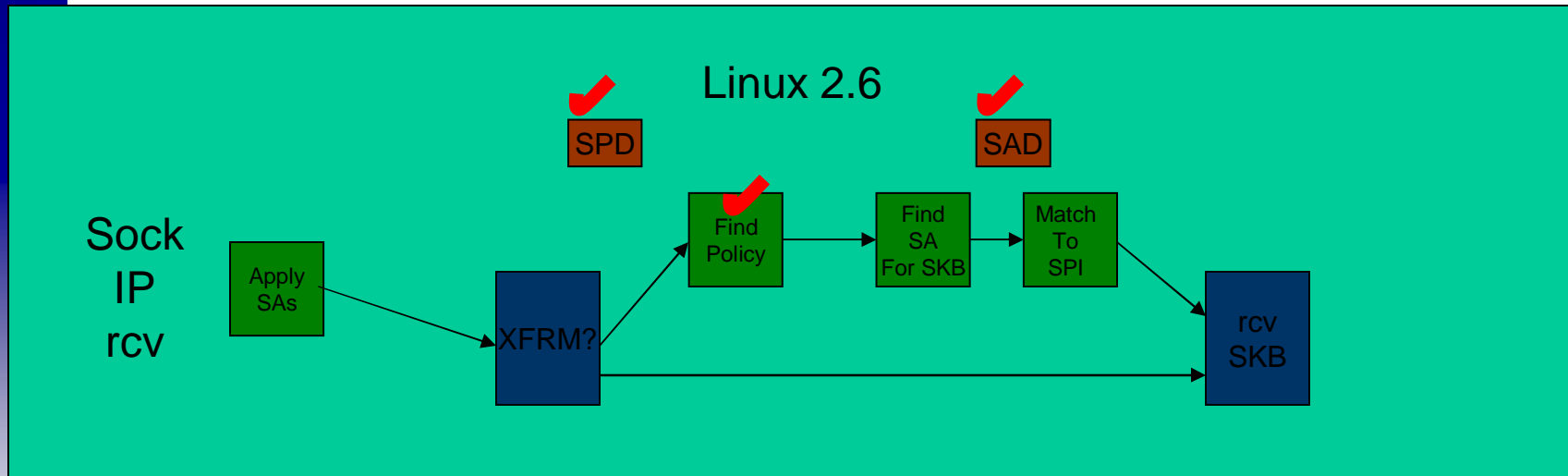


# Overall LSM Network Control (Output)





# Overall LSM Network Control (Input)





# Overall control (rcv\_skb and postroute\_last)

~~/\* if authorized\_xfrm then already authorized~~





# Issues

- ❑ Policy specification
  - sk\_policy vs. manual policy
  - Set by racoon for ISAKMP messages – can use unlabelled
  - Will get rejected unless unlabelled access is allowed
- ❑ No sock in some cases
  - E.g., ping and packet forwarding
  - Kernel is the subject in these cases
- ❑ Breadth of IPsec use tested
  - Transport for TCP, UDP, ICMP
  - Tunnel
- ❑ Patch acceptance
  - 15 files modified: 5 security; 5 net; 5 includes
  - IPsec-tools patch supports these changes
- ❑ Inter-system policy management
  - Single domain policy distribution to setkey (in addition to SELinux policy)
  - Cross-domain limited policy use
  - Applications use SSL