Concordia University
Engineering and Computer Science

Advanced Host-Based Anomaly Detection for Cyber Security

Wahab Hamou-Lhadj, PhD, ing.
Software Behaviour Analysis (SBA) Research Lab
ECE, Concordia University
Montreal, QC, Canada
wahab.hamou-lhadj@concordia.ca

Apr. 17, 2014
UQO, QC, Canada

Université Concordia University

Recent Significant Cyber Incidents according to CSIS

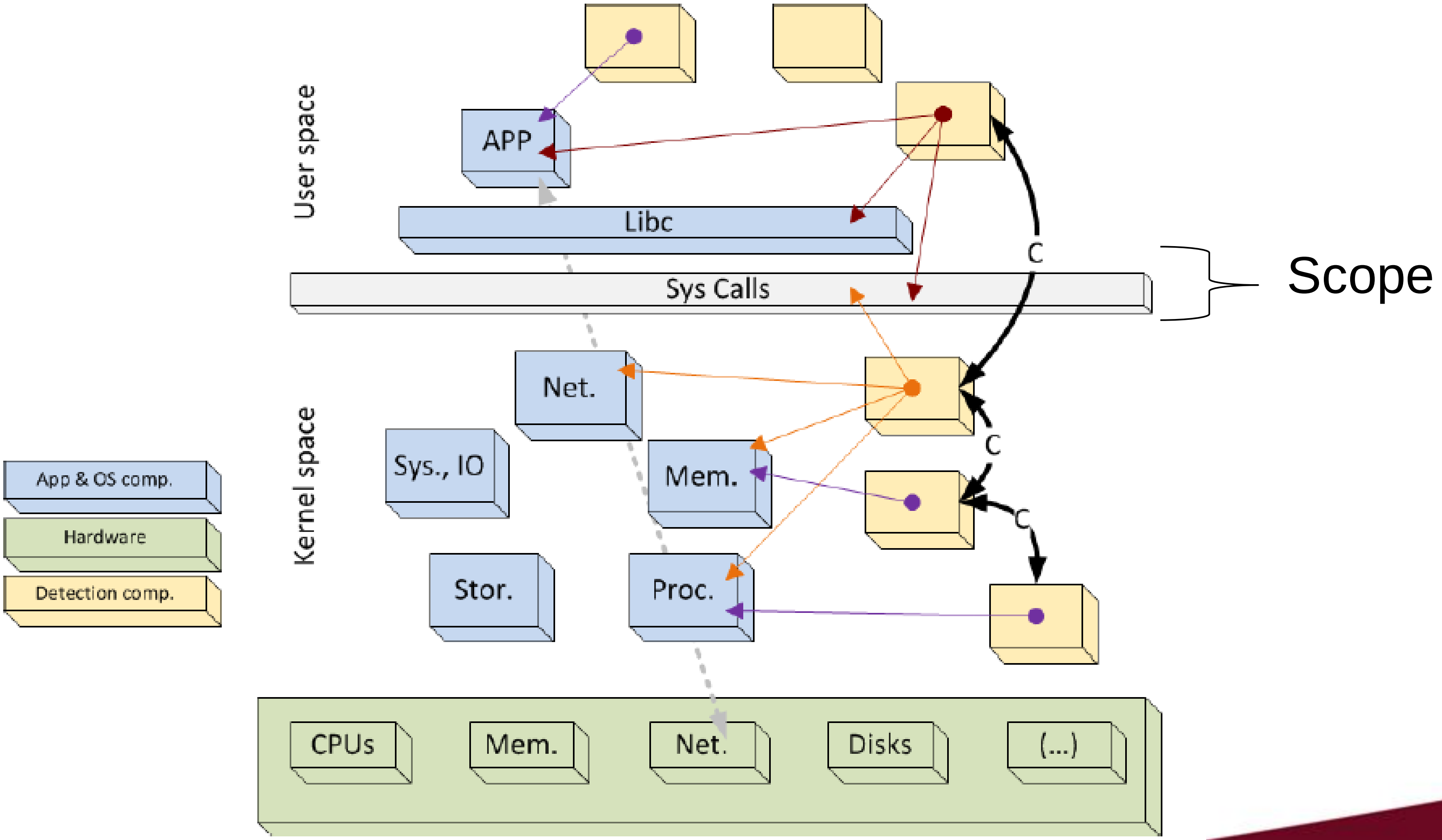
- 2011. The Canadian government reported a major cyber attack against its agencies, including Defence Research and Development Canada, a research agency for Canada‘s Department of National Defence.
- 2011. Cybercriminals masquerading as member of the hacktivist group “Anonymous” penetrated the Play Station network. Sony estimated that personal information for more than 80 million users was compromised and that the cost of the breach at over $170 million.
- 2011. Australia’s Defense Signals Directorate says that defense networks are attacked more than 30 times a day, with the number of attacks increasing by more than 350 percent by 2009.
- 2012. The Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) reported that two power plants in the U.S. suffered sophisticated malware infections.
- The head of the UK Security Service stated that a London-listed company lost an estimated £800m ($1.2 billion) as a result of state cyber attacks.
- 2013. Chinese hackers breach the Federal Election Commission’s networks while it is closed during the U.S. government shutdown.
- 2013. An estimated 40 million holiday shoppers at a major U.S. retail Chain have debit and credit card credentials stolen by hackers.
- 2013. Russian hackers steal personal data from 54 million Turks.
- 2014. Indian defense sources say classified material may have been compromised when around 50 computers from the armed forces and the Indian defense research organization were hacked.

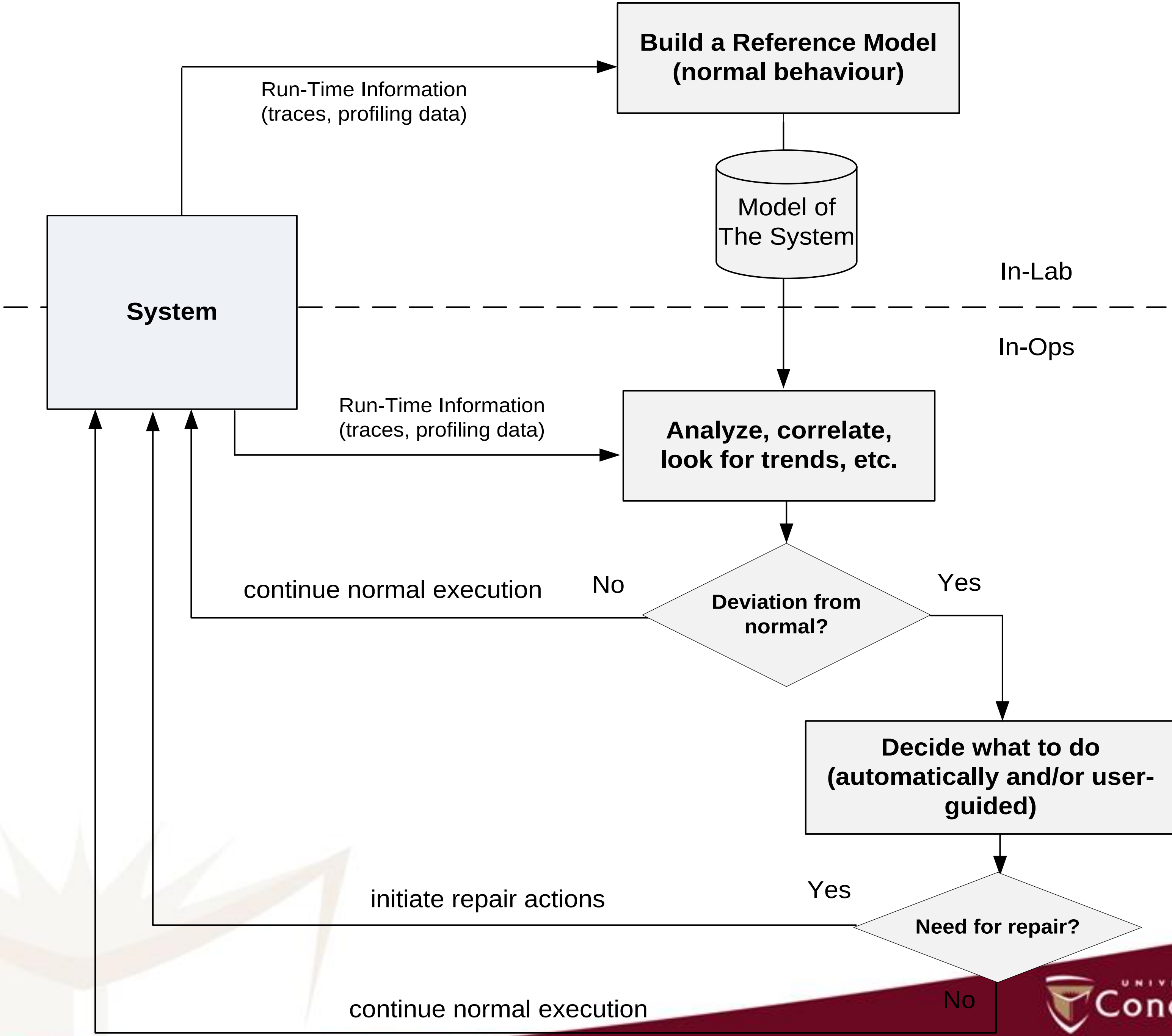
http://csis.org/files/publication/140310_Significant_Cyber_Incidents_Since_2006.pdf

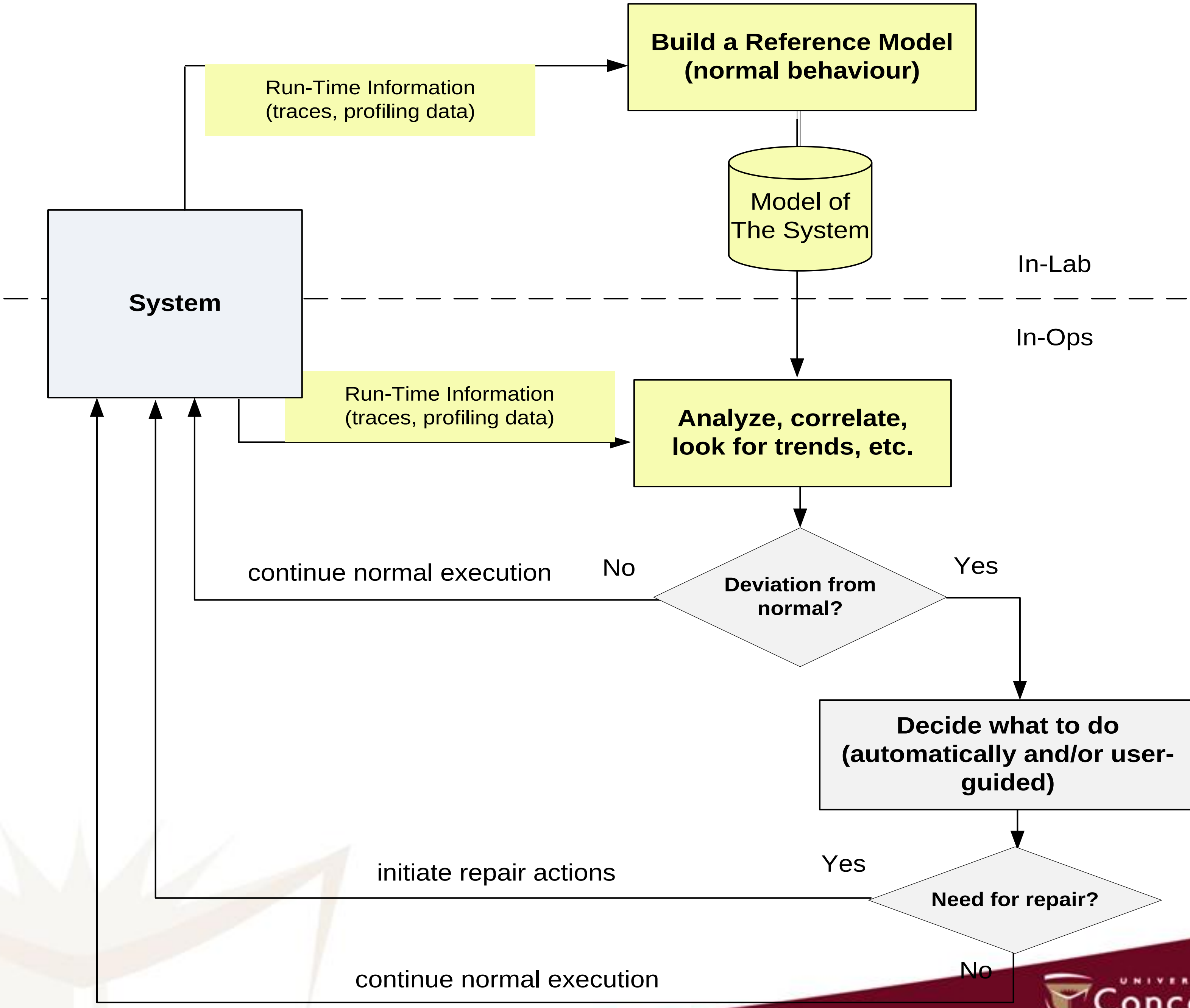
Intrusion Detection Systems

- Monitor computer or network activity for signs of intrusions and alert administrator
- Signature based Detection
  - Looks for known patterns
  - Detects only known attacks
- Anomaly Detection
  - Looks for deviations from normal behavior
  - Detects even unknown attacks

Detection layers




Build a Reference Model (normal behaviour)
Run-Time Information (traces, profiling data)
Model of The System
In-Lab
System
In-Ops
Run-Time Information (traces, profiling data)
Analyze, correlate, look for trends, etc.
continue normal execution
No
Deviation from normal?
Yes
Decide what to do (automatically and/or user-guided)
initiate repair actions
Yes
Need for repair?
No
continue normal execution

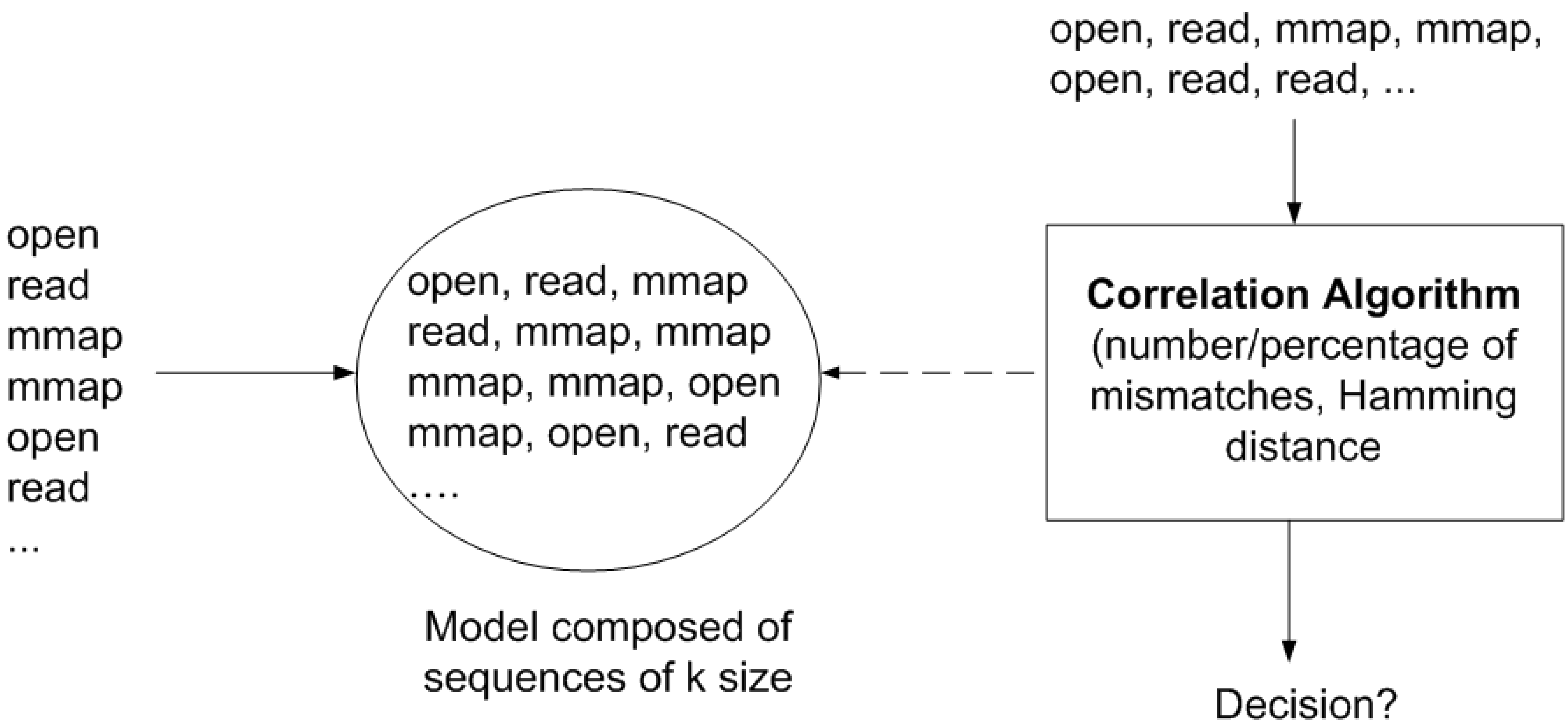


Build a Reference Model (normal behaviour)
Run-Time Information (traces, profiling data)
Model of The System
In-Lab
System
In-Ops
Run-Time Information (traces, profiling data)
Analyze, correlate, look for trends, etc.
continue normal execution
No
Deviation from normal?
Yes
Decide what to do (automatically and/or user-guided)
Yes
initiate repair actions
Need for repair?
No
continue normal execution

Existing Work

- Several techniques have been used to model the normal behaviour of a system
  - Sliding window technique
  - HMM
  - Neural networks (two-class)
  - Clustering
  - Varied length n-gram technique
  - Context Free Grammar

Example: Sliding Approach



Challenges – False alarms

- ADSs generate large numbers of false alarms
  - → Misclassify normal events as anomalous

- Frequent false alarms reduce the confidence and could lead to deactivation of the ADS

Challenges – False alarms

- False alarms are caused by several reasons including:
  - Unrepresentative normal data for training and attack data for validation and testing
  - Inappropriate model or feature selection
  - Poor optimization of models parameters
  - Over fitting (leads to poor generalization)
  - Inadequate assumptions such as static environments

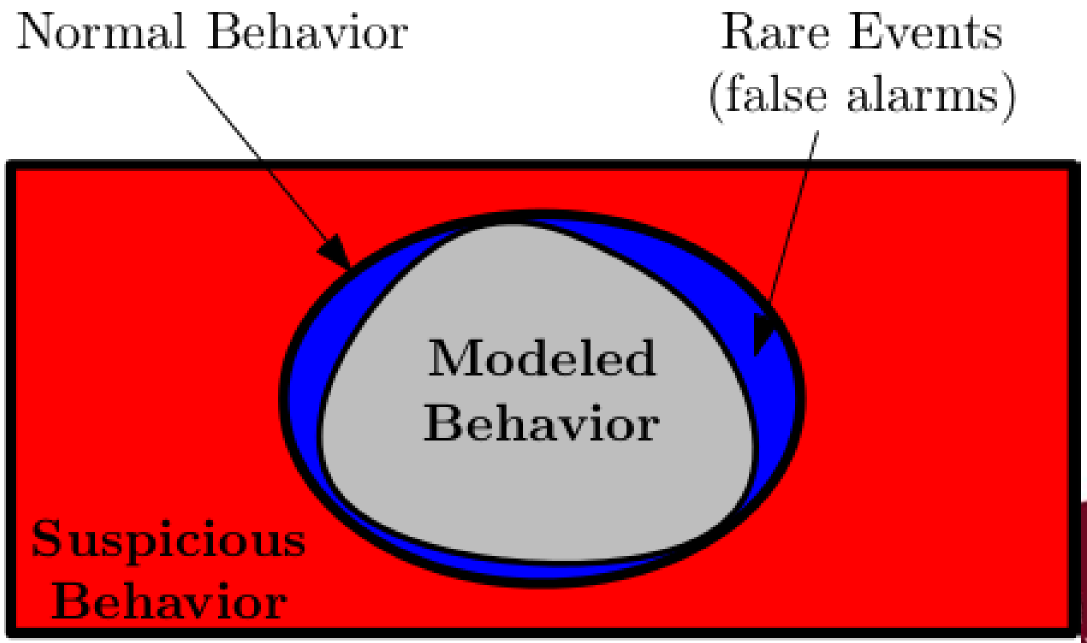
Assumptions

- Most of the work found in related literature assumes:
  - Representative amount of normal data provided for training
  - Static environments: normal behavior will not change over time

In Practice

- ADSs are often designed using limited data
  - collection and analysis of representative data from each process (different version, OS, etc.) is costly

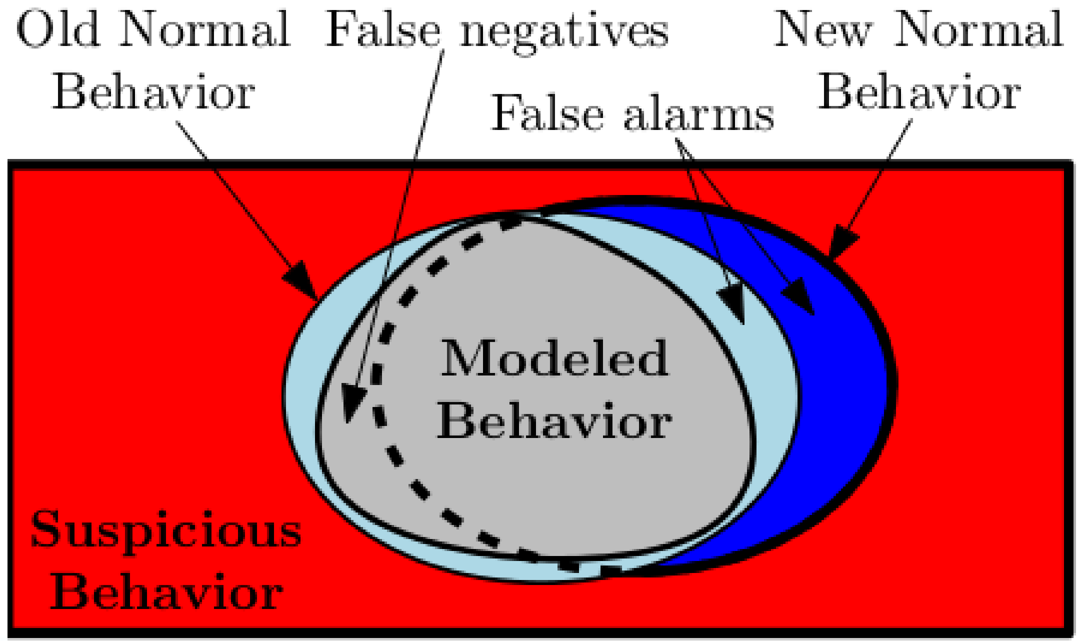
Anomaly detector will have **incomplete** view of normal system behavior



In Practice

- Dynamic environment
  - Changes in normal process behavior due, for instance, to application update

Internal model of normal behavior **diverges** with respect to the underlying data



ADS Requirements

- ADSs should be able to:
  - Account for rare normal events (false alarms)
  - Scalable and modular: can add, replace or remove models or features over time
  - Handle large data spaces
  - Accommodate new data

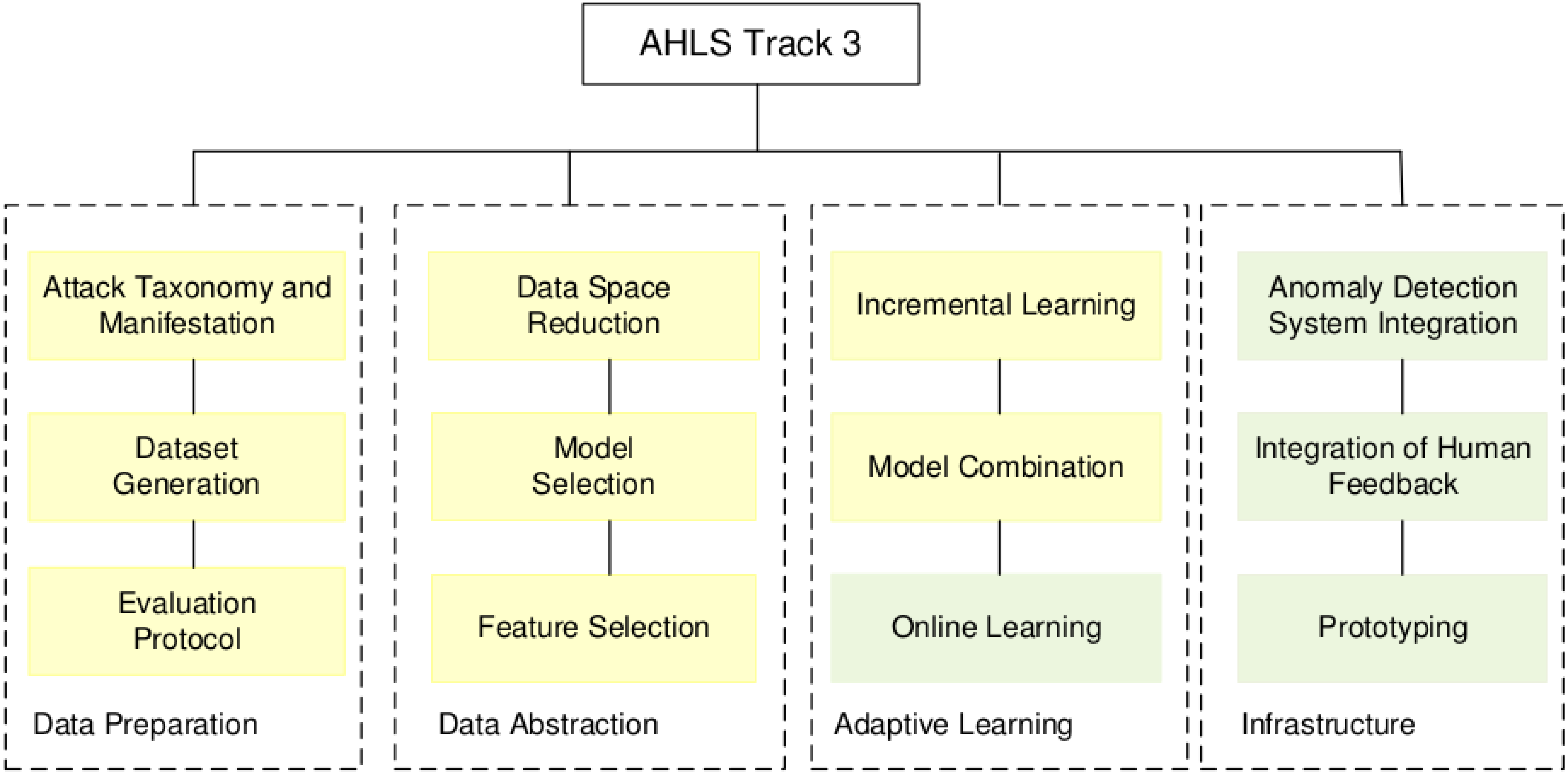
Advanced Host-Level Project

- Four-year NSERC-DND project (2012-2015)
- 6 PhDs, 8 Masters, 2 Postdocs, 2 RAs

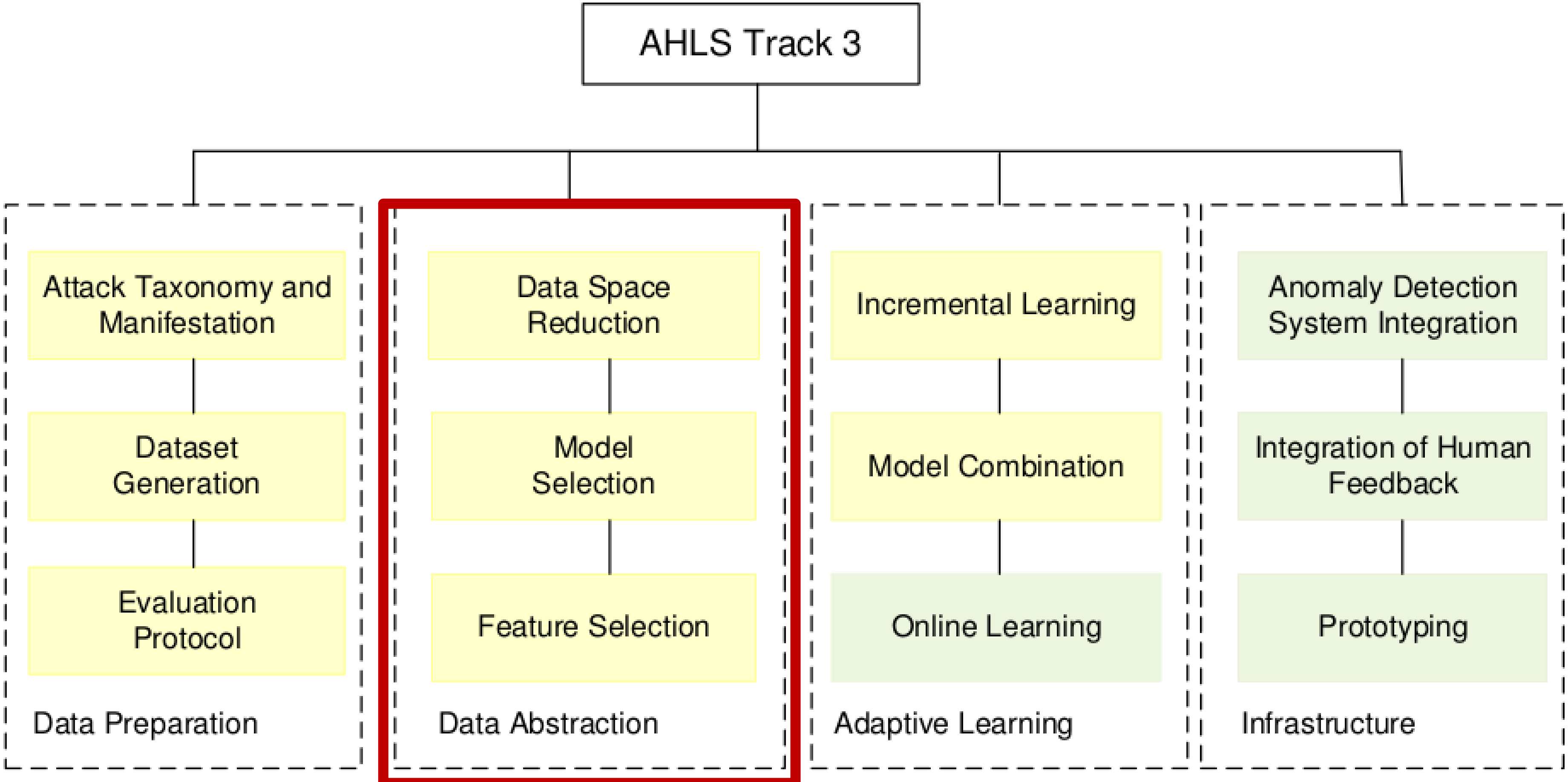
Objectives of Concordia Research Thread

- Protect **host systems against cyber-attacks**
- Develop **modular, adaptive,** and **scalable** Anomaly Detection Systems (ADS) at the **system call level**
- Reduce **false positives (alarms)** and improve the **true positives**
- Develop comprehensive **test beds** and **evaluation protocols**
- Provide preliminary analysis/recommendations for future research and directions

Advanced Host-Level Surveillance



Advanced Host-Level Surveillance



Kernel State Modeling (KSM)

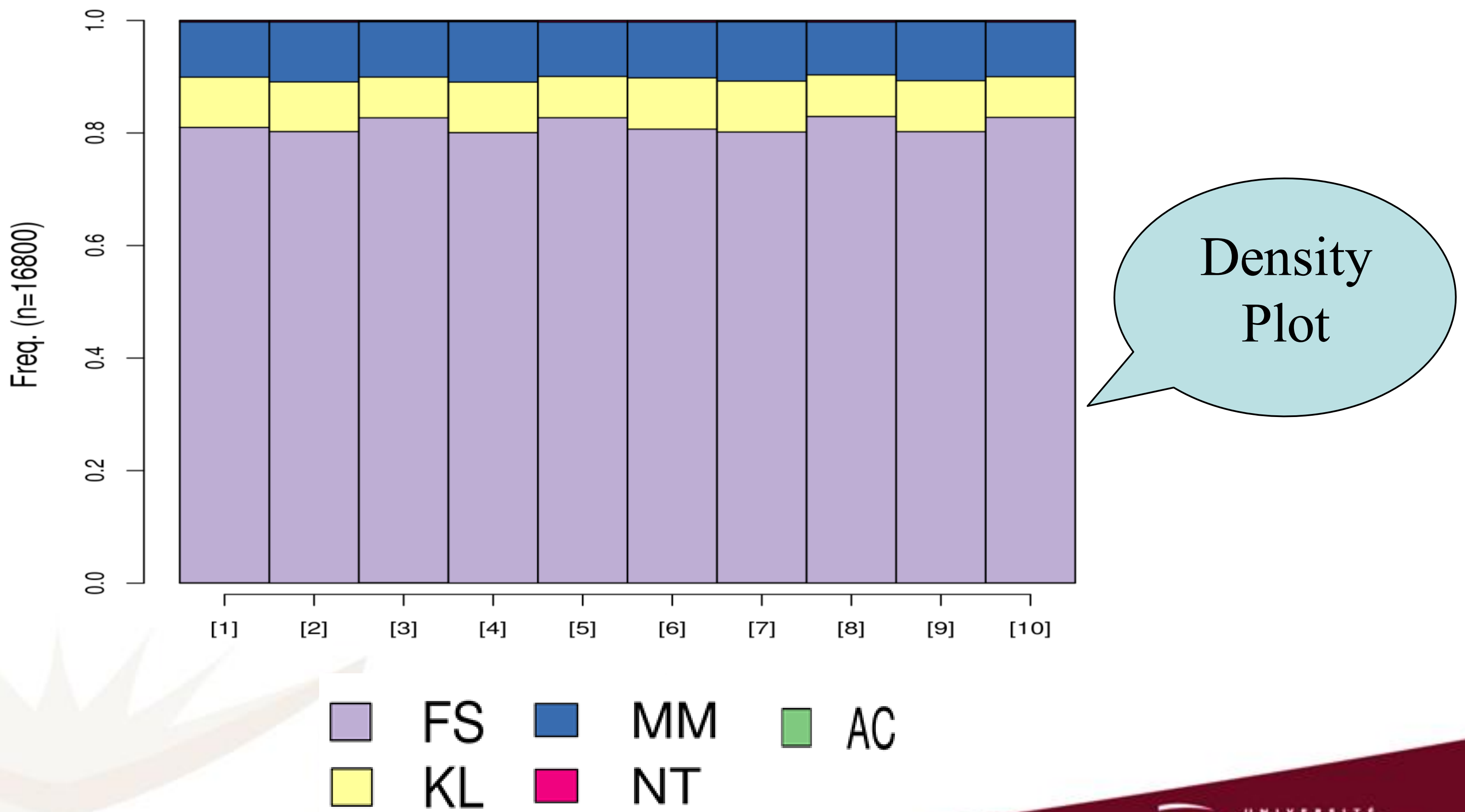
- KSM is an anomaly detection technique
  - Transforms system calls into kernel modules, called states
  - Detect anomalies at the level of interaction of kernel states
  - Reduces data space used in training and testing
  - Favors efficiency while keeping accuracy

Transforming System Calls into States of Kernel Modules

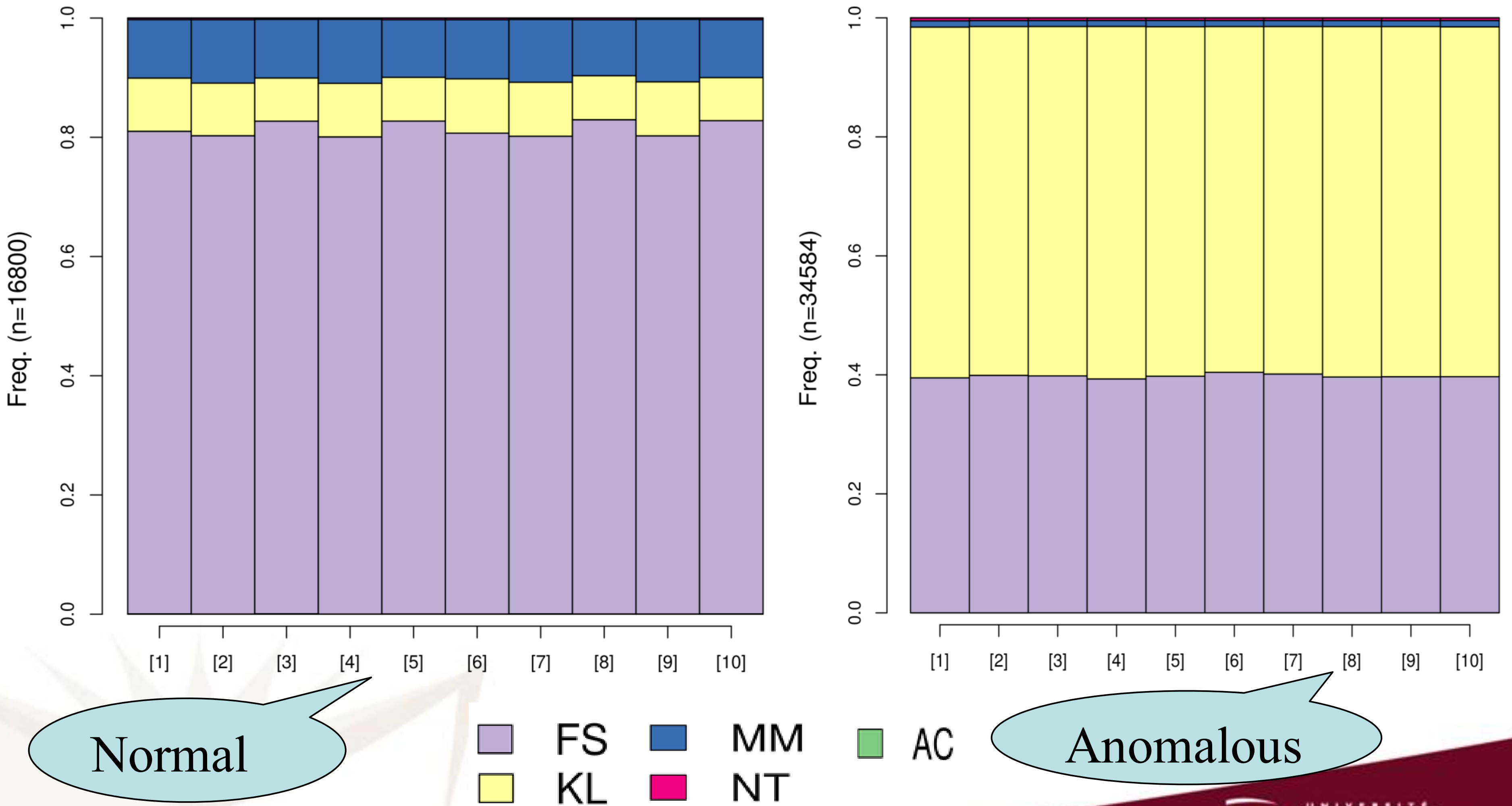
State	Module in Linux Source Code	# of System Calls
AC	Architecture	10
FS	File System	131
IPC	Inter Process Communication	7
KL	Kernel	127
MM	Memory Management	21
NT	Networking	2
SC	Security	3
UN	Unknown	37

[Source]: http://syscalls.kernelgork.com

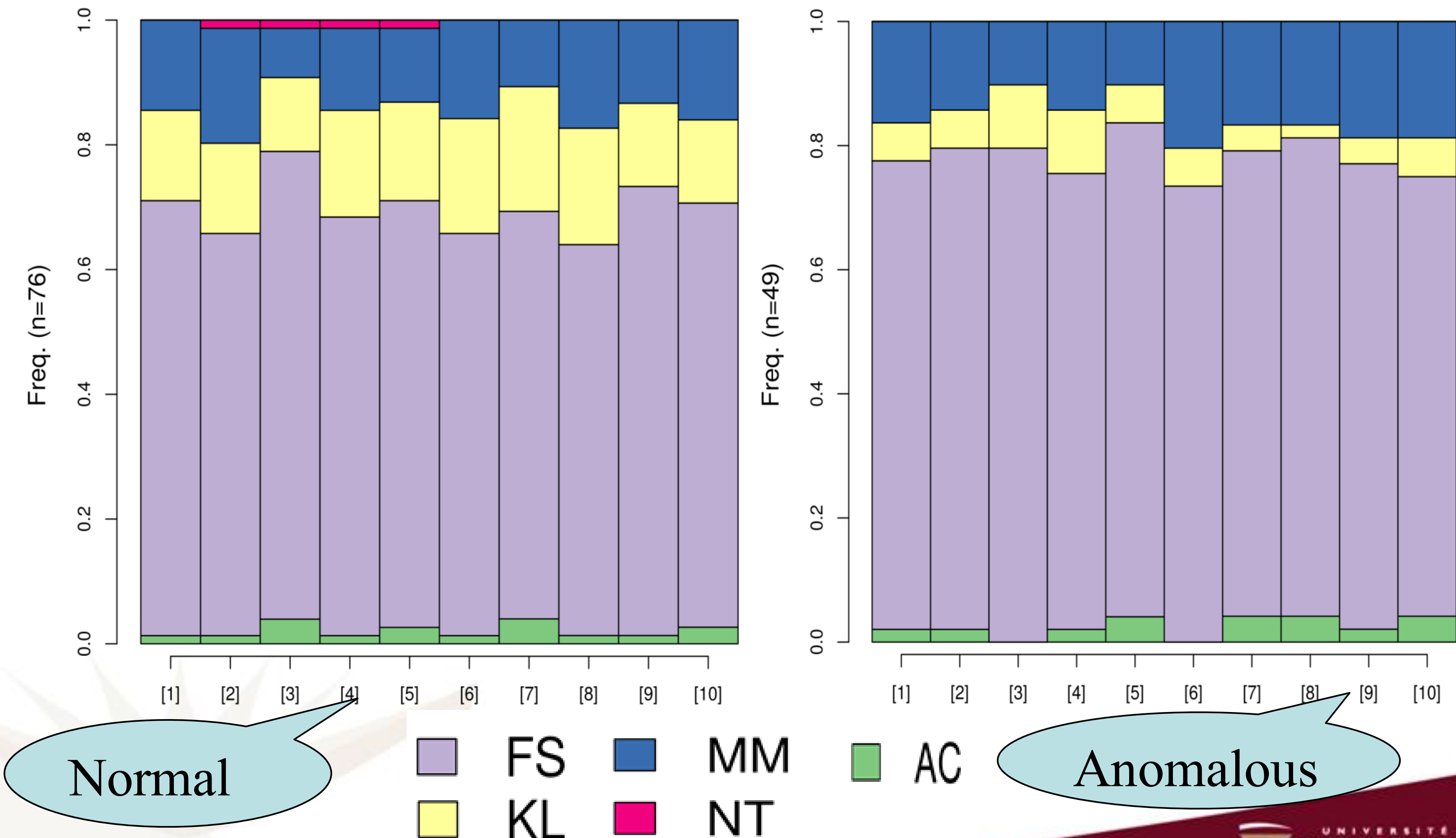
KSM and Density Plots



Anomaly Detection in Firefox



Anomaly Detection in Login Utility



Automatically Detecting Anomalies

Trace #	FS	KL	MM	Type
1	0.60	0.20	0.00	Normal
2	0.54	0.06	0.40	Normal
3	0.73	0.04	0.23	Normal
4	0.74	0.05	0.03	Normal
5	0.82	0.01	0.03	Normal
6	0.82	0.03	0.11	Normal
7	0.55	0.15	0.19	Anomalous
8	0.53	0.16	0.20	Anomalous

Xlock
Program

Automatically Detecting Anomalies

- To determine significant deviation threshold (alpha):
  - Divide normal dataset into training set, validation set, and testing set
  - Extract probabilities from training set
  - Evaluate on validation set and adjust alpha
  - Measure accuracy on testing set

Case Study 1: Dataset

Program	# Normal Traces			#Attack Types	#Attack Traces
	Training	Validation	Testing		
Login	4	3	5	1	4
PS	10	4	10	1	15
Stide	400	200	13126	1	105
Xlock	91	30	1610	1	2
Firefox	125	75	500	5	19

Case Study 1: Results

Program	Technique	TP rate	FP rate
Login	KSM (alpha=0.00)	100%	0.00%
	Stide (win=6)	100%	40.00%
	Stide (win=10)	100%	40.00%
	HMM (states=10)	100%	40.00%
PS	KSM (alpha=0.02)	100%	10.00%
	Stide (win=6)	100%	10.00%
	Stide (win=10)	100%	10.00%
	HMM (states=5)	100%	30.00%
Xlock	KSM (alpha=0.04)	100%	0.00%
	Stide (win=6)	100%	1.50%
	Stide (win=10)	100%	1.50%
	HMM (states=5)	100%	0.00%

Université Concordia University

Case Study 1: Results

Program	Technique	TP rate	FP rate
Stide	KSM (alpha=0.06)	100%	0.25%
	Stide (win=6)	100%	4.97%
	Stide (win=10)	100%	5.25%
	HMM (states=5)	100%	0.25%
Firefox	KSM (alpha=0.08)	100%	0.60%
	Stide (win=6)	100%	44.60%
	Stide (win=10)	100%	49.20%
	HMM (states=5)	100%	1.40%

$$TP = \frac{Number\ of\ detected\ attacks\ (anomalies)}{Total\ number\ of\ attacks\ (anomalies)} \times 100$$

Equation 1. True positive rate

$$FP = \frac{Number\ of\ normal\ traces\ detected\ as\ anomalous}{Total\ number\ of\ normal\ traces} \times 100$$

Case Study 1: Execution Time

	Size of All Traces	KSM	Stide	HMM
Login	26.2KB	4.46 sec	0.03 sec	56.43 min
PS	29.6KB	5.14 sec	0.11 sec	46.24 min
Xlock	47.4MB	1.51 min	12.3 min	13.37 hr
Stide	36.2MB	5.85 min	8.53 min	2.3 day
Firefox	270.6MB	9.35 min	4.17 hr	4.03 day

Case Study 2: ADFA Linux Dataset

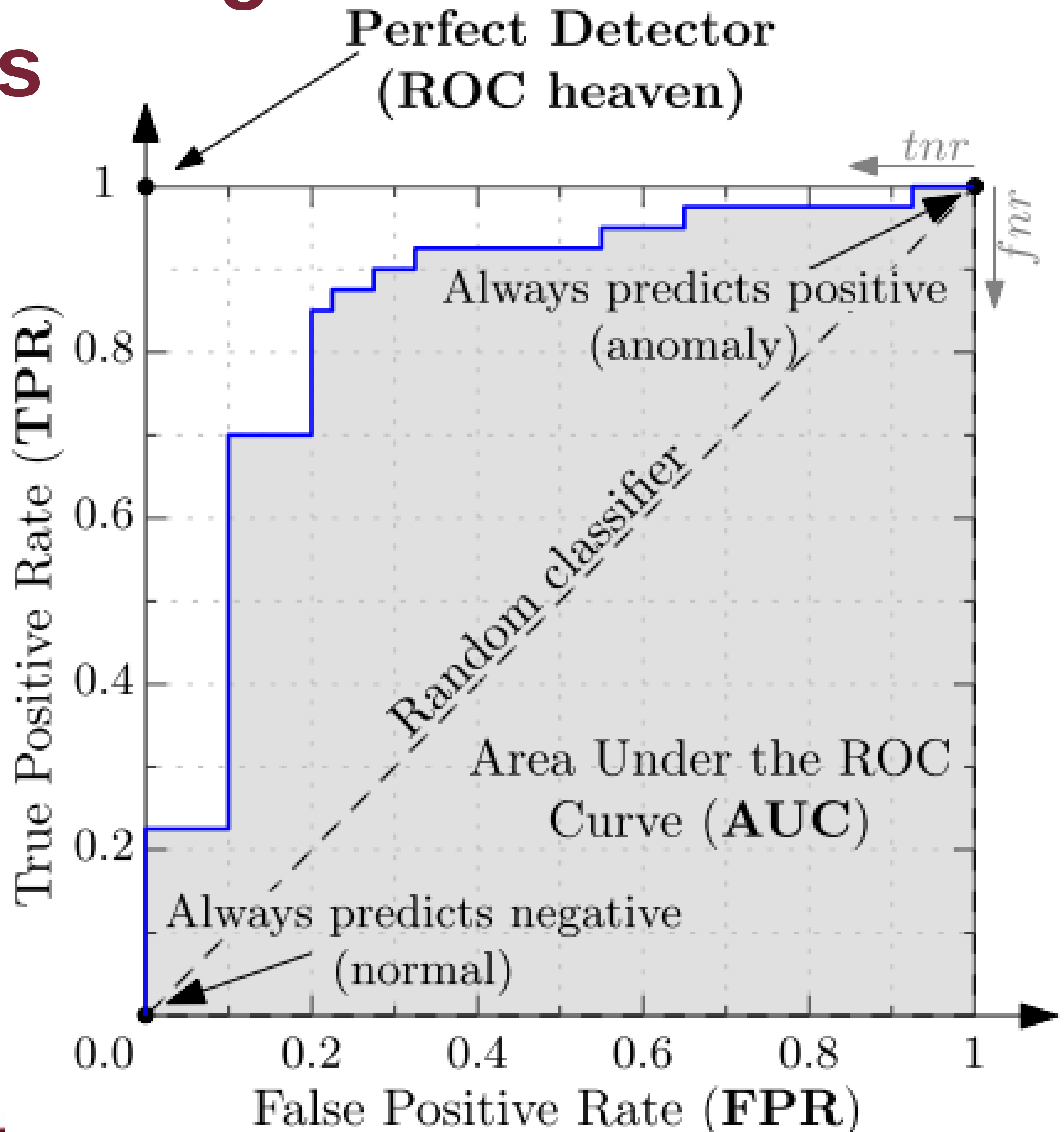
- A host with Ubuntu 11.04, Apache 2.2.17, PHP 5.3.5, TikiWiki 8.1, FTP server, MySQL 14.14 and an SSH server
  - web-based exploitation
  - simulated social engineering
  - poisoned executable,
  - remotely triggered vulnerabilities,
  - remote password brute force attacks
  - system manipulation

Case Study 2: ADFA Linux Dataset

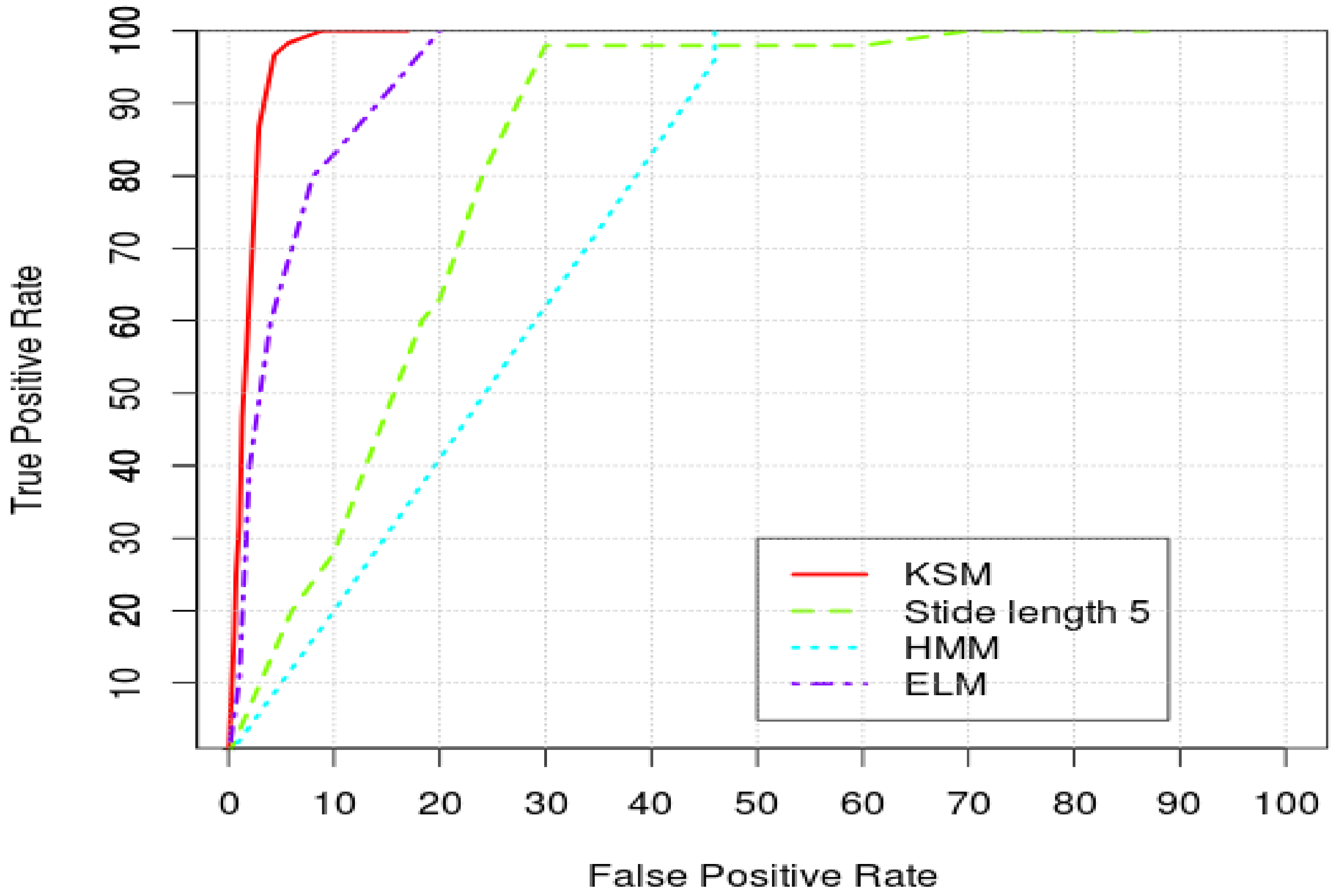
Training Set	
# of training traces	**833**
Validation Set	
# of attacks	**20**
# of normal traces	**1000**
Testing Set	
# of attacks	**40**
# of normal traces	**3373**

Receiver Operating Characteristics (ROC) Curves

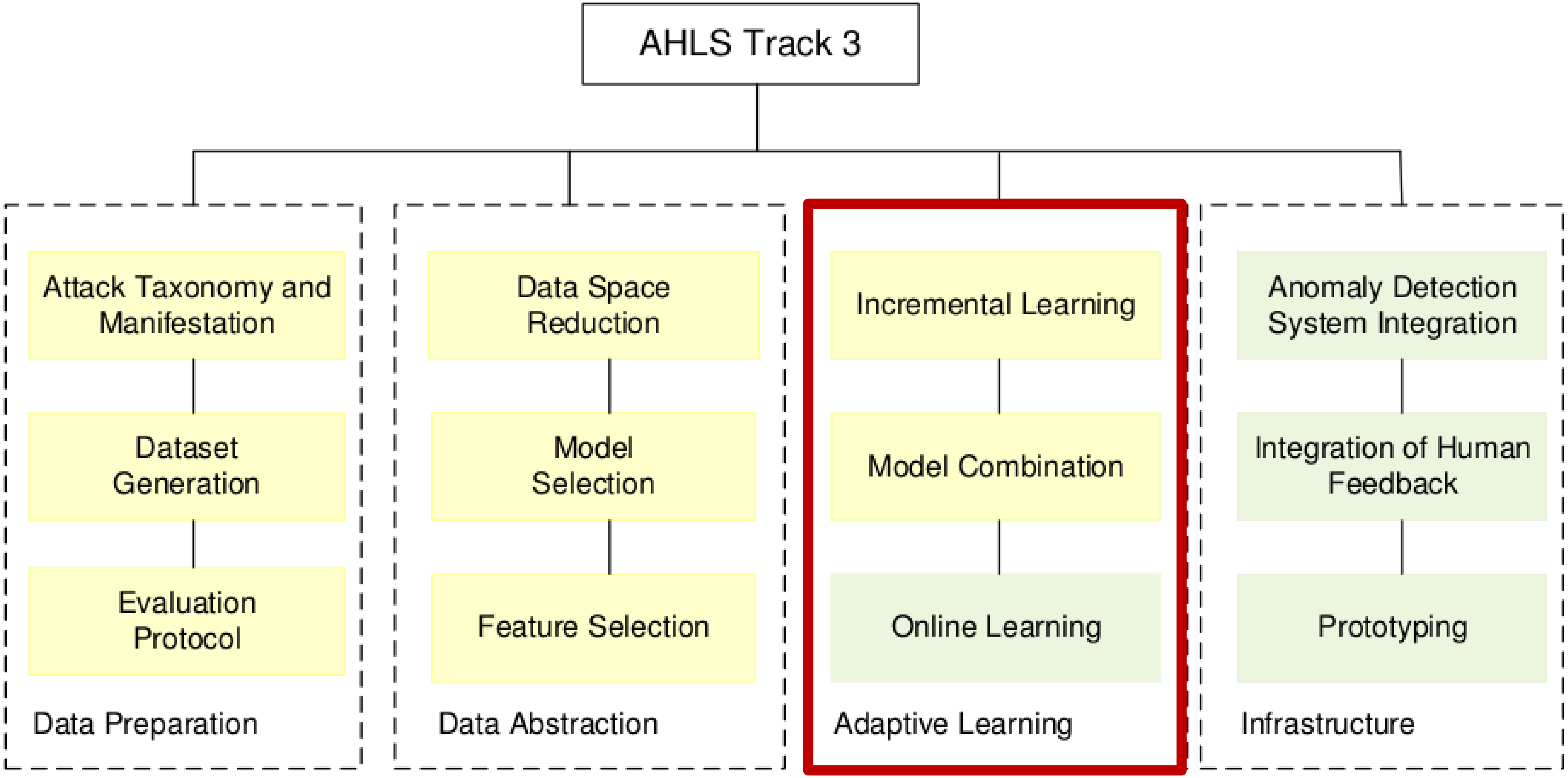
- True Positive: anomaly detected as anomaly
- False Positive: normal detected as anomaly



Case Study 2: ADFA Linux Dataset



Research Threads



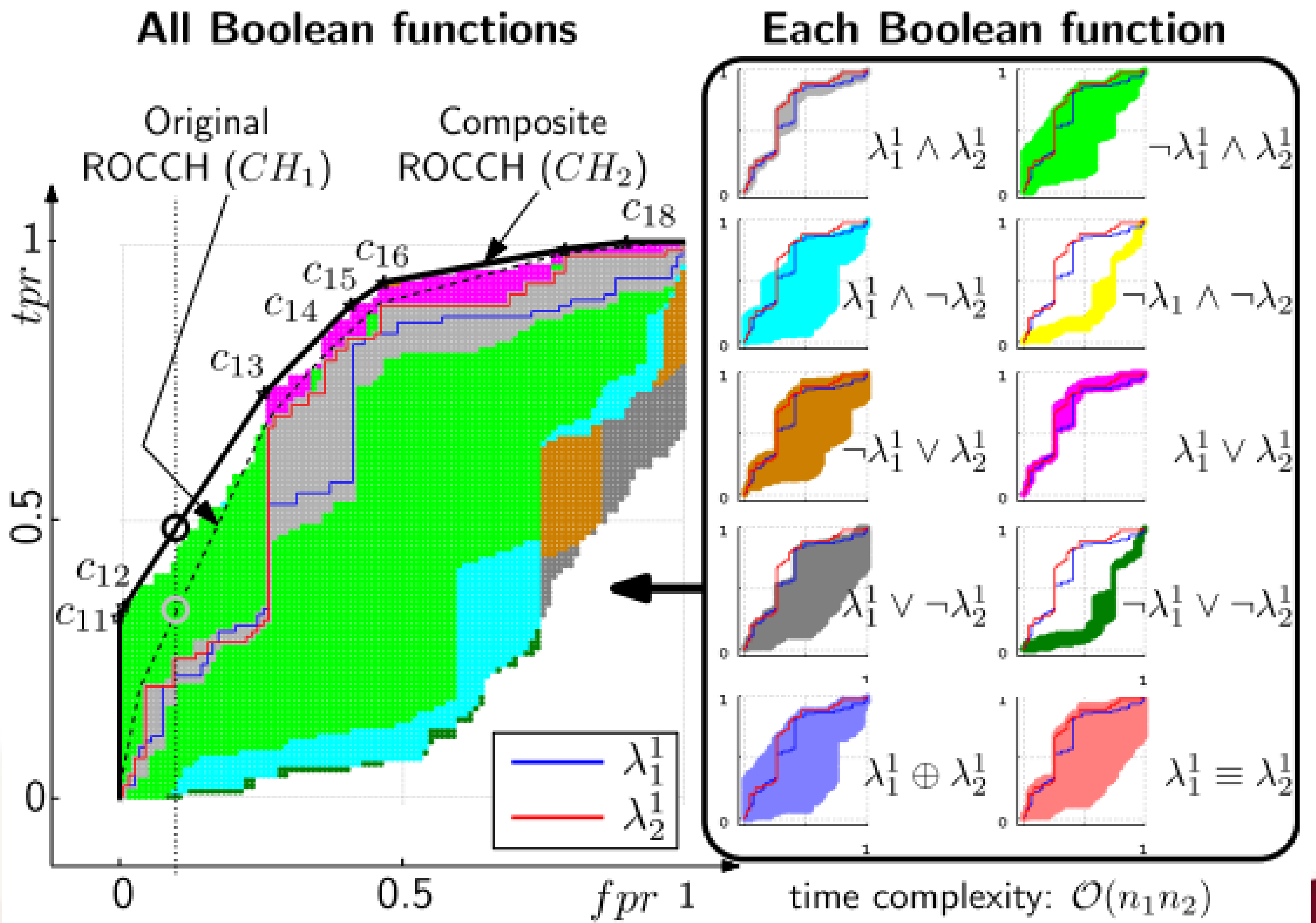
Model Combination

- A single classifier or model may not provide a good approximation to the underlying data structure or distribution
  - No dominant classifier for all data distributions (“no free lunch” theorem)
  - True data distribution is usually unknown
  - Limited amount of (labeled) data is typically provided during training

IBC: Iterative Boolean Combination in the ROC Space

- For each threshold from the first detector and each threshold from the second detector:
  - Combine the responses using all Boolean functions
  - Select thresholds and Boolean functions that improve the ROC space

IBC - Example

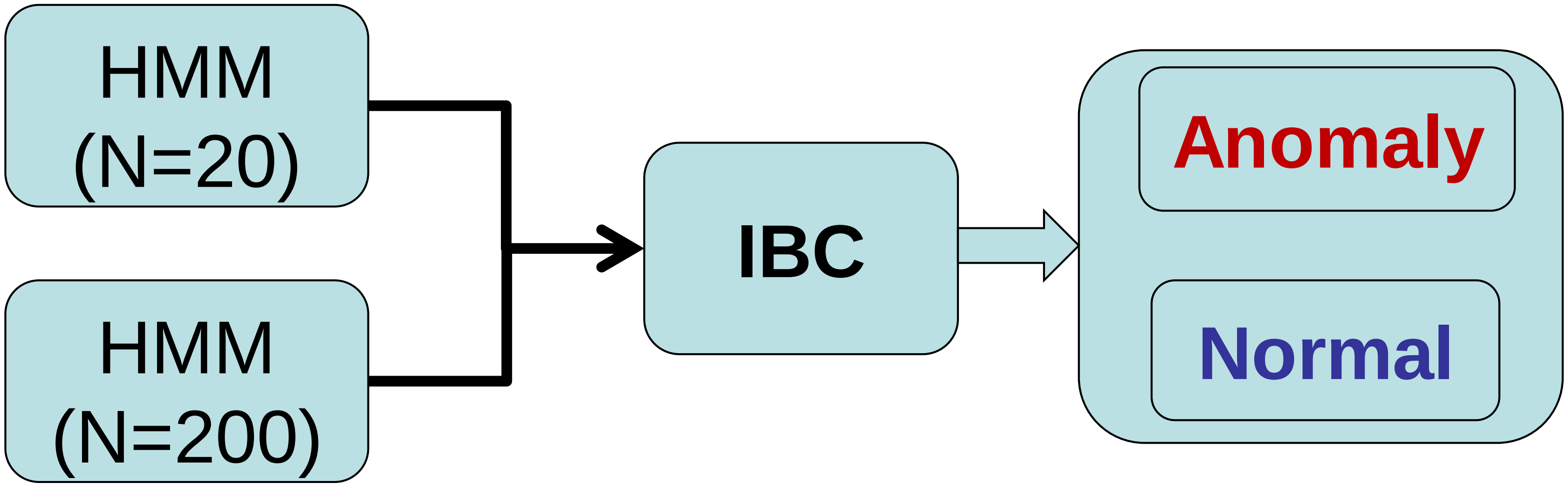


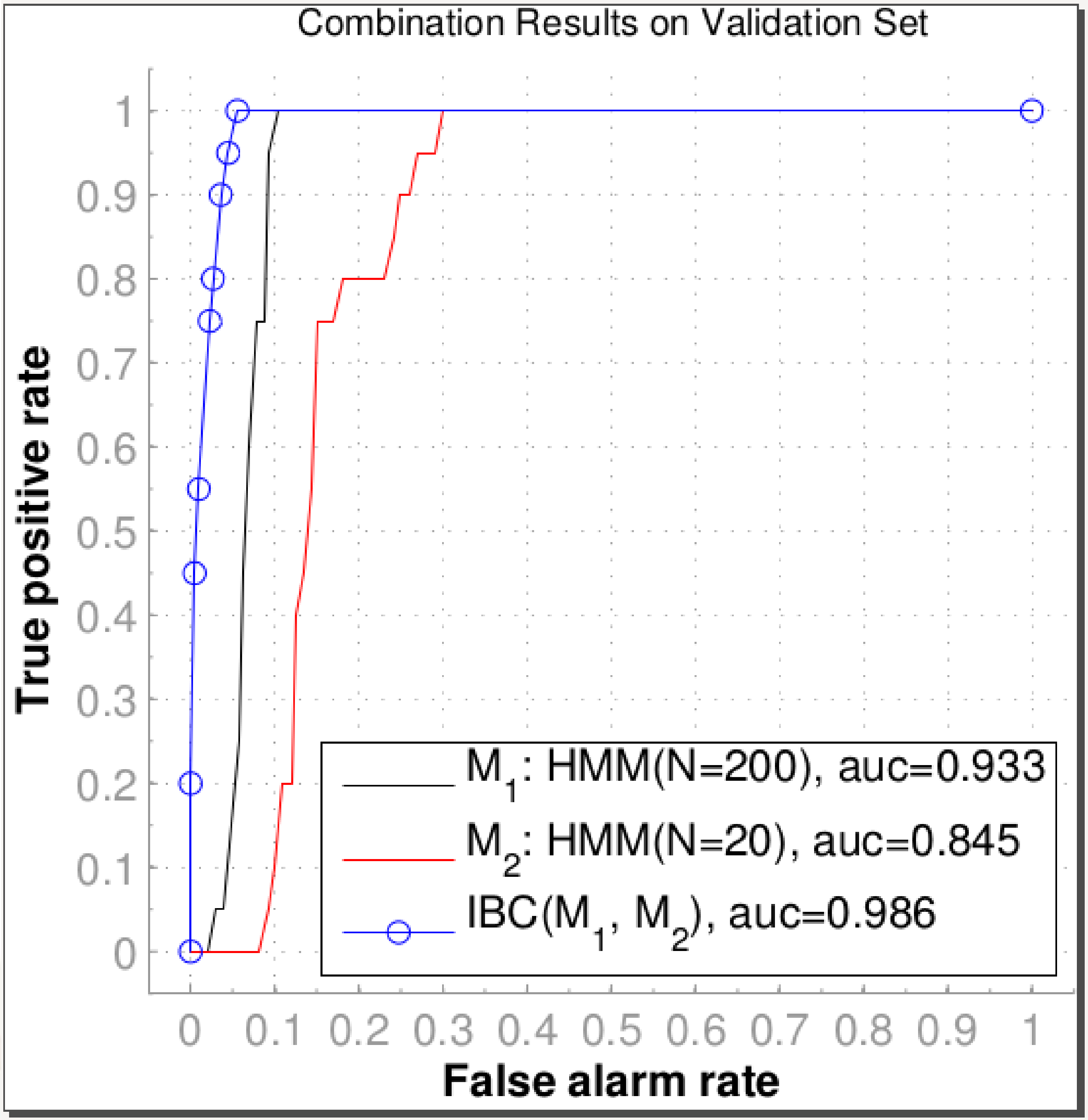
All Boolean functions
Each Boolean function
Original ROCCH (CH_1)
Composite ROCCH (CH_2)
c_{11}
c_{12}
c_{13}
c_{14}
c_{15}
c_{16}
c_{18}
tpr
fpr
0
0.5
1
λ_1^1
λ_2^1
$\lambda_1^1 \wedge \lambda_2^1$
$\neg\lambda_1^1 \wedge \lambda_2^1$
$\lambda_1^1 \wedge \neg\lambda_2^1$
$\neg\lambda_1 \wedge \neg\lambda_2$
$\neg\lambda_1^1 \vee \lambda_2^1$
$\lambda_1^1 \vee \lambda_2^1$
$\lambda_1^1 \vee \neg\lambda_2^1$
$\neg\lambda_1^1 \vee \neg\lambda_2^1$
$\lambda_1^1 \oplus \lambda_2^1$
$\lambda_1^1 \equiv \lambda_2^1$
time complexity: $\mathcal{O}(n_1 n_2)$

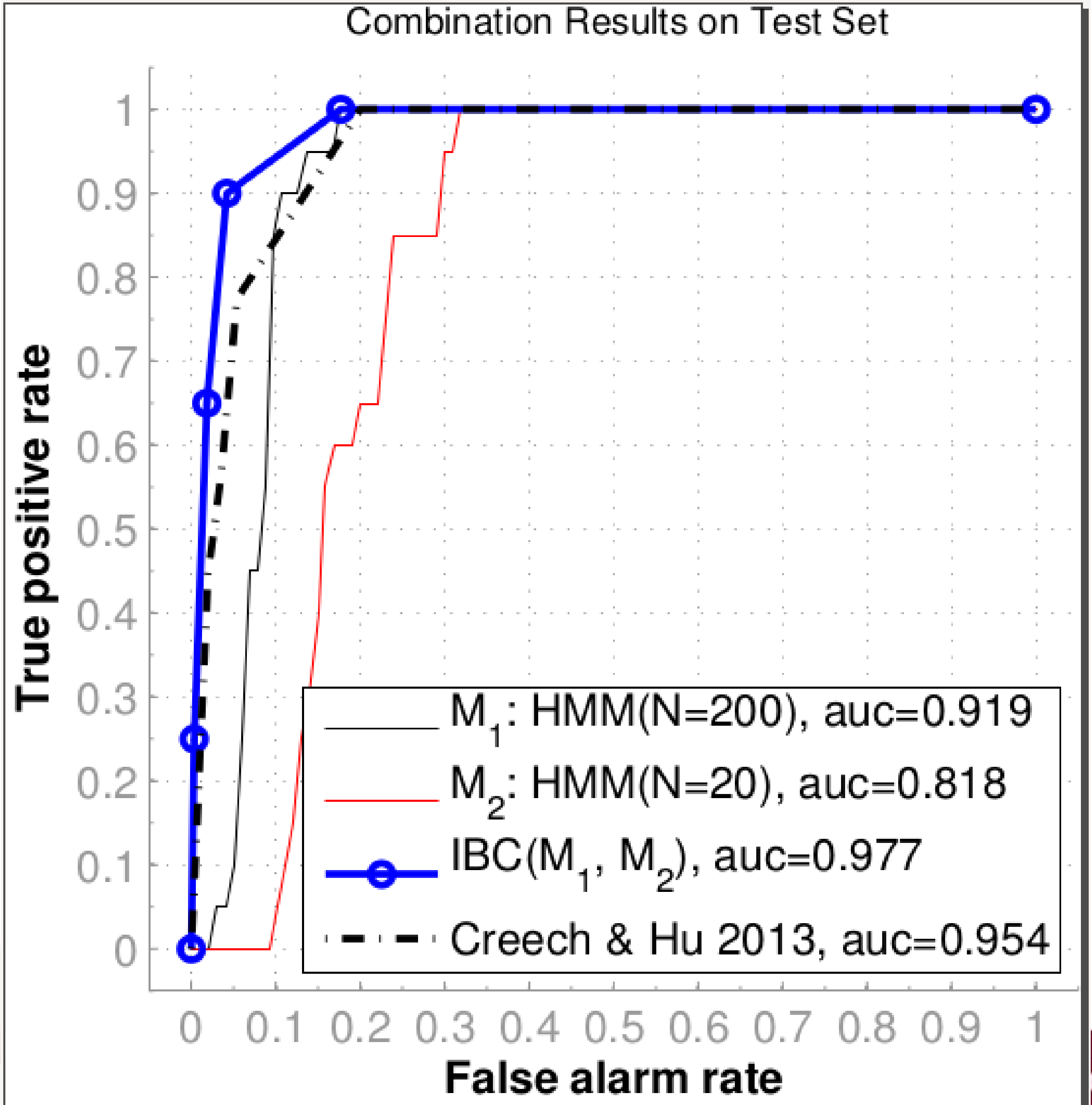
Experimental Methodology

Training Set	
# of training traces	**833**
Validation Set	
# of attacks	**20**
# of normal traces	**1000**
Testing Set	
# of attacks	**40**
# of normal traces	**3373**

Combination of Responses from Different HMMs

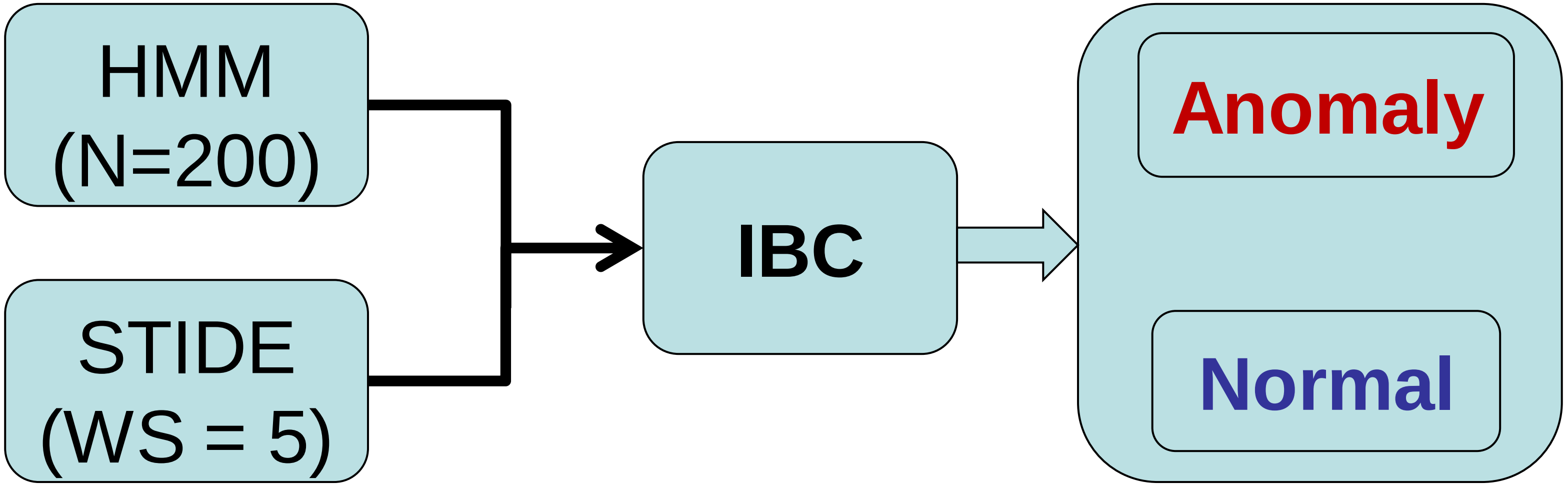


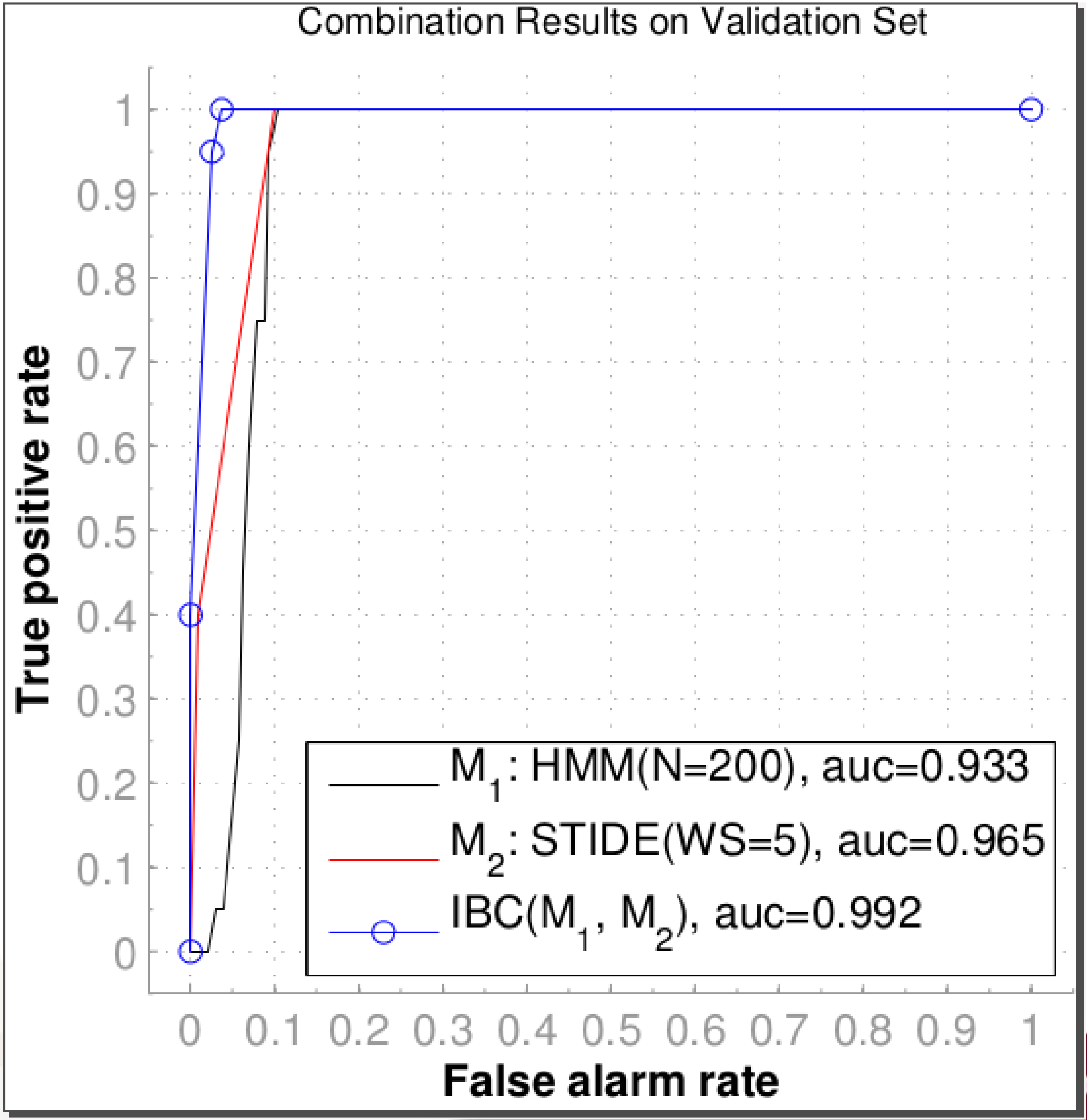

Combination Results on Validation Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M1: HMM(N=200), auc=0.933
M2: HMM(N=20), auc=0.845
IBC(M1, M2), auc=0.986

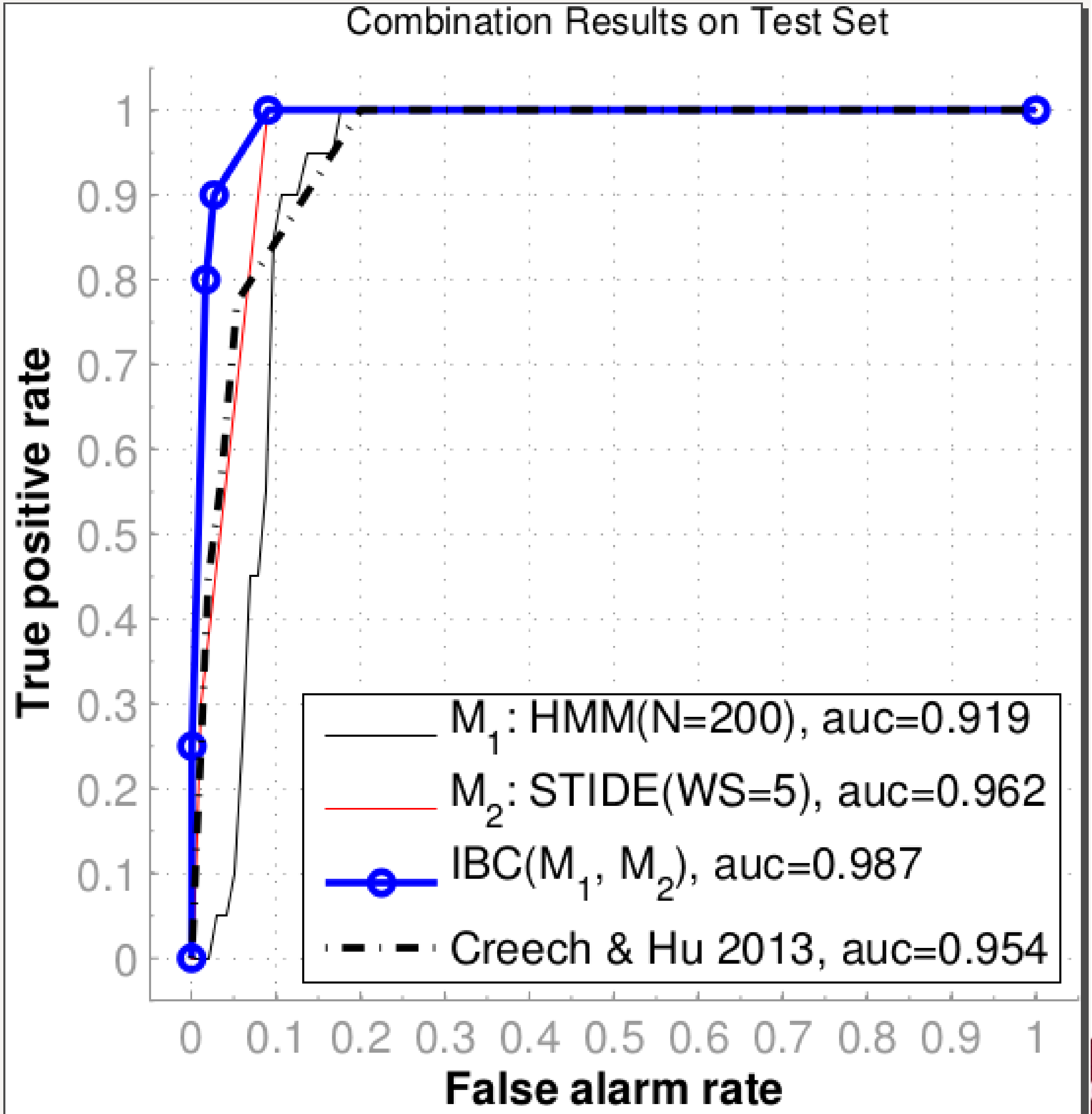


Combination Results on Test Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M1: HMM(N=200), auc=0.919
M2: HMM(N=20), auc=0.818
IBC(M1, M2), auc=0.977
Creech & Hu 2013, auc=0.954

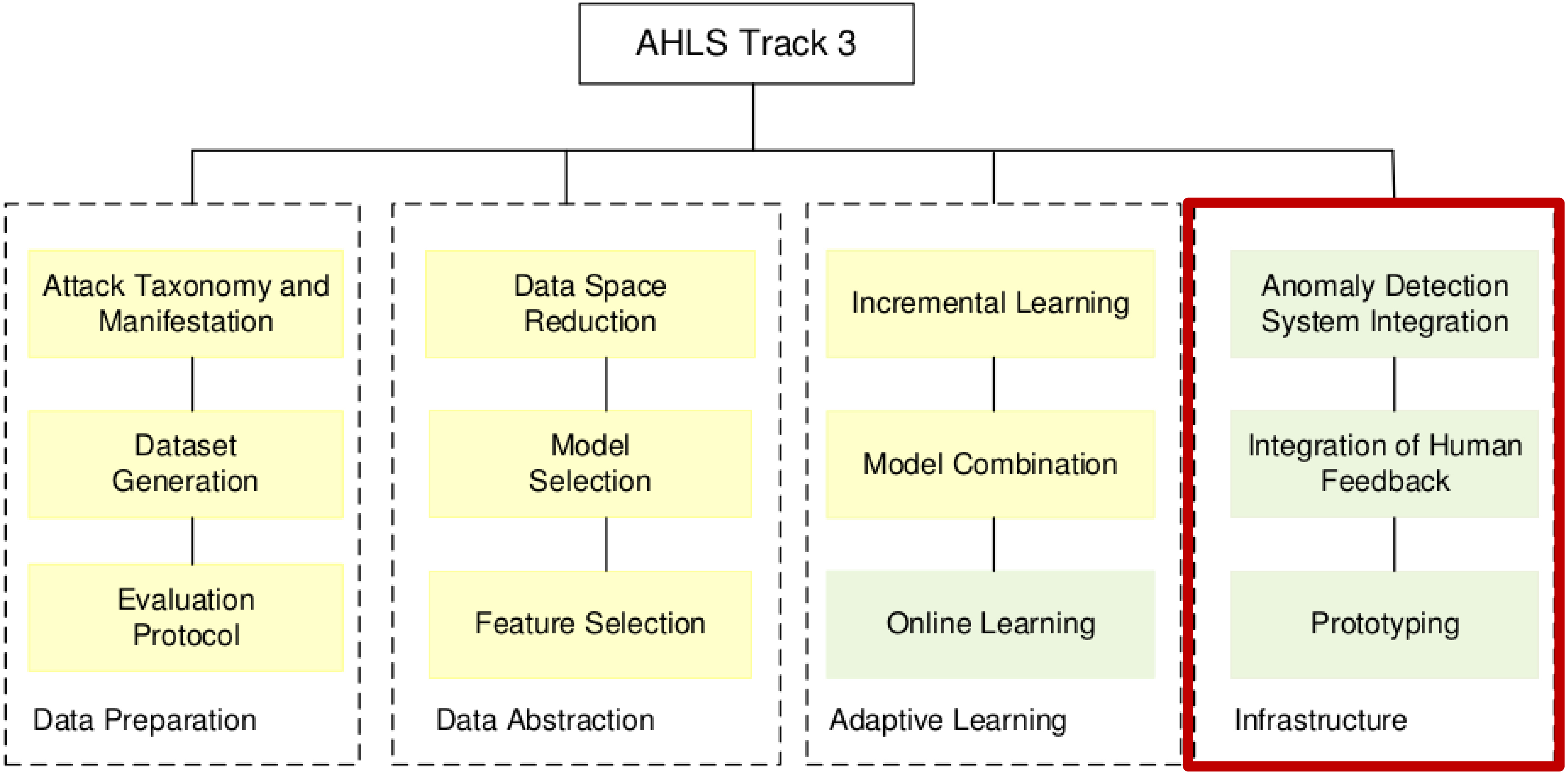
Combination of HMM and STIDE Responses




Combination Results on Validation Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M_1: HMM(N=200), auc=0.933
M_2: STIDE(WS=5), auc=0.965
IBC(M_1, M_2), auc=0.992


Combination Results on Test Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M1: HMM(N=200), auc=0.919
M2: STIDE(WS=5), auc=0.962
IBC(M1, M2), auc=0.987
Creech & Hu 2013, auc=0.954

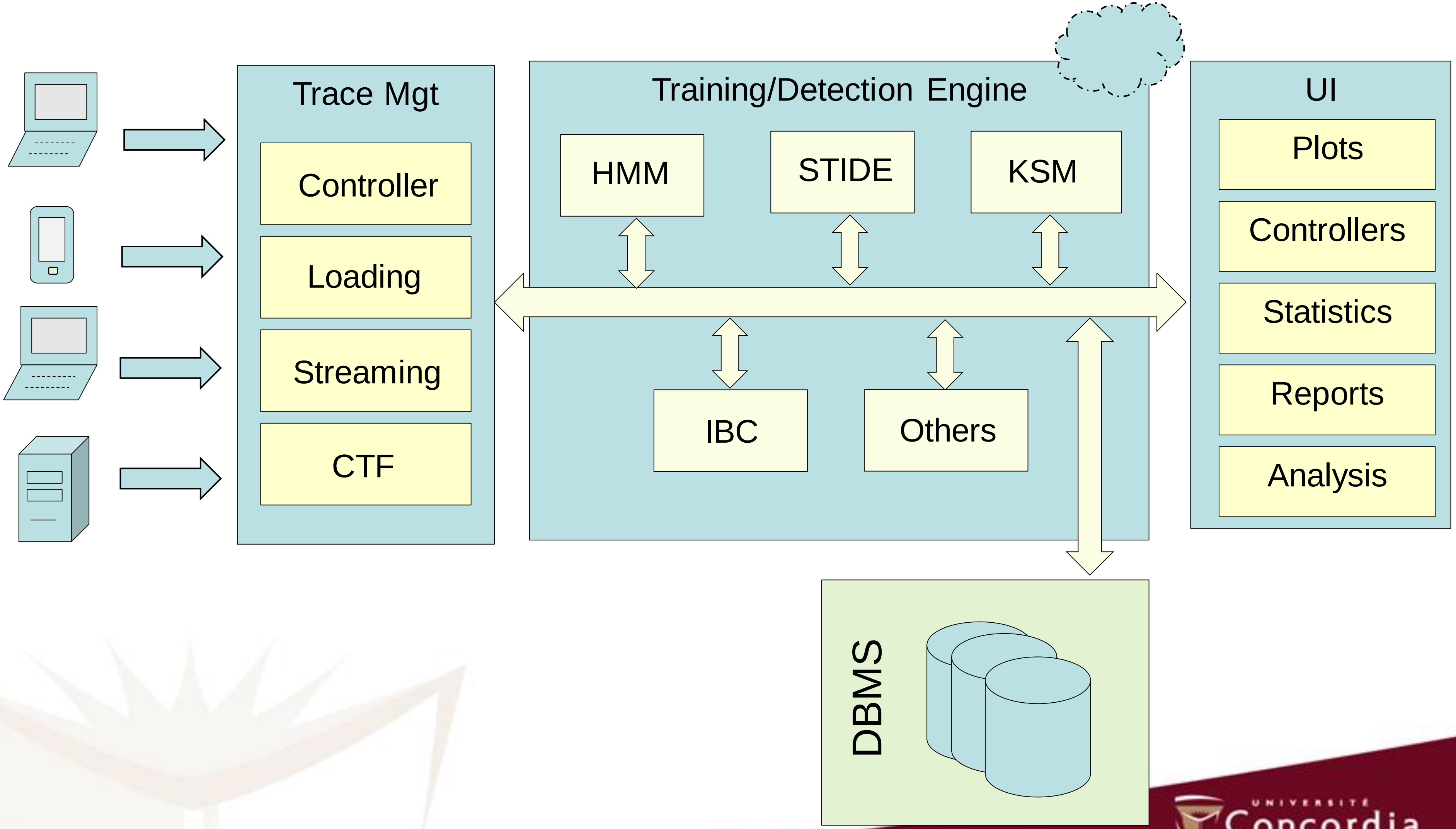
Research Threads

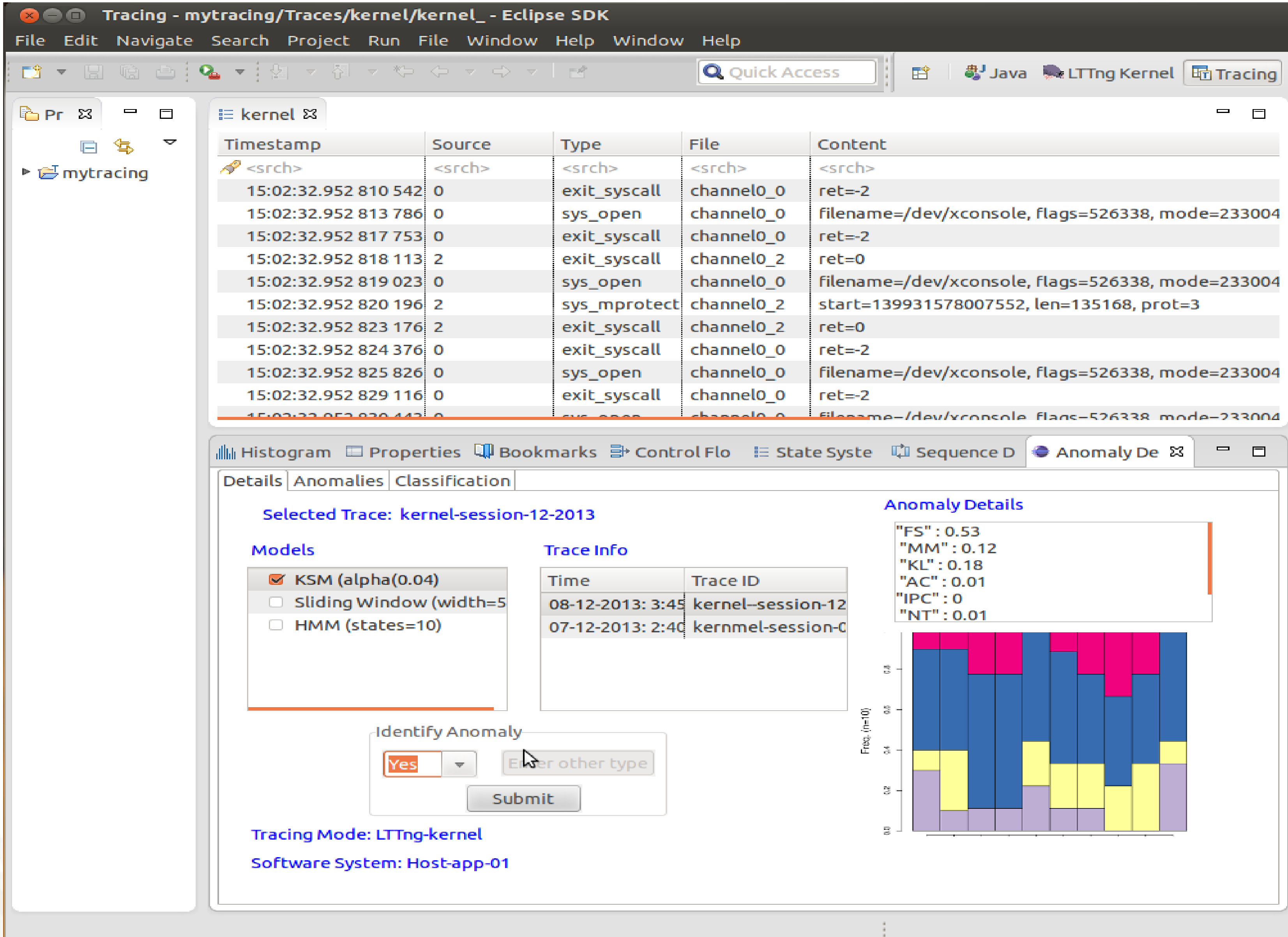


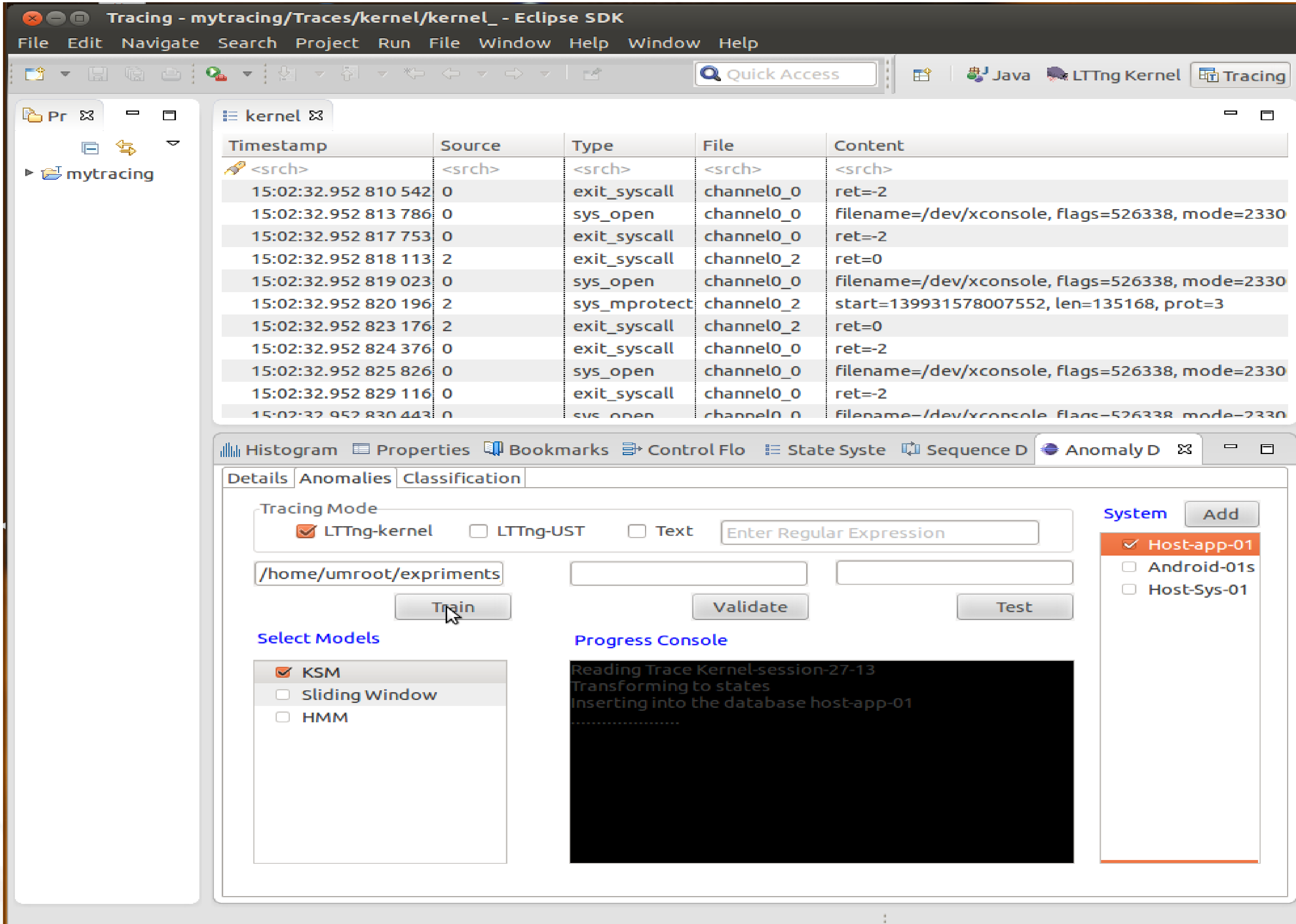
TotalADS

- TotalADS is an integrated Anomaly Detection System Environment
  - Eclipse Plug-in,
  - Open Source
  - Based on TMF (Tracing and Monitoring Framework)
  - Supports STIDE, HMM, KSM, IBC
  - Supports a combination of classifiers
  - Supports trace analysis and forensic analysis
  - Supports CTF (Common Trace Format)

Architecture




Tracing - mytracing/Traces/kernel/kernel_ - Eclipse SDK
File Edit Navigate Search Project Run File Window Help Window Help
Quick Access
Java
LTTng Kernel
Tracing
Pr
mytracing
kernel
Timestamp Source Type File Content
15:02:32.952 810 542 0 exit_syscall channel0_0 ret=-2
15:02:32.952 813 786 0 sys_open channel0_0 filename=/dev/xconsole, flags=526338, mode=233004
15:02:32.952 817 753 0 exit_syscall channel0_0 ret=-2
15:02:32.952 818 113 2 exit_syscall channel0_2 ret=0
15:02:32.952 819 023 0 sys_open channel0_0 filename=/dev/xconsole, flags=526338, mode=233004
15:02:32.952 820 196 2 sys_mprotect channel0_2 start=139931578007552, len=135168, prot=3
15:02:32.952 823 176 2 exit_syscall channel0_2 ret=0
15:02:32.952 824 376 0 exit_syscall channel0_0 ret=-2
15:02:32.952 825 826 0 sys_open channel0_0 filename=/dev/xconsole, flags=526338, mode=233004
15:02:32.952 829 116 0 exit_syscall channel0_0 ret=-2
Histogram Properties Bookmarks Control Flo State Syste Sequence D Anomaly De
Details Anomalies Classification
Selected Trace: kernel-session-12-2013
Models
KSM (alpha(0.04)
Sliding Window (width=5
HMM (states=10)
Trace Info
Time Trace ID
08-12-2013: 3:45 kernel--session-12
07-12-2013: 2:40 kernmel-session-0
Identify Anomaly
Yes
Submit
Tracing Mode: LTTng-kernel
Software System: Host-app-01
Anomaly Details
"FS" : 0.53
"MM" : 0.12
"KL" : 0.18
"AC" : 0.01
"IPC" : 0
"NT" : 0.01


Tracing - mytracing/Traces/kernel/kernel_ - Eclipse SDK
File Edit Navigate Search Project Run File Window Help Window Help
Quick Access
Java
LTTng Kernel
Tracing
Pr
mytracing
kernel
Timestamp
Source
Type
File
Content
<srch>
15:02:32.952 810 542 0 exit_syscall channel0_0 ret=-2
15:02:32.952 813 786 0 sys_open channel0_0 filename=/dev/xconsole, flags=526338, mode=2330
15:02:32.952 817 753 0 exit_syscall channel0_0 ret=-2
15:02:32.952 818 113 2 exit_syscall channel0_2 ret=0
15:02:32.952 819 023 0 sys_open channel0_0 filename=/dev/xconsole, flags=526338, mode=2330
15:02:32.952 820 196 2 sys_mprotect channel0_2 start=139931578007552, len=135168, prot=3
15:02:32.952 823 176 2 exit_syscall channel0_2 ret=0
15:02:32.952 824 376 0 exit_syscall channel0_0 ret=-2
15:02:32.952 825 826 0 sys_open channel0_0 filename=/dev/xconsole, flags=526338, mode=2330
15:02:32.952 829 116 0 exit_syscall channel0_0 ret=-2
Histogram
Properties
Bookmarks
Control Flo
State Syste
Sequence D
Anomaly D
Details
Anomalies
Classification
Tracing Mode
LTTng-kernel
LTTng-UST
Text
Enter Regular Expression
/home/umroot/expriments
Train
Validate
Test
Select Models
KSM
Sliding Window
HMM
Progress Console
Reading Trace Kernel-session-27-13
Transforming to states
Inserting into the database host-app-01
System
Add
Host-app-01
Android-01s
Host-Sys-01

Future Plans

- Continue experimenting with KSM and IBC on other datasets (preferably generated at DRDC)
- Combine additional detectors using IBC
- Start working on adaptive/incremental learning
- Continue improving the maturity level of TotalADS
- Integrate this work with work done at other universities
- Transfer knowledge to DRDC & Ericsson

Thank You

Wahab Hamou-Lhadj, PhD, ing.

Software Behaviour Analysis (SBA) Research Lab
Concordia University
Montreal, QC, Canada

www.ece.concordia.ca/~abdelws/sba