iCAST / TRUST Collaboration
Year 2 - Kickoff Meeting

Robin Sommer

International Computer Science Institute

robin@icsi.berkeley.edu
http://www.icir.org

ICSI
INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Projects Overview

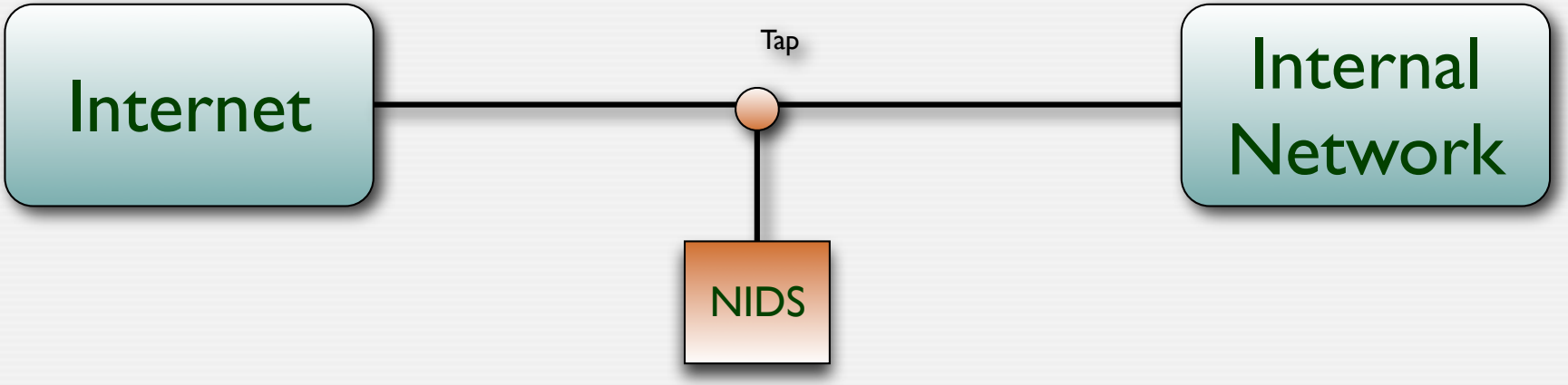
- **Project 1**
  NIDS Evasion Testing in a Live Context

- **Project 2**
  Efficient Archival & Retrieval of Network Activity

NIDS Evasion Testing in a Live Context

Network Intrusion Detection

- Network Intrusion Detection Systems (NIDS) monitor traffic passively for security violations



- NIDS reconstructs activity from network packets
- Needs to track the state of endpoints closely
  - May miss attacks if it gets desynchronized from endpoints
- However, there are fundamental limitations

Evasion Opportunities

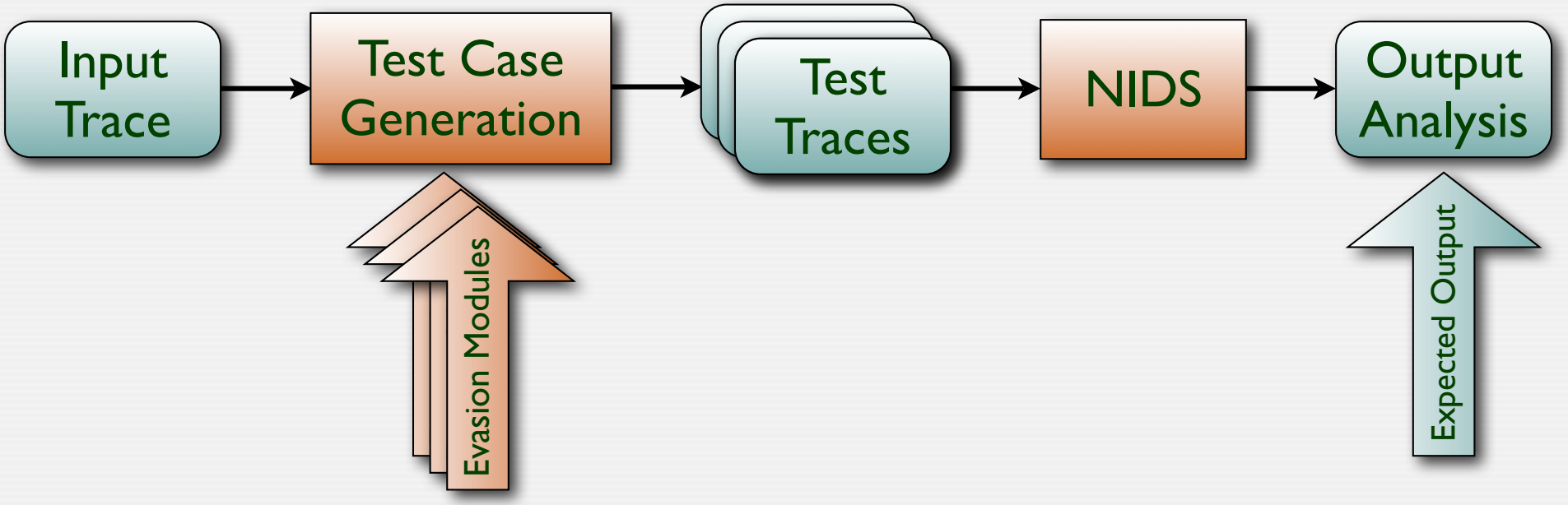
- NIDS may interpret data differently than endpoints
  - Protocol implementations often differ for corner-case semantics
  - RFCs hardly define how to respond to non-conforming traffic
    (*"Be liberal in what you accept, and conservative in what you send."*)
- Endpoints may actually see *different* data than NIDS
  - Packets may, e.g., be dropped before they reach their destination
- Even benign traffic contains tons of "crud"
- Attacker can craft traffic to exploit such ambiguities

Building an Evasion Test-suite

- Handling of ambiguities crucial for accuracy
  - No perfect solution but many cases can be mitigated or at least detected
- To the user, a NIDS is often just a black-box
  - Hard to understand & *validate* the internal operation
  - Typically its unclear how the NIDS handles ambiguities
- We built a framework to test NIDS for their evasion resilience in our year one effort.
  - While commercial labs perform such tests, no open platform existed so far

The *idsprobe* Framework

- *idsprobe is* as a modular & extensible framework to
  1. Systematically generate test-cases
  2. Evaluate a NIDS



Works well, but is restricted to *offline* analysis.

Year 2: Testing in a Live Context

- Goal of Year 2: Extend *idsprobe* to incorporate live traffic
- Not everything can be evaluated offline on traces:
  1. Attacks which depend on live conditions
     - Especially overload attack seeking to exhaust CPU or memory resources
  2. Active evasion counter-measures
     - Active Mapping, Normalization, Hybrid IDS

Example: Resource Exhaustion

- Algorithmic Complexity Attacks force worst-case behavior
- Crosby & Wallach presented hash collision attack
  - Usenix Security 2003
  - Attack weak hash functions by generate lots of collisions with low-volume input
  - Attacked Perl, Squid, and the Bro NIDS
- Bro could be overloaded with the bandwidth of a modem
- More generally, packet floods can exhaust CPU budget
- Only detectable when running the NIDS live

Extending *idsprobe*

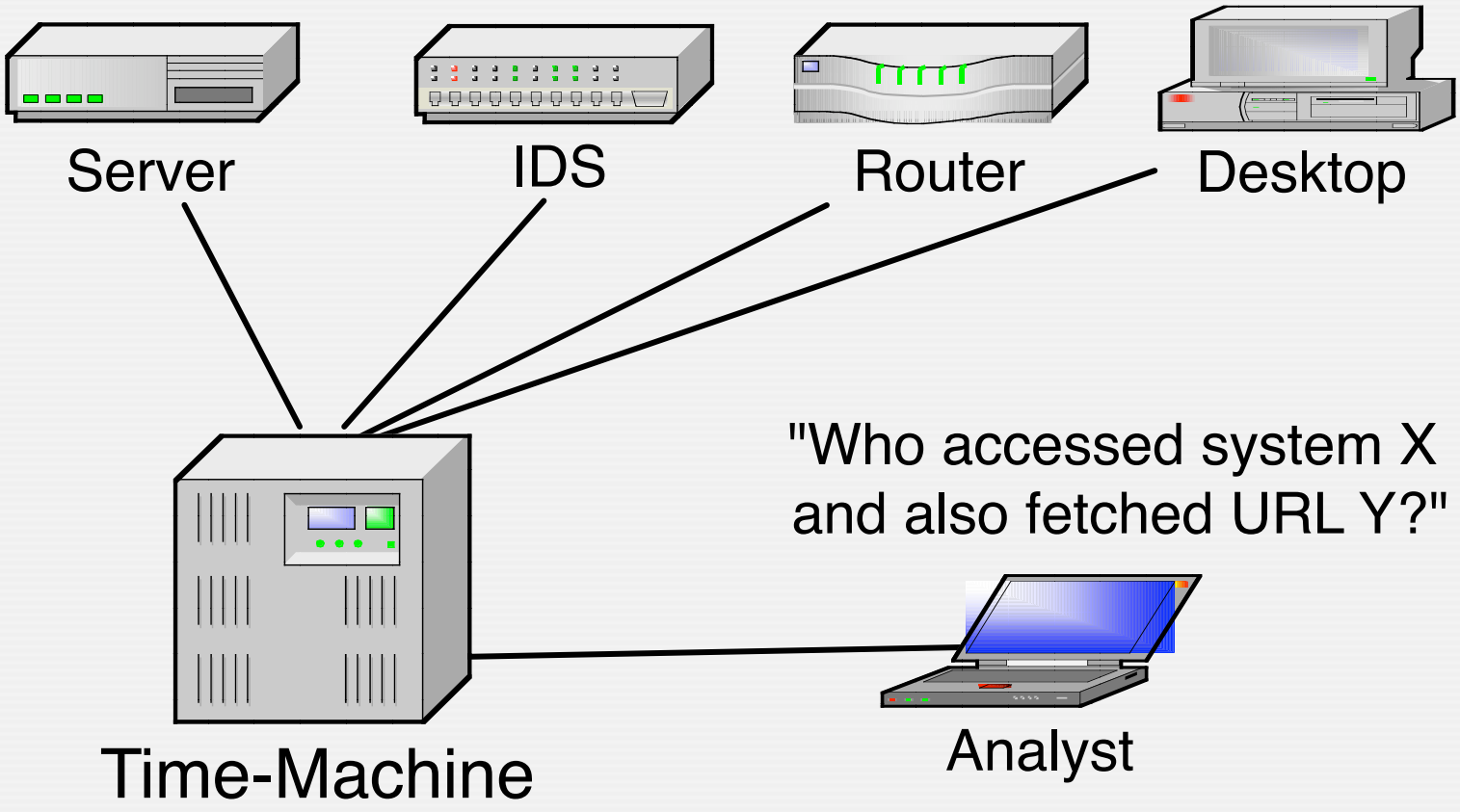
- Goal for Year 2: Extending *idsprobe* to handle live traffic
- This involves
  - Setting up a test-lab network (including live "victims")
  - Replaying traffic into the network
  - Ensuring consistency between runs of the test-suite
  - Automating the process
- Result will be a comprehensive online/offline framework

Efficient Archival & Retrieval of Network Activity

Motivation

- Network operators need to understand network's activity
  - For, e.g., troubleshooting and tracking security breaches
  - Typical information sources are firewalls, IDS, NetFlow, logging daemons, etc.
- Requires analysis in both time and space
  - Temporal: Understanding past & future activity
  - Spatial: Incorporating heterogeneous sources
  - Today, this is essentially a manual process
- Goal: Build a "*Time Machine*" which unifies these analyses
  - Integrates a multitude of different sources in a coherent way
  - Provides an interactive query interface for past & future activity
  - Operates in a policy-neutral fashion
  - Can coalesce, age, sanitize, and also reliably delete data

Application Scenario



Time Machine for Network Traffic

- We have already built a Time Machine for *network packets*
  - Efficient packet bulk-recorder for high-volume network streams
  - Taps into a network link and records packets in their entirety
  - Provides efficient query interface to retrieve past traffic
  - Proven to be extremely valuable for network forensics in operational use at LBL
- Heuristics for volume reduction
  - Cut-off: For each connection, TM stores only the first few KB
  - Expiration: Once available space is exhausted, TM expires oldest packets
- Simple yet very effective scheme
  - Even in large networks we can go back in time for several days

Generalizing the Time Machine

- We now aim to build a generalized Time Machine
  - Incorporates arbitrary network activity rather than just packets
  - Allows live queries for future activity
- The packet-based TM relies on three concepts
  1. A high-volume stream of input that we want to archive & query
  2. A rule how to separate more important input from less important input
  3. An aging mechanism to expire old information when storage fills up
- These concepts are independent of packets
- Need to find a suitable data model to network activity

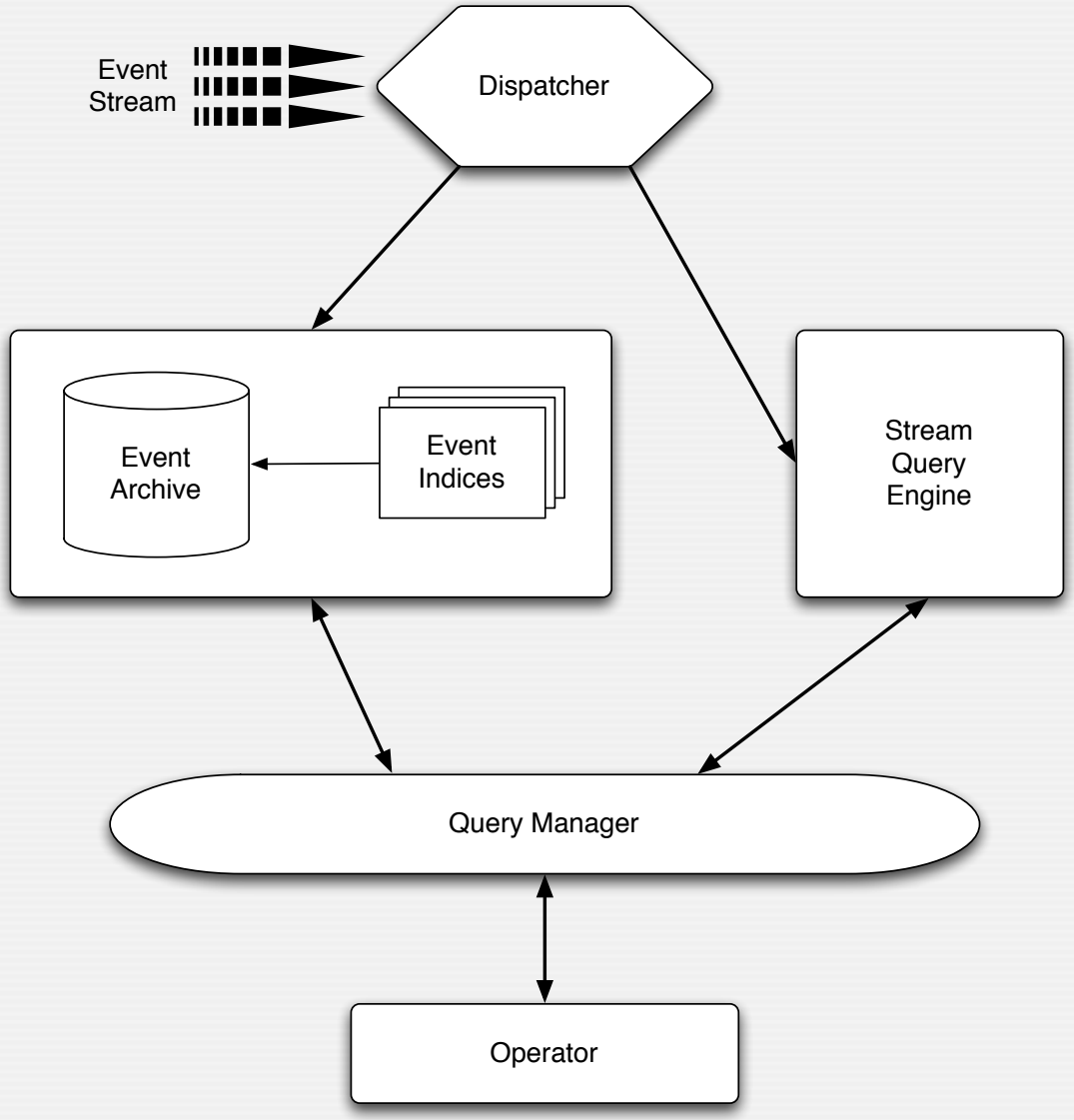
Event Model

- *Events* are well-suited to represent network activity
- Our Bro NIDS already relies on an event model
  - Abstracts network packets into high-level events, e.g., `connection_attempt`, `http_request`, `ssh_login`
  - Security policies are specified on top of these event
  - Events are policy-neutral (no notion "good" or "bad")
- Network can be instrumented to provide events
  - Network components can provide activity in form of event streams
  - We have already done that in the Bro context for, e.g., syslog, Apache, OpenSSH
  - Machinery for instrumentation exists (*libbroccoli*)
- Time Machine archives events instead of packets
  - Event archive provides a picture of the network's activity

Event Model (2)

- *Aggregation* gives a powerful aging scheme
- Instead of deleting old data, we can aggregate
  - From low-level & high-volume to high-level & low-volume
  - E.g., from NetFlow records to traffic matrices
- Aggregation schemes are event-specific
  - Time Machine will provide extensive hooks for flexibility
- Events also allow sanitizing data for archival
  - E.g., anonymization before storage

Time Machine Architecture



Project Tasks

- Instrumenting network components
  - Developing event specifications and aggregation schemes
  - Leveraging existing technology for implementation
  - Extending tools where necessary (e.g., scripting language interfaces)
- Building the Time Machine platform
  - Designing the user interface
    - Devising example queries to understand applications
  - Designing the storage backend for efficient archival & retrieval
    - Evaluating existing database backends
  - Designing the live query engine
    - Evaluating existing stream databases

Year 2 Projects

- **Project 1**
  NIDS Evasion Testing in a Live Context

- **Project 2**
  Efficient Archival & Retrieval of Network Activity

Any questions ...?

Robin Sommer
International Computer Science Institute

robin@icsi.berkeley.edu
http://www.icir.org