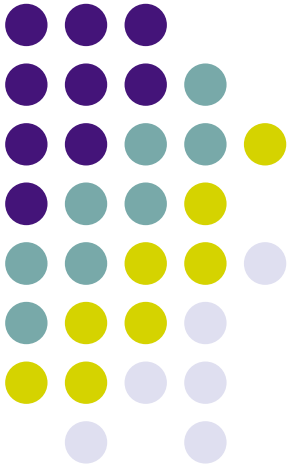
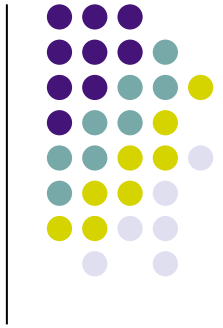
Adaptive Anomaly Detection via Self-Calibration and Dynamic Updating

Gabriela F. Cretu-Ciocarlie
Department of Computer Science
Columbia University

Joint work with Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo



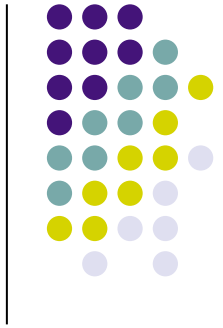
Motivation and Problem



- Current attack methods, such as polymorphic engines, will overwhelm signature-based anomaly detectors [Song07, Crandall05]
- Relying on anomaly-based detection (AD) sensors to detect 0-day attacks has become a necessity

BUT….

Motivation and Problem

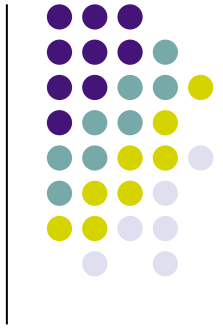


- Current attack methods, such as polymorphic engines, will overwhelm signature-based anomaly detectors [Song07, Crandall05]
- Relying on anomaly-based detection (AD) sensors to detect 0-day attacks has become a necessity

BUT….

- There is a **major hurdle** in the deployment, operation, and maintenance of AD systems
  - Calibrate them
  - Update their models when changes appear in the protected system

Contributions

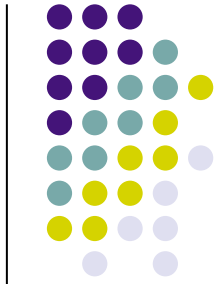


- Identifying the intrinsic characteristics of the training data (*i.e.* **self-calibration**)

- Cleansing a data set of attacks and abnormalities by automatically selecting an adaptive threshold for the voting (*i.e.* **automatic self-sanitization**)

- Maintaining the performance we gained by applying the sanitization methods beyond the initial training phase and extending them throughout the lifetime of the sensor (*i.e.* **self-update**)

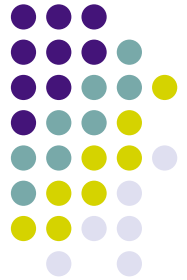
Training Dataset Sanitization



- Attacks and accidental malformed requests/data cause a local "pollution“ of training data
  - An attack can pass as normal traffic if it is part of the training set

- We seek to remove both malicious and abnormal data from the training dataset

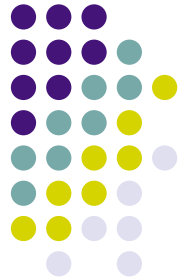
Training Strategies



- Divide data into multiple blocks
  - automatic selection of the **optimal time granularity**

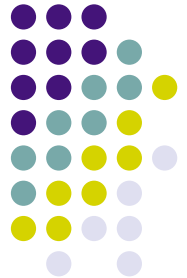
$$T = \{md_1, \ldots md_N\}$$

Time Granularity Characteristics



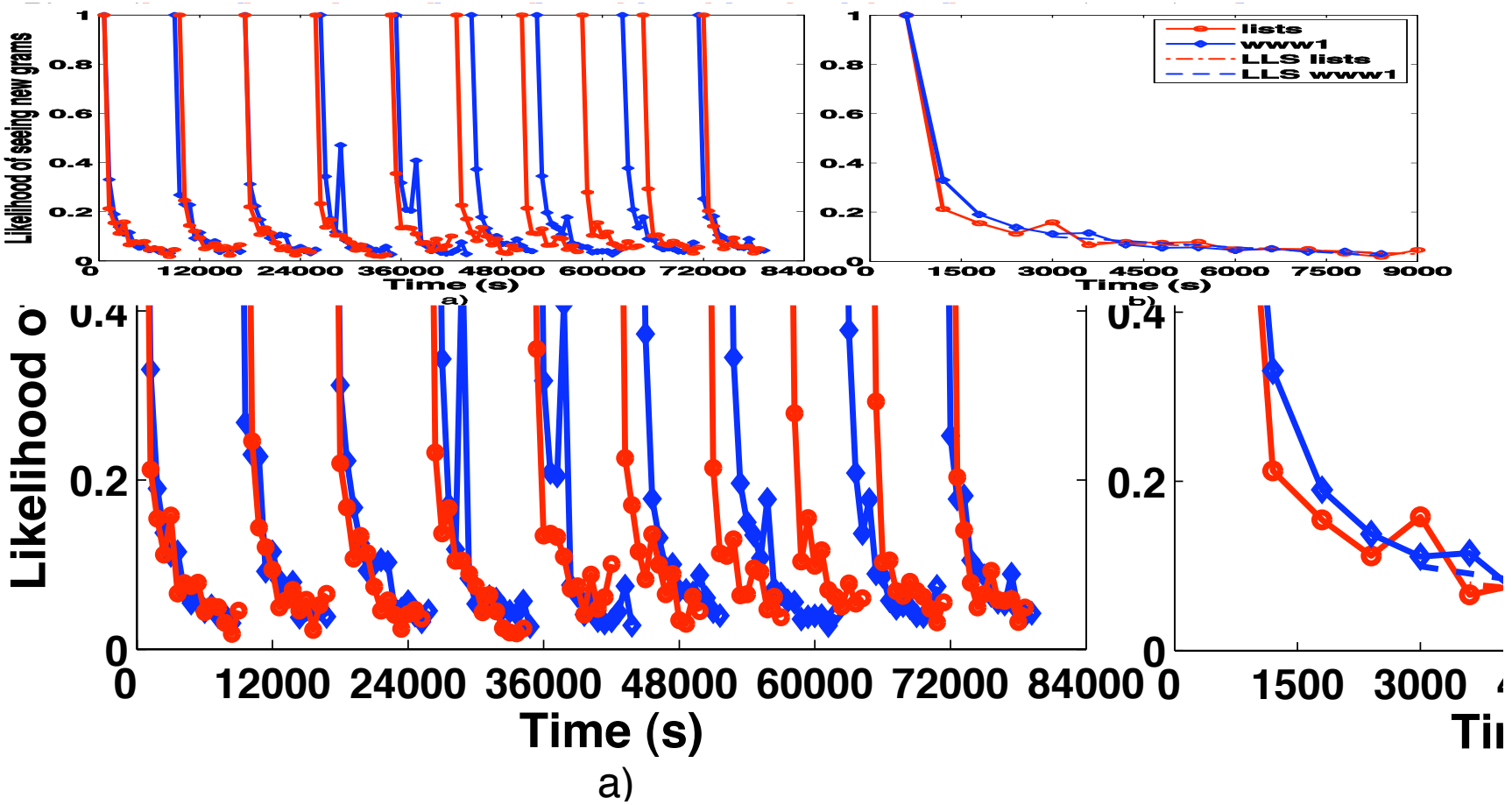
- Smaller value of the time granularity g = > confines the effect of an individual attack to a smaller neighborhood of micro-models
- Excessively small values can lead to *under-trained* models
- Automatically determine when a model is **stable**

How to Determine g?

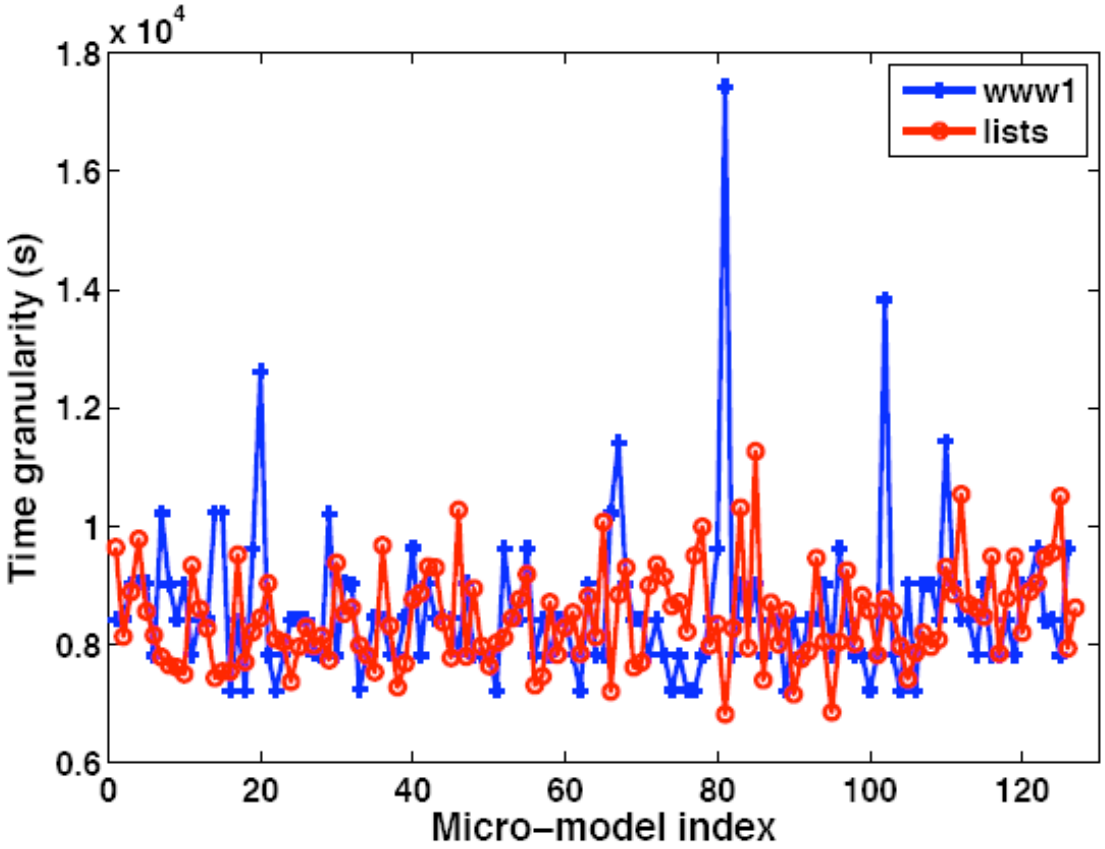


- Compute the likelihood L of seeing new n-grams
- Use a linear least squares approximation over a sliding window of points to detect the stabilization point
- When the stabilization point is found, reset L and start a new model

Time Granularity Detection



Automatic Time Granularity



www1 g ≈ 2 hours and 22 minutes std ≈ 21 minutes
lists g ≈ 2 hours and 20 minutes std ≈ 13 minutes)

Adaptive Training using Self-Sanitization

- Divide data into multiple blocks (*automatically*)
- Build **micro-models** for each block
- Test all models against a smaller dataset
- Build **sanitized** and **abnormal** models
  - sanitized model:

$$T_{san} = \bigcup \{P_j \mid SCORE(P_j) \leq V\}$$

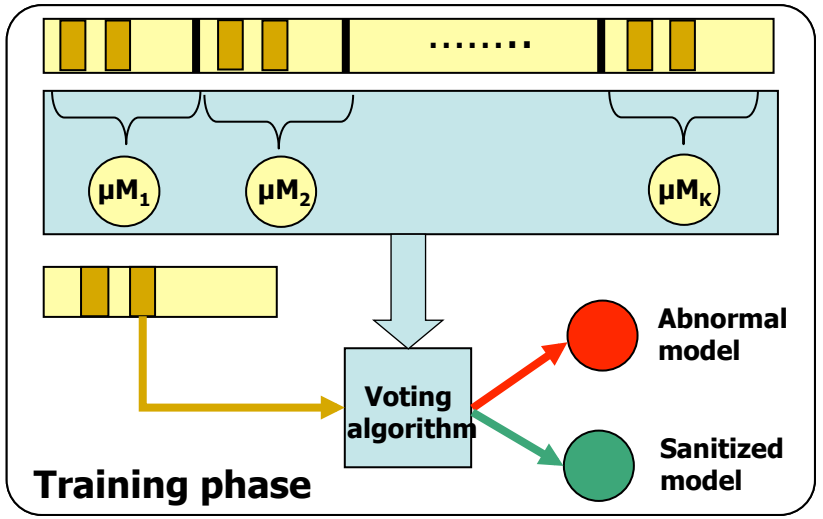
$$M_{san} = AD(T_{san})$$

  - abnormal model:

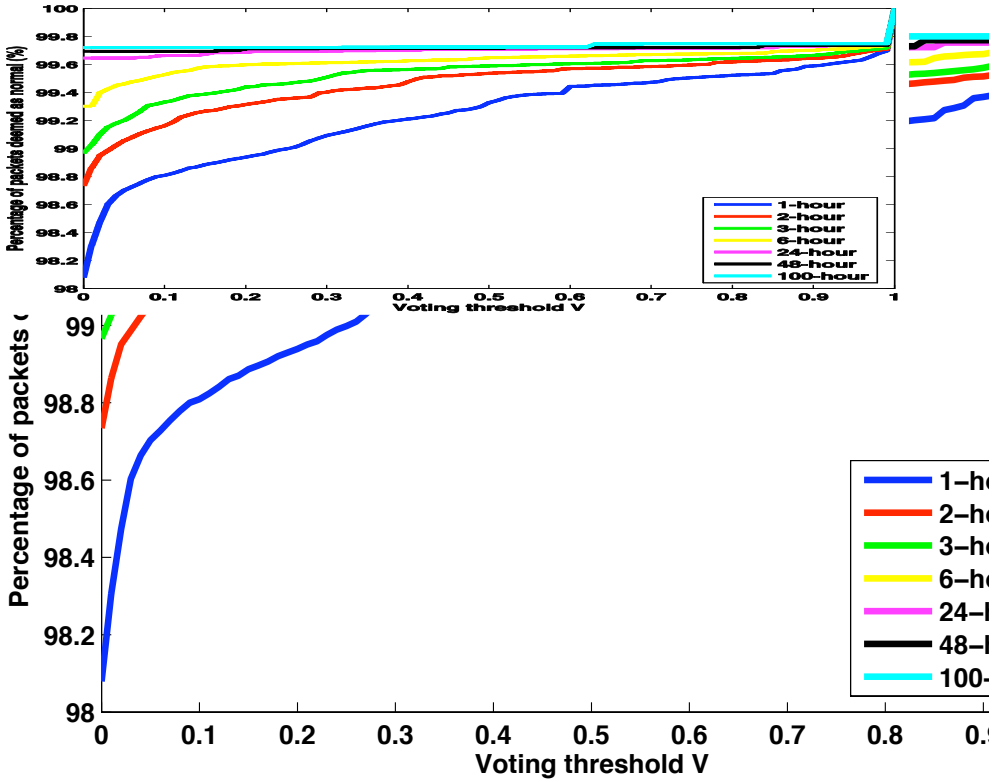
$$T_{abn} = \bigcup \{P_j \mid SCORE(P_j) > V\}$$

$$M_{abn} = AD(T_{abn})$$

  - **V** = *automatically* determined *voting threshold*

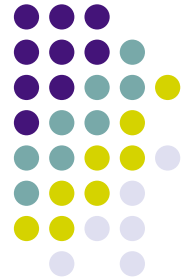


Automatic Detection of Voting Threshold



- Analyze how the different values of the micro-model ensemble score are distributed across the tested dataset
- V=0: a packet must be approved by all micro-models in order to be deemed normal
- V=1: a packet is deemed as normal as long as it is accepted by at least one micro-model.

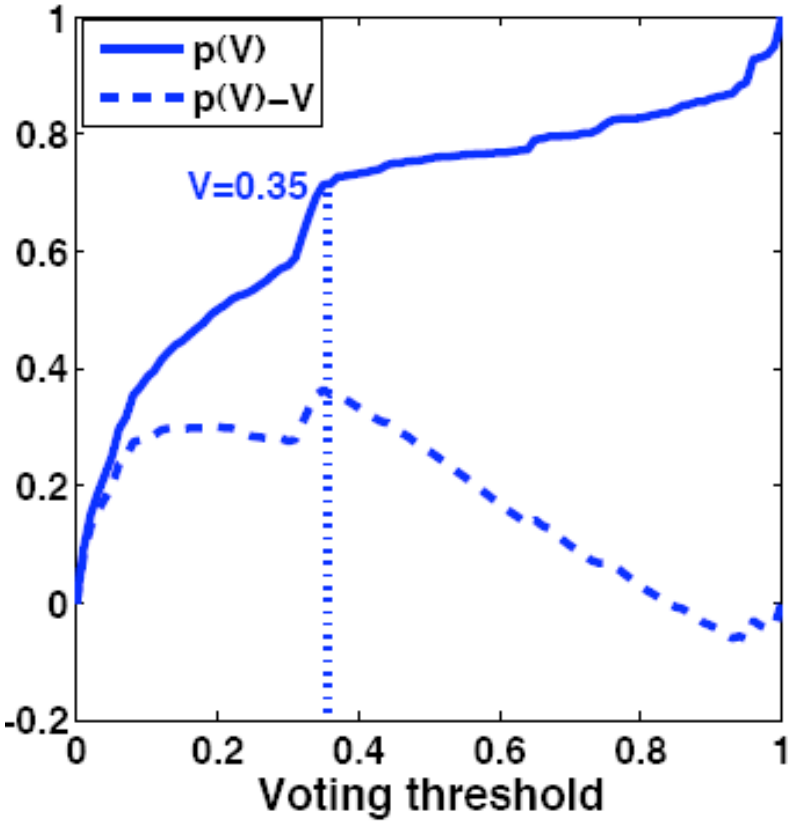
Voting Threshold Detection



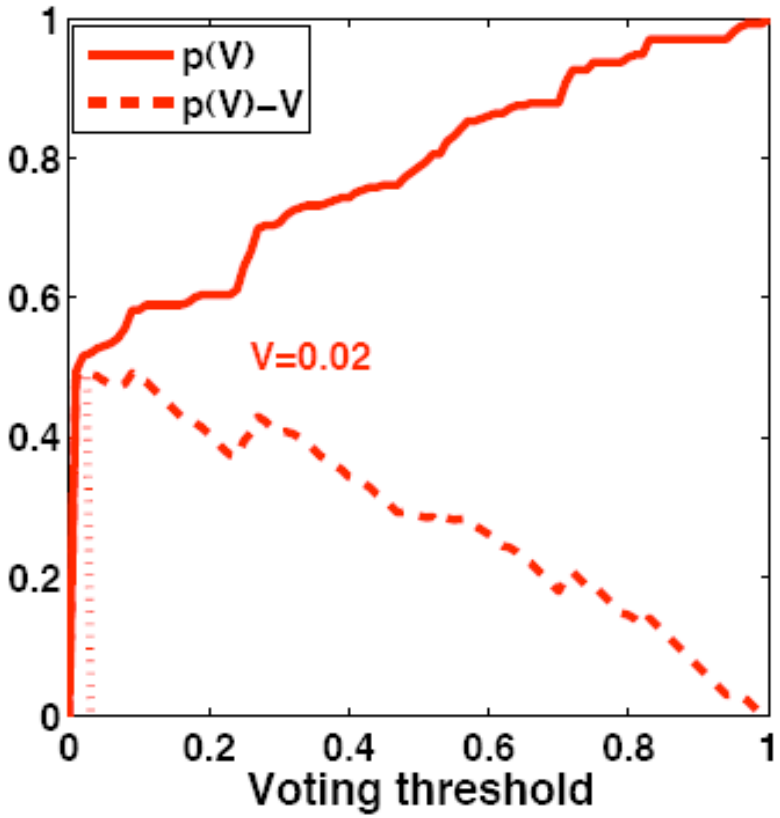
- $p(V_i) = \frac{P(V_i) - P(0)}{P(1) - P(0)}$ where P(V) – number of packets ned normal

- *Separation problem:*
  - finding the smallest threshold (minimize V) that
  - captures as much of the data (maximize p(V))

Example of Voting Threshold Detection

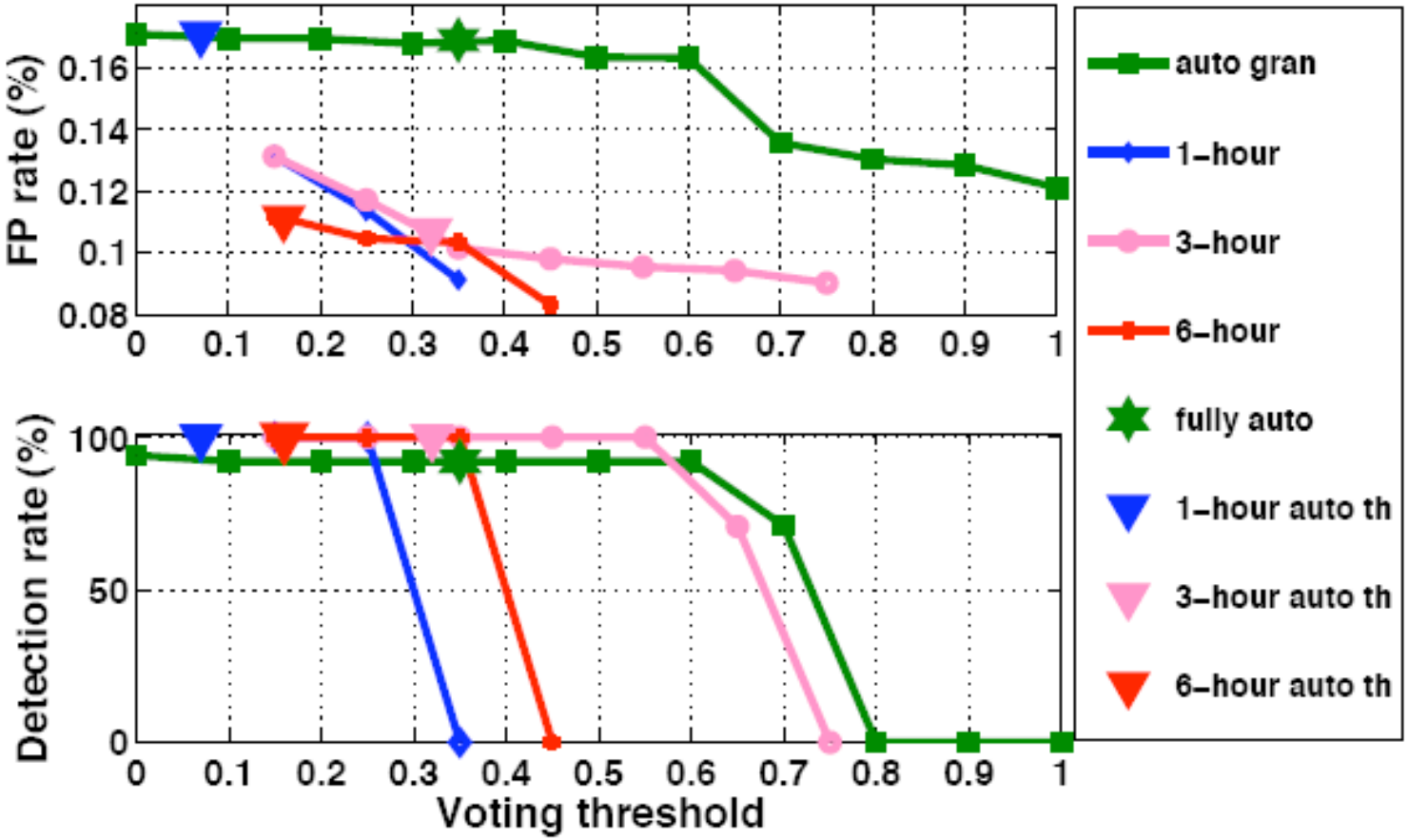


(a) *www1*

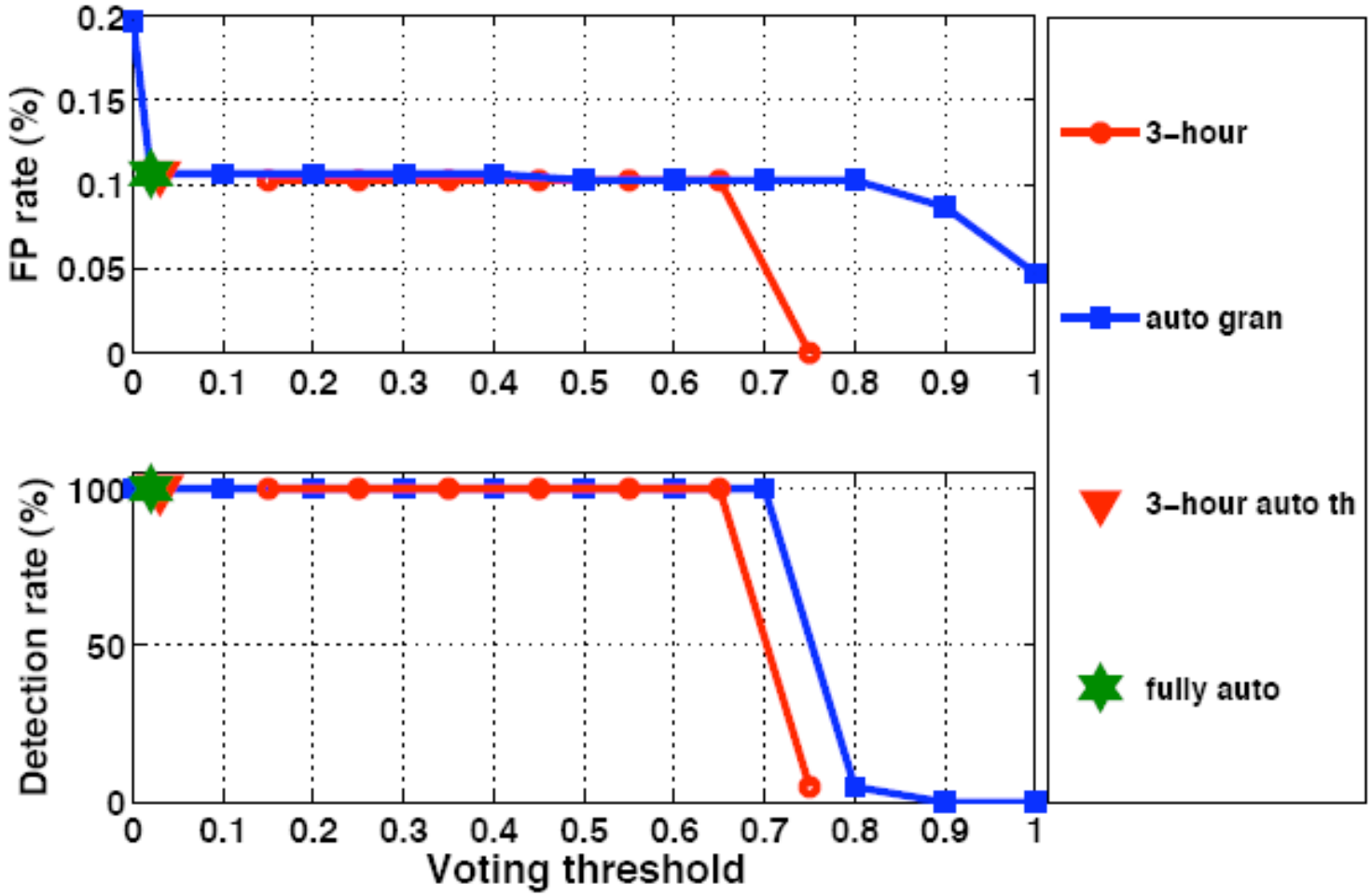


(b) *lists*

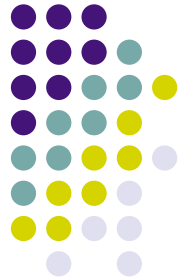
Automated vs. Empirical (*www1*)



Automated vs. Empirical (*lists*)

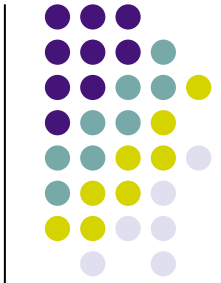


Overall Performance

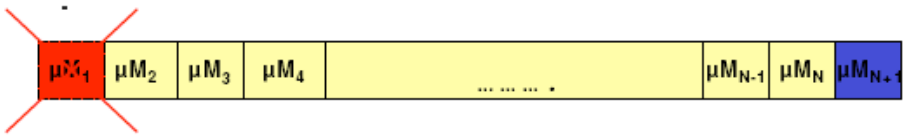


Parameters	www1		lists	
	FP(%)	TP(%)	FP(%)	TP(%)
N/A(no sanitization)	0.07	0	0.04	0
empirical	0.10	100	0.10	100
fully automated	0.16	92.92	0.10	100

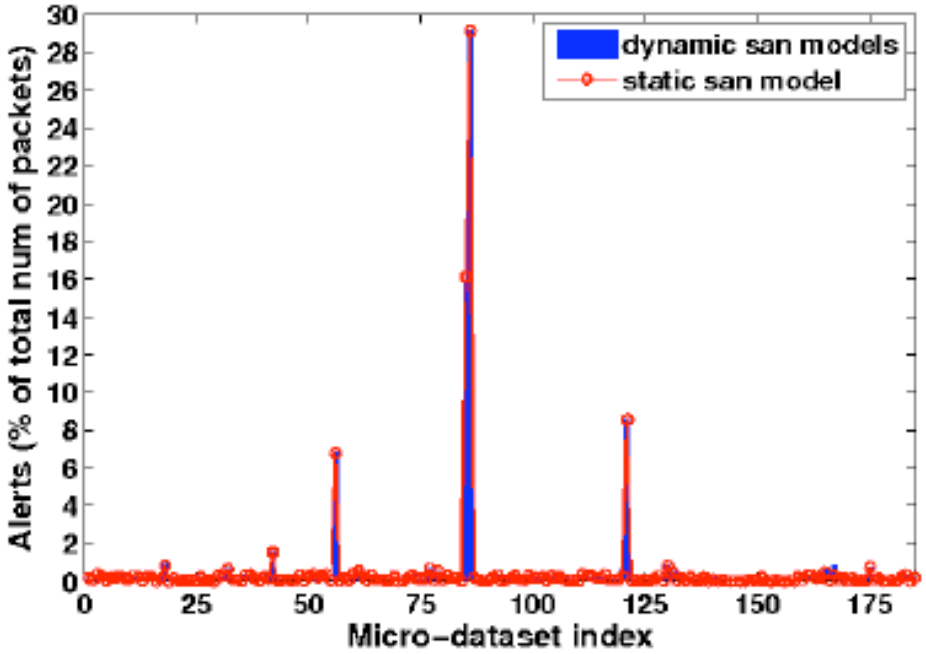
Self-Updating AD Models

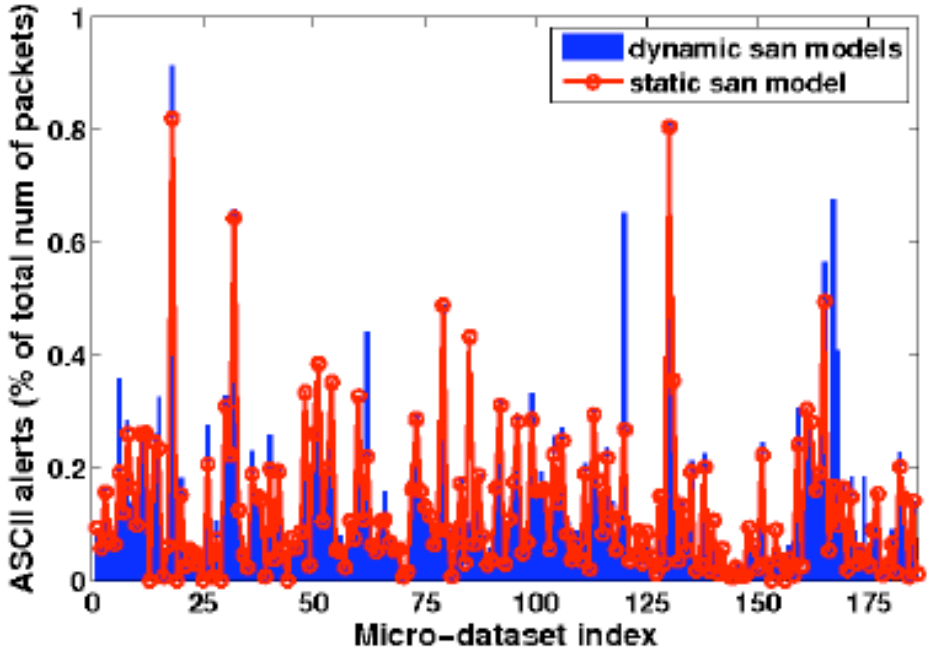


- The way users interact with systems can evolve over time, as can the systems themselves.
  - AD models need to adapt to *concept drift*
- Online learning can accommodate changes in behavior of computer users [Lane99]
- Continuously create micro-models and sanitized models
  - *Use introspection*: the micro-models are engaged in a voting scheme against their own micro-datasets



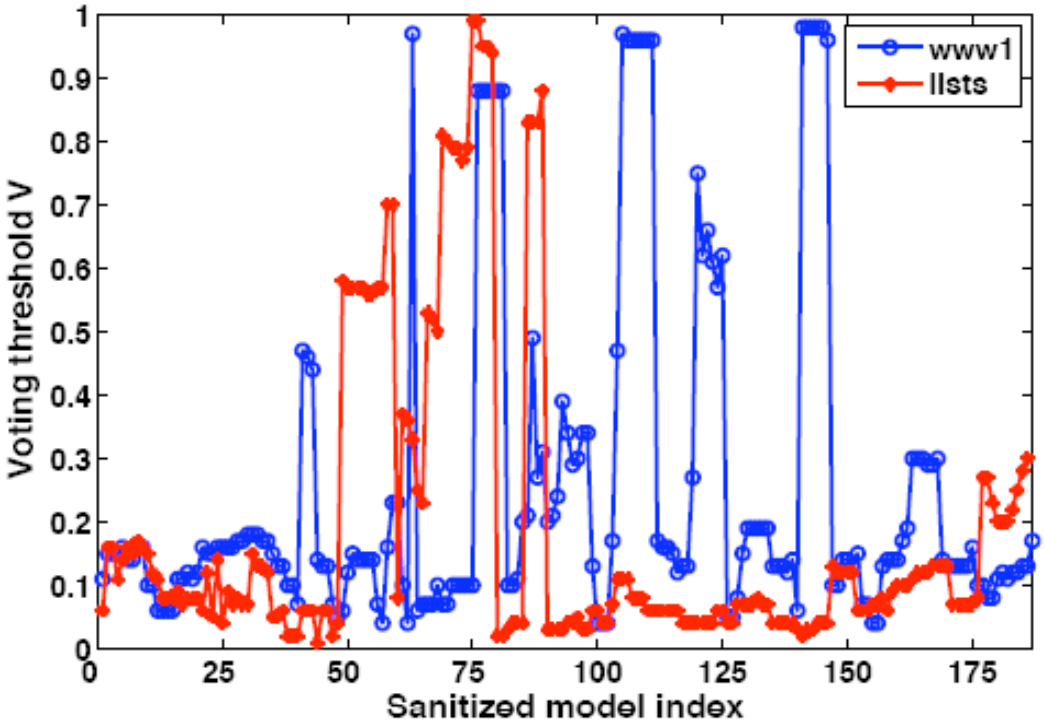
Alert Rate For *www1*



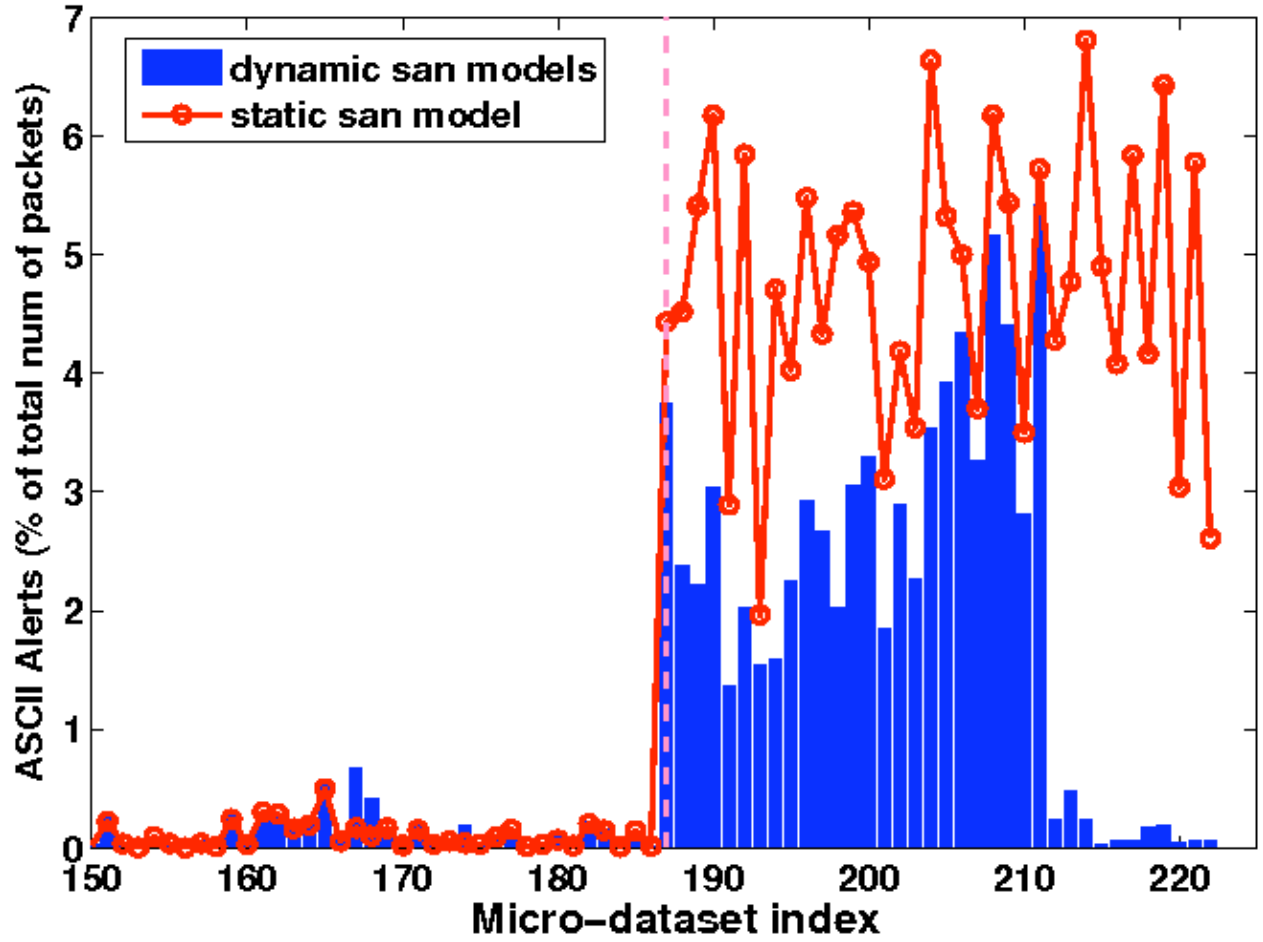


Self-Update Performance

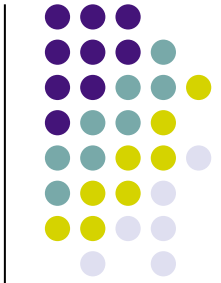
Model	www1		lists	
	FP(%)	TP(%)	FP(%)	TP(%)
static model	0.61	94.68	0.13	100
dynamic models	0.62	98.37	0.26	100



Concept Drift at Larger Scale



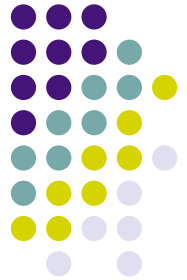
Computational Performance



Task	Time to process
build and save a new micro-model	7.34 s
test its micro-dataset against the older micro-models	1 m 12 s
test the old micro-datasets against the new micro-model	1 m 58 s
rebuild and save the sanitized model	3 m 03 s

- 25 micro-models
- Each micro-model size is 483KB on average (10.98 MB of traffic)

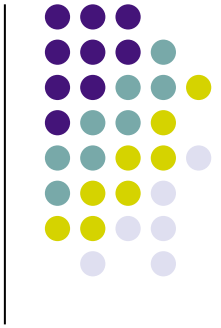
Possible Improvements



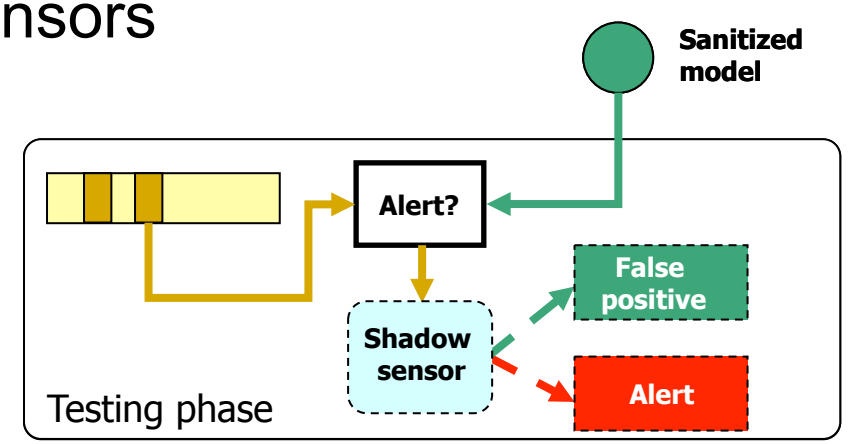
- Parallelization
  - Multiple datasets can be tested against multiple models in parallel
  - The test for each dataset-model pair is an independent operation

- Faster test for the bloom filters

Testing Strategies: Shadow Sensor Redirection

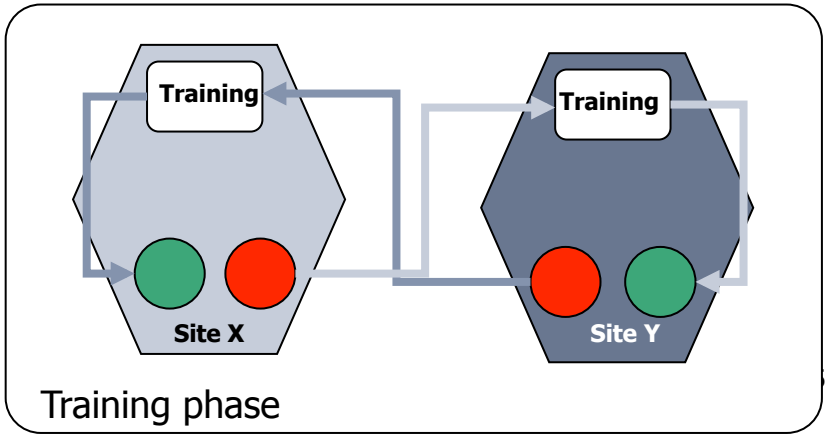


- **Shadow sensor**
  - Heavily instrumented host based anomaly detector akin to an "**oracle**"
  - Performs substantially **slower** than the native application
- Use the shadow sensor to **classify or corroborate** the alerts produced by the AD sensors
- Feasibility and scalability depend on the number of alerts generated by the AD sensor

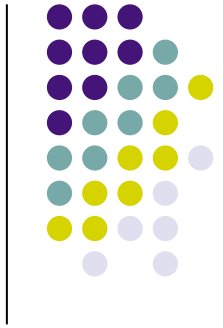


Distributed Sanitization

- Use **external knowledge** (models) to generate a better local normal model
- Abnormal models are exchanged across collaborative sites [Stolfo00]
  - Re-evaluate the locally computed sanitized models
- Apply **model differencing**
  - Remove remote abnormal data from the local normal model



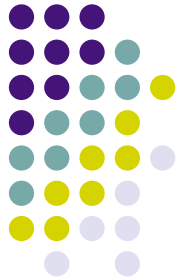
Conclusions



- We propose a fully automated framework that allows the AD sensor to adapt to the characteristics of the protected host while maintaining high performance

- We believe that our system can help alleviate some of the challenges faced as AD is increasingly relied upon as a first-class defense mechanism

Future Work



- Combine the strengths of multiple sensors under a general and unified framework, following the directions traced out in this study
- The temporal dimension of our online sanitization process can be complemented by a spatial one
- Use feedback information for concept drift
  - The error responses returned by the system under protection

Thank you!

Questions?