Technical Overview of PolyScore™

Introduction

PolyScore is a single, authoritative score that provides the probability a given file contains malware.

Clear decision-making in an uncertain environment is one of the most important traits a security professional can possess. Companies today rely on a variety of structured methods for collecting and aggregating IOCs to inform these critical decisions.

PolyScore simplifies this by providing an authoritative metric for predicting the likelihood of malware. A single metric like PolyScore is important to humans and automations within a SOC that must make sound, often binary, defense decisions from multiple sources of information. Today these decisions are often made through a combination of inconsistent analyst intuition and simple majority rule of engine verdicts from sources like VirusTotal. Analyst intuition does not scale and can not be automated. Weighing automated results by majority role or other simple approaches is not accurate enough.

PolyScore aims to address these issues and is built from millions of PolySwarm's assertions, analyzed by an ensemble of both traditional statistics and semi-supervised machine-learning models.

This paper introduces the terminology, challenges, and approaches to aggregating engine verdicts into an authoritative measure of malice. It uses simplified examples to illustrate how PolyScore helps solve practical challenges and drives better decision-making by CTI and SOC teams.

Background

Since PolySwarm's inception, we've wanted engines to render the best possible verdicts and we are always considering how to build systems that reward engines, arbiters and bounty-participants for long-term correctness. To that end, we undertook an internal investigation focused on studying and modeling strategic participant behavior in PolySwarm, with the goal of establishing guidelines for fair dealing and principled engine ratings.

One question core to this investigation was: "*How can we ensure engines receive rewards proportional to their novelty, while preserving other desirable properties such as equitable participation rewards?*". One such solution came in the form of a system that scaled rewards proportionally to the engine's historical performance. This system rewards participants willing to first stake minority verdicts, while ensuring engines who

substantiate those claims in later rescans[1] receive some reward for bringing quick convergence to a strong consensus. To achieve this we employed a framework that takes into account ‘speed’ in assertions, relevant connections between meta-characteristics of engines and the likelihood of observing a particular set of verdicts given instance ‘malice’.

One problem stood out: aggregating verdicts into a single authoritative malice metric would be required in any scheme, and the accuracy of our system could decrease significantly if our methodology was faulty.

Assigning Ground Truth

In constructing PolyScore, it was important to have a training dataset that included ground truth about files and their malintent. *Labeling*, the identification of the true malware class of files, is unusually difficult. There is no "golden goose" for labeling novel malware which anti-malware vendors and malware authors haven’t already explored and exploited. The efforts we’ve seen to gather training data can be summarized in the following approaches, each with their own downsides:

- Determine malice manually with reverse engineered samples. This approach requires skilled human intervention, making it unsuitable at scale.
- Use malicious software dumps from a single, trusted source. Many are ‘old’ files from the most clear-cut cases, heavily biasing results.
- Combine malice labels from multiple anti-malware vendors. This inevitably favors some approaches to malware detection while reproducing the bias of each engine’s shared threat intelligence sources.
- Backdating "future" malware with old signatures and engines. This is a lagging and sometimes dangerous indicator, where accidental good performance is over-inforced.

Remaining conscious of the distribution of malicious and benign files and with no prior knowledge of engine performance, scale, and homogeneity, we approached this problem from the top. We conducted our own independent analysis, diving into disassemblies occasionally, and gathering only never-before-seen data to benchmark future model assessments.

The most important (perhaps unsurprising) finding about this population is that it has, by far, the most variance and the lowest malware detection rates. Further, the benign binaries in this sample population also have the most false positives[2].

[1] Made before arbitration
[2] Installers, particularly those that decompress themselves in memory, are big here.

Our initial dataset for testing and training included 21 engine verdicts (a boolean value indicating that engine's findings) over the sample described below:

	Benign	Malicious	Total
Number of samples	966	1306	2272

Explored Approaches

Majority Rule

The naive (and often best) approach is to assign equal weights to each engine, without considering the varying accuracy of engines. For this approach, you don't need to determine the strengths and weaknesses of engines in majority rule schemes - the only relevant parameter being the threshold that serves as the cut-off to classify malware as an absolute number or percentage.

You may have already perceived a conceptual flaw from the outset - majority rule systems are unable to incorporate engines with relatively high or low false positive rates - reducing the threshold to take advantage of the former as an additional information source collapses the entire model's false positive rate and vice versa. This makes simple majority rule a poor method of detecting malware.

A more sophisticated approach: Brier Scoring

Brier scoring is an approach for verifying accuracy of event forecasts after the outcome of the event is known. It is used for events with exclusive and discrete outcomes, such as malicious or benign assertions on a file. Brier scoring is sensitive to the underlying probabilities of outcomes, so it is able to provide more information than flat weights provide.

We used an iterative approach to optimize the Brier Scoring function for the weights of our engines. The update method for each weight when using this scoring function is as follows:

The method we used is similar to gradient descent on the weights using the Brier scoring rule as an objective function. This is an intuitive method of evaluating an engine's performance, with a strong foundation in information theory. This process

returns a list of engine weights that we can use to evaluate the maliciousness of a sample based on the previous verdicts they've rendered. We used this approach to perform a 5-fold cross-validation in our dataset. The following table shows the results for this trained model.

	precision	**recall**	**f1-score**
Benign	0.84	0.74	0.78
Malicious	0.82	0.89	0.86
Accuracy			0.83
Scale of 0 to 1, higher score indicates greater accuracy.			

Even though this is an improvement over a naive majority-rule classifier[3], 83% accuracy is still woefully inadequate.

Models limitations

There are few fine points that make linear solutions like majority rule and Brier scoring difficult. In technical terms: a simple hyperplane can not separate malicious and benign verdicts well.

These schemes assign equal or proportional importance to all engines, they only differ in the degree they suffer from the existence of 'specialists'. Put another way, the majority of generalist engines dominate summary numbers and mute specialist engines that have important predictions in some cases.

These weaknesses are the direct reality of their construction's arithmetic[4]. Engines that operate as detectors with low false positive and high false negative rates that are scored pairwise to only provide an incremental improvement, detecting benign files and

$$S_{\text{quadratic/Brier}}(p) = \begin{cases} -(1-p)^2 & \text{if the verdict is } malicious \\ -p^2 & \text{if the verdict is } benign \end{cases}$$

letting malicious files go undetected. ¼ of malware whose first sighting was less than a

[3] A classifier which always predicts an artifact to be malicious would have 57% accuracy in this sample.
[4] While in theory Logarithmic Pooling can model any linear function, in practice it possesses the same weaknesses of our original scheme to assign equal weight to all engines and differs only in the *number* or *percentage* of engines in agreement.

year old was undetected by at least one of five engines in a study by Micheal Bailey and colleagues. [5]

Minority Advantage

The figure below shows a histogram of malicious detections from initial and final rescans.
Early assertions are reported by few engines, as time goes on, the consensus gap closes. By attending to early assertions appropriately, we can strengthen our response time and accuracy.

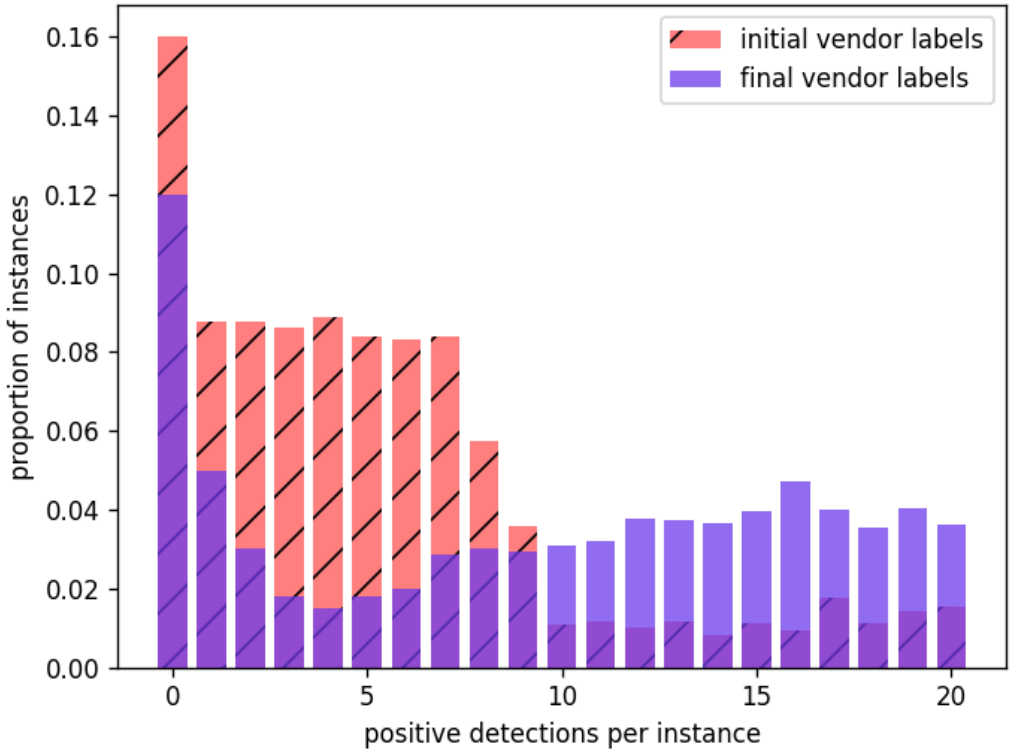

Figure 3: Benign & Malicious Verdict Histogram

We can use the output of a receiver operator characteristic (ROC) curve at various thresholds to compute the odds of a false-positive to scale various engines, but the overarching problem is best solved by introducing smaller, more focused engines which are less likely to experience false positives. We can put more emphasis on engines that may be in the minority of verdicts if they've been correct in similar situations before.

Engine Similarity

[5] Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., and NazarioBailey, Michael, et al. "Automated classification and analysis of internet malware." *International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, Heidelberg, 2007.

Early on, we realized there are other factors we must consider before the model itself. This was mainly borne from an analysis of engine similarities and differences. We realized that we had to relax our assumption of independence because engines often share methodologies and threat intelligence. Engines can be grouped into cliques – groups of engines which respond more often together than they would be expected to be by chance.

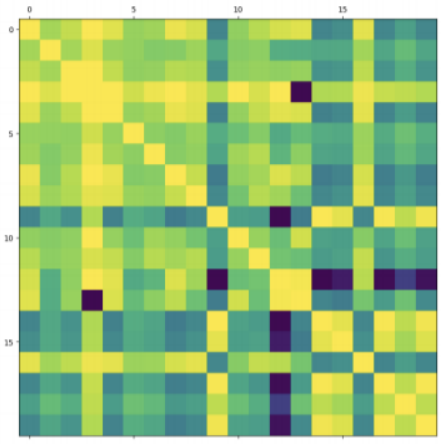

Figure 2: A similarity matrix of PolySwarm's engine population during a training run. Clique-y engines are similarly colored as they often agree.

Many statistical methods can account for similarity or dependence between features, but having a common scale of similarity between engines allows us to extract more performance from our classifier. This common scale allows us to:

- **Identify** shared threat intelligence and detection architecture between engines. This avoids over-weighting large cliques of engines while also being more agile when responding to a threat.
- **Inform** classifier predictions with statistical measures of likelihood given rates of historical agreement.

Classification Strings

Engines often generate 'classification strings' alongside their verdict, which is unstructured meta-data about the artifact being scanned. Often, there are patterns across engines that can be used to improve classification and malware detection.

PolyScore takes these into account. We integrate malware classification names into the model by normalizing strings from more than a dozen malware classification schemes and fragment hash tables built from vendor identifiers. These are learned from more than 10 million label changes between scan features in our data set.

This cross-engine data gathering and analysis leverages a significant space of features that improve our model, and PolySwarm has access to this information which is not available from other providers.

PolyScore as a New Metric for Malware Detection

After examining the problem of weighting malware detectors and the many shortcomings of current approaches, we decided to build a semi-supervised model from a class of algorithms used in crowdsourcing information under uncertainty. It would encode malware-specific domain knowledge from our observations across our minority advantage and engine similarity data to address many of the issues with simple classification methods, and leverage the unique data that is available in the PolySwarm platform. Some key factors that distinguish PolyScore from other methods are:

- **PolyScore Scales**: we used a variety of methods that are prepared to handle a huge amount of data from multiple engines.
- **Incorporates Engine Similarity:** pairwise engine similarity plays a significant role in malware detection
- **Utilizes Classification Strings**: Cross-engine family features, as well as other malware classification strings, improve our classification performance.

These advantages are unique to PolySwarm. We gather information from millions of assertions over dozens of engines - with real value staked on verdicts. This provides for a uniquely rich source of model features that don't rely on the individual detectors. As more engines choose to participate in the marketplace, these cross-engine features will grow, further improving PolyScore.

Today, PolyScore outperforms ***any*** unweighted majority voting scheme and beats out linear optimization models.

Our most-recent validation shows 97% accuracy, a significant improvement over other methods.

	precision	**recall**	**f1-score**
Benign	0.96	1.00	0.98
Malicious	1.00	0.91	0.96
Accuracy			0.97
Scale of 0 to 1, higher score indicates greater accuracy.			

Scale

PolyScore was designed to scale with the demands of a growing dataset and shifting malware ecosystem, incorporating more engine verdicts to continuously improve over time. It started from an initial model built on fundamental results: the confidence of an engine in their verdict, bounty-specific characteristics, pairwise correlations between engines, and historical performance of engines, to name a few. Our model also does smart segmentation of the malware space, meaning we have fine-tuned models with increased performance based on malware families and other characteristics extracted from the artifacts. We've conducted hundreds of research trials to systematize our approach and validate our thinking on candidate relationships for our models to employ.

PolyScore's design was inspired by research incorporating detection metadata in an investigation of the delay between heuristic detection and signature introduction[6]. We've normalized strings from more than a dozen malware classification schemes, and learned from more than 10 million label changes to distinguish heuristic based verdicts. This in turn allows us to focus on real malware by collapsing family aliases and excluding 'paramalware' categories like AV test files and *Greyware*.

Other Desirable Features

PolyScore produces a wealth of descriptive statistics about engine performance. Internally, we use over a dozen engine strength rankings to explore candidate results. Our team uses this information to highlight interesting engine verdicts and identify emerging threats in a firehouse of daily samples processed.

Malware authors are always innovating, that is why we define engine **agility,** which is the time from sample submission to an engine marking it as malicious. We use agile engines as our early warning system, when they move to convict and when new convictions are wide-spread, we pay attention to that data.

The proportion of malicious artifacts that an engine correctly identifies are key to understanding engine signal versus noise, and we define this as the engine's **sensitivity**.

Naming and classification of malware has been a controversial topic from the birth of the AV industry. Far from solved, we look to the bright side of how much engines agree with each other in their tagging and naming schemes. That is why we define **lexism**, which tells us how much engines agree with each other on malware tags and names, and allows us to form the basis for PolyUnite, our tagging framework that gives consensus based names to malware and families.

---

[6]Moheeb Abu Rajab, Lucas Ballard, Noe Lutz, Panayiotis Mavrommatis, Niels Provos ´ Google Inc.. CAMP: Content-agnostic malware protection. In 20th Annual Network and Distributed System Security Symposium, NDSS (2013).

Conclusion

PolyScore is a single, authoritative score that provides the probability a given file contains malware. It is designed to address the main shortcomings of crowdsourced models and existing multiscanners:

- Multiple and often conflicting binary opinions require additional, intuition-based work from analysts; which is time intensive, produces inconsistent results and can not be automated.
- Scores found in solutions like VirusTotal use basic models that simply summarize results by aggregating opinions; a suboptimal approach for identifying new and emergent threats.

PolyScore 's algorithm filters the noise and amplifies the signal by weighting engine's opinions based on past performance, strengths, confidence levels, and other rich contextual threat indicators built from millions of daily assertions generated inside PolySwarm.

PolyScore uses a semi-supervised machine learning model to continuously improve over time, and already outperforms any other malware scoring methods by a significant margin, currently yielding a 97% accuracy rate.

It can easily be automated into SOC and CTI team workflows, enabling enterprise security teams to make fast and accurate defense decisions, at scale.