Detecting the Malicious Use of Legitimate Credentials

Austin Brown, Andy Brown, and Kendall Land
Faculty advisor: Maanak Gupta
Department of Computer Science, Tennessee Tech University, Cookeville, TN

Background

Any organization that owns or has access to valuable data is at risk of being attacked by cyber criminals who obtain legitimate credentials and use them to steal or destroy. If a nefarious actor can access a network with legitimate credentials there are currently very limited ways to determine the legitimacy of the activity.

Existing intrusion detection systems often use signature-based algorithms that are designed to detect known attacks, but attackers with valid credentials can bypass these systems by looking like normal user traffic. An anomaly detection system could potentially thwart these attackers by detecting the difference between a user's normal behavior and the behavior of the attacker, who is likely to perform actions a regular user would not normally perform.

Motivations and Approach

- Credential theft is a pervasive problem in enterprise environments
- Credential theft is difficult for signature-based intrusion detection systems to detect
- Rapidly detecting the use of stolen credentials will allow security teams to quickly revoke access to the account and prevent damage
- Machine learning offers great anomaly detection possibilities
- The goal of this research is to determine a method to profile user behavior and detect actions that deviate from the expected
- Whereas other approaches to this problem look at events that happen before a login, we used events that happen after a login

Dataset

Content
- 58 days of sanitized data: 100GB uncompressed
- Represents host, network, authentication, and process events
- Collected from an enterprise network at Los Alamos National Lab
- Approximately 1.6 billion events with 689 "red-team" records indicating which events were the result of simulated malicious activity

Manipulation
- Size of dataset made simple processing prohibitively slow
- Data was ingested into the Elastic stack to enable fast querying and exploration of the dataset
- Features for machine learning were generated from statistics retrieved from Elastic

Feature Extraction

To build a feature vector for the model to ingest, we focused on activity that occurred soon after a login. For each user, we started from each successful NTLM login and collected 500 seconds of data including:
- The first 4 programs that users started
- Number of DNS requests
- Network activity – bytes transferred by direction per standard/nonstandard port
- The collected features were hash encoded into feature vectors
- Each feature vector holds 124 feature hashes representing 19 original features
- A large number of feature hashes is needed to avoid feature hash collision

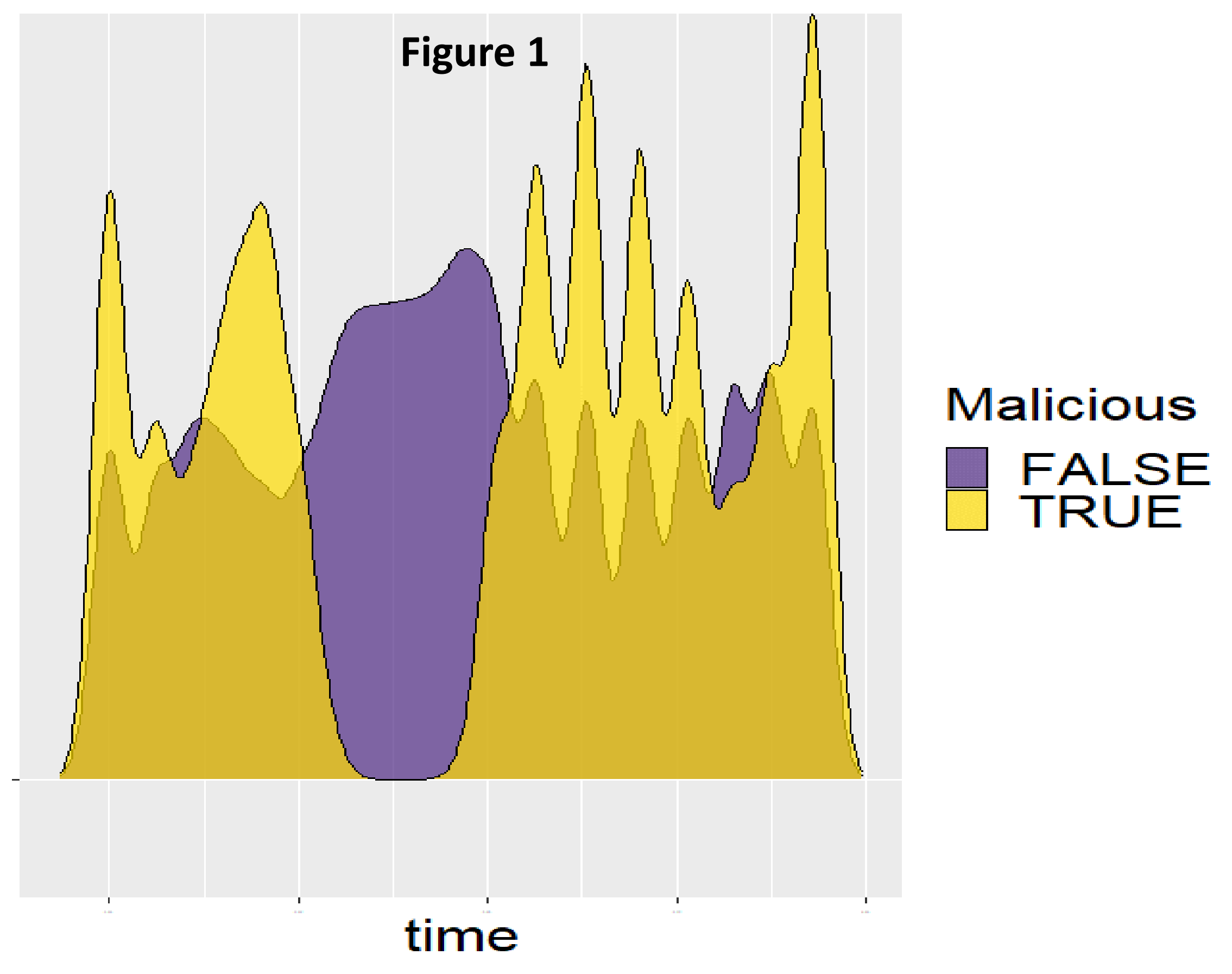


Figure 1 shows that the hour of the day that an event occurs can influence how the model classifies it. The other features also influence the model in this way. The model considers the value of all the features in the set when making its decision on how to classify an event.

Machine Learning

We chose a random forest binary classification model for classifying events. Random forests provide accurate classification for data with high numbers of variables and data with missing values. This was appealing for our use case as enterprise logs are quite large and user behavior can generate data with "missing" values if users do not take certain actions. The forest was implemented using these technologies and parameters:
- 500 estimators (Decision Trees)
- The classifier was trained on a sample dataset with known malicious events
  - Sample sets contained 10 thousand benign and 3.5 thousand malicious events
    - Malicious events were replicated fivefold to account for class imbalance
  - Results derived from an average of 50 samples from dataset

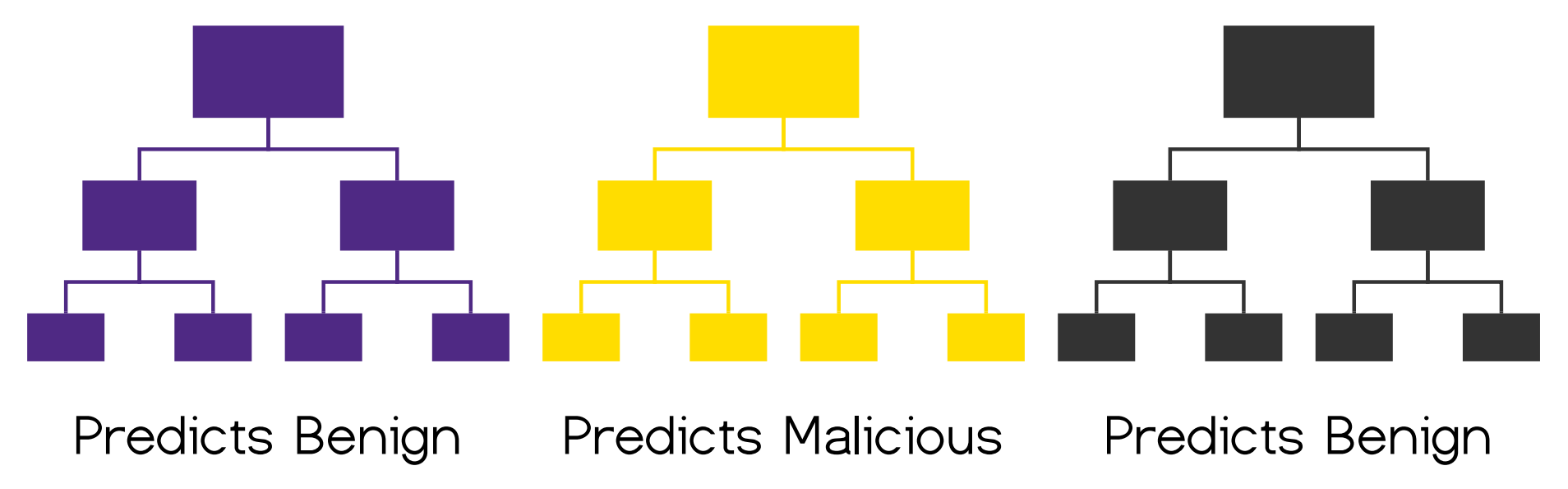


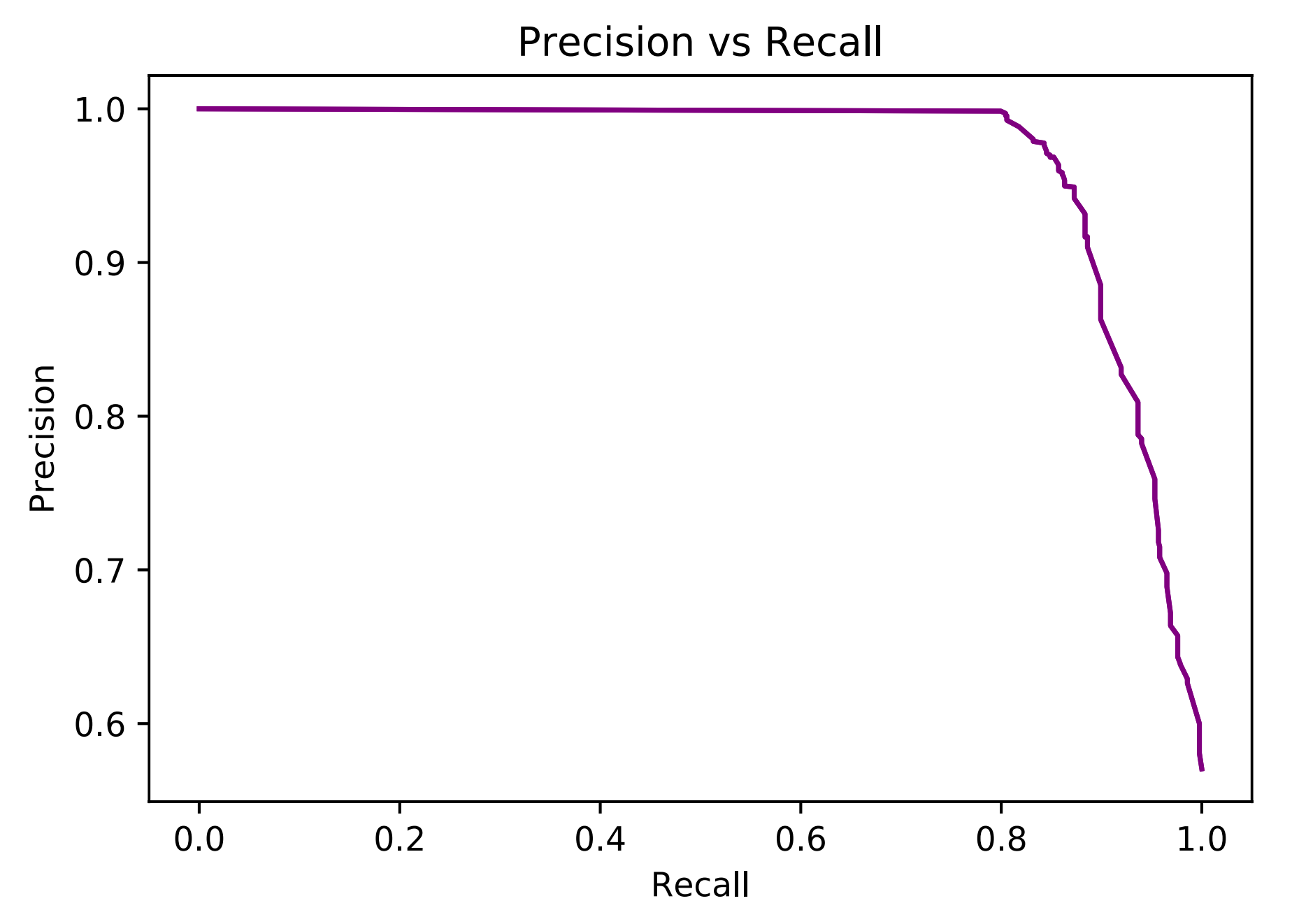
Figure 2: Three decision trees vote on an event

Results

The table to the right shows the average classification results over fifty samples of two different detection thresholds. A threshold represents how confident the model needs to be about something being malicious for it to alert us. Setting a higher threshold will yield a lower false positive rate but miss more true positives because the detector only alerts when it is very certain that an event is malicious.

	Threshold = 0.25		Threshold = 0.5		
	Alert	No Alert	Alert	No Alert	Total
Malicious	794.48	38.52	729.92	103.08	833
Benign	251.82	2277.18	46.36	2482.64	2529
Total	1046.3	2315.7	776.28	2585.72	3362

A classifier with low false positive rate may still generate huge numbers of false positives when the number of events is very high. Precision-Recall plots are better than ROC curves at visualizing data with large class imbalances, as they emphasize completeness and accuracy of detection over proportional rates of detection. The precision-recall plot of our model indicates that precision drops sharply as recall increases.



Conclusions

Using a detection threshold of 25%, our classifier identified 95% of malicious events in a random sample data set with a 25% false positive rate over our dataset with the replicated red team events.

If the detection threshold is lowered to 8% certainty, we can theoretically achieve 100% detection of malicious events with a 50% false positive rate. This rate may be acceptable if the number of true positives is low enough or the protected systems are valuable enough.

Overcoming the class imbalance in log data will be a significant problem to overcome. Our model relies on the duplication of knows red-team events to balance the classes. Without the class balancing our model accuracy drops significantly.

Increasing the time window for feature extraction would increase model efficacy with a proportional increase in compute intensity.

Future Work

Overcoming the class imbalance between malicious and benign events is a topic of interest. We overcame the class imbalance by artificially replicating the number of red-team events. This is not possible on a live data feed, so future work should address tuning the model to better handle the imbalance.

If the classifier considered log events in context of recent preceding log events, the classifier could learn user behavior in a much more contextual way. Representing this data in a time sequence may yield better results. LSTM autoencoders paired with modified GANs have shown promising results in similar classification problems of time series data with extreme class imbalances

Acknowledgements and References

Dataset:
A. D. Kent, "Comprehensive, Multi-Source Cybersecurity Events," Los Alamos National Laboratory, http://dx.doi.org/10.17021/1179829, 2015.

Inspiration:
https://www.rsaconference.com/usa/us-2018/agenda/detection-of-authentication-events-involving-stolen-enterprise-credentials

Implementation:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html