A Secure Programming Paradigm for Internet Virtualization

Sonia Fahmy

Department of Computer Science
Purdue University

http://www.cs.purdue.edu/~fahmy/

Joint work with Ana Milanova, David Musser, Bulent Yener (Rensselaer Polytechnic Institute)
Motivation

• DDoS attacks, malware, spam, ... are rapidly increasing
• Overlay and other end system solutions are insufficient and inefficient
• New applications require network support
• Active networks efforts failed due to security and scalability concerns
• Can we resolve these security and scalability concerns, and allow users to customize and virtualize the Internet?
Virtual Internet (VInet) Client Module at end user

Virtual Internet (VInet) Server Module at Edge Router: PCC or static analysis

Middleware
VInet Kernel & System Library

Proof Compiler
Proof(P)
Program(P)

Safety Policy
Proof(P)
Program(P)

Proof Checker

To network

From network

Dedicated port

BGP router

Edge Router

AS

users
Research Agenda

• Programmability
  – A library of packet manipulation routines
  – Language to compose routines
• Verification technology
  – Extend PCC with a safety policy (to check security and scalability)
  – Generate compact, composable, easy-to-check proofs
• Client-server protocols and scheduling
  – Securely transmit program, proof, payment
• Execution environment
  – Constrain programmability and use adaptive optimizations
• Emulation technology
  – Study performance, security, scalability limits, reliability, placement, partial deployment, under realistic scenarios
Packet Manipulation Routines

- DISCARD(Packets)
- GENERATE(SrcIP,DestIP,Content)
- GROUP_BY_DESTINATION(Packets,DestIP): Packets
- GROUP_BY_DESTINATION(Packets): Packets(i), DestIP(i)
- GROUP_BY_SOURCE(Packets,SrcIP): Packets
- GROUP_BY_SOURCE(Packets): Packets(i), SrcIP(i)
- GROUP_BY_CONTENT(Packets): DestIPs, Content
- CONTAINS_STRING(Packets,String): Boolean
Example: Virtual Firewall

- LIFETIME = 2 days $\leftarrow f(\text{Complexity}, \text{lifetime}) \leq \text{Cost}^{up}$
- MyPackets = GROUP_BY_DESTINATION(AllPackets, THIS_IP);
- BadPackets = GROUP_BY_SOURCE(MyPackets, SrcIP);
- DISCARD(BadPackets);
Example: Virtual Spam Filter

- LIFETIME = 5 days
- MyPackets = GROUP_BY_DESTINATION(AllPackets, THIS_IP);
- SourceIP(i), Packets(i) = GROUP_BY_SOURCE(MyPackets);
- Foreach i do
  - If CONTAINS_STRING(Packets(i), Word1) and CONTAINS_STRING(Packets(i), Word2)
    - DISCARD(Packets(i));
Example: Multicast

- LIFETIME = 5 hours
- MyPackets = GROUP_BY_DESTINATION(AllPackets, THIS_SUBNET_IP);
- Broadcast = GROUP_BY_SOURCE(MyPackets, BroadcastIP);
- Destinations, Content = GROUP_BY_CONTENT(Broadcast);
- GENERATE(BroadcastIP, THIS_IP, Destinations);
- GENERATE(BroadcastIP, THIS_IP, Content);
- DISCARD(Broadcast);
Example Safety Policy

![Formal Requirements and Safety Policy for Packet Filtering Programs.](image-url)
Fig. 3. A model of a filtering program and the proof of the Safety Policy in Figure 2.
Dynamic Execution Environment

Program Space with Capacity

The sources hashtable

The destinations hashtables

Program classes and adaptive optimizations
Source: DETER USC-ISI team; DETER is based on U. of Utah Emulab
Related Work

- Active networks, e.g., ANTS, PLAN, …
  http://nms.lcs.mit.edu/activeware/
  - Need useful programmability, while balancing security and scalability
- Liquid software project http://www.cs.arizona.edu/liquid/
  - We restrict programs to compositions of well-defined routines, and reuse lemma library
- Model checking
  - We place less responsibility on servers and use a higher level programming language
- Overlay networks
  - We allow efficient, network-level, operations
- Emulation, e.g., Click modular routers
  - Does not address remote programmability, security, or scalability
Conclusions and Planned Work

- Virtualizing the Internet enables several exciting services and a better Internet
- Security and scalability constraints and a library of lemmas enable efficient and secure virtualization using PCC
- Planned research on programmability, verification technology, client-server protocols and scheduling, execution environment, and emulation technology