Intrusion Detection

Wenke Lee

Computer Science Department
Columbia University

Intrusion and Computer Security

- Computer security: confidentiality, integrity, and availability
- Intrusion: actions to compromise security
- Why are intrusions possible?
  - protocol and system design flaws
  - implementation (programming) errors
  - system administrative security “holes”
  - people (users) are naive

Design Flaws

- Security wasn't a "big deal"
  - ease of use (by users) and communications (among systems) more important
- Operating systems (next guest lecture)
- TCP/IP
  - minimal or non-existent authentication
    - relying IP source address for authentication
    - some routing protocols don't check received information

Example: IP Spoofing

- Forge a trusted host's IP address
- Normal 3-way handshake:
  - C-> S: SYN (ISNc)
  - S-> C: SYN (ISNs), ACK (ISNc)
  - C-> S: ACK (ISNs)
  - C-> S: data
  - and/or
  - S-> C: data

Example: IP Spoofing (cont'd)

- Suppose an intruder X can predict ISNs, it could impersonate trusted host T:
  - X-> S: SYN (ISNx), SRC=T
  - S-> T: SYN (ISNs), ACK (ISNx)
  - X-> S: ACK (ISNs), SRC=T
  - X-> S: SRC=T, nasty data
- First put T out of service (denial of service) so the S->T message is lost
- There are ways to predict ISNs

Implementation Errors

- Programmers are not educated with the security implications
- People do make mistakes
- Examples:
  - buffer overflow:
    - strcpy (buffer, nasty_string_larger_than_buffer)
  - overlapping IP fragments, “urgent” packets, etc.

System Holes

- Systems are not configured with clear security goals, or are not updated with "patches"
- The user-friendly factors: convenience is more important
  - e.g., "guest" account

4 Main Categories of Intrusions

- Denial-of-service (DOS)
  - flood a victim host/port so it can't function properly
- Probing
  - e.g. check out which hosts or ports are "open"
- Remote to local
  - illegally gaining local access, e.g., "guess passwd"
- Local to root
  - illegally gaining root access, e.g., "buffer overflow"

Intrusion Prevention Techniques

- Authentication (e.g. biometrics)
- Encryption
- Redesign with security features (e.g., IPSec)
- Avoid programming error (e.g., StackGuard, HeapGuard, etc.)
- Access control (e.g. firewall)
- Intrusion prevention alone is not sufficient!

Intrusion Detection: Overview

- Main Benefits:
  - security staff can take immediate actions:
    - e.g., shut down connections, gather legal evidence for prosecution, etc.
  - system staff can try to fix the security “holes”
- Primary assumptions:
  - system activities are observable (e.g., via tcpdump, BSM)
  - normal and intrusive activities have distinct evidence (in audit data)

Intrusion Detection: Overview (cont'd)

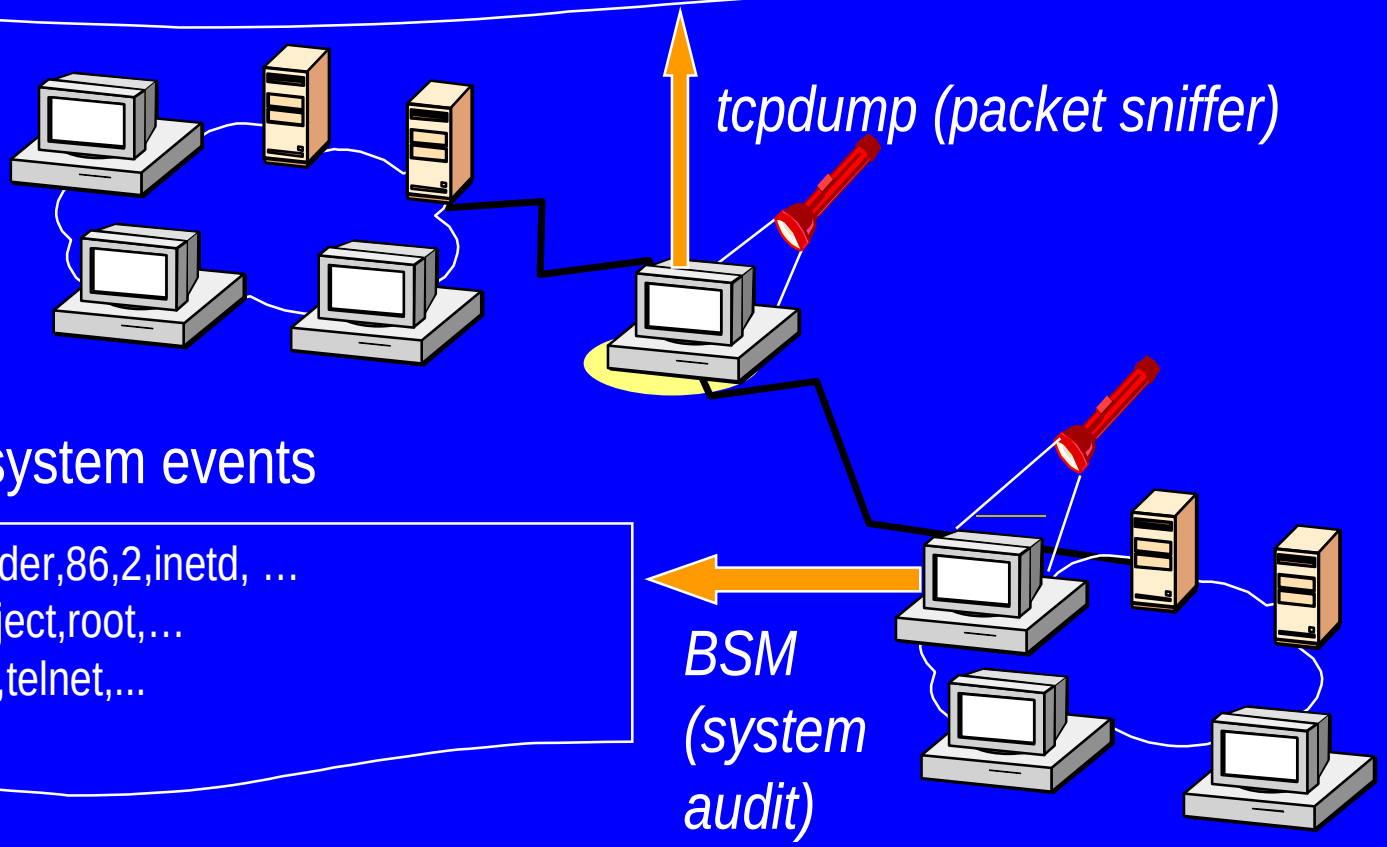
- Main Difficulties:
  - network systems are too complex
    - too many "weak links"
  - new intrusions methods are discovered continuously
    - attack programs are available on the Web

Intrusion Detection: Overview (cont'd)

- Issues:
  - Where?
    - gateway, host, etc.
  - How?
    - rules, statistical profiles, etc.
  - When?
    - real-time (per packet, per connection, etc.), or off-line

network traffic

10:35:41.5 128.59.23.34.30 > 113.22.14.65.80 : . 512:1024(512) ack 1 win 9216
10:35:41.5 102.20.57.15.20 > 128.59.12.49.3241: . ack 1073 win 16384
10:35:41.6 128.59.25.14.2623 > 115.35.32.89.21: . ack 2650 win 16225



system events

header,86,2,inetd, …
subject,root,…
text,telnet,...
...

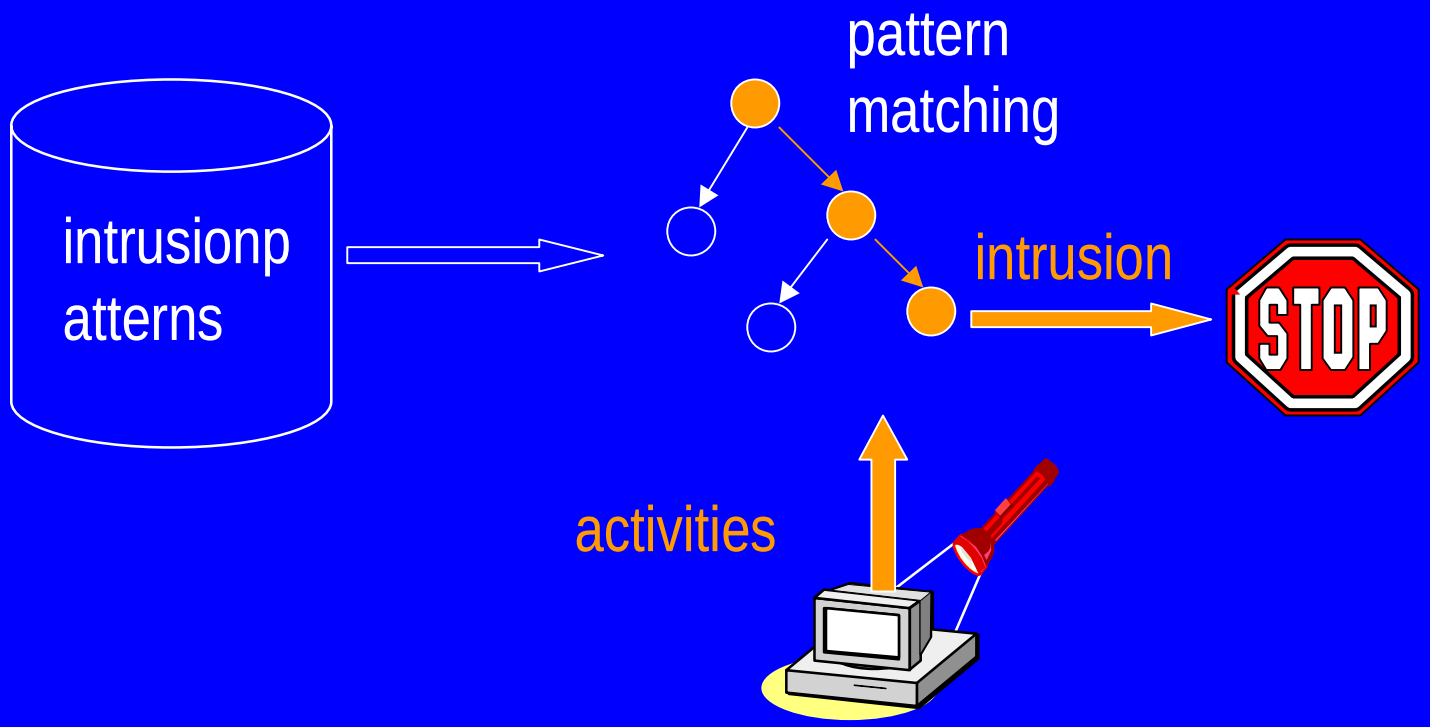
Audit Data

- Ordered by timestamps
- Network traffic data, e.g., tcpdump
  - header information (protocols, hosts, etc.)
  - data portion (conversational contents)
- Operating system events, e.g. BSM
  - system call level data of each session (e.g., telnet, ftp, etc.)

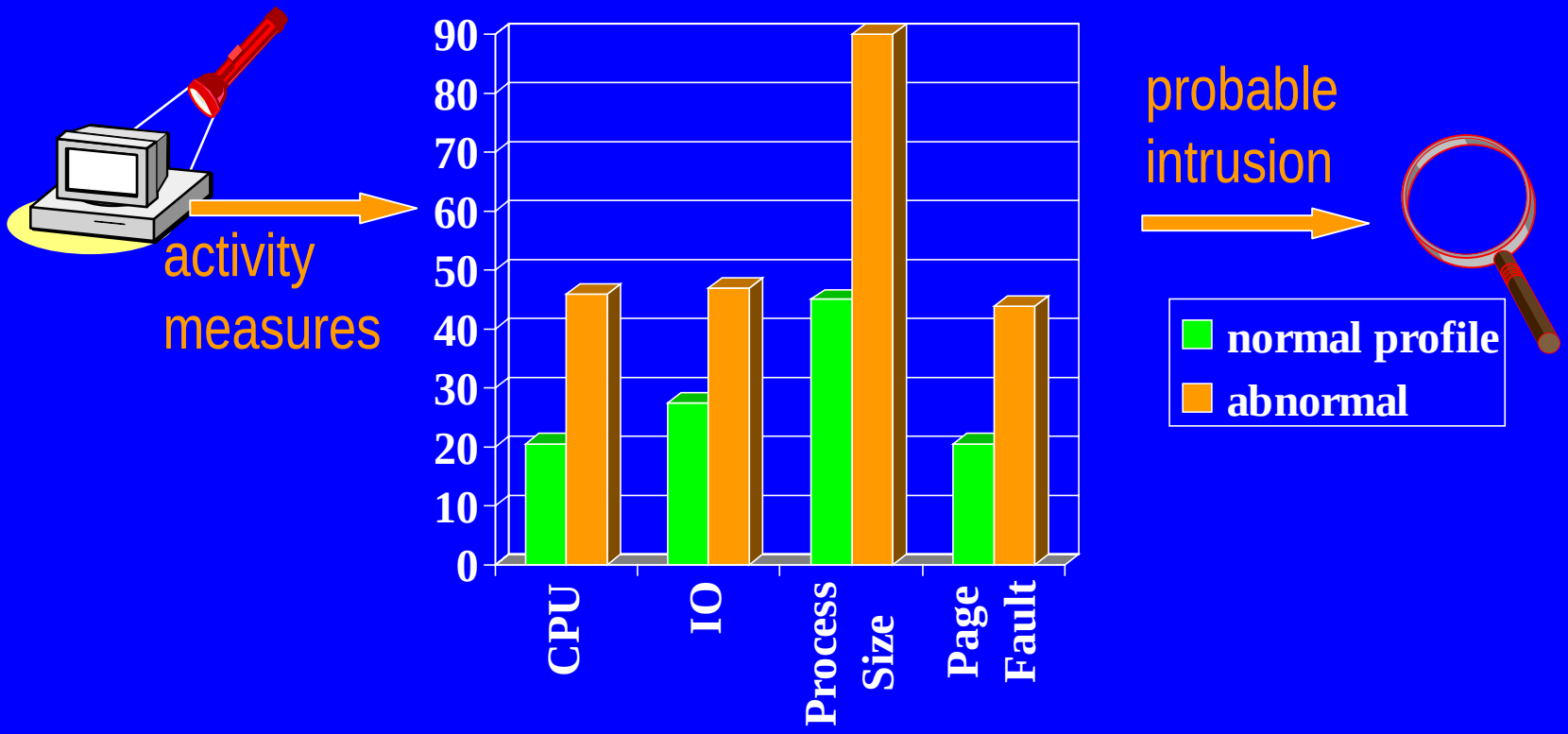
Intrusion Detection Techniques

- Many IDSs use both:
  - Misuse detection:
    - use patterns of well-known attacks or system vulnerabilities to detect intrusions
    - can't detect "new" intrusions (no matched patterns)
  - Anomaly detection:
    - use "significant" deviation from normal usage profiles to detect "abnormal" situations (probable intrusions)
    - can't tell the nature of the anomalies

Misuse Detection



Anomaly Detection



Current Intrusion Detection Systems (IDSs)

- "Security scanners" are not
- Naïve Keyword matching
  - e.g. no packet filtering, reassembling, and keystroke editing
- Some are up-to-date with latest attack "knowledge-base"

Requirements for an IDS

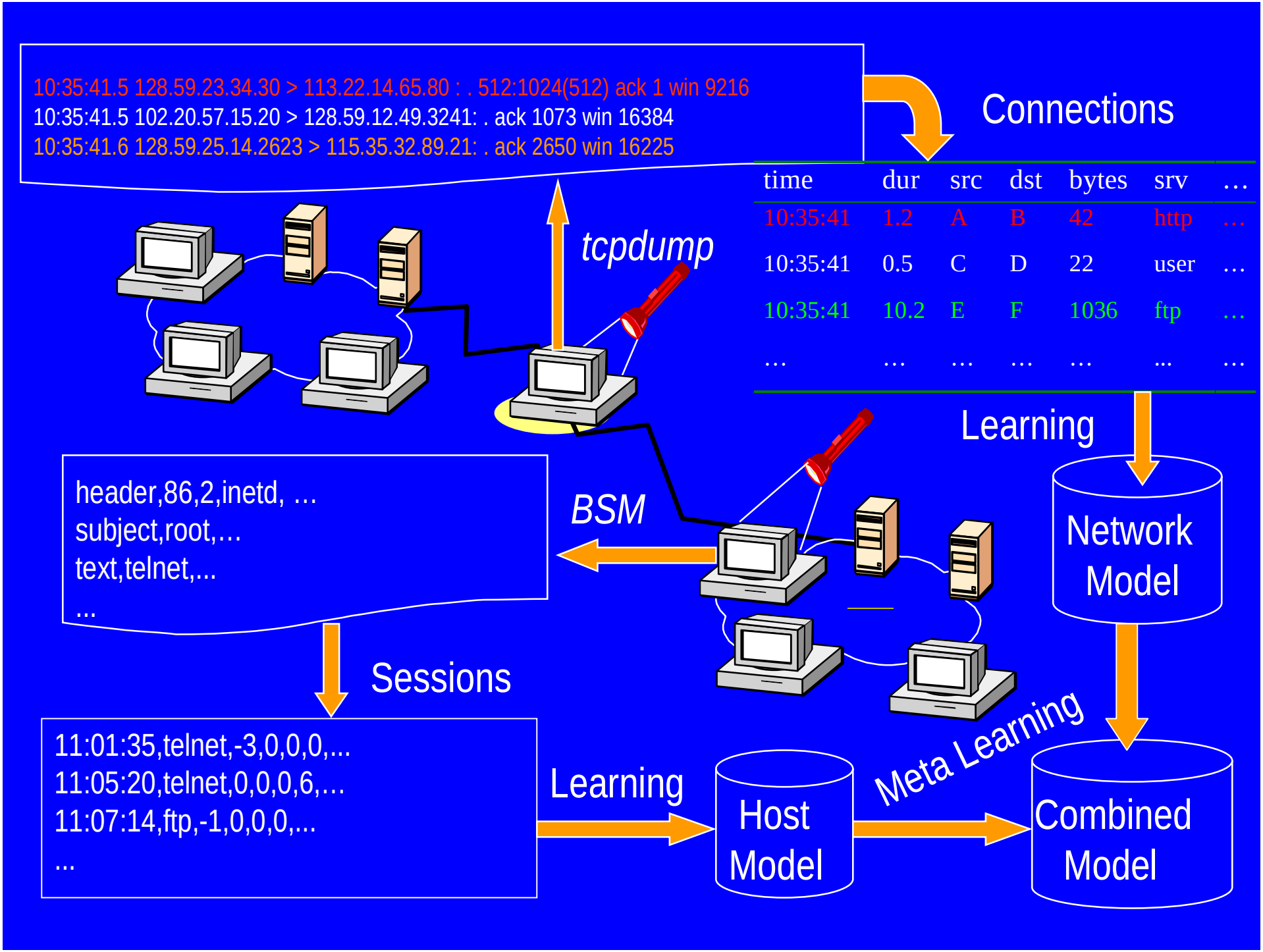
- Effective:
  - high detection rate, e.g., above 95%
  - low false alarm rate, e.g., a few per day
- Adaptable:
  - to detect “new” intrusions soon after they are invented
- Extensible:
  - to accommodate changed network configurations

Traditional Development Process

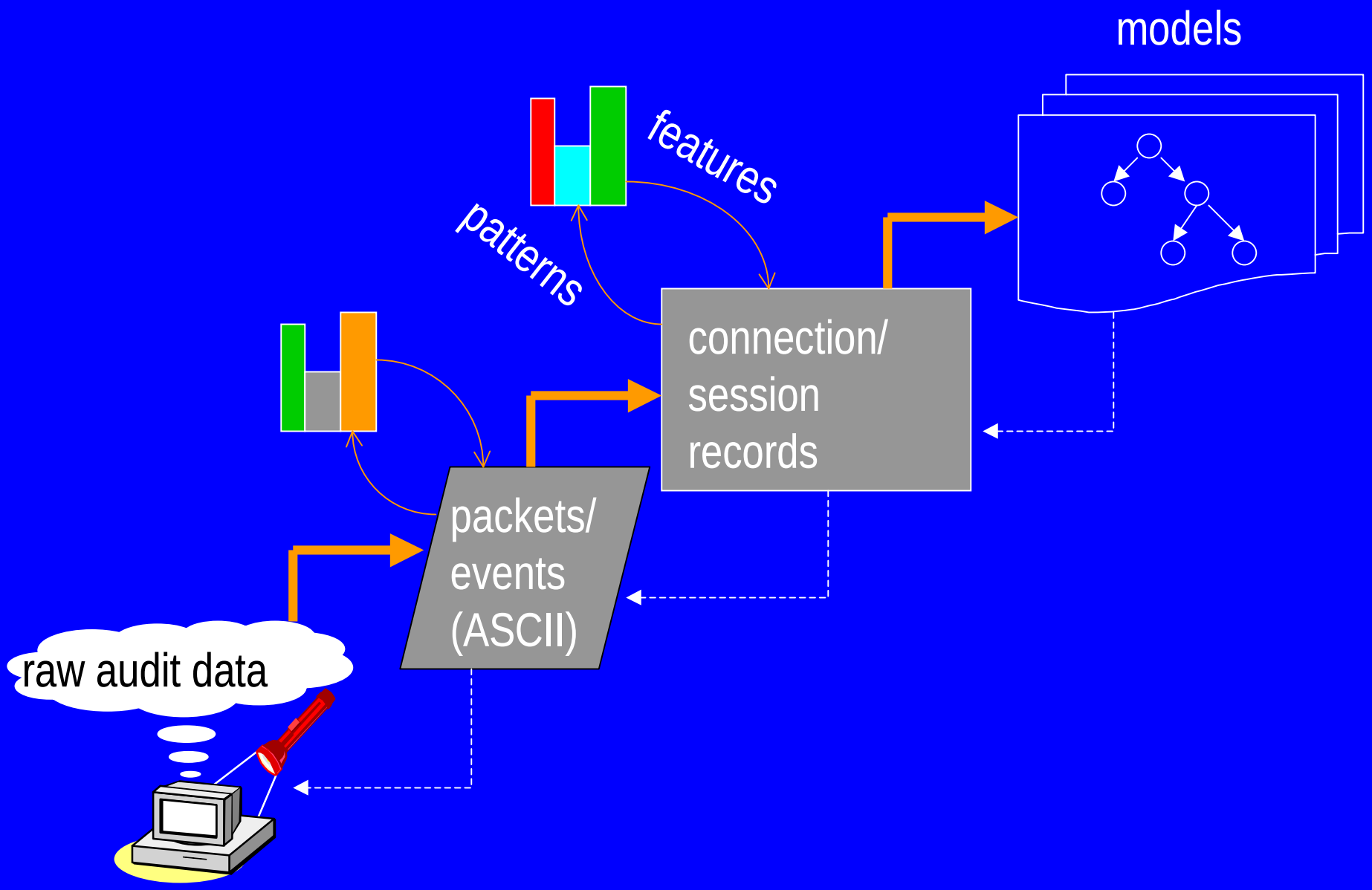
- Pure knowledge engineering approach:
  - Misuse detection:
    - Hand-code patterns for known intrusions
  - Anomaly detection:
    - Select measures on system features based on experience and intuition
  - Few formal evaluations

A New Approach

- A systematic data mining framework to:
  - Build effective models:
    - inductively learn detection models
    - select features using frequent patterns from audit data
  - Build extensible and adaptive models:
    - a hierarchical system to combine multiple models


10:35:41.5 128.59.23.34.30 > 113.22.14.65.80 : . 512:1024(512) ack 1 win 9216
10:35:41.5 102.20.57.15.20 > 128.59.12.49.3241: . ack 1073 win 16384
10:35:41.6 128.59.25.14.2623 > 115.35.32.89.21: . ack 2650 win 16225
Connections
time dur src dst bytes srv …
10:35:41 1.2 A B 42 http …
10:35:41 0.5 C D 22 user …
10:35:41 10.2 E F 1036 ftp …
… … … … … ... …
tcpdump
Learning
Network Model
BSM
header,86,2,inetd, …
subject,root,…
text,telnet,...
...
Sessions
11:01:35,telnet,-3,0,0,0,...
11:05:20,telnet,0,0,0,6,…
11:07:14,ftp,-1,0,0,0,...
...
Learning
Host Model
Meta Learning
Combined Model

The Data Mining Process of Building ID Models



Data Mining

- Relevant data mining algorithms for ID:
  - Classification: maps a data item to a category (e.g., normal or intrusion)
    - RIPPER (W. Cohen, ICML'95): a rule learner
  - Link analysis: determines relations between attributes (system features)
    - Association Rules (Agrawal et al. SIGMOD'93)
  - Sequence analysis: finds sequential patterns
    - Frequent Episodes (Mannila et al. KDD'95)

Classifiers as ID Models

- RIPPER:
  - Compute the most **distinguishing** and **concise** attribute/value tests for each class label
- Example RIPPER rules:
  - pod :- wrong_fragment ≥ 1, protocol_type = icmp.
  - smurf :- protocol = ecr_i, host_count ≥ 3, srv_count ≥ 3.
  - ...
  - normal :- true.

Classifiers as **EFFECTIVE** ID Models

- Critical requirements:
  - Temporal and statistical features
    - How to automate ***feature selection***?
  - Our solution:
    - Mine frequent sequential patterns from audit data

Mining Audit Data

- Basic algorithms:
  - Association rules: intra-audit record patterns
  - frequent episodes: inter-audit record patterns
  - Need both
- Extensions:
  - Consider characteristics of system audit data (Lee et al. KDD'98, IEEE SP'99)

Association Rules

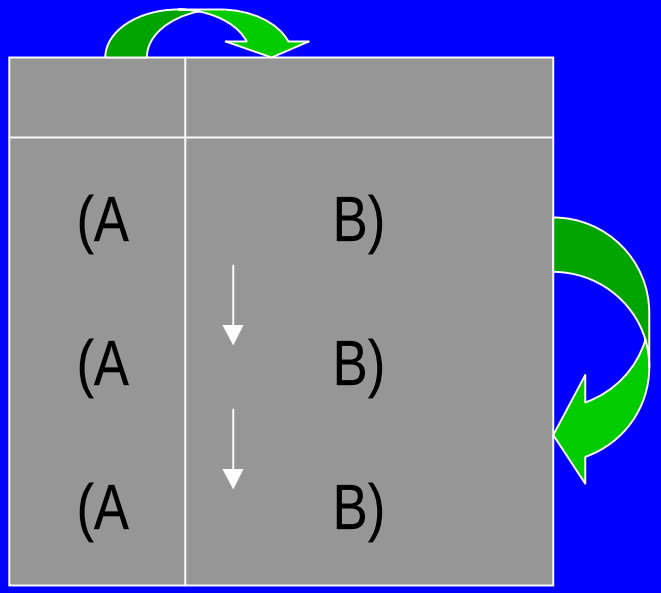
- Motivation:
  - Correlation among system features
- Example from shell commands:
  - $mail \rightarrow am, hostA$ [0.3, 0.1]
  - Meaning: 30% of the time when the user is sending emails, it is in the morning and from host A; and this pattern accounts for 10% of all his/her commands

Frequent Episodes

- Motivation:
  - Sequential information (system activities)
- Example from shell commands:
  - (*vi, C, am*) $\rightarrow$ (*gcc, C, am*) [0.6, 0.2, 5]
  - Meaning: 60% of the time, after *vi* (edits) a C file, the user *gcc* (compiles) a C file within the window of next 5 commands; this pattern occurs 20% of the time

Mining Audit Data (continued)

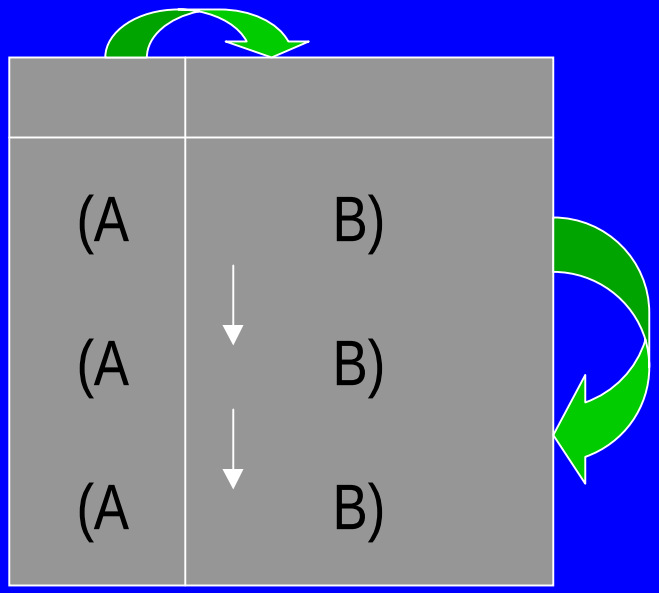
- Using the *Axis* Attribute(s)
  - Compute sequential patterns in two phases:
    - associations using the axis attribute(s)
    - serial episodes from associations



Example (*service* is the axis attribute):
(*service* = telnet, *src_bytes* = 200, *dst_bytes* = 300, *flag* = SF), (*service* = smtp, *flag* = SF) $\rightarrow$ (*service* = telnet, *src_bytes* = 200).

Mining Audit Data (continued)

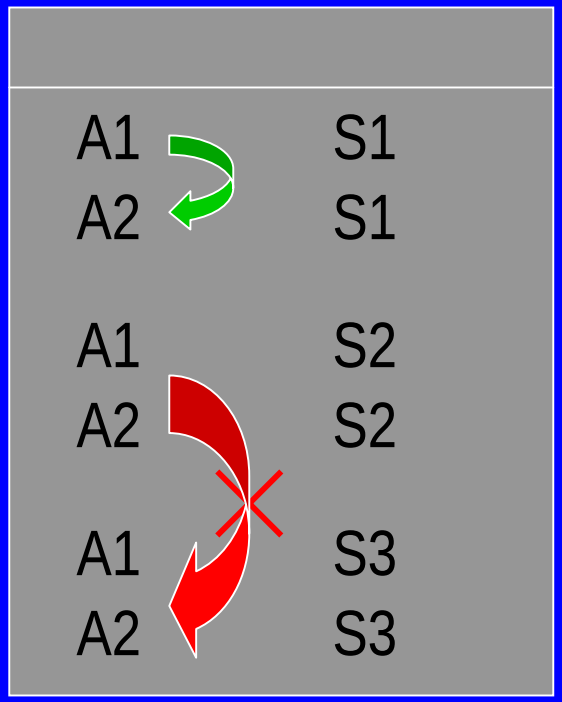
- Using the *Axis* Attribute(s)
  - Compute sequential patterns in two phases:
    - associations using the axis attribute(s)
    - serial episodes from associations



Axis attributes are the “essential” attributes of audit records, e.g., service, hosts, etc.

Mining Audit Data (continued)

- "reference" relations among the attributes
  - reference attribute(s): "subject", e.g., *dst_host*
  - others, e.g., *service* : "actions" of "subject"
  - "actions" pattern is frequent, but not "subject"



reference attribute(s) as an item constraint:

records of an episode must have the same *reference* attribute value

Connection Records (port scan)

...

17:27:57 1234 priv_19 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 priv_18 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 priv_17 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 priv_16 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 netstat 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 priv_14 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 daytime 192.168.1.10 172.16.114.50 ? ? REJ ...

17:27:57 1234 priv_12 192.168.1.10 172.16.114.50 ? ? REJ ...

...

Frequent Patterns (port scan)

- Use dst_host is as both the *axis* and *reference* attribute to find the "same destination host" frequent sequential "destination host" patterns:
  - (dst_host =172.16.114.50, src_host = 192.168.1.10, flag = REJ), (dst_host =172.16.114.50, src_host = 192.168.1.10, flag = REJ) $\rightarrow$ (dst_host =172.16.114.50, src_host = 192.168.1.10, flag = REJ) [0.8, 0.1, 2]
  - ...

Connection Records (syn flood)

...

11:55:15 19468 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

11:55:15 19724 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

11:55:15 18956 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

11:55:15 20492 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

11:55:15 20748 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

11:55:15 21004 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

11:55:15 21516 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

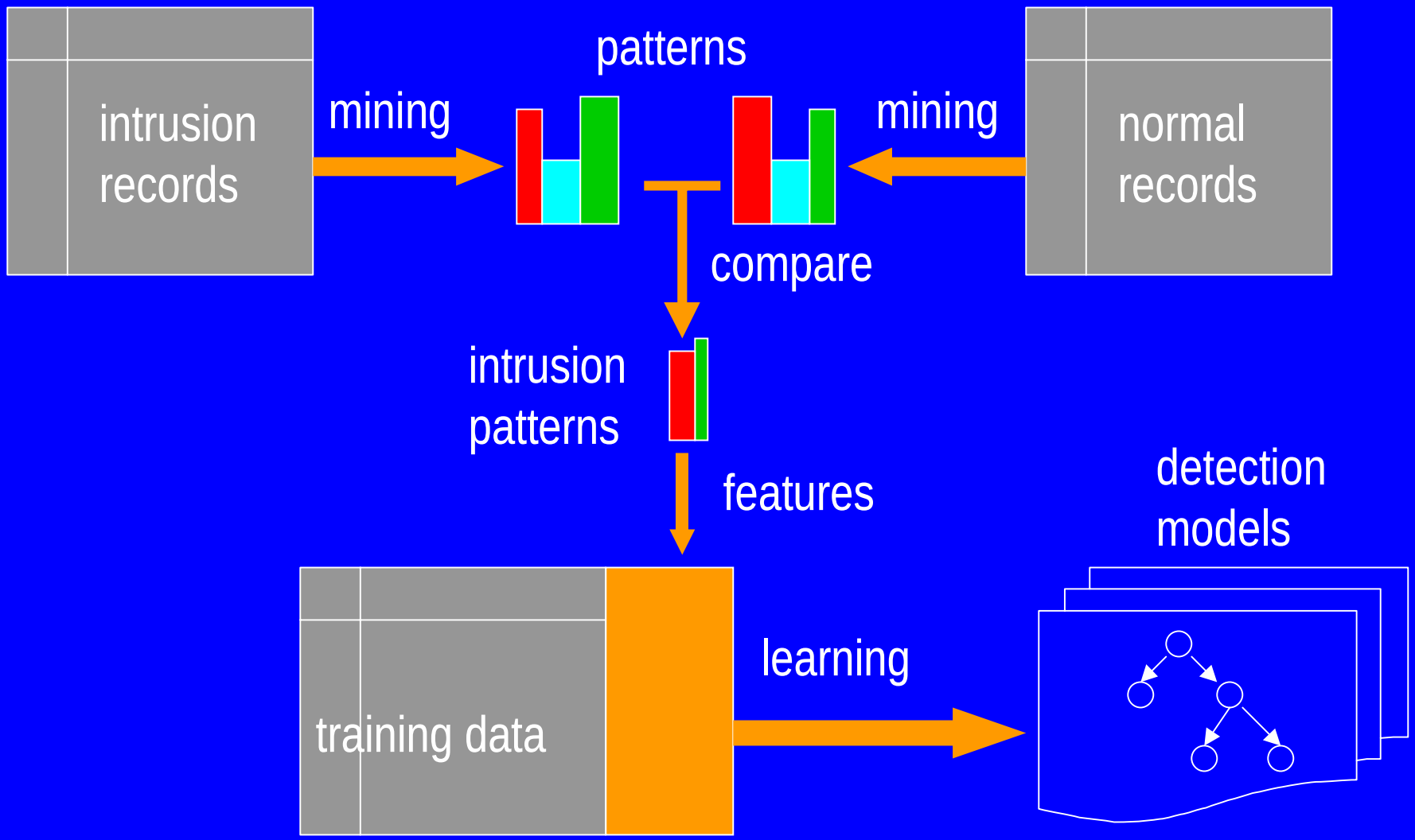
11:55:15 21772 telnet 1.2.3.4 172.16.112.50 ? ? S0 ...

...

Frequent Patterns (syn flood)

- Use service as the *axis* attribute and dst_host is *reference* attribute to find "same destination host" frequent sequential "service" patterns:
  - (service = telnet, flag = S0), (service = telnet, flag = S0) $\rightarrow$ (service = telnet, flag = S0) [0.6, 0.1, 2]
  - ...

Feature selection/construction



Feature selection/construction

- An example: “syn flood” patterns (dst_host is *reference* attribute):
  - (service = telnet, flag = S0), (service = telnet, flag = S0) $\rightarrow$ (service = telnet, flag = S0) [0.6, 0.1, 2]
  - add features:
    - count the connections to the same *dst_host* in the past 2 seconds, and among these connections,
    - the # with the same *service*,
    - the # with S0

1998 DARPA ID Evaluation

- The plan:
  - Seven weeks of labeled training data, *tcpdump* and *BSM* output
    - normal traffic and intrusions
    - participants develop and tune intrusion detection algorithms
  - Two weeks of unlabeled test data
    - participants submit “list” files specifying the detected intrusions
    - ROC (on TP and FP) curves to evaluate

DARPA ID Evaluation (cont'd)

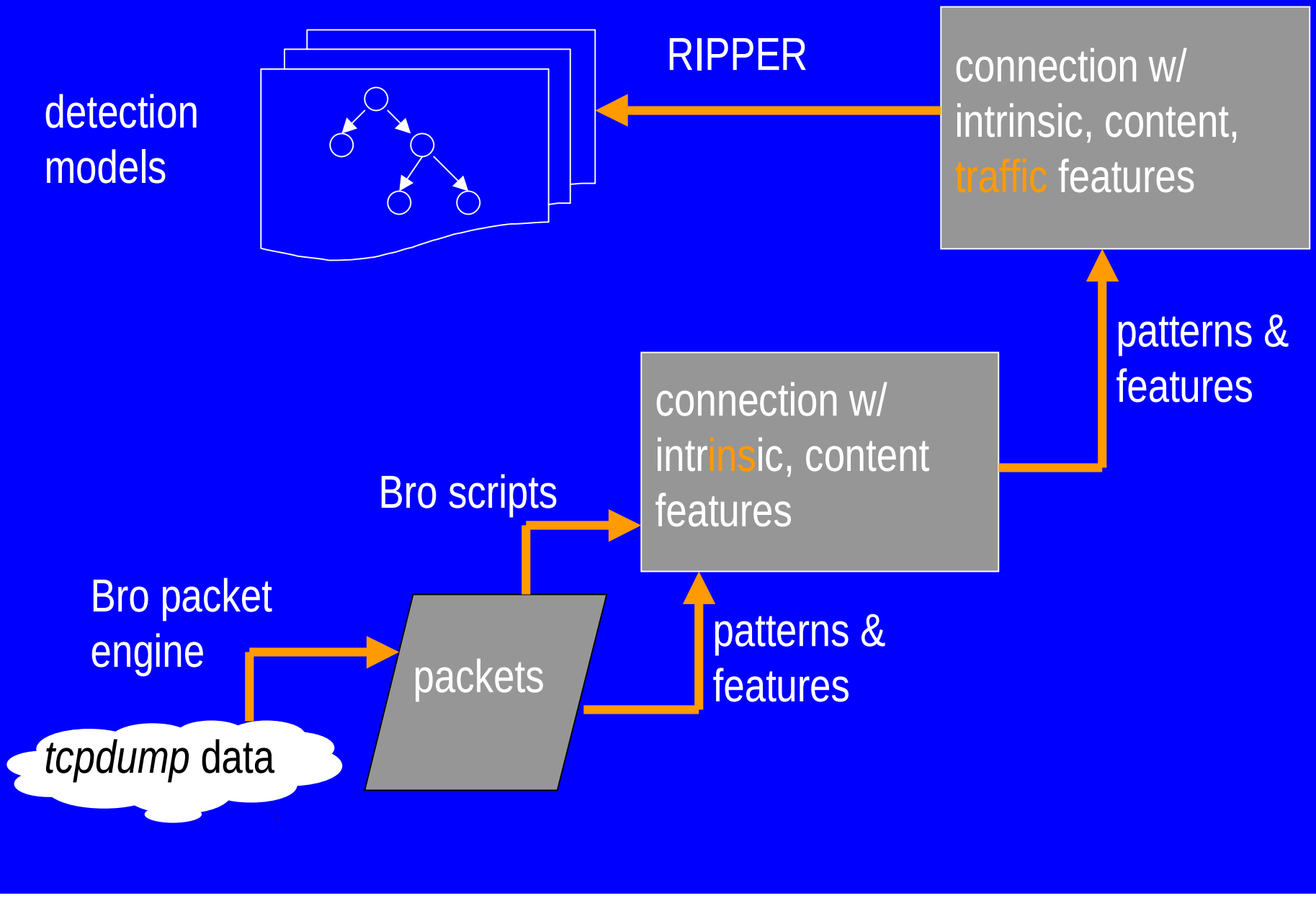
- The data:
  - Total 38 attack types, in four categories:
    - DOS (denial-of-service), e.g., syn flood
    - Probing (gathering information), e.g., port scan
    - r2l (remote intruder illegally gaining access to local systems), e.g., guess password
    - u2r (user illegally gaining root privilege), e.g., buffer overflow
  - 40% of attack types are in test data only, i.e., "new" to intrusion detection systems
    - to evaluate how well the IDSs generalized

38 Attack Types in 1998 Test Data

	Solaris Server (audited)	SunOS internal	Linux internal	Cisco Router
DENIAL OF SERVICE (11 Types, 43 Instances)	•back •Neptune •Ping of death •Smurf •Syslogd •land •Apache2 •Mailbomb •Process Table •UDP Storm	•back •Neptune •Ping of death •Smurf •land •Apache2 •Mailbomb •Process Table •UDP Storm	•back •Neptune •Ping of death •Smurf •Teardrop •land •Apache2 •Mailbomb •Process Table •UDP Storm	•snmpgetattack
REMOTE TO USER (14 Types, 16 Instances)	•dictionary •ftp-write •guest •phf •ftp-write •httptunnel •xlock •xsnoop	•dictionary •ftp-write •guest •phf •httptunnel •xlock •xsnoop	•dictionary •ftp-write •guest •imap •phf •httptunnel •named •sendmail •xlock •xsnoop	
USER TO ROOT (7 Types, 38 Instances)	•eject •ffbconfig •fdformat •ps	•loadmodule •ps	•perl •xterm	
SURVEILLANCE /PROBE (6 Types, 17 Instances)	•ip sweep •nmap •port sweep •satan •mscan •saint	•ip sweep •nmap •port sweep •satan •mscan •saint	•ip sweep •nmap •port sweep •satan •mscan •saint	•ip sweep •nmap •port sweep •satan •mscan •saint

• 114 Attacks in 2 Weeks of Test Data ■ ≡ test only

Building ID models for DARPA Data



DARPA ID Evaluation (cont'd)

- Features from Bro scripts:
  - "intrinsic" features:
    - protocol (service),
    - protocol type (tcp, udp, icmp, etc.)
    - duration of the connection,
    - flag (connection established and terminated properly, SYN error, rejected, etc.),
    - # of wrong fragments,
    - # of urgent packets,
    - whether the connection is from/to the same ip/port pair.

DARPA ID Evaluation (cont'd)

- "content" features (for TCP connections only):
  - # of failed logins,
  - successfully logged in or not,
  - # of root shell prompts,
  - "su root" attempted or not,
  - # of access to security control files,
  - # of compromised states (e.g., "Jumping to address", "path not found" ...),
  - # of write access to files,
  - # of outbound commands,
  - # of hot (the sum of all the above "hot" indicators),
  - is a "guest" login or not,
  - is a root login or not.

DARPA ID Evaluation (cont'd)

- Features constructed from mined patterns:
  - temporal and statistical "traffic" features that describe **connections** within a time window:
    - # of connections to the **same *destination host*** as the current connection in the past 2 seconds, and among these connections,
    - # of rejected connections,
    - # of connections with "SYN" errors,
    - # of different services,
    - % of connections that have the same service,
    - % of different (unique) services.

DARPA ID Evaluation (cont'd)

- Features constructed from mined patterns:
  - temporal and statistical "traffic" features (cont'd):
    - # of connections that have the ***same service*** as the current connection, and among these connections,
    - # of rejected connections,
    - # of connection with "SYN" errors,
    - # of different destination hosts,
    - % of the connections that have the same destination host,
    - % of different (unique) destination hosts.

DARPA ID Evaluation (cont'd)

- Learning RIPPER rules:
  - the "content" model for TCP connections:
    - detect u2r and r2l attacks
    - each record has the "intrinsic" features + the "content" features, total 22 features
    - total 55 rules, each with less than 4 attribute tests
    - total 11 distinct features actually used in all the rules

DARPA ID Evaluation (cont'd)

- example "content" connection records:

dur	p_type	proto	flag	l_in	root	su	compromised	hot	…	label
92	tcp	telnet	SF	1	0	0	0	0	…	normal
26	tcp	telnet	SF	1	1	1	0	2	…	normal
2	tcp	http	SF	1	0	0	0	0	…	normal
149	tcp	telnet	SF	1	1	0	1	3	…	buffer
2	tcp	http	SF	1	0	0	1	1	…	back

- example rules:
  - buffer_overflow :- hot ≥ 3, compromised ≥ 1, su_attempted ≤ 0, root_shell ≥ 1.
  - back :- compromised ≥ 1, protocol = http.

DARPA ID Evaluation (cont'd)

- Learning RIPPER rules (cont'd):
  - the "traffic" model for all connections:
    - detect DOS and Probing attacks
    - each record has the "intrinsic" features + the "traffic" features, total 20 features
    - total 26 rules, each with less than 4 attribute tests
    - total 13 distinct features actually used in all the rules

DARPA ID Evaluation (cont'd)

- example "traffic" connection records:

dur	p_type	proto	flag	count	srv_count	r_error	diff_srv_rate	…	label
0	icmp	ecr_i	SF	1	1	0	1	…	normal
0	icmp	ecr_i	SF	350	350	0	0	…	smurf
0	tcp	other	REJ	231	1	198	1	…	satan
2	tcp	http	SF	1	0	0	1		normal

- example rules:
  - smurf :- protocol = ecr_i, count ≥ 5, srv_count ≥ 5.
  - satan :- r_error ≥ 3, diff_srv_rate ≥ 0.8.

DARPA ID Evaluation (cont'd)

- Learning RIPPER rules (cont'd):
  - the host-based "traffic" model for all connections:
    - detect slow probing attacks
    - sort connections by destination hosts
    - construct a set of host-based traffic features, similar to the (time-based) temporal statistical features
    - each record has the "intrinsic" features + the host-based "traffic" features, total 14 features
    - total 8 rules, each with less than 4 attribute tests
    - total 6 distinct features actually used in all the rules

DARPA ID Evaluation (cont'd)

- example host-based "traffic" connection records:

dur	p_type	proto	flag	count	srv_count	srv_diff_host_rate	…	label
2	tcp	http	SF	0	0	0	…	normal
0	icmp	eco_i	SF	1	40	0.5	…	ipsweep
0	icmp	ecr_i	SF	112	112	0	…	normal

- example rules:
  - ipsweep :- protocol = eco_i, srv_diff_host_rate $\geq$ 0.5, count $\leq$ 2, srv_count $\geq$ 6.

DARPA ID Evaluation (cont'd)

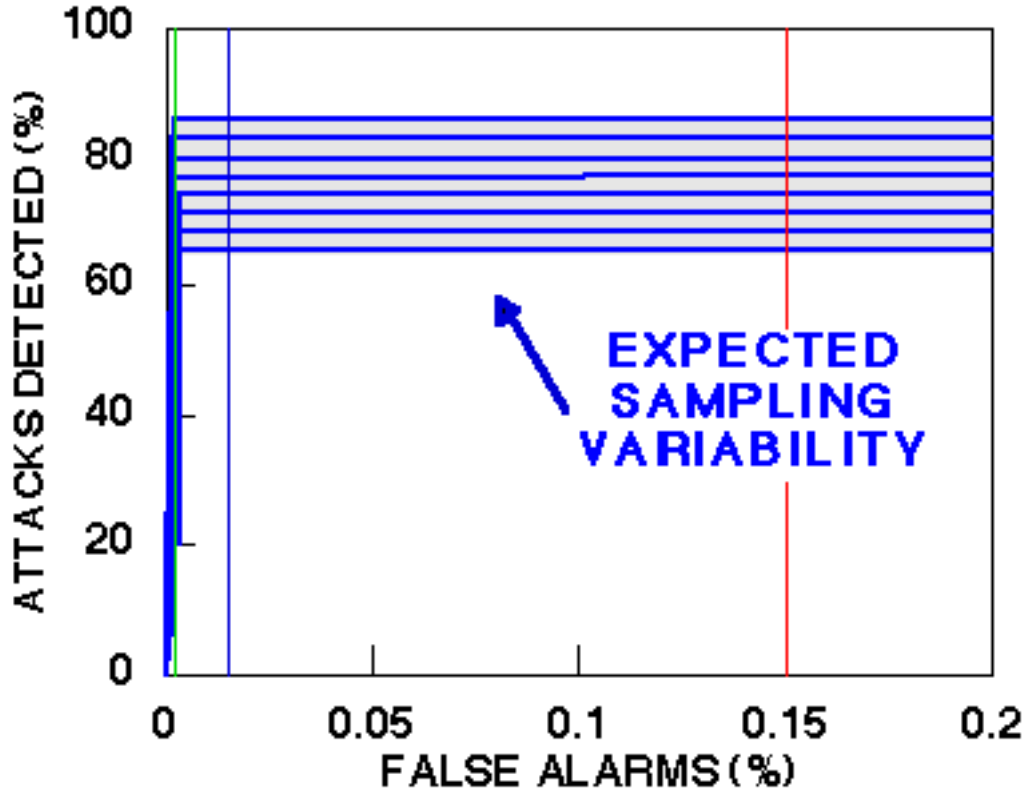
- Learning RIPPER rules - a summary:

Models	**Attacks**	**Features**	**# Features in training**	**# Rules**	**# Features in rules**
content	u2r, r2l	intrinsic + content	22	55	11
traffic	DOS, probing	intrinsic + traffic	20	26	4 + 9
host traffic	slow probing	intrinsic + host traffic	14	8	1 + 5

DARPA ID Evaluation (cont'd)

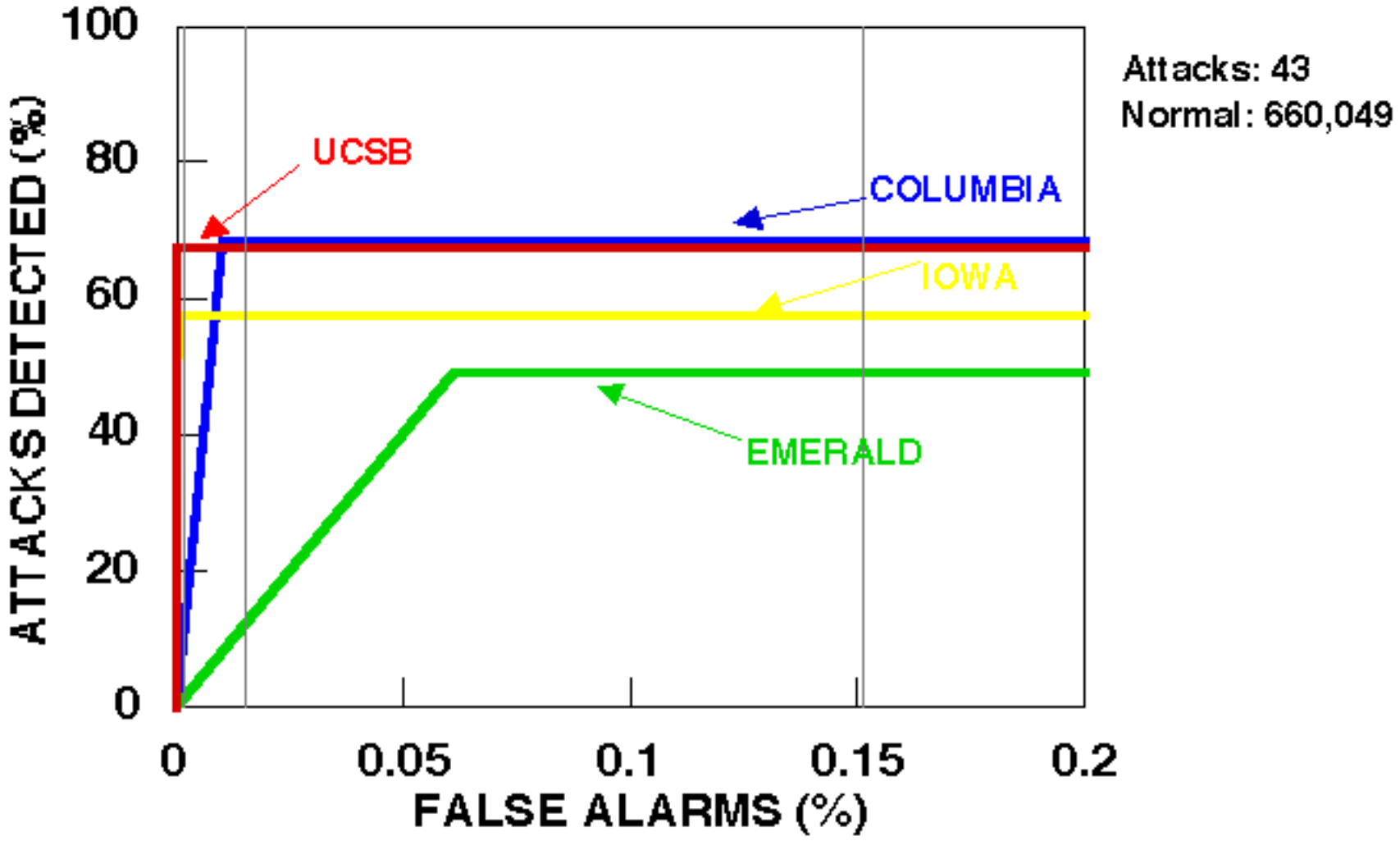
- Results evaluated by MIT Lincoln Lab
  - Participants:
    - Columbia
    - UCSB
    - SRI (EMERALD)
    - Iowa State/Bellcore
    - Baseline Keyword-based System (Lincoln Lab)

Statistical Sampling Variability In ROCs

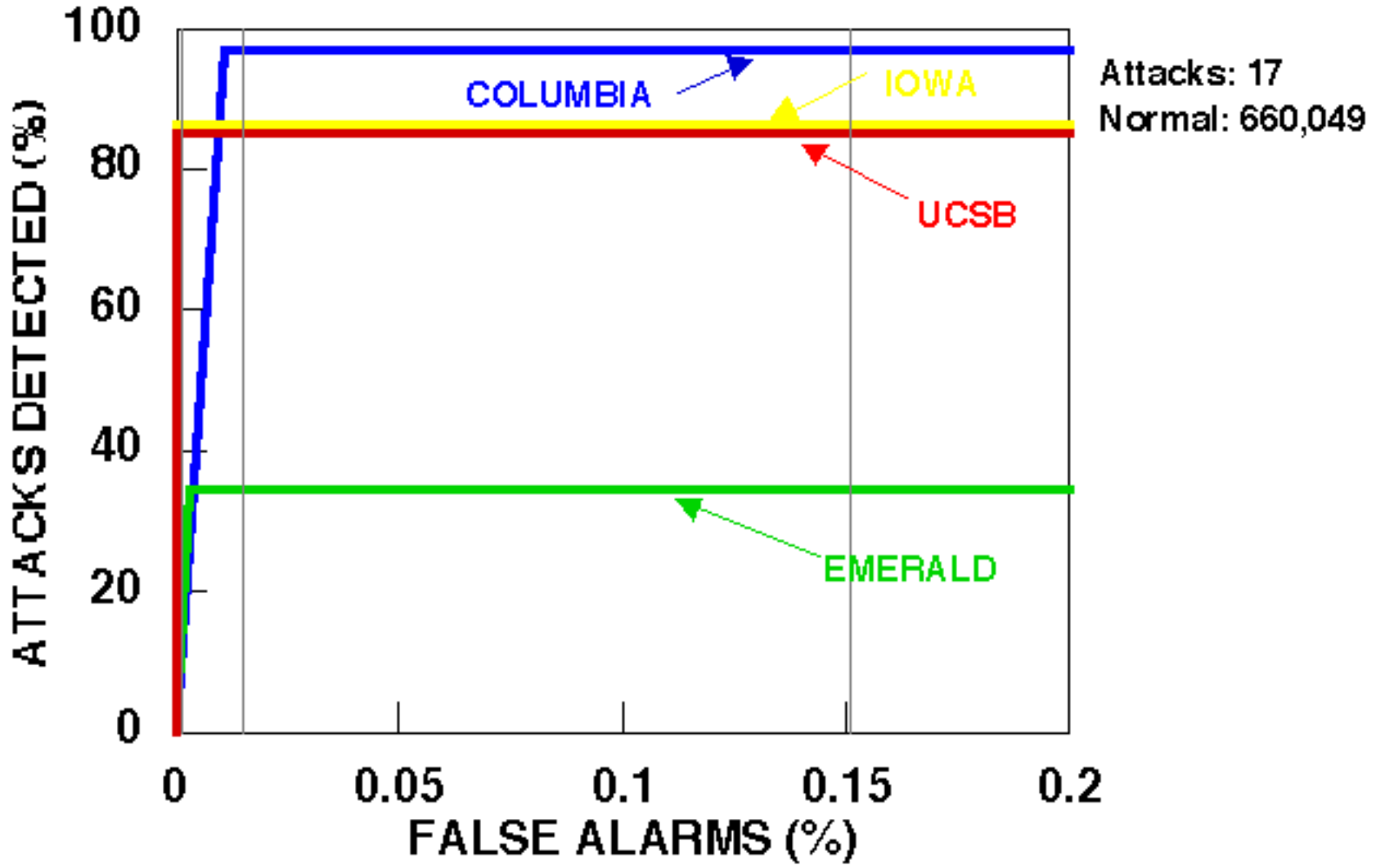


- Don't Inflate Small Differences Between ROCs
- Two Standard Deviations are 12 to 18 Percentage Points on ROC Plots for Separate Attack Categories (20 Bootstrap ROC's Above Confirm This)
- This is the First Evaluation, Small Changes in Scoring, and Attack Composition Can Lead to Similar Small Differences

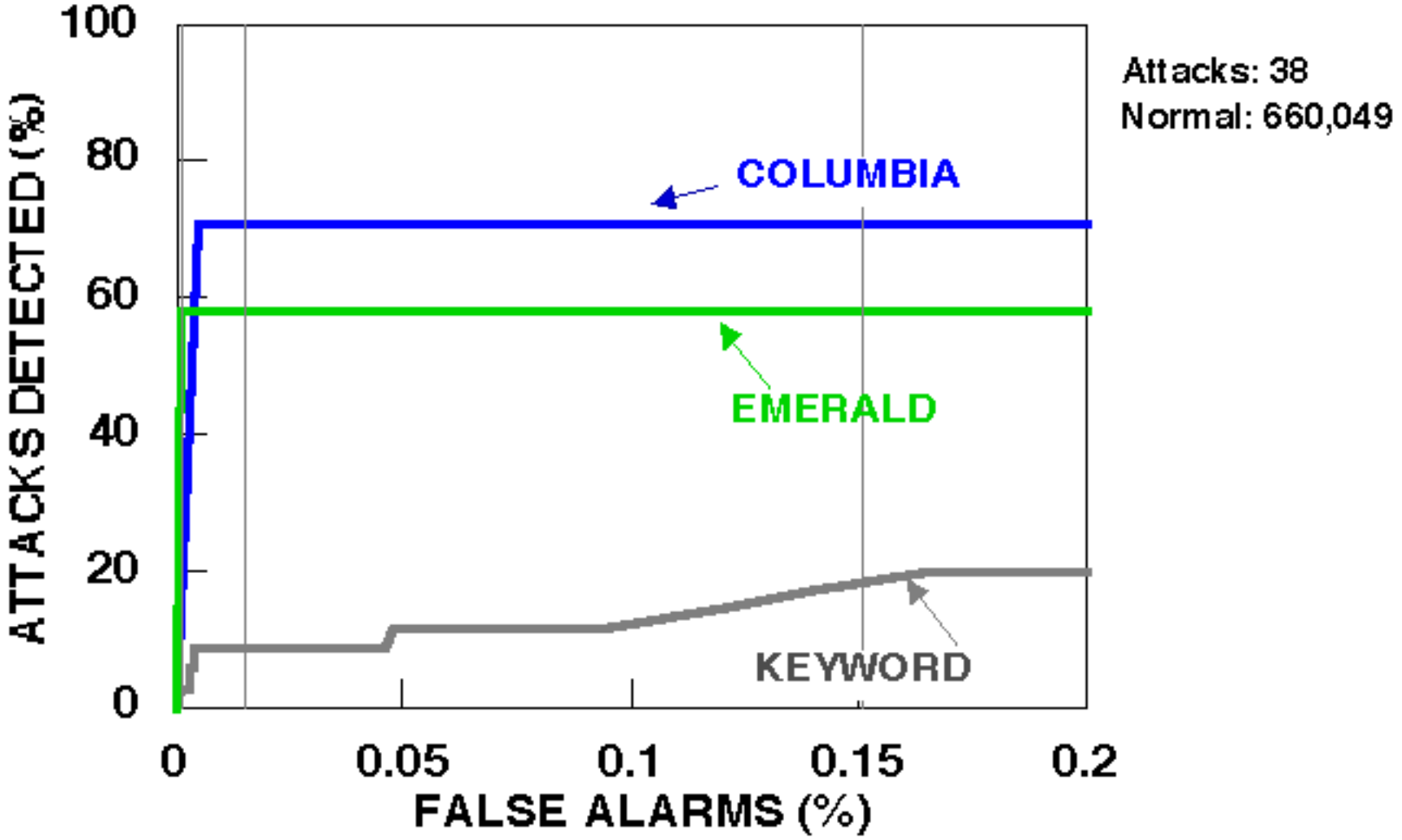
TCPDump - Denial of Service ROC



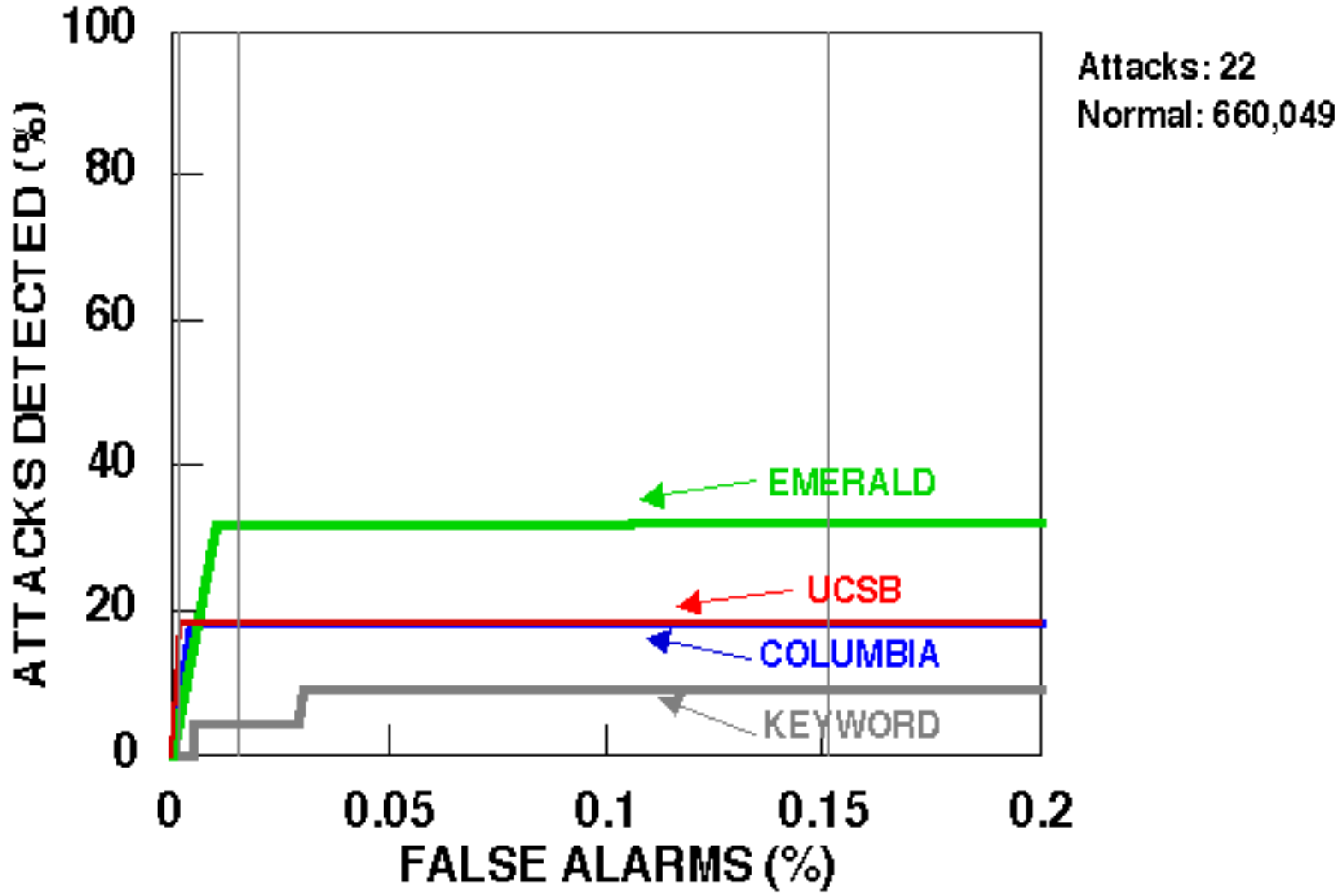
TCPdump: Probe



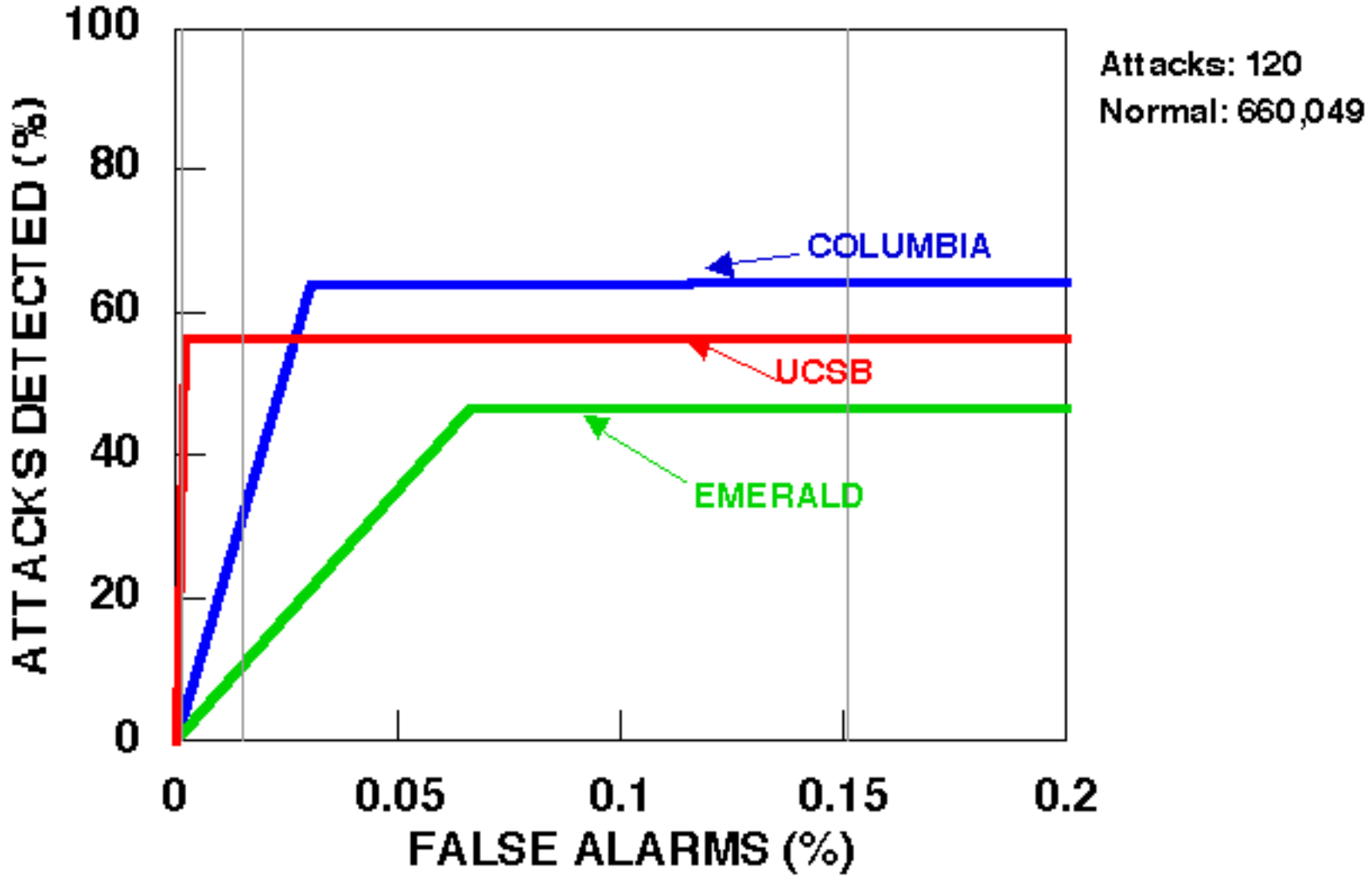
TCPdump: User to Root



TCPdump: Remote to Local



TCPDump Overall



DARPA ID Evaluation (cont'd)

- Our results:
  - Very good detection rate for probing, and acceptable detection rates for u2r and DOS attacks
    - predictive features are constructed
    - variations of the attacks are relatively limited
    - training data contains representative instances
  - Poor detection rate for r2l attacks
    - too many variations
    - lack of representative instances in training data

Open Problems

- Anomaly detection for network traffic
- Real-time ID systems:
  - translate learned rules into real-time detection modules
  - optimize algorithms and data structures
  - more intelligent/efficient auditing

Resources

- Intrusion detection research:
  - www.cs.purdue.edu/coast/intrusion-detection
- Attack programs
  - www.rootshell.com
- Intrusion detection systems:
  - www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html
  - NFR (www.nfr.com)
  - Bro (ftp.ee.lbl.gov)