Network Intrusion Detection: Capabilities & Limitations

Vern Paxson

International Computer Science Institute
Lawrence Berkeley National Laboratory

vern@icsi.berkeley.edu

November 16, 2005

Outline

- What problem are we trying to solve?
- Why network intrusion detection? Why not?
- Styles of approaches.
- Architecture of a network intrusion detection system (NIDS).
- The fundamental problem of evasion.
- Detecting *activity*: scanners, stepping stones.

What Problem Are We Trying To Solve?

- A crucial basic question is *What is your threat model?*
  - What are you trying to protect?
  - Using what sort of resources?
  - Against what sort of adversary who has what sort of goals & capabilities?

- It's *all* about shades of grey, policy decisions, limited expenditure, risk management

Types of Threats

- In general, two types of threats: *insider* and *outsider*.

Types of Threats

- In general, two types of threats: *insider* and *outsider*.
- Insider threat:
  - Hard to detect $\Rightarrow$ hard to quantify
  - Can be *really* damaging
  - In many contexts, apparent prevalence: *rare*

Types of Threats

- In general, two types of threats: *insider* and *outsider*.
- Insider threat:
  - Hard to detect $\Rightarrow$ hard to quantify
  - Can be *really* damaging
  - In many contexts, apparent prevalence: *rare*
- Outsider threat:
  - Attacks from over the Internet: *ubiquitous*.
  - Internet sites are *incessantly* probed:
    - *Background radiation*: on average, Internet hosts are probed every 90 sec
  - Medium-size site: 10,000's of remote scanners each day.
    - What do they scan for? A wide and changing set of services/vulnerabilities, attacked via "auto-rooters" or *worms*.
  - Increasingly, not just "over the Internet":
    - Laptops, home machines erode notion of "perimeter"

What Are They After?

- Short answer: Not Us.
  - Most attacks are not targeted.
- They seek bragging rights:
  - E.g., via IRC or Web page defacement
- They seek *zombies* for:
  - DDOS slaves
  - Spamming
  - Bots-for-sale
  - Finding more targets
- They seek more of themselves (worms).
- Most don't cause damage beyond cleanup costs.
- But: this is *changing* with the *commercialization of malware*

What can you learn watching a network link?

- Far and away, most traffic travels across the Internet unencrypted.
- Communication is <u>layered</u> with higher layers corresponding to greater semantic content.
- The entire communication between two hosts can be reassembled: individual *packets* (e.g., TCP/IP headers), application *connections* (TCP byte streams), user *sessions* (Web surfing).
- You can do this in real-time.

Tapping links, con't:

- Appealing because it's *cheap* and gives broad coverage.
- You can have <u>multiple</u> boxes watching the same traffic.
- Generally (not always) undetectable.
- Can also provide insight into a site's general network use.

Problems with passive monitoring

- Reactive, not proactive
  - However, this is changing w/ intrusion *prevention* systems
- Assumes network-oriented (often “external”) threat model.
- For high-speed links, monitor may not keep up.
  - Accordingly, monitors often rely on filtering.
  - Very high speed: beyond state-of-the-art.
- Depending on “vantage point”, sometimes you see only one side of a conversation (especially inside backbone).
- Against a skilled opponent, there is a fundamental problem of evasion: confusing / manipulating the monitor.

Styles of intrusion detection — *Signature-based*

- Core idea: look for specific, known attacks.
- Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139
  flow:to_server,established
content:"|eb2f 5feb 4a5e 89fb 893e 89f2|"
msg:"EXPLOIT x86 linux samba overflow"
reference:bugtraq,1816
reference:cve,CVE-1999-0811
classtype:attempted-admin
```

Signature-based, con’t:

- Can be at different semantic layers, e.g.: IP/TCP header fields; packet payload; URLs.
- Pro: good attack libraries, easy to understand results.
- Con: unable to detect new attacks, or even just variants.

Styles of intrusion detection — *Anomaly-detection*

- Core idea: attacks are *peculiar*.

- Approach: build/infer a profile of “normal” use, flag deviations.
- Example: “user `joe` only logs in from host A, usually at night.”

- Note: works best for *narrowly-defined* entities
  - Though sometimes there’s a sweet spot, e.g., *content sifting* or *scan detection*

- Pro: potentially detects wide range of attacks, including novel.
- Con: potentially misses wide range of attacks, including known.
- Con: can potentially be “trained” to accept attacks as normal.

Styles of intrusion detection — *Specification*-based

- Core idea: codify a specification of what a site's *policy* permits; look for patterns of activity that deviate.
- Example: "user `joe` is *only* allowed to log in from host A."

- Pro: potentially detects wide range of attacks, including novel.
- Pro: framework can accommodate signatures, anomalies.
- Pro: directly supports implementing a site's policy.

- Con: policies/specifications require significant development & maintenance.
- Con: hard to construct attack libraries.

Some general considerations about the problem space

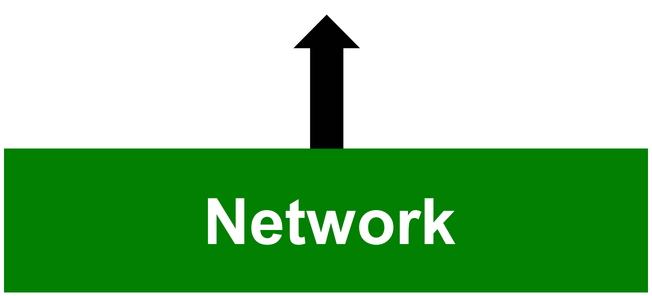
- Security is about *policy*.
- The goal is risk management, not bulletproof protection.

- All intrusion detection systems suffer from the twin problems of *false positives* and *false negatives*.
- These are not minor, but an Achilles heel.

- Scaling works against us: as the volume of monitored traffic grows, so does its diversity.

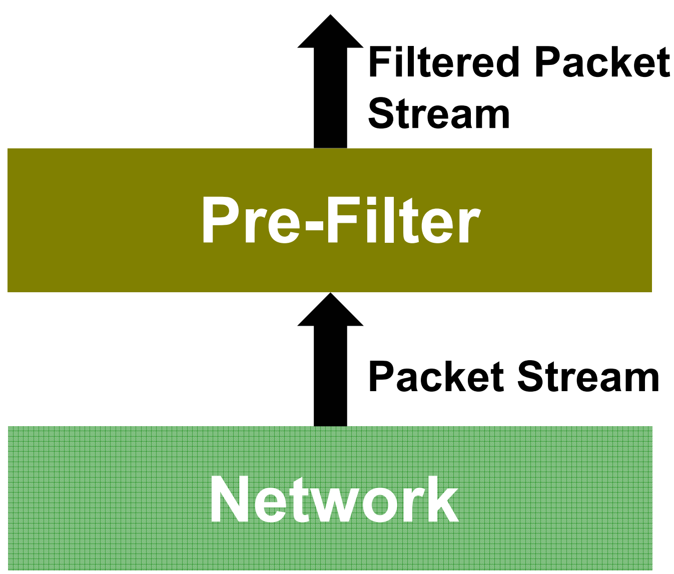
- Much of the state of the art is at the level of *car alarms*
  - Sure, for many attackers, particularly unskilled ones, they go off …
  - … but they also go off inadvertently a whole lot too

General NIDS Structure



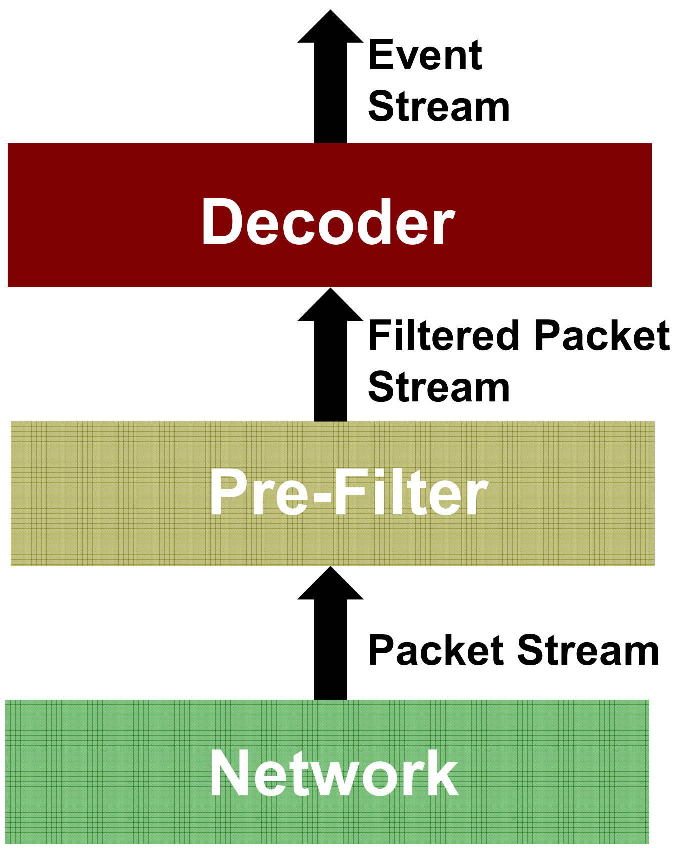
- Taps link passively, sends up a copy of all network traffic.

General NIDS Structure



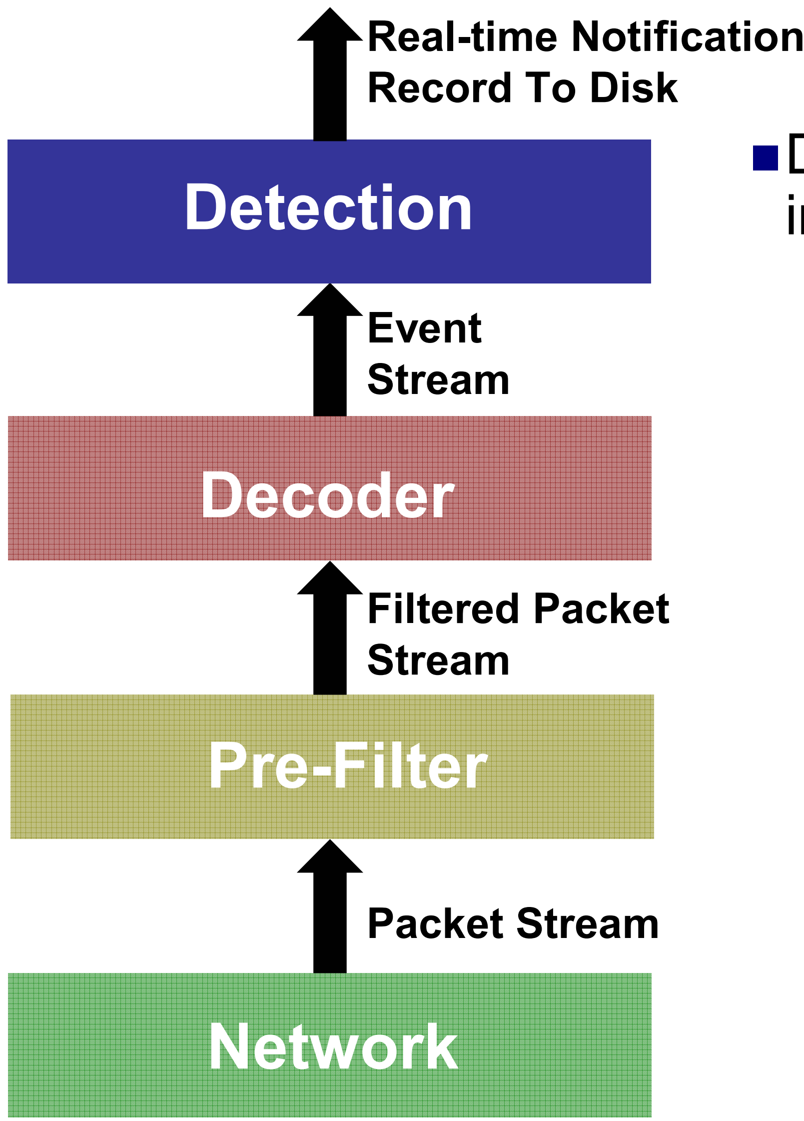
- Reduces high-volume stream via static filter to subset of main interest

General NIDS Structure



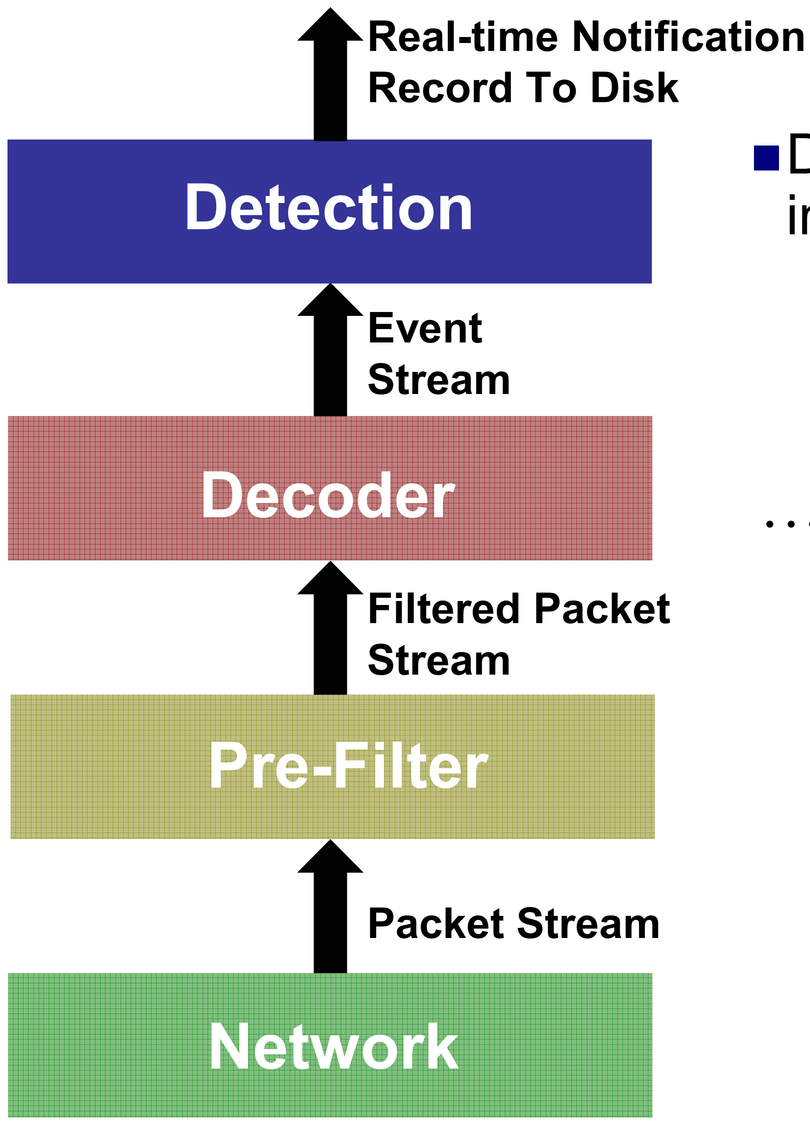
- Distills filtered stream into high-level, *policy-neutral* elements reflecting underlying network activity
  - E.g., connection attempt, Web request, user logged in

General NIDS Structure



- Detection logic processes event stream, incorporates:
  - Context from past analysis
  - Site's particular policies

General NIDS Structure



- Detection logic processes event stream, incorporates:
  - Context from past analysis
  - Site's particular policies

… and *takes action*:

Records forensic information to disk
Generates alarms
Executes response

A Stitch in Time: Prevention instead of Detection

- Big *win* to not just detect an attack, but **block it**
- However: *Big lose* to block **legitimate traffic**

- Mechanisms:
  - NIDS spoofs connection tear-down/denial messages
  - NIDS contacts firewall/router, requests block (race condition)
  - NIDS is *in-line* and itself drops offending traffic (no race, but performance and robustness issues)

- Increasing trend in industry …
- … but requires highly accurate algorithms

The Problem of *Evasion*

- Consider the following attack URL:

  http://…./c/winnt/system32/cmd.exe?/c+dir

- Easy enough to scan for (say, “cmd.exe”), right?

The Problem of *Evasion*

- Consider the following attack URL:

  http://…./c/winnt/system32/cmd.exe?/c+dir

- Easy enough to scan for (say, “cmd.exe”), right?

- But what about

  http://…./c/winnt/system32/cm%64.exe?/c+dir

The Problem of *Evasion*

- Consider the following attack URL:

  http://…./c/winnt/system32/cmd.exe?/c+dir

- Easy enough to scan for (say, “cmd.exe”), right?

- But what about

  http://…./c/winnt/system32/cm%64.exe?/c+dir

- Okay, we need to handle % escapes.

The Problem of *Evasion*

- Consider the following attack URL:

  http://…./c/winnt/system32/cmd.exe?/c+dir

- Easy enough to scan for (say, “cmd.exe”), right?

- But what about

  http://…./c/winnt/system32/cm%64.exe?/c+dir

- Okay, we need to handle % escapes.
- But what about

  http://…./c/winnt/system32/cm%25%54%52.exe?/c+dir

- Oops. Will recipient double-expand escapes … or not?

The Problem of *Evasion*, con't

- More generally, consider *passive measurement*: scanning traffic for a particular string ("`USER root`")

The Problem of *Evasion*, con’t

- More generally, consider *passive measurement*: scanning traffic for a particular string (“`USER root`”)
- Easiest: scan for the text in each packet
  - No good: text might be split across multiple packets

The Problem of *Evasion*, con't

- More generally, consider *passive measurement*: scanning traffic for a particular string ("`USER root`")
- Easiest: scan for the text in each packet
  - No good: text might be split across multiple packets
- Okay, remember text from previous packet
  - No good: out-of-order delivery

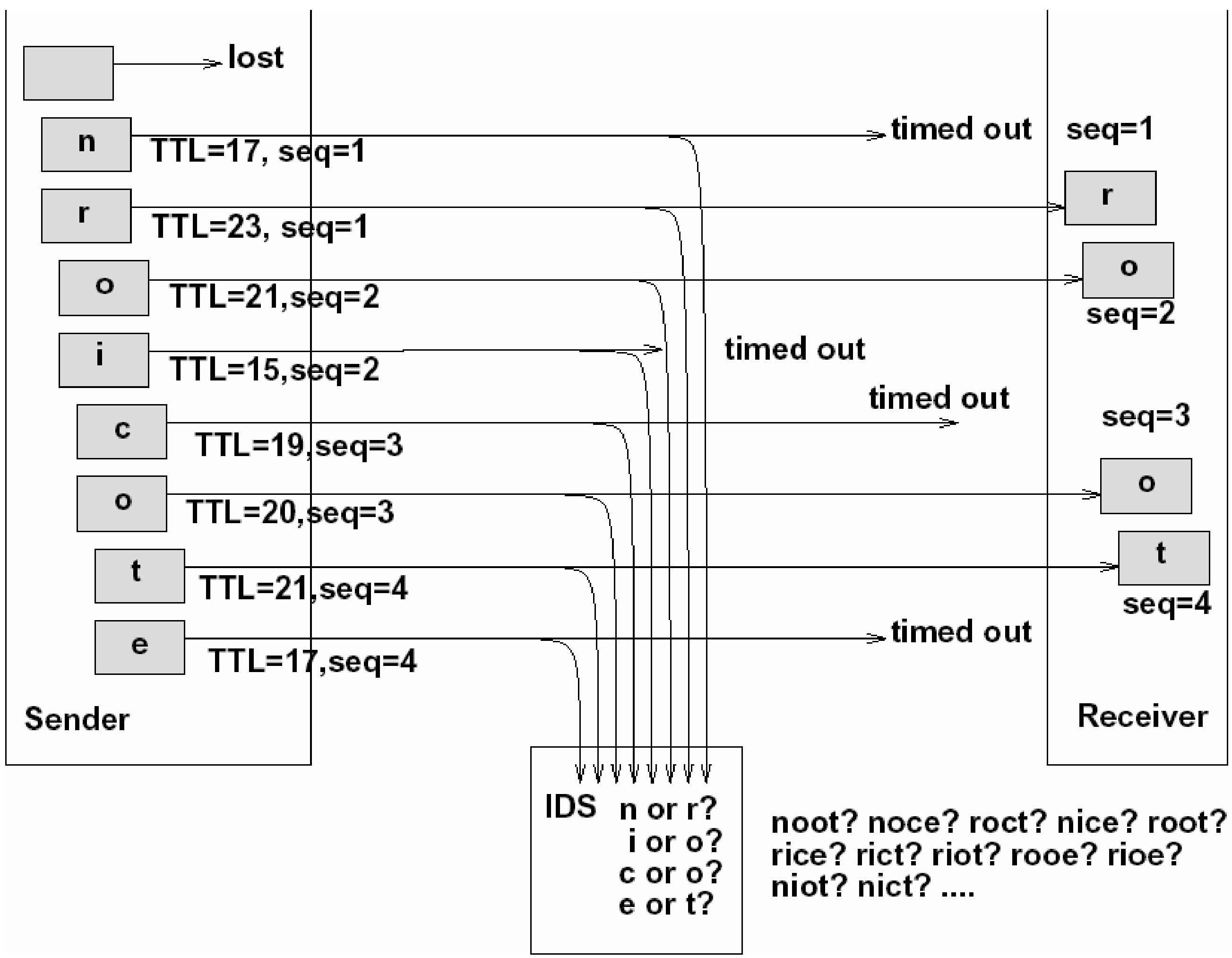
The Problem of *Evasion*, con't

- More generally, consider *passive measurement*: scanning traffic for a particular string ("`USER root`")
- Easiest: scan for the text in each packet
  - No good: text might be split across multiple packets
- Okay, remember text from previous packet
  - No good: out-of-order delivery
- Okay, fully reassemble byte stream
  - Costs **state** ….

The Problem of *Evasion*, con't

- More generally, consider *passive measurement*: scanning traffic for a particular string ("`USER root`")
- Easiest: scan for the text in each packet
  - No good: text might be split across multiple packets
- Okay, remember text from previous packet
  - No good: out-of-order delivery
- Okay, fully reassemble byte stream
  - Costs **state** ….
  - …. and *still* evadable

Evading Detection Via Ambiguous TCP Retransmission



The Problem of Evasion

- Fundamental problem passively measuring traffic on a link: Network traffic is *inherently* <u>ambiguous</u>
- Attackers can craft traffic to confuse/fool monitor
- Okay, can’t you then generate an alarm when you see ambiguous traffic?

The Problem of Crud

- Real network traffic, especially in high volume environments, inevitably has a steady stream of strange/broken traffic:
  - Storms of useless packets, due to implementation/protocol bugs.
  - Unroutable addresses leaking out.
  - Data split into overlapping pieces
  - Direct violations of protocol standards
  - Senders that acknowledge data that was never sent
  - Senders that retransmit different data than sent the first time
- E.g., LBL's *Bro* NIDS logged 76,610 "weird" events yesterday
  - Out of 13,261,129 connections analyzed
- Plenty of background noise in which an attacker can hide :-(

Countering Evasion-by-Ambiguity

- Involve end-host: have it *tell you* what it saw, or do the analysis itself
- Probe end-host in advance to resolve vantage-point ambiguities
  - E.g., how many hops to it?
  - E.g., how does it resolve double %-escapes?
- *Change the rules - perturb*
  - Introduce a network element that “normalizes” the traffic passing through it to eliminate ambiguities
  - Approach works for some ambiguities, but not all
- *Change the rules - become the recipient*
  - Run proxies for allowed services

Detecting activity — scanners

- How do you detect if someone is probing your site?
- How quickly can you do this?

- Classic approach: look for a source attempting to contact N different hosts in T seconds
  - Easy to evade if attacker knows (or can measure) N and T
  - Hard to set N and T to rapidly detect scanners but not misdetect hosts whose traffic fans out

Detecting activity — scanners, con't

- Idea: leverage the fundamental fact that the scanner *doesn't know what's there*
- Ergo: scanner's attempts are *more likely to fail*

- Posit two hypotheses:
  - H0: source is benign. Connection attempts fail with probability P0.
  - H1: source is a scanner. Attempts fail with probability P1 > P0.

- For each new connection attempt, use *sequential hypothesis testing* to see if either H0 or H1 is strongly more consistent with observations.

- Very effective at finding scanners after 4-5 attempts.
- Framework gives *bounds* on false positives & negatives.

Detecting activity — stepping stones

- Interactive attacks invariably do not come from the attacker's own personal machine, but from a *stepping-stone*: an intermediary previously compromised.

- Furthermore, usually it is a *chain* of stepping stones.
- Manually tracing attacker back across the chain is virtually impossible.

- So: want to detect that a connection going into a site is closely related to one going out of the site.

Detecting stepping stones

- Approach:
  - Leverage unique on/off pattern of user login sessions.
  - Look for connections that end <u>idle periods</u> at the same time.
  - Two idle periods correlated if ending time differ by $<= \Delta$sec.
- If enough idle periods coincide ⇨ stepping stone pair.
  - For A → B → C stepping stone, just 2 correlations suffices.
  - For A → B → . . . → C → D, 4 suffices.

Detecting stepping stones

- Works very well, *even for encrypted traffic*.
- But: easy to evade, if attacker cognizant of algorithm.
- And: also turns out there are frequent *legit* stepping stones.
  - This is a *broad theme* of detecting high-level activity. Often if you manage to do it, you find it indeed occurs, but for benign reasons.

Some Summary Points

- Security is not about bullet-proof; it's about *policies* and *tradeoffs*.
- You can detect a whole lot by piecing together judiciously filtered network traffic into events reflecting activity …
- … but this raises difficult issues with *managing state* …
- … and there are significant problems with evasion, potentially leading to an arms race.
- Traffic contains much more diversity/junk than you'd think, including *incessant scanning* for vulnerabilities.
- The endpoint host is a great location to look for attacks, if you can “manage” it.
- There is increasing pressure to bolster network intrusion detection with in-line forwarding elements that constrain/alter the traffic on behalf of the NIDS.