# It's a Beautiful Day in the Malware Neighborhood

Matt Maisel

Manager of Security Data Science @ Cylance

DEF CON AI Village, August 2018

# Motivation

- Search and retrieval of similar malware samples provides context to analysts and systems
  1. Relate previously analyzed samples with unknowns
  2. Prioritize outliers for manual analysis and reverse engineering
  3. Process samples in incoming alerts and route to other workflows
- Indexing samples by cryptographic or fuzzy hash is standard approach

# Problem Statement

- Malware similarity is performed through comparison of raw bytes or extracted static and dynamic features that distill semantic characteristics
- Represent samples in a n-dimensional feature space

## Please won't you be my neighbor?

- **Nearest Neighbor (NN) Search**: Given a set of $n$ samples $X$, return the $k$ nearest neighbors for query sample $x_q$ according to a distance function $d(x_q, x_n)$.
- Approximate variant allows some error threshold $\epsilon$ that satisfies: $d(x_q, x_n) \leq (1+\epsilon)d(x_q, x_n)$

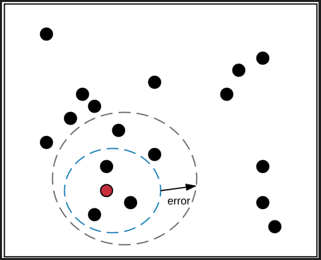



Figure 1: K = 3

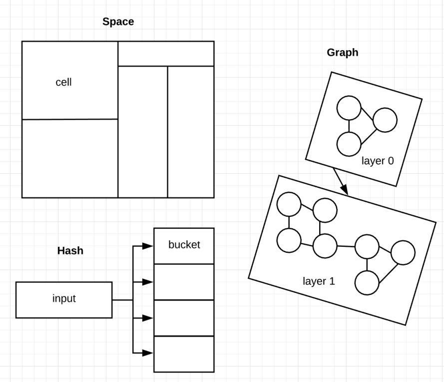


CYLANCE

# Theory and Literature Review

## Methods

- Tree
- Hashing
- Graph



# NN Methods

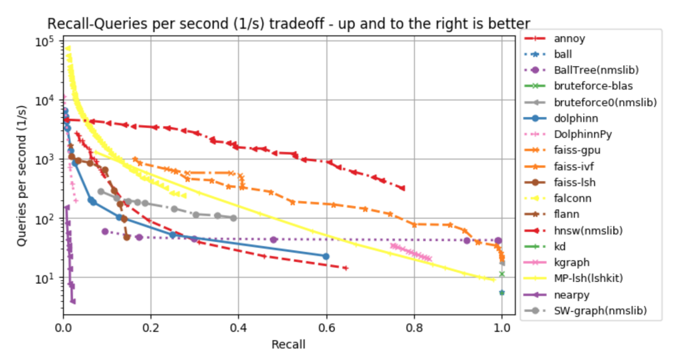



Figure 2: NYTimes @ $k = 100$(ANN Benchmarks)[1]

[1] https://github.com/erikbern/ann-benchmarks

# Hierarchical Navigable Small World (HNSW)

- Fu et al. (2017) use a multi-layer graph and greedily identifies candidate samples for comparison
  - Construct graph during an offline phase
  - Query candidate neighbors via traversal mechanism
  - Iteratively search neighboring nodes until stopping criteria

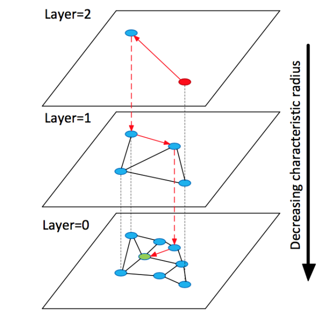



Figure 3: Sketch of query from top to bottom layers (Fu et al. (2017))

# Prioritized Dynamic Continuous Indexing (PDCI)

- Li and Malik (2017) design an exact randomized algorithm that avoids partitioning samples by vector space
  1. Construct multiple indices that order samples along random directions
  2. Visit samples in index in order of distance from query
  3. If sample retrieved from all indices, add to candidate set for distance comparison

# Related Malware Similarity Systems

- VirusTotal (2018) offers similarity search based feature hashing structural data
- Wallace (2015) provides an implementation of indexed ssdeep[2] and Abrahamy (2017) extends to use Elasticsearch[3]
- BitShred by Jang et al. (2011) perform pairwise Jaccard similarity in hadoop
- Upchurch and Zhou (2016) use MinHash in the Malware Provenance system which uses a sliding window hash on n-gram features from blocks of a disassembled sample

[2] https://github.com/bwall/ssdc

[3] https://github.com/intezer/ssdeep-elastic

# Related Malware Similarity Systems

- Rieck et al (2011) released Malheur[4] which uses a sequence representation of behavior extracted from sandbox reports to identify prototypes
- SARVAM[5] indexes raw bytes as gray-scale images and compares the distance of computer vision features

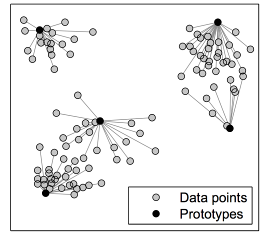



Figure 4: Malheur Prototype Selection

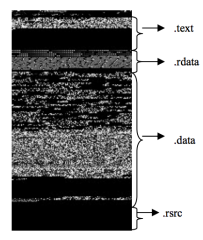



Figure 5: SARVAM image

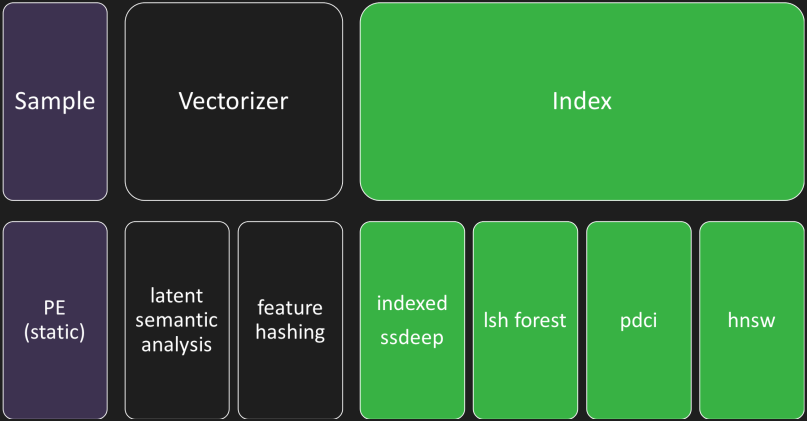
[4] http://www.mlsec.org/malheur/

[5] http://sarvam.ece.ucsb.edu

# System Design

1. **Extract** and store sample metadata and raw feature data
2. **Transform** data via feature vectorization pipeline
3. **Fit** indexes for NN methods on feature matrices
4. **Query** index with an input sample and return $k$-nearest neighbors along with relevant contextual features

# System Design



# Experiments

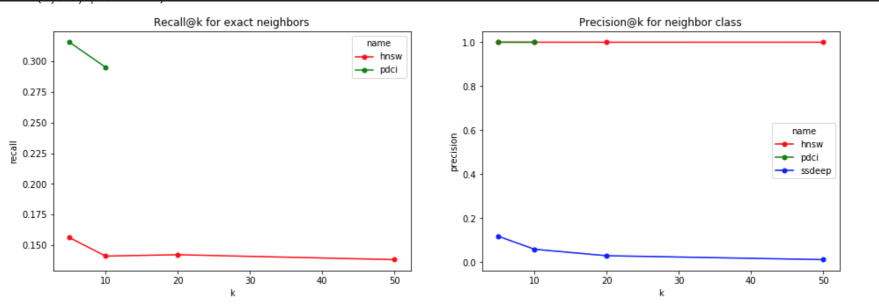



Figure 6: Results on vtcluster-jan2018 dataset, $n = 27000$, 15 classes

$$Precision@k = \frac{\text{relevant} \ \cap \ \text{retrieved}}{k}$$

$$Recall@k = \frac{\text{relevant} \ \cap \ \text{retrieved}}{\text{total relevant}}$$



CYLANCE

# Remarks and Future Work

- **Feature Engineering**
  - Add support for more file type vectorizers beyond PE
  - Extract multiple modalities, e.g. dynamic
  - Feature selection and learning representations
- **Experiments**
  - Large-scale parameter Optimization
  - Additional Benchmarks
  - Evaluation of difference distance metrics
- **Use Cases**
  - Indexing of benign samples?
  - Partial Fit

# Questions?

- https://github.com/cylance/rogers
- Pull request are welcome!
- mmaisel@cylance[.]com

# Appendix A - Feature Engineering

| Modality | Variable Type | Examples |
|---|---|---|
| Raw Bytes | Continuous | entropy of byte ngrams, similarity hash digest (e.g. ssdeep, tlsh) |
| Static | Continuous | file size, PE image size, code size, # of sections, compile timestamp |
| Static | Categorical | import symbols, import dlls, exported symbols, opcodes |
| Dynamic | Categorical | system API calls, spawned processes, network activity |
| Dynamic | Continuous | # of registry operations, # of file system operations, # of network operations |
| Contextual | Categorical | AV and Yara detection names, observed hostnames, file path names , user account |

Table 1: Examples of Feature by Modality and Type

# Appendix B - Protocol Buffers

```
message Feature {
    // type of variable
    message Variable {
        enum Type {
            CATEGORICAL = 0; // values in a specific category
            CONTINUOUS  = 1; // infinite number of values
            DISCRETE    = 2; // limited to certain number of values
            ORDINAL     = 3; // ordered variable, expects int value,
        }
    }
    // type of feature space
    message Modality {
        enum Type {
            BYTES       = 0; // raw byte features
            STATIC      = 1; // structural / static features
            DYNAMIC     = 2; // behavioral / dynamic features
            CONTEXTUAL  = 3; // contextual features
        }
    }

    Variable.Type type = 1;
    Modality.Type mode = 2;
    Value value        = 3;
}
```

Figure 7: Protocol buffer message definition for Feature