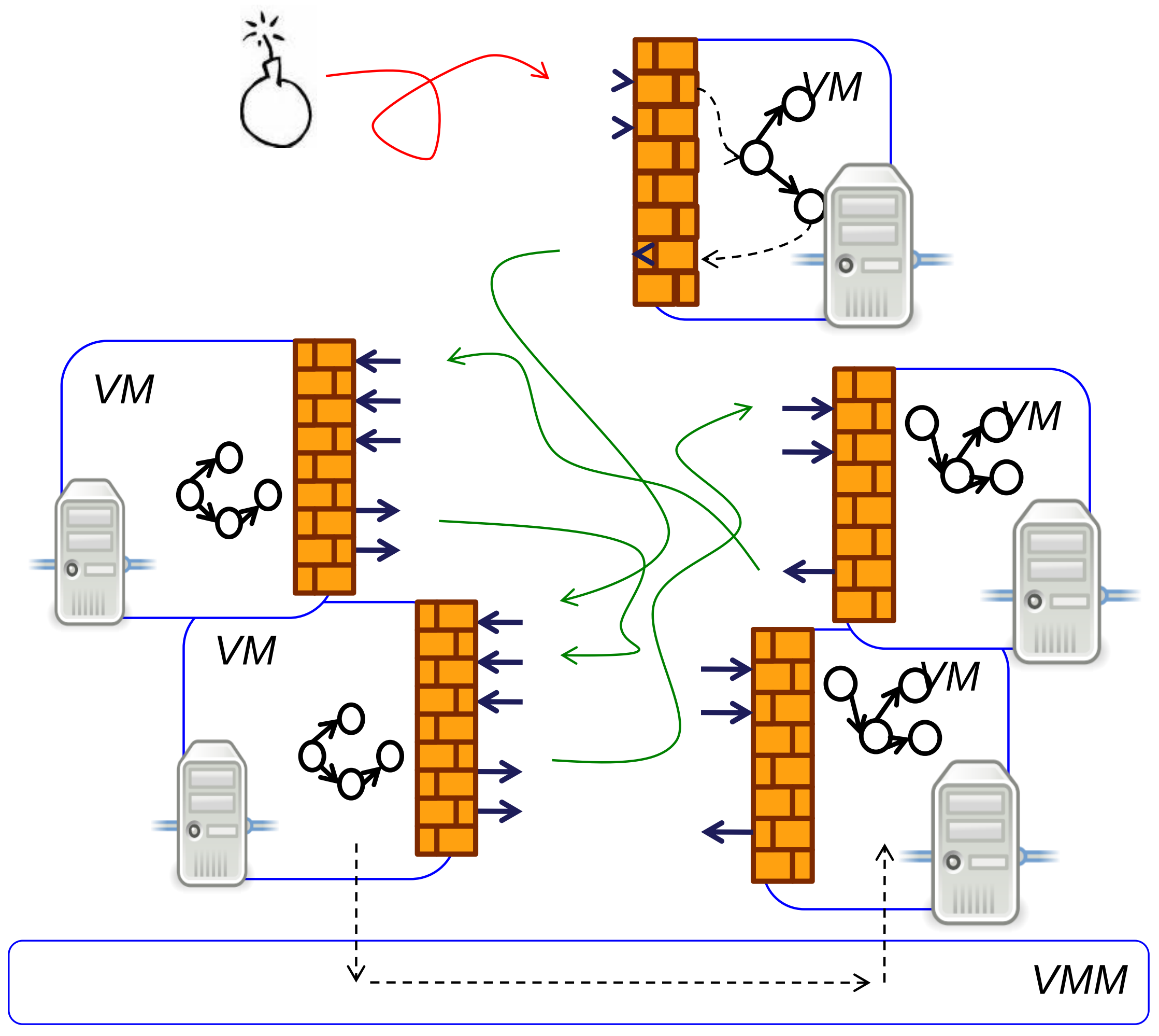
Detecting Attack Risks in Virtual Machine Environments

Sandra Rueda, Hayawardh Vijayakumar, Divya Muthukumaran,
Joshua Schiffman, Trent Jaeger and Swarat Chauduri

- ➢ Mandatory Access Control (MAC) enforcement mechanisms are now available at multiple layers: virtual machines (VMs), network policy, virtual machine monitors, and even user-level applications.
- ➢ How do we determine whether this set of policies when acting together prevent attackers from gaining control of a system?
- ➢ Attack graphs for this kind of environment are large and complex. What would administrators look for ?



OS MAC Policies

```
allow kernel_t proc_t:dir { read getattr lock search ioctl };
allow kernel_t proc_t:{ lnk_file file } { read getattr lock ioctl };
allow kernel_t proc_net_t:dir { read getattr lock search ioctl };
allow kernel_t proc_net_t:file { read getattr lock ioctl };
allow kernel_t proc_mdstat_t:file { read getattr lock ioctl };
allow kernel_t proc_kcore_t:file getattr;
allow kernel_t proc_kmsg_t:file getattr;
allow kernel_t sysctl_kernel_t:file { read getattr lock ioctl };
```

Network MAC Policies

```
iptables -t mangle -A INPUT -p udp --dport 22 -s 130.203.32.171 -d 130.203.32.56 -j SECMARK --selctx
system_u:object_r:ssh_server_packet_t:s0
iptables -t mangle -A INPUT -p udp --sport 67 --dport 68 -s 130.203.32.173 -d 130.203.32.56 -j SECMARK --selctx
system_u:object_r:dhcpc_client_packet_t:s0
iptables -t mangle -A OUTPUT -p udp --sport 68 --dport 67 -s 130.203.32.56 -d 130.203.32.173 -j SECMARK --selctx
system_u:object_r:dhcpc_client_packet_t:s0
iptables -t mangle -A INPUT -p udp --sport 53 -s 130.203.32.173 -d 130.203.32.56 -j SECMARK --selctx
system_u:object_r:dns_client_packet_t:s0
iptables -t mangle -A OUTPUT -p udp --dport 53 -s 130.203.32.56 -d 130.203.32.173 -j SECMARK --selctx
system_u:object_r:dns_client_packet_t:s0
iptables -t mangle -A INPUT -p tcp --dport 80 -d 130.203.32.56 -j SECMARK --selctx system_u:object_r:http_server_packet_t:s0
iptables -t mangle -A OUTPUT -p tcp --sport 80 -s 130.203.32.56 -j SECMARK --selctx system_u:object_r:http_server_packet_t:s0
iptables -t mangle -A INPUT -p tcp --sport 3306 -s 130.203.32.61 -d 130.203.32.56 -j SECMARK --selctx
system_u:object_r:mysqld_client_packet_t:s0
iptables -t mangle -A OUTPUT -p tcp --dport 3306 -s 130.203.32.56 -d 130.203.32.61 -j SECMARK --selctx
system_u:object_r:mysqld_client_packet_t:s0
```

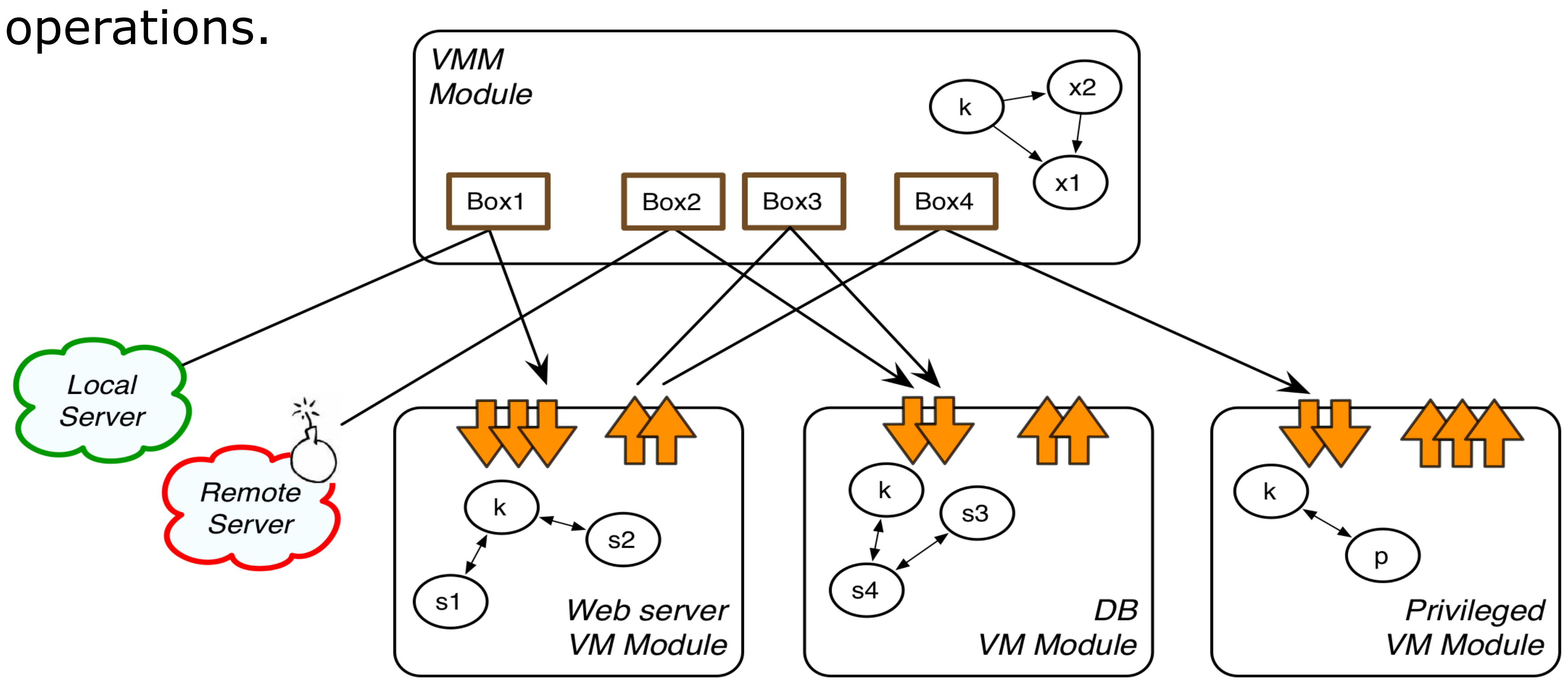
Approach

We automatically look for ATTACK RISKS: processes in attack paths that enable attackers to expand the set of processes under their control. To detect attack risks, we create an information flow graph of the system, trace information flows, and highlight processes that receive data with different integrity requirements.

- ➢ Compute information flow graph:
  - • VM MAC policy ➔ information flows inside VMs
  - • VMM and network ➔ information flows across VMs
- ➢ Assign integrity levels
  - • Manual assignment for the inputs
  - • Predefined rules for some processes
- ➢ Intransitive flows:
  - • some processes are expected to enforce access control for others
- ➢ Compute reachability and look for processes that receive multiple integrity levels.

Model

- ➢ Hierarchical State Machine (HSM) model:
  - • Informally, it is a collection of modules (each one describing a system), the instances of those modules, and their relationships.
  - • We use it to represent the information flow graph that models the composition of multiple independent MAC policies.
- ➢ Policy Summary:
  - • It succinctly represents the behavior of a policy, and improves performance of some operations.

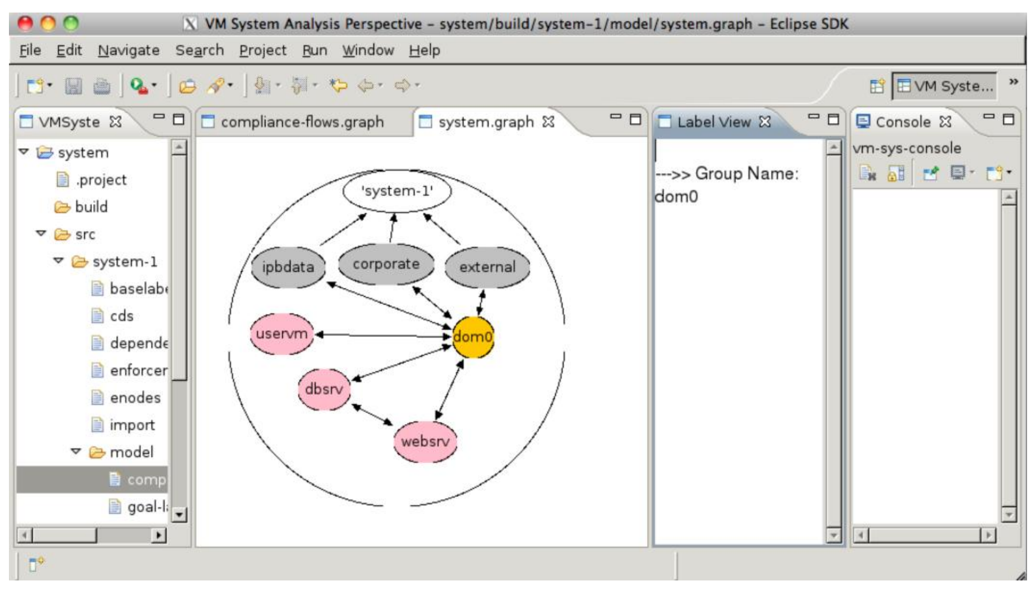


Refinement

- ➢ Processes that receive multiple integrity levels are attack risks. If we confine these processes, we cut the attacker's path.
  - • Some processes are designed to handle data with multiple integrity levels.
  - • If a process cannot handle data with different integrity levels, we recursively check if its children can.
- ➢ We aim to assess administrators in the configuration of a system where all attack risks are identified.

Analysis Tool

- ➢ The interface is an Eclipse plug-in that enables administrators to define VM systems, and load the corresponding MAC policies.
- ➢ The backend programs are C and Prolog. Our current implementation handles XSM/Flask MAC policies (VMM), SELinux (VM), and iptables with the SECMARK extension (network). We expect to program additional parsers to support other MAC policies.

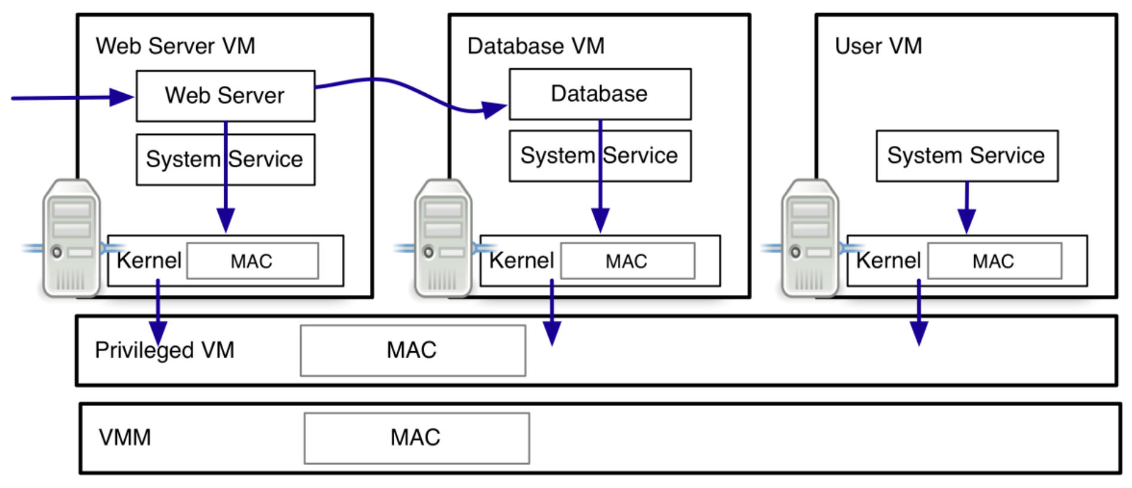


Results

- ➢ Since we consider network policy, administrators do not need to manually define an attack surface for every VM.
- ➢ The number of attack risks that we identify, based on OS and network MAC policies, is less than the number of nodes to consider in an attack graph.
- ➢ Results depend on access control policies. A permissive policy will yield a higher number of attack risks.

➢ *Managing Attack Risks in Virtual Machine Systems. S. Rueda, H. Vijayakumar, D. Muthukumaran, J. Schiffman, T. Jaeger, and S. Chauduri. Technical Report NAS-TR-0137-2010, Networking and Security Research Center, Department of Computer Science and Engineering, The Pennsylvania State University. July 2010.*

- ➢ Case Study: dom0, database server, web server, and user VMs. The web server receives requests that may require DB processing. The user VM only runs local applications.



VM	Policy Size (lines + network ports)	Processing time (secs)	Reachability evaluation (secs)	Network Facing Daemons	Attack Risks
dom0	283993 + 6	6.8	0.24	70	3
websrv	275738 + 9	6.3	0.23	60	34
dbsrv	256190 + 7	5.5	0.19	60	3
uservm	234316 + 4	5.2	0.17	60	0

Sponsored by AFRL