

NetSTAT

A Network-based Intrusion Detection Approach

ACSAC '98

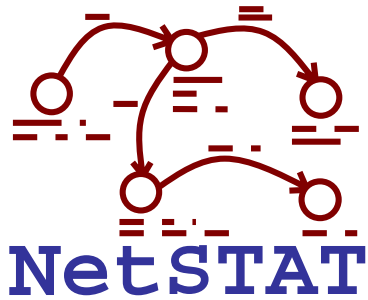
Giovanni Vigna and Richard A. Kemmerer

Reliable Software Group

University of California Santa Barbara

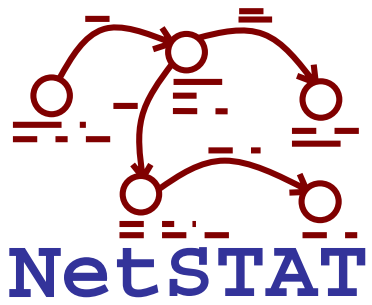
[vigna,kemm]@cs.ucsb.edu

<http://www.cs.ucsb.edu/~kemm/NetSTAT/>

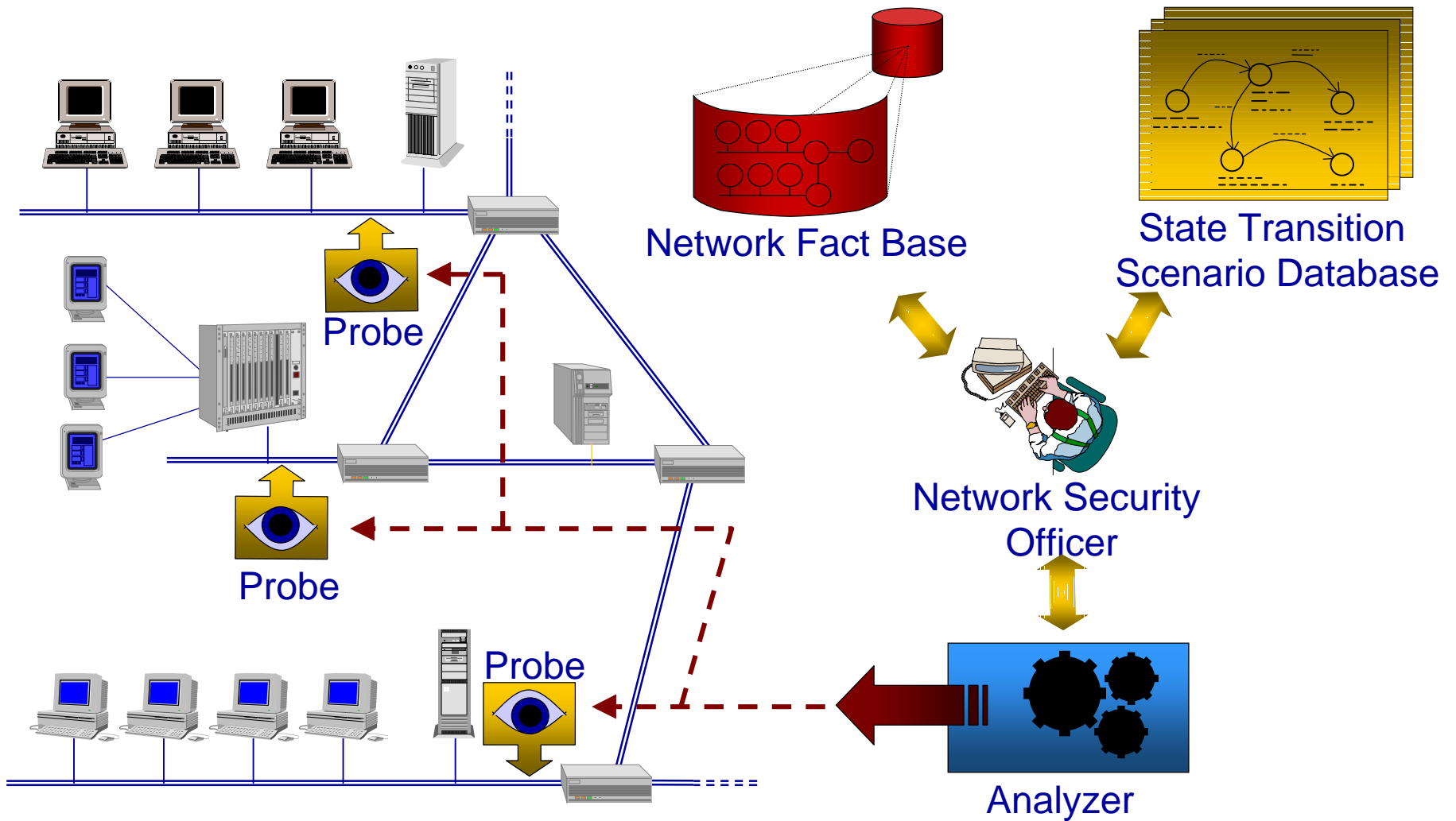


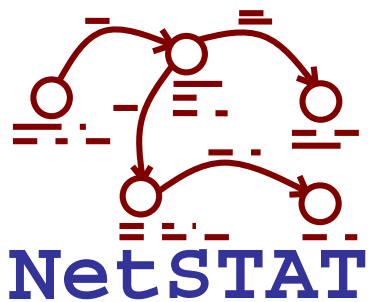
NetSTAT

- Network-based real-time intrusion detection system
- Extends the *State Transition Analysis Technique* to represent attacks in a networked environment
- Uses *Network Hypergraphs* to represent the network topology and services
- Customizes generic attack scenarios to a target network (*what* to look for and *where* to look for it)
- Features distributed architecture with decentralized processing of events



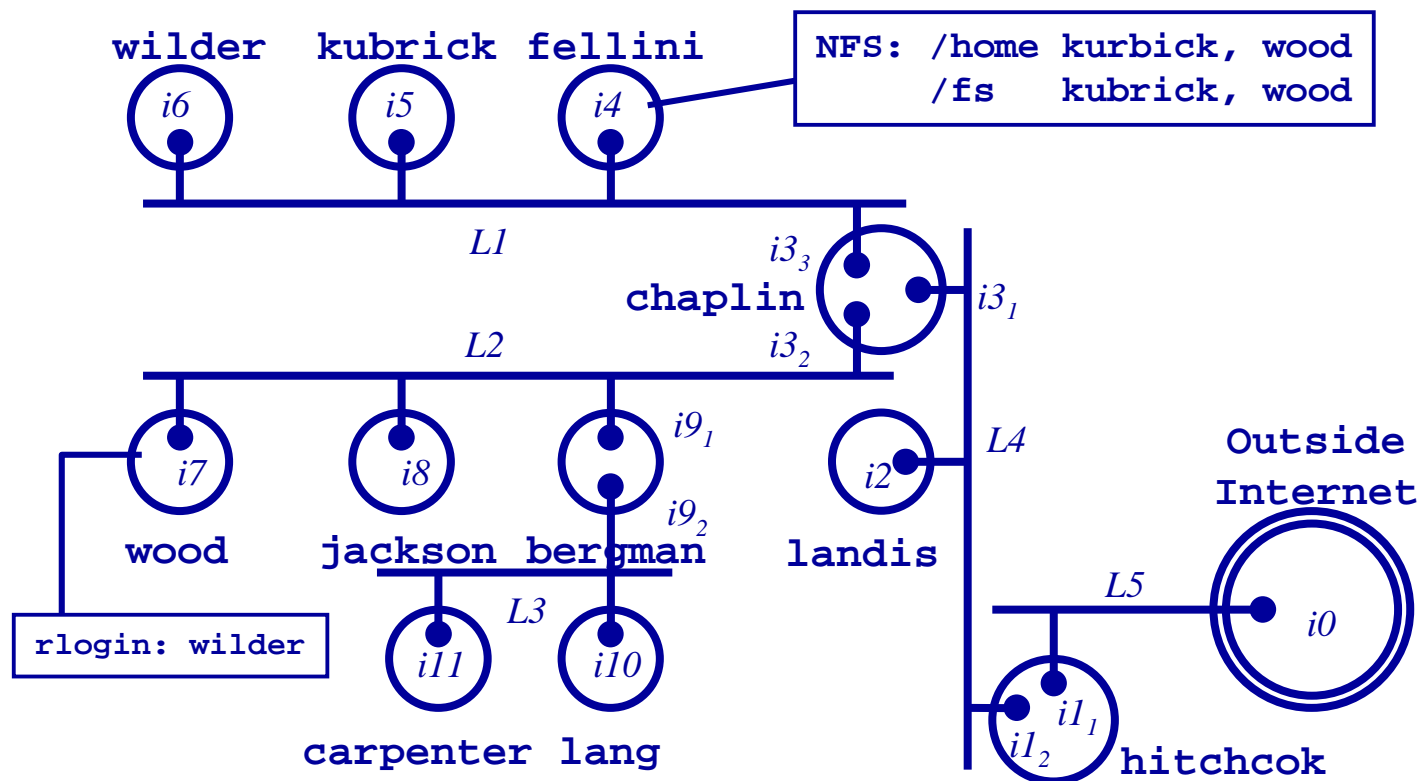
Overview

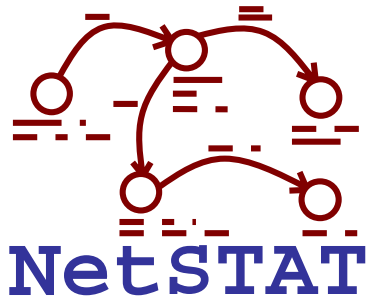




Network Fact Base

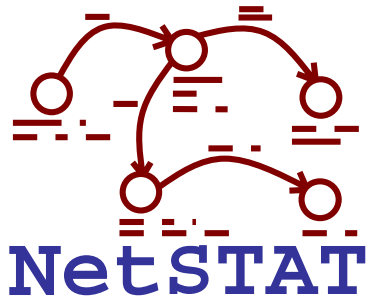
- Contains information about the network topology and the services deployed





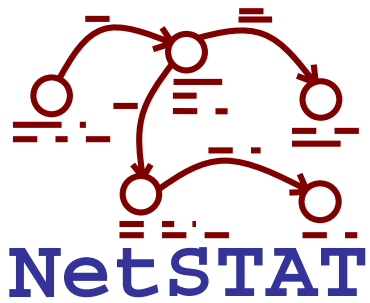
Advantages

- Provides well-defined semantics
- Supports reasoning and automation
- Models hosts and links in a uniform way (sets of interfaces)
- Allows natural modeling of shared-bus links

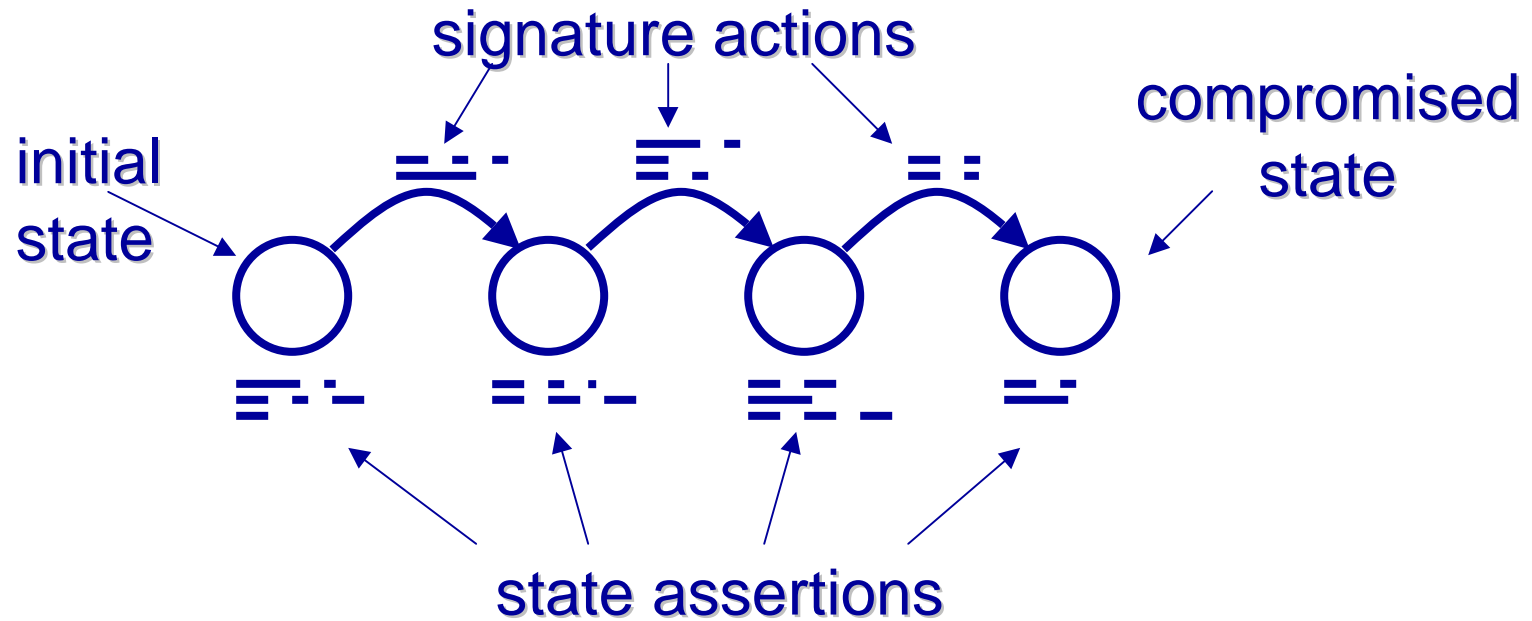


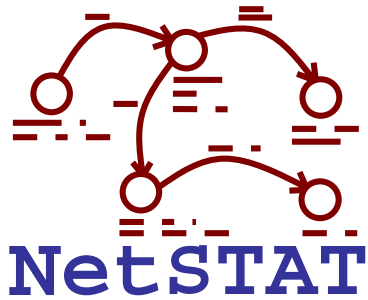
State Transition Scenario Database

- Manages the state/transition representations of the intrusion scenarios to be detected
- Representations based on the State Transition Analysis Technique (STAT)
- STAT born to represent intrusions in host-based IDSs (USTAT) and extended to distributed IDSs (NSTAT)
- Extended to represent intrusions in a networked environment
- State Transition Diagram: from a safe state to a compromised state through a series of signature actions



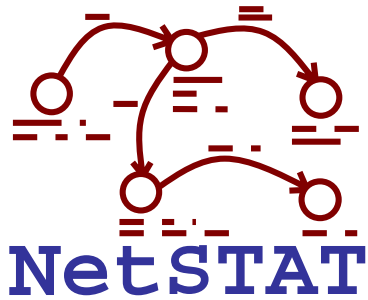
State Transition Diagrams





States and Assertions

- State of the network
 - Active connections (connection-oriented services)
 - State of interactions (connectionless services)
 - Tables
- Assertions
 - Static assertions
 - Dynamic assertions



Static and Dynamic Assertions

- *Static assertions*

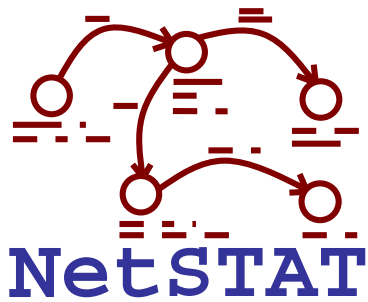
- Verified by examining the Network Fact Base
- Used to customize state transition representation for particular scenarios

```
Service s in server.services |  
    s.name == "www" and  
    s.application.name == "CERN httpd";
```

- *Dynamic assertions*

- Verified by examining the state of the network
- Used to determine what relevant network state events should be monitored

```
ConnectionEstablished(addr1, port1, addr2, port2)
```

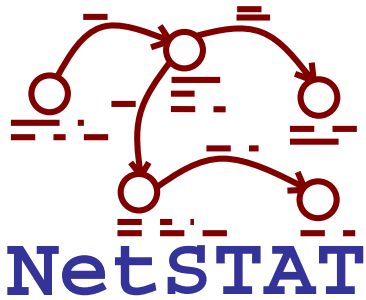


Signature Actions

- Signature actions represent critical changes in the security state of the system
- For NetSTAT signature actions leverage off of an *event model*
- Basic event: *link-level message*

```
Message m {i_x, i_y} |  
    m.length > 512;
```

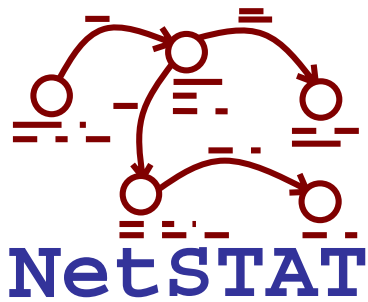
- Composite events:
 - IP datagrams (sequence of link-level message used for delivery)
 - UDP datagrams and TCP segments (encapsulated in IP datagrams)
 - Application-level events (encapsulated in UDP datagrams or TCP streams)



Examples

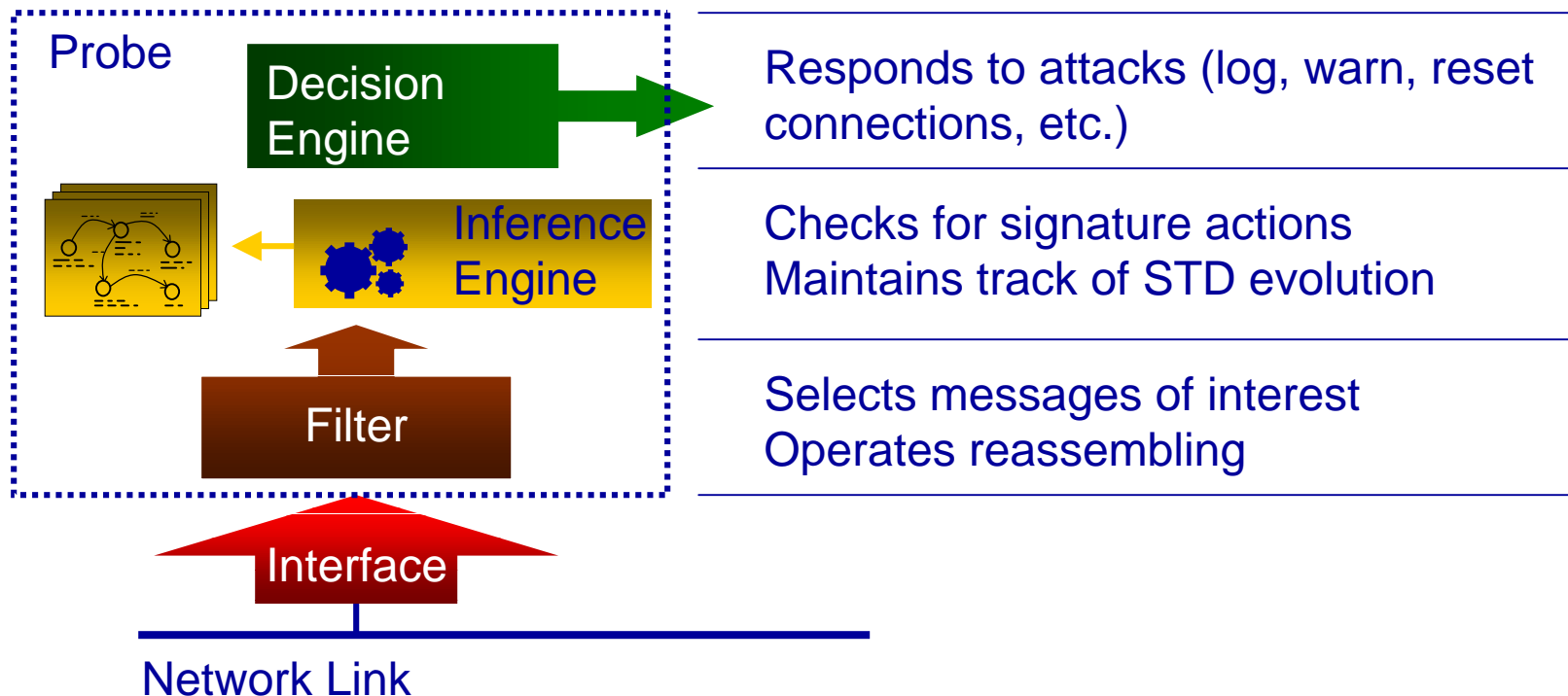
```
[IPDatagram d [UDPDatagram u [RPC r]]] {i_x, i_y} |  
  d.dst == a_y and  
  u.dst == 2049 and  
  r.type == CALL and  
  r.proc == MKDIR;
```

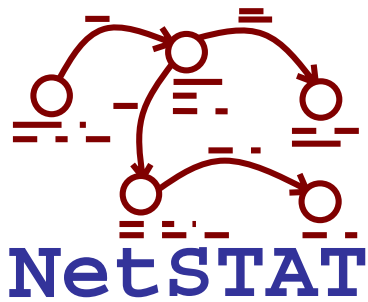
```
TCPSegment t in [VirtualCircuit c] {i_x, i_y} |  
  c.dstIP == a_y and  
  c.dstPort == 80 and  
  t.syn == true;
```



Probes

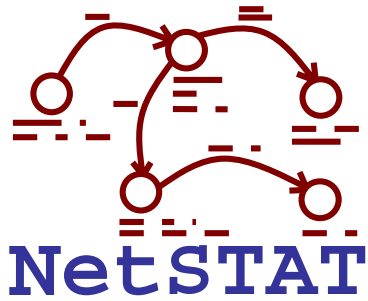
- Responsible for analyzing the stream of network events
- Configured and positioned by the Analyzer





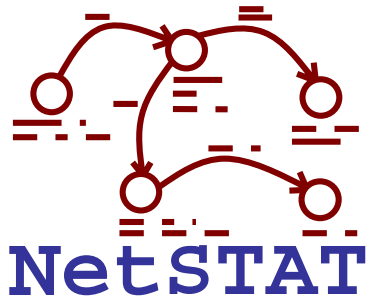
Analyzer

- Takes as input the Network Fact Base and the State Transition Scenario Database
- Determines:
 - Which events have to be monitored
 - Where the events have to be monitored
 - What information about the topology of the network is required to perform detection
 - What information about the state of the network must be maintained to be able to verify state assertions
- Produces a set of *probe configurations* and a *deployment plan*



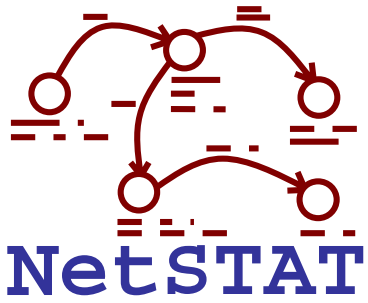
Configuration and Deployment

- NSO builds a database of attack scenarios using the State Transition Scenario Database
- NSO builds a network description using the Network Fact Base
- NSO selects a set of scenarios to be detected on the network
- The Analyzer performs the analysis and customization process
 - Mostly automated
 - May require help from the NSO in specific situations
- Configuration files are sent to probes

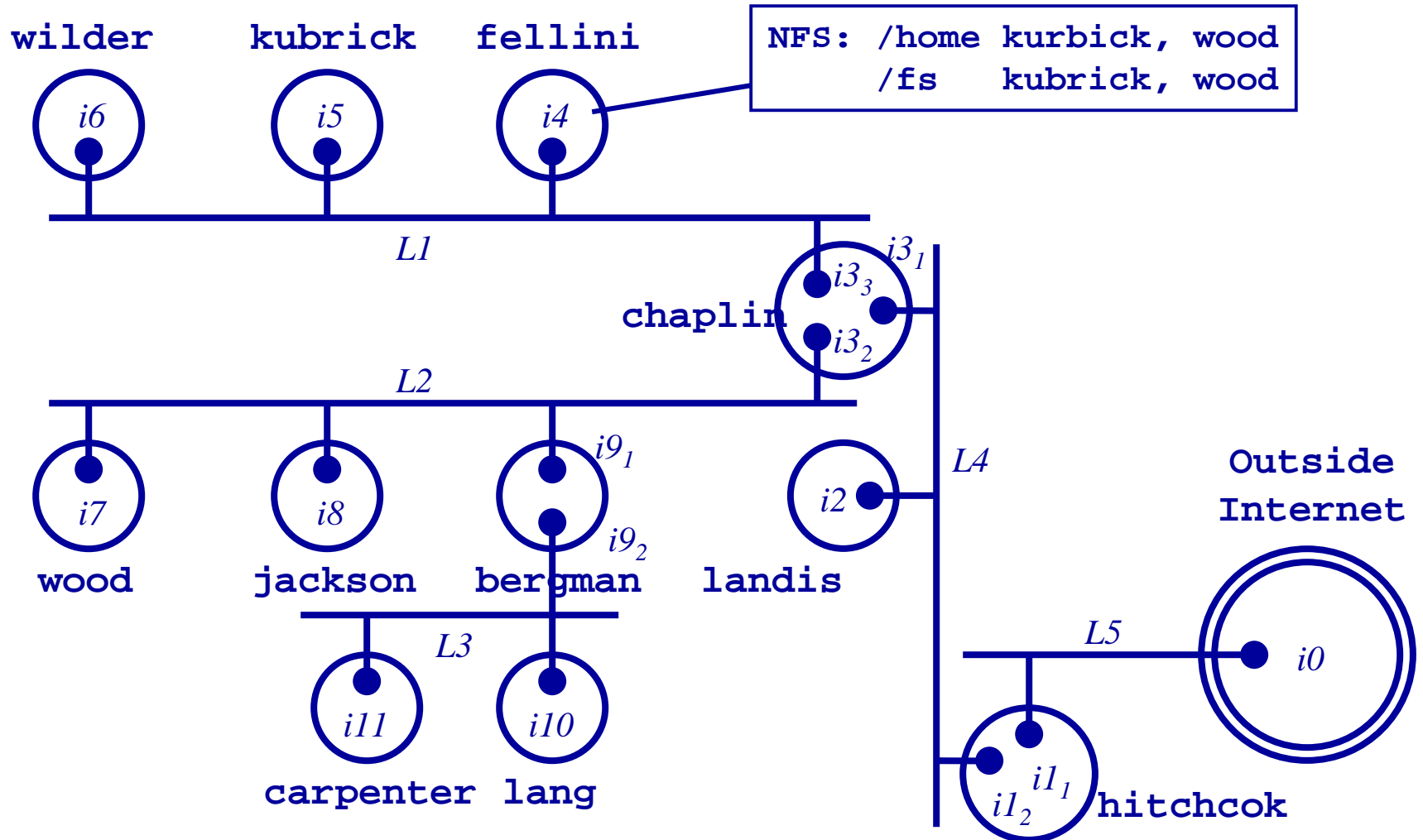


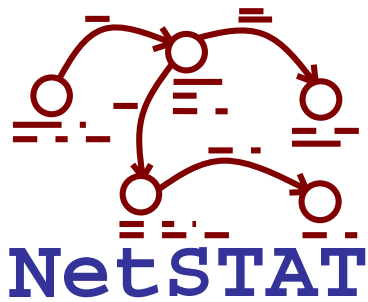
Example: UDP spoofing

- Service authentication based on source IP address
- An attacker tries to access a UDP-based service by pretending to be one of its trusted clients
- Attacker sends a forged UDP-over-IP datagram
- Attack detectable in particular topologies only



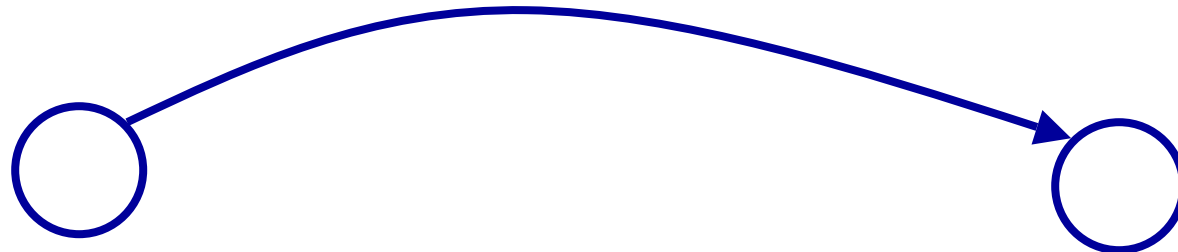
Sample Network





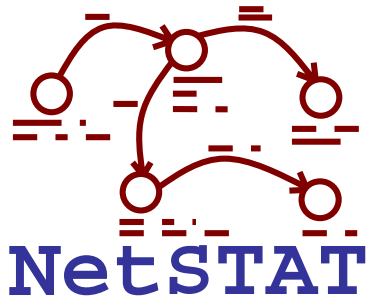
Attack Representation

```
Message m in [IPDatagram d [UDPDatagram u]] {i, a_v.interface}|  
  d.src == a_t and  
  d.dst == a_v and  
  u.dst == s.port and  
  not (Network.detachFromLink(m.src)).  
    existsPath(m.src, d.src.interface);
```



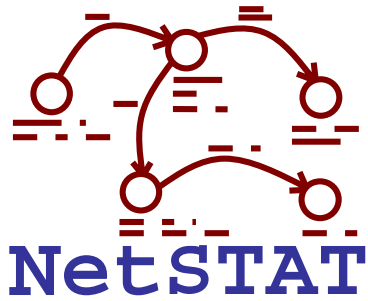
```
Host victim in ProtectedNetwork.hosts;  
Service s in victim.services |  
  s.protocol == "UDP" and  
  s.authentication == "IPaddress";  
IPAddress a_v in s.addresses;  
IPAddress a_t in s.trustedAddr;  
Host attacker in Network.hosts |  
  attacker != victim and  
  not attacker.Ipaddresses.contains(a_t);  
Interface i in attacker.interfaces;
```

Compromised



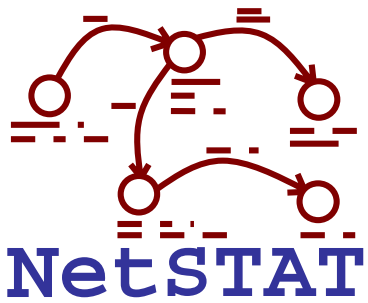
Customization to Target Network

victim	s	a_v	a_t	attacker	i
fellini	NFS	a4	a5	Outside	i0
fellini	NFS	a4	a7	Outside	i0
fellini	NFS	a4	a5	hitchcock	il ₁
fellini	NFS	a4	a7	hitchcock	il ₁
...
fellini	NFS	a4	a5	lang	i10
fellini	NFS	a4	a7	lang	i10
fellini	NFS	a4	a5	carpenter	il1
fellini	NFS	a4	a7	carpenter	il1

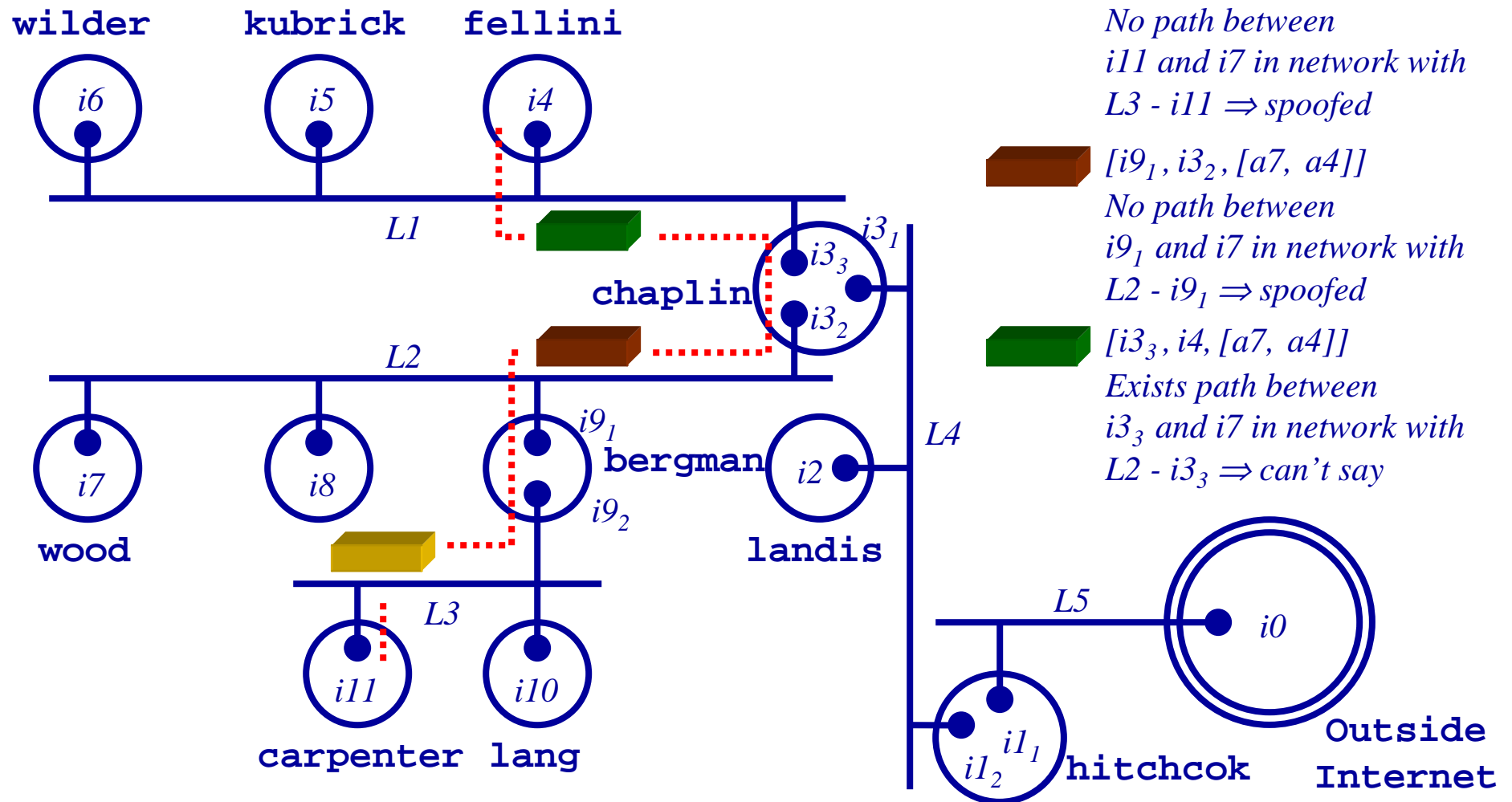


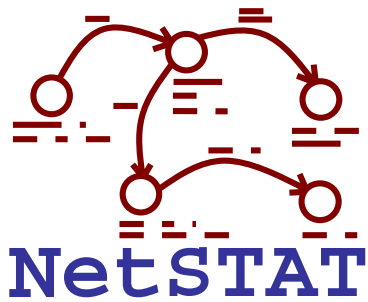
Probe placement and configuration

- For each scenario, the analyzer simulates the link-level messages that would be used to execute the attack
- The messages are then matched against the predicates contained in the attack's signature action
- The messages that satisfy the predicate can be used as a basis for attack detection
- One possible probe placement is chosen



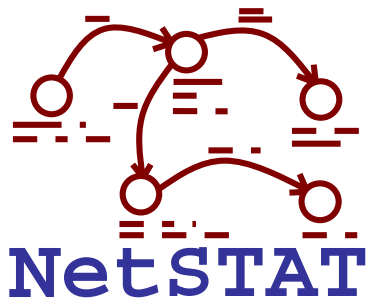
Example





Conclusions

- Focused and efficient real-time network-based intrusion detection in complex topologies
- Attacks represented as STDs
- Networks represented as hypergraphs
- Detection mechanisms highly tailored to target networks
- Automated tool support for the Network Security Officer
- Distributed, modular architecture with local processing
- Scalable and interoperable



Status

- Generic probe module completed
- Automatic deployment mechanisms for dynamic probe configuration developed
- More than 30 attacks analyzed and corresponding detection plug-ins developed
- Recently successfully completed participation to LL/AFRL IDSs evaluation
- People involved:
 - Richard Kemmerer, Steve Eckmann, Giovanni Vigna