Beyond the blacklists: Detecting malicious URL through machine learning

Hao Dong, Jin Shang,
David Yu, Chenghuai Lu

Hillstone Networks

About Us

Hao Dong

- Senior Principle Engineer
  @Hillstonenet.com
- Interested in computer system and network security

Jin Shang, Ph.D.

- Chief Scientist & Fellow
  @Hillstonenet.com
- Marathon Runner In-Training

Hillstone NETWORKS

About Us

David Yu

- Distinguished Engineer @Hillstone Networks
- BS and MS in Electrical & Computer Engineering
- Over 20 years experiences in computer networking and security industry

Chenghuai Lu

- Senior Principle Engineer
  @Hillstonenet.com
- Lead efforts in defending against advanced threats in next generation firewall
- Afternoon Tea Party Ping Pong Champion

Hillstone™ NETWORKS

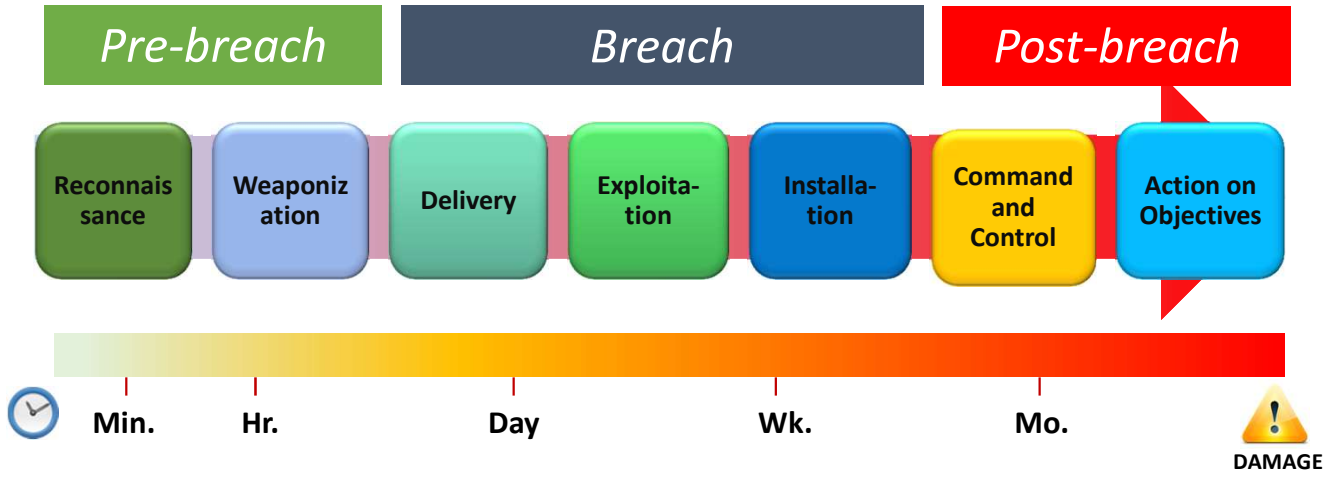
Agenda

- Introduction and problem overview
- Malicious HTTP traffic detection methods
- Using machine learning techniques
- The results
- Beyond the malicious URLs
- Black Hat Sound Bytes
- Q & A

Overview

WHAT: *Problem to Solve*

- To detect malicious URL connections, esp. in post-breach advanced threat protection.



Overview

WHAT: *Problem to Solve*

- Limitation of signature based blacklist
- To know unknowns from knowns

"There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know."

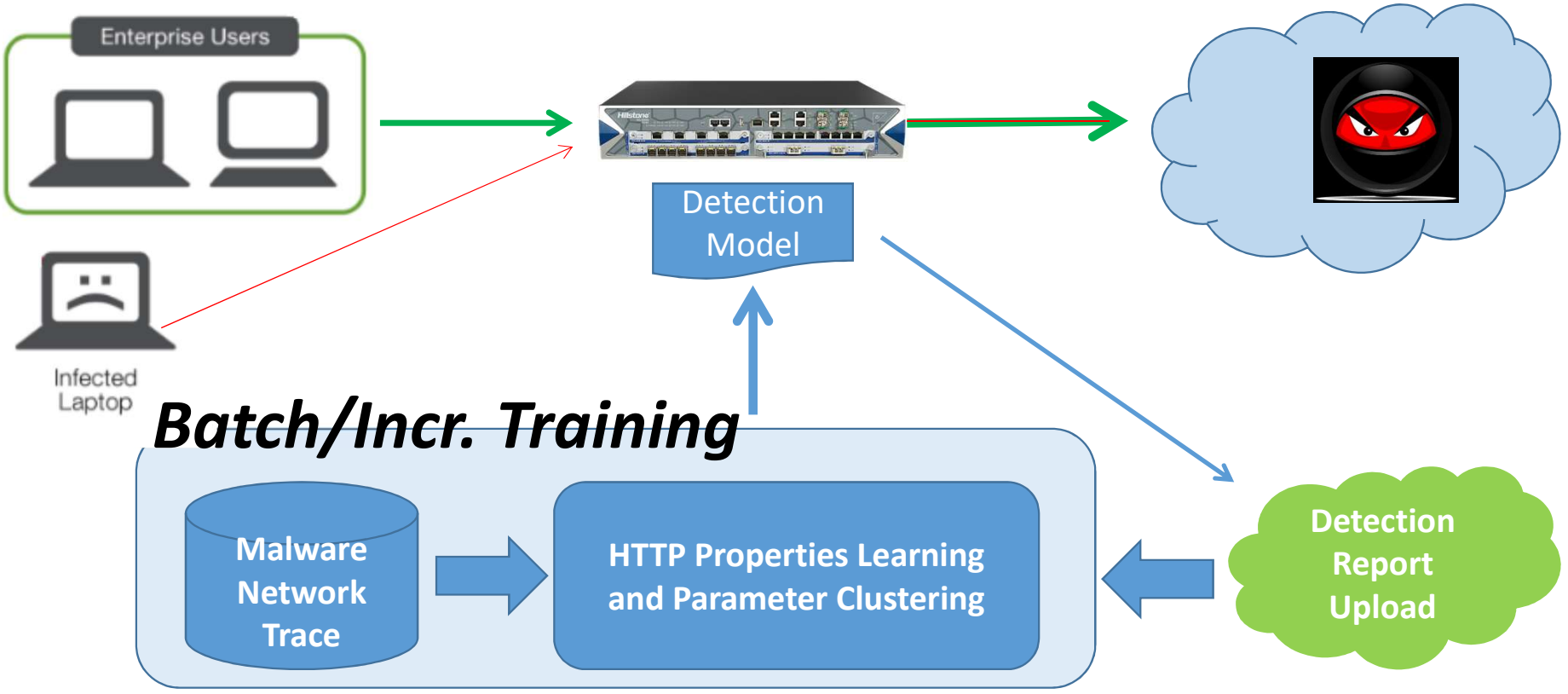
-- Donald Rumsfeld

Overview

WHERE: *Deployment*

- Network-level detection.
- Detection mechanism is typically deployed on network perimeter.
  - E.g. integrated with a NG Firewall appliance.
  - Server-centric intranet deployment is another option.
- Signature/Knowledge-Base are generated offline (or in cloud), using learning methods from malicious sample network trace data feeds.
- Device statistics and feedbacks are sent to cloud, where they are analyzed and used to update signature models.

Overview


Enterprise Users
Infected Laptop
Detection Model
Batch/Incr. Training
Malware Network Trace
HTTP Properties Learning and Parameter Clustering
Detection Report Upload

Overview

HOW: *methods*

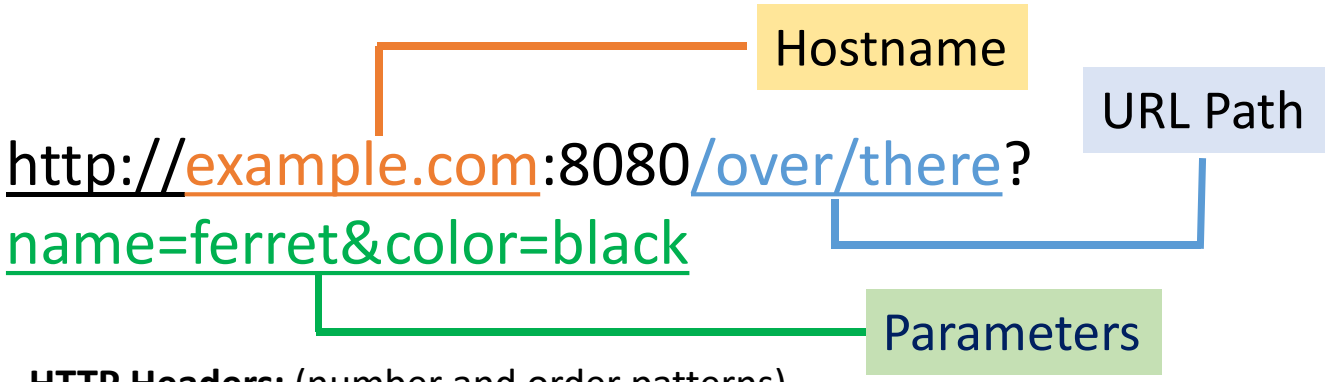
- Based on network traffic trace analysis, mainly HTTP traffics.
  - Compare with system-level behavioral model.
- We examine HTTP host, URL path, header fields, and URL parameters.
  - Focus on solving parameters complex issues.
  - Catch commonality among parameters and be general enough to detect variants.
- Based on URL lexical features
  - vs. contents of pages

Method Overview

- A classifier based on HTTP connections
  - A binary classification problem
  - Multi-class classification (malware families)
- Study features that can separate malicious and benign URLs
- Based on malicious sample network traffic collection
  - Establish mathematical representation of features, and methods to extract similarities.
  - Aggregate commonalties and select malicious commonalties
  - Generate model as output
- Apply the output (model) to real time network traffics, detect known and unknown(derivative) malware connections

HTTP detection modules

- **User-Agent**
- **Host Domain Name**
- **URL Path**
- **URL Parameters**
- **HTTP headers patterns**



HTTP Headers: (number and order patterns)

Content-Type:
application/x-www-form-urlencoded

Origin:
https://www.youtube.com

Referer:
https://www.youtube.com/embed/LSPJaxHMam4

User-Agent:
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36

Put URL Under Microscope

- **Host Domain Name**
  - Known malicious domains
  - DGA (Domain generation algorithms)
- **User-Agent**
  - Known signatures
- **URL Path**
  - http://surusegitimmerkezi.com/administrator/components/com_akeeba/akeeba/engines/proc/mzsystem.php (from ransomware tracker)
- **URL parameters**
  - http://malwaresite.com/contact/info?id=1234&os_type=linux&ver=3.0&state=1
  - Dynamic, extremely large amount of texts hard for efficient signatures

Put URL Under Microscope

HTTP Headers Pattern (browser finger print)

- Mostly useful to reduce false positive
  - Based on common http header fields
  - Number of fields and order of fields of common web browsers
- Observations
  - Malicious connection tender to have less number of header fields
  - Typical web browser traffic have fixed set of headers and orders
- So we set rules
  - Threshold number of header fields
  - Order of header fields match a connection of common web browsers

ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
ACCEPT_ENCODING	gzip, deflate, sdch, br
ACCEPT_LANGUAGE	en-US,en;q=0.8
CONNECTION	keep-alive
DNT	1
HOST	www.whatismybrowser.com
REFERER	https://www.google.com/
USER_AGENT	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537…
COOKIE	…

URL Parameter Features

- Challenges:
  - Dynamic, huge amount of text strings and keep increasing.
  - Unlimited amount of strings, thus huge high feature dimensions.
- We treat parameters as collection of key/value pairs
- The way to reduce dimensions:
  - Sorting method vs. hashing.
  - Sorting to the count occurrences of each string token, and pick most frequent as features, e.g. top 5000. Features after top 5000 will be lost.
  - Hashing include all features, though it may have hashing collisions.

URL Parameter Features

- Sorting method
  - Parameters sorted by hit numbers (total hits, malware hits, etc.)
  - Select top N as feature set
- Hashing method
  - Transform a parameter to multiple features to emphasize field weight.
  - Fixed size of feature space
  - Features are combinations of field value, type and length
    - Numeric - 123456
    - Alphabets - abcdefg
    - NumAlpha - abc123
    - Base64 - d2VsY29tZSBibGFja2hhdCBhdHRlbmRlZXMh
    - Etc.

Transforming parameters

User_id=blackhat_2017

```
User_id=blackhat_2017      ->   1
Alphabet7=blackhat_2017    ->   1
User_id=AlphaNum13         ->            1
Alphabet7=AlphaNum13       ->        1
```

One http URL is converted to a sparse vector

```
..1    1        ..        11 ..          1   1       1       1..
```

Feature Space

Machine Learning

- HTTP URL collections
  - From malware network trace PCAPs
  - From clean legitimate traffic collections
- Choose learning method
  - Supervised vs. Unsupervised
  - Pros and Cons

Supervised Machine Learning

Supervised learning is the machine learning task of inferring a function from *labeled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias).

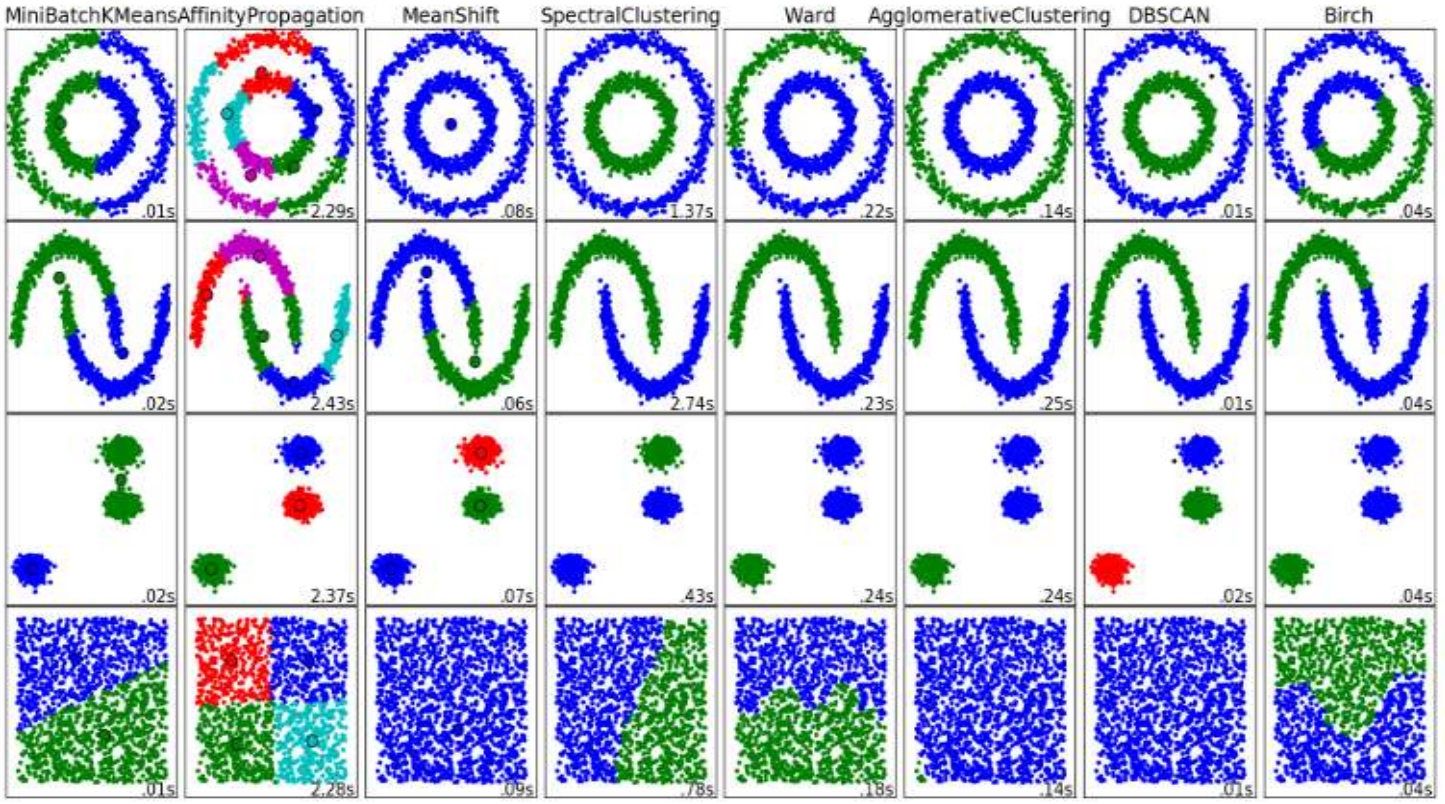
Un-supervised Clustering

- **Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

A comparison of the clustering algorithms

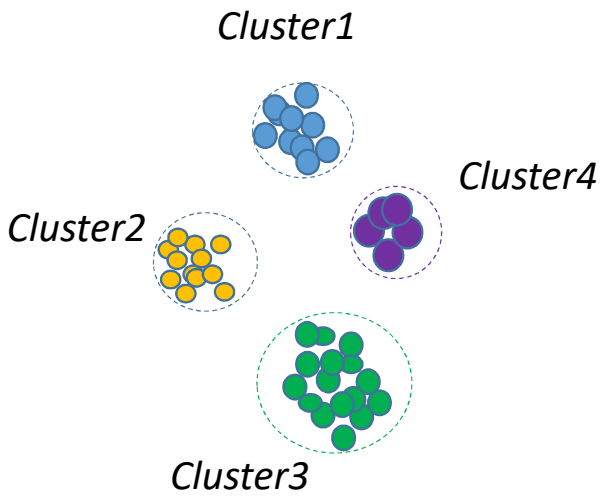
Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters withMiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clustersand n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

A comparison of the clustering algorithms

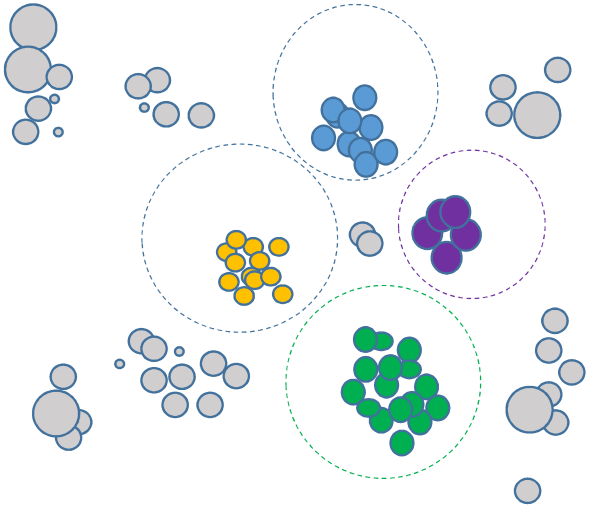


ML Methods Comparison

Unsupervised Clustering



Supervised machine learning

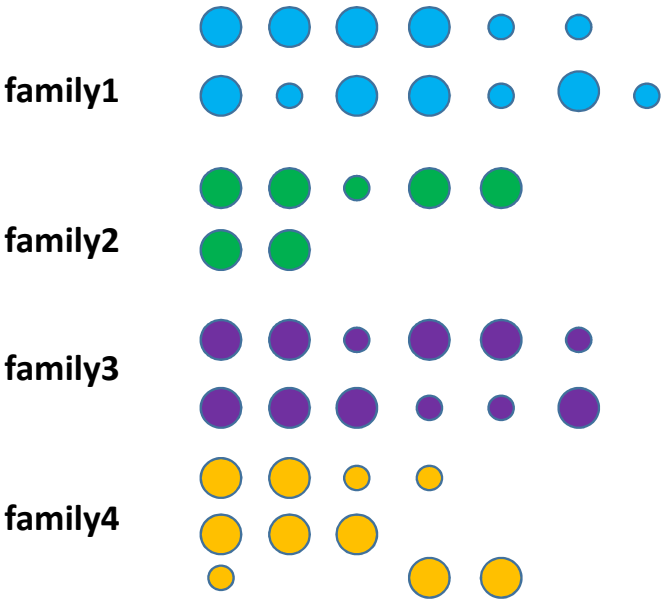


Benign

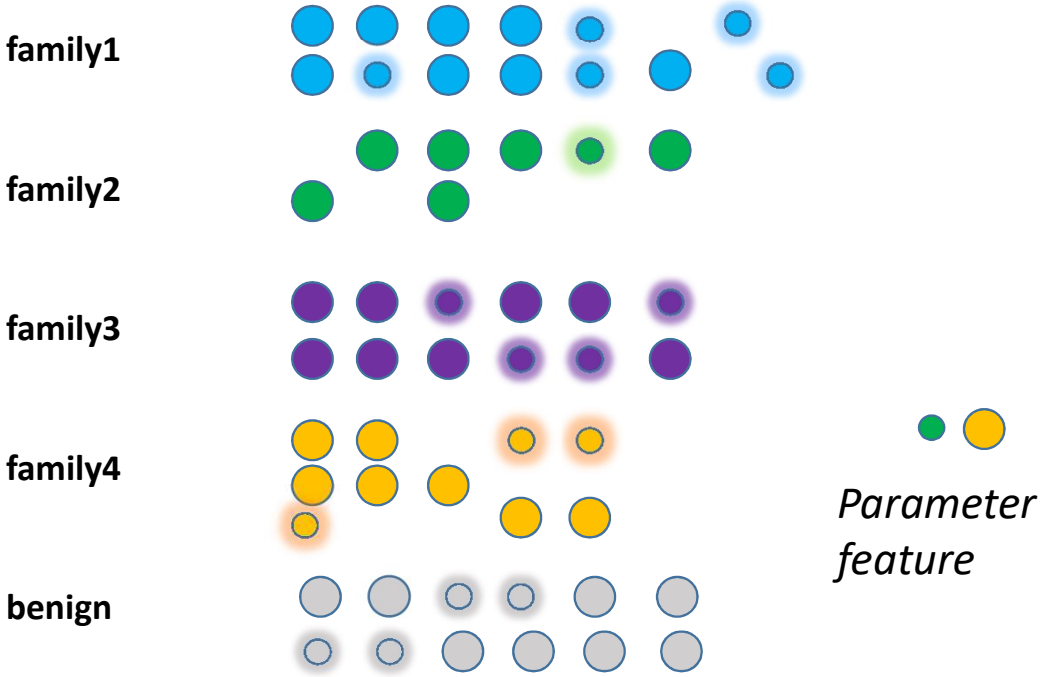
Malicious

ML Methods Comparison

Signatures are commonalities extracted from malware samples

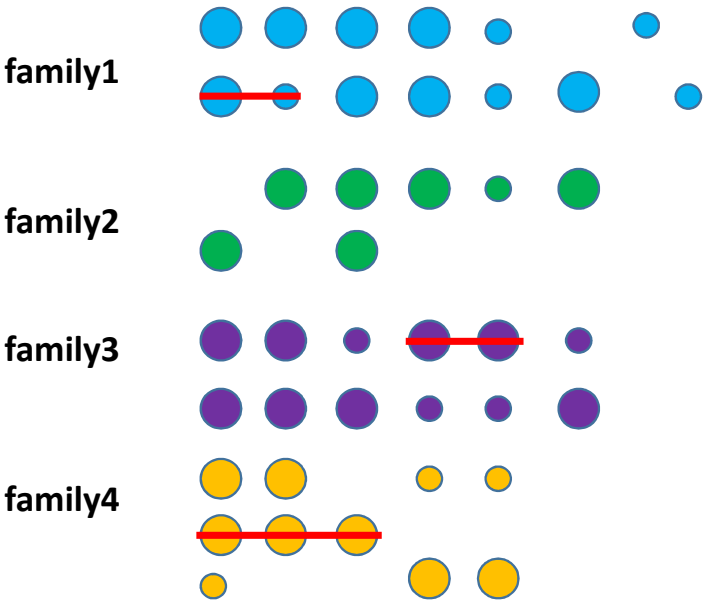


Signatures are learned from both malicious and benign samples

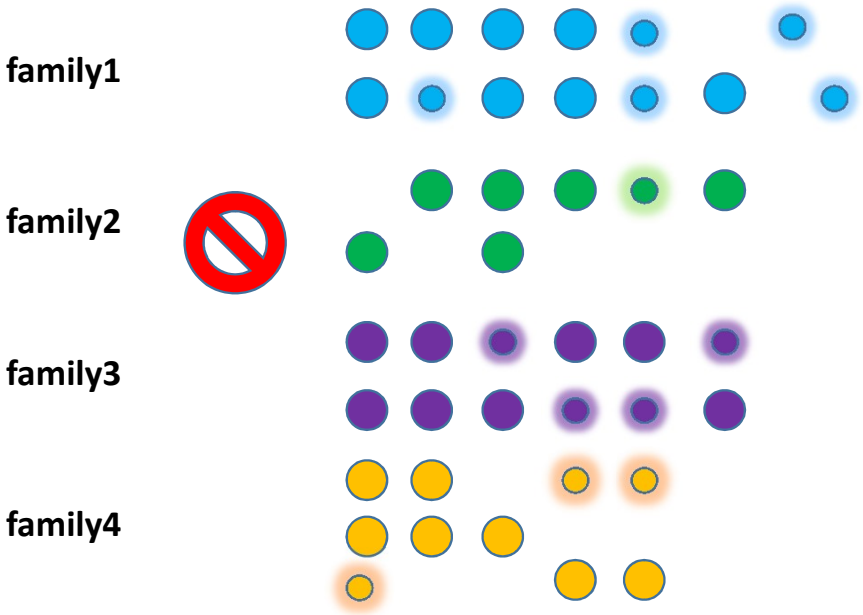


ML Methods Comparison

Adjust models with false positive feedbacks



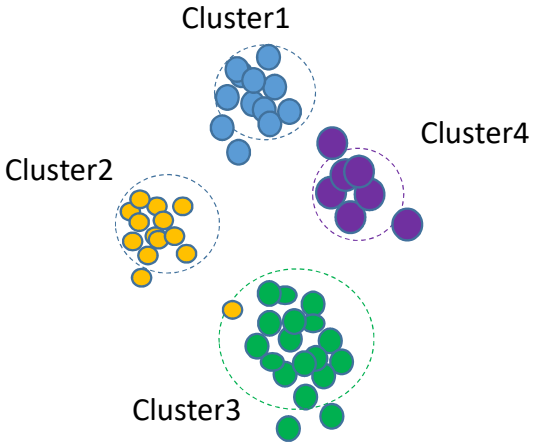
Hard to change models. Need take feedback and re-learn. Or use whitelist



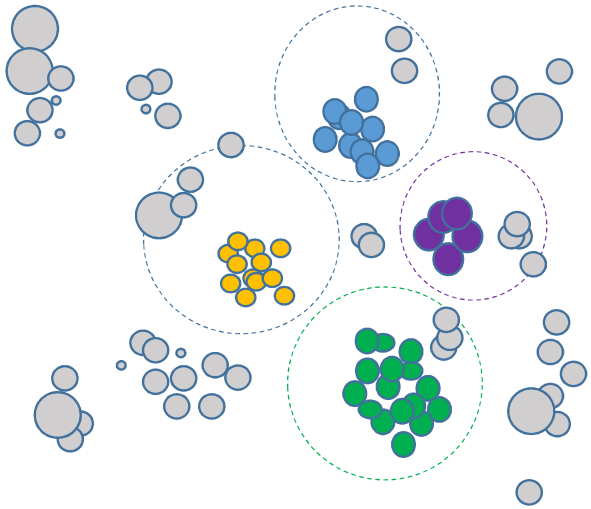
Whitelist to bypass

ML Methods Comparison

Tight compact clusters. High precision, possible low recall

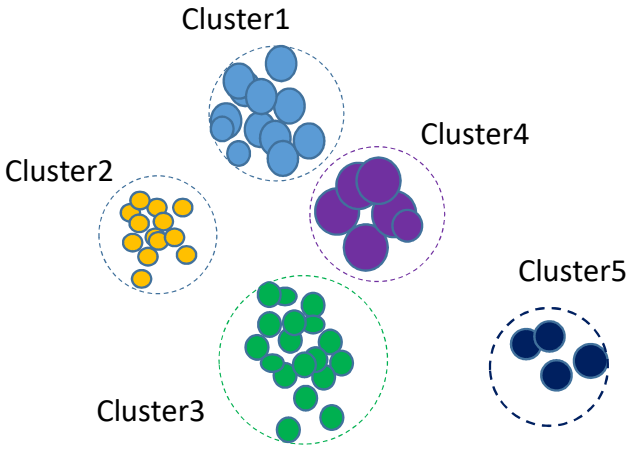


Relatively loose. Low precision, high recall

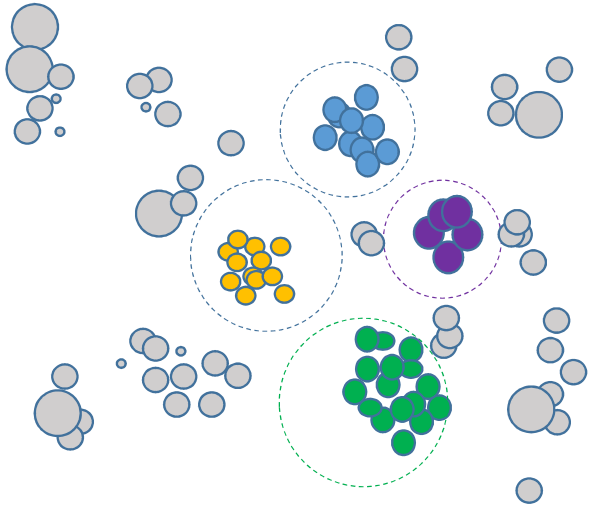


ML Methods Comparison

Easy for incremental update (add and remove)



Add new data and labels, and re-learn the whole model



ML Methods Comparison

- Supervised ML
  - Logistic Regression. Small model size, good for binary classification, e.g. black and white
  - Random Forrest. Big model size, good result for multi-family classification.
  - Ground truth labeling of malware family is important. Model can be biased.
- Unsupervised ML
  - Need only malware (black) samples. The clustering method is to extract common similarities.
  - Sample noises (mis-labeling) can be mitigated.
  - Benign (white) samples are not needed in clustering, but for clusters cleanup (pruning).
  - True positive is easy to explain.
- We chose semi-supervised clustering method for our work
  - Clustering of labeled data

Our Work – Data Source

- Malware Samples:
  - Third party vendor, Hillstone Network security lab
  - Network trace captured in PCAP when run in sandboxes
- Number of samples:
  - 300,000 initial samples, and 10k updates/week
  - ~4500 malware families (variants are merged)
- Malware samples and their PCAPs are labeled by malware families
  - Family variants are grouped under malware family label.
  - Collect HTTP traces from PCAPs. HTTPs inherit malware family labels.

Our Work – Data Source

- Clustering based on URL parameters
  - HTTP GET
- HTTP POST method
  - Some suspicious content is stored in http post body
  - Private format/encoding
  - Some are simple key: value pairs
  - Analyze POST body content for certain patterns. And treat them the same as parameters

Our Work – Data Source

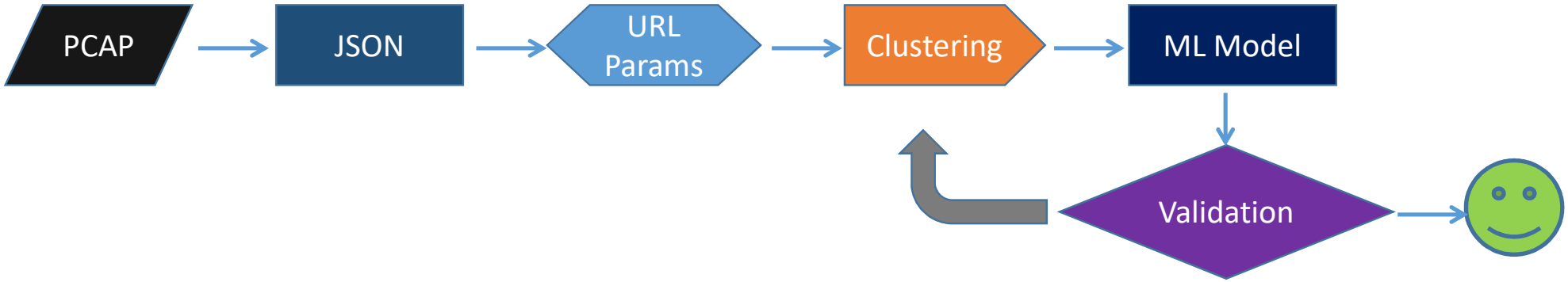
- HTTP POST Examples
  - http://imp.myappz02.com/impression.do

    event=loader_start&implementation_id=min.0.0.30&user_id=ef2443da-1705-4437-8280-878105582da1&adprovider=treasure&source=RedUKBeaconRON&page=Workstation
  - http://stan.mxp30.com/__dmp__/

    data={"msg":";PAYLOAD NOT FOUND","url":"PAYLOAD NOT FOUND","WaitCreateFile":"","MethodTrace":"returnNewPayload","Language":"Chinese","jscript":"","jscript":"Notfind","lno":"","version":"1.7.7","ieversion":"6.0.2900.5512","trace":"U3RhcnRTZXRVbmhhbmRsZWRFeGNlcHRpb25GaWx0ZXJnZXQgcGF5bG9hZFBBWUxPQUQgTk9UIEZPVU5E","osname":"Microsoft Windows XP Professional 5.1.2600 x86","av":"","method":"PAYLOAD NOT FOUND"}

Our Work – Process Flow

- Parsing PCAP to JSON text
- HTTP properties
  - Host, User-Agent, URL path, headers
  - URL parameters
  - Clustering to generate model. Model validation and redo.



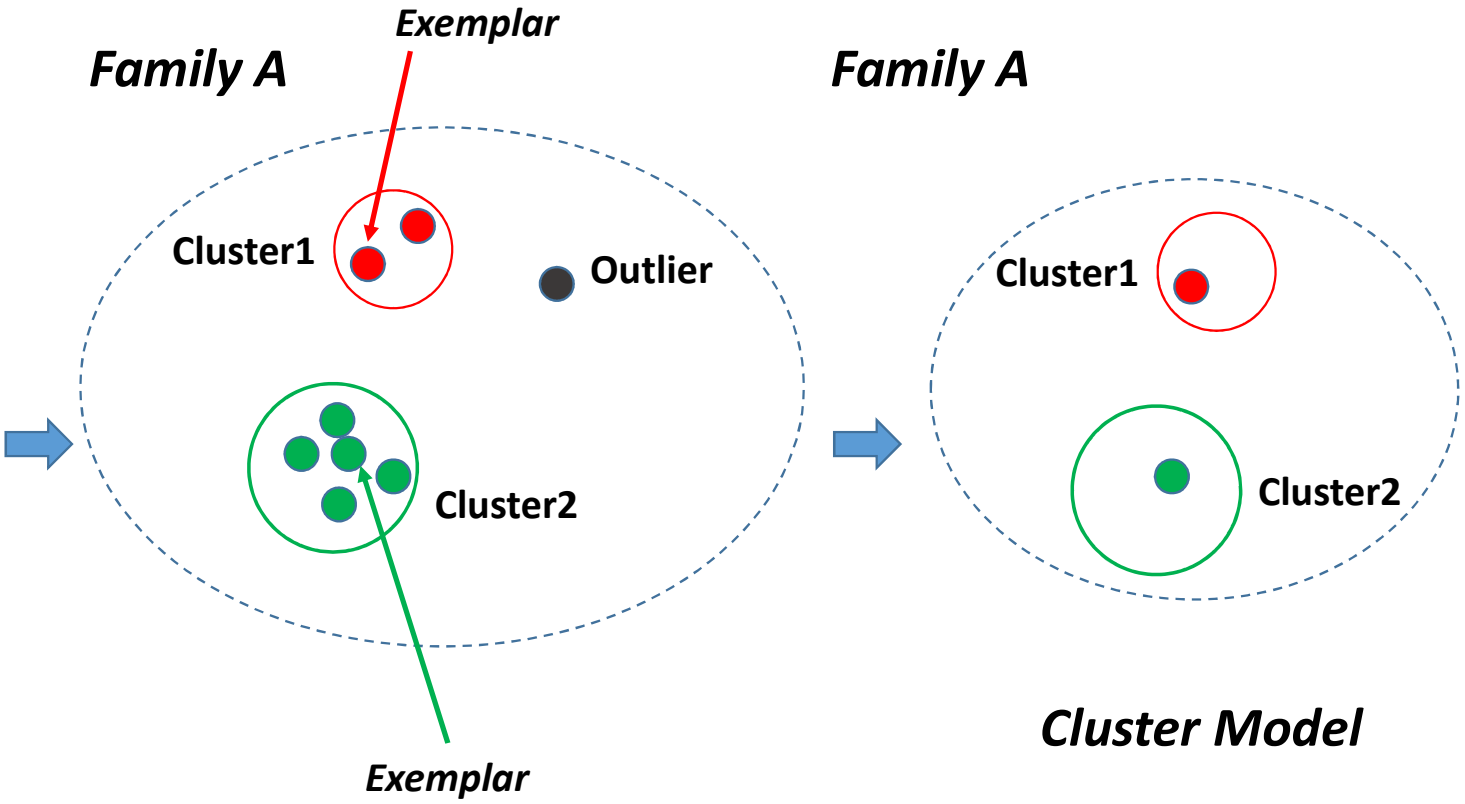
Clustering Method

- Coarse-grained grouping by malware families
- Fine-grained clustering within families based on URL parameters

  Similarity between two http url parameters:
  - Bag-of-words approach: Set of key names and set of values. Jaccard distance between sets
  - Our approach: feature vector and distance function
- Clustering algorithm
  - Density based DBScan
  - Similarity, Distance (or cost) function $D_{i,j} = f(V_i, V_j)$
  - Minimal number of URLs in a cluster. Auto removal of noises
  - Cluster centroid representation (picking an exemplar)

●	1 11 1 111 1…
●	11 1 1 1 1…
●	11 11 1 1 …
●	1111 1 1…
●	1 1 1 11 1…
●	1 1 1 111 …
●	1111 11 …
●	1 1 1 11 1…

Feature Matrix

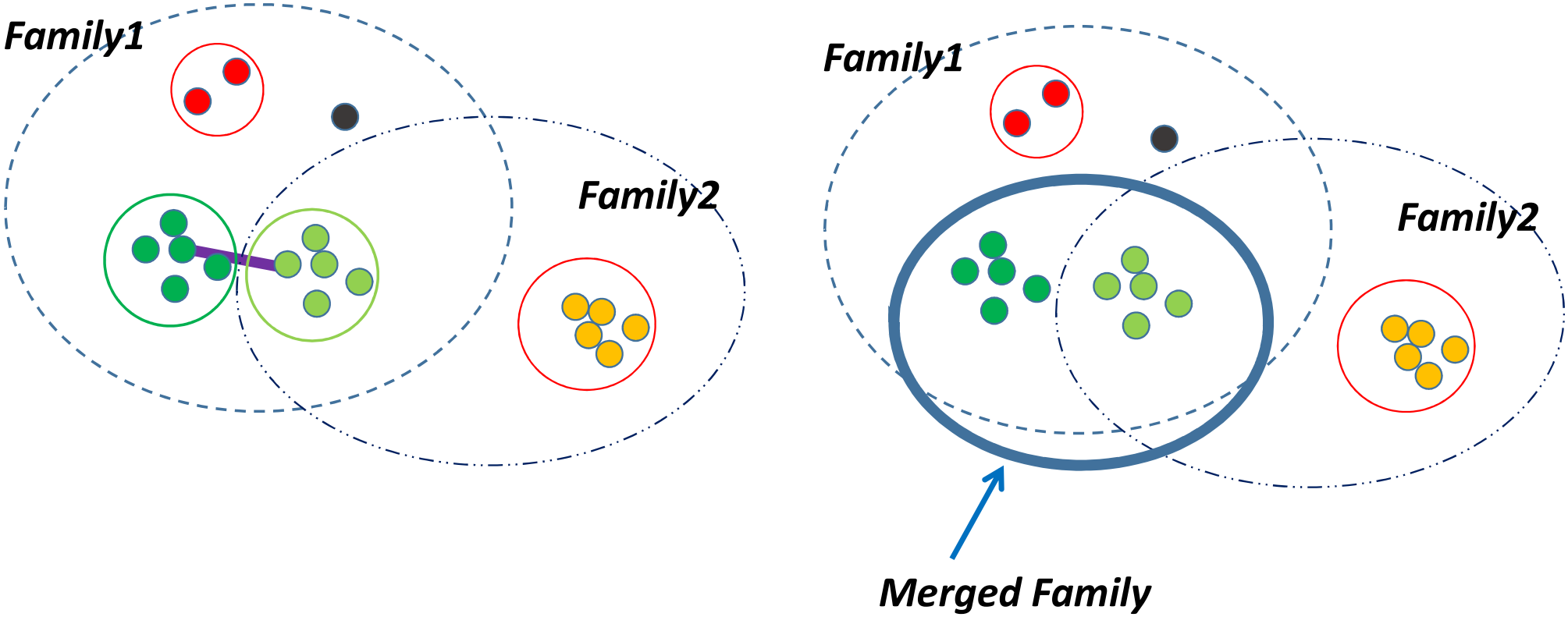


● Sample feature vector

Clustering Method

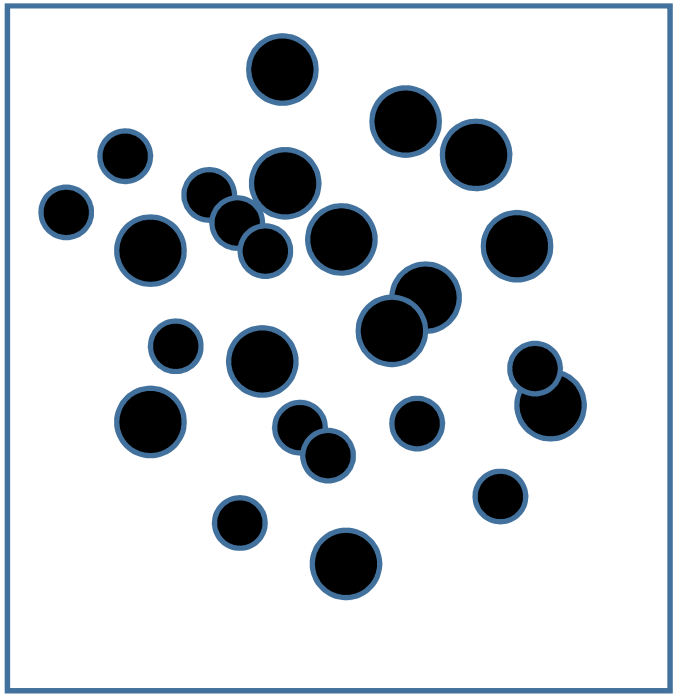
- Cross-family clusters merge
  - The intuition: code re-use in malware
  - For collection of cluster exemplars, do a second-round clustering.
  - For example, f1c1, f1c2, f2c1, f3c1, f4c1, f4c2, are four clusters from families f1, f2, f3, f4.
  - Cluster centroids can be further merged to (f1c1, f2c1, f4c2), (f1c2), (f3c1, f4c1) three clusters
  - Reduce redundancy

Cross-Family Cluster Merging

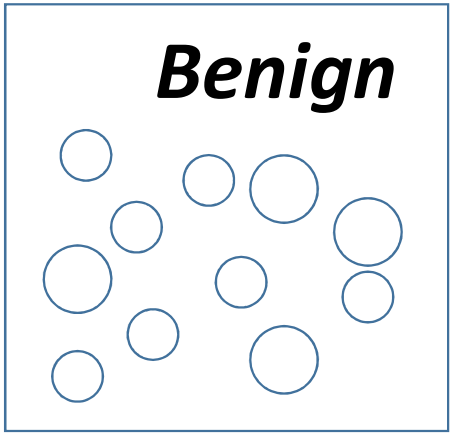


Clustering Method

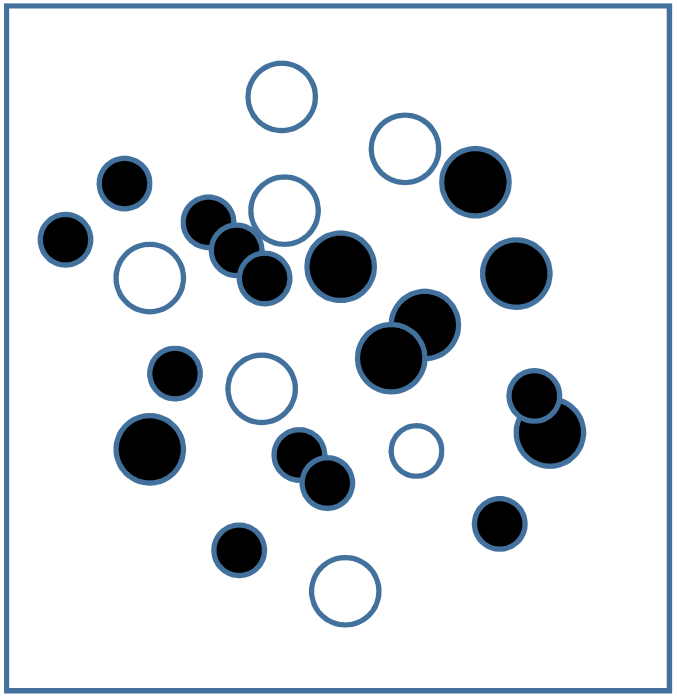
- Cluster Cleanup (pruning)
  - Malware network trace may contain benign legitimate URL connections
  - Clusters are pruned with benign URL parameters
  - Reduce false positives



Clusters



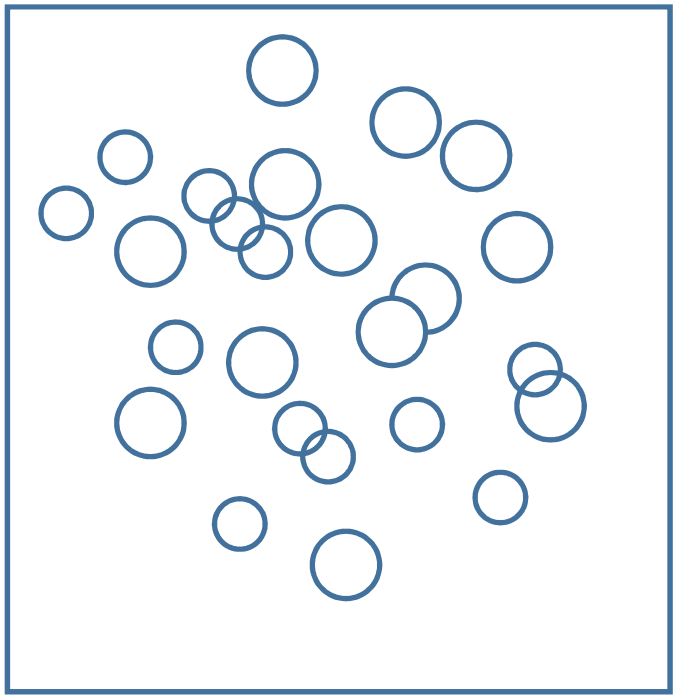
Again, the same distance function matching method



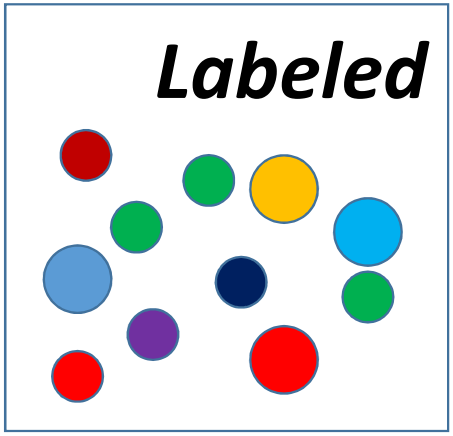
White Pruned

Clustering Method

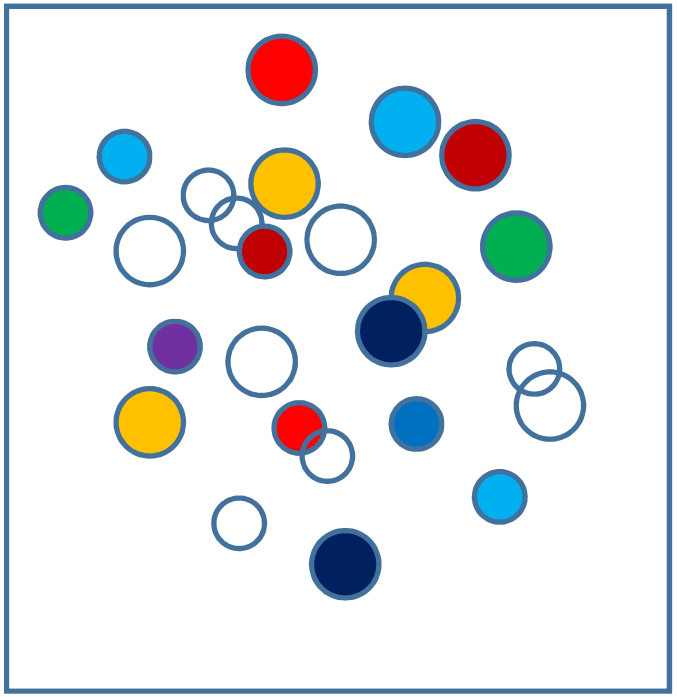
- Cluster Categorization
  - Known URL categories
  - Apply to clusters and mark similar URLs with the same category type
  - C&C, remote access, advertisement, etc.



Cluster Centroids



Again, the same distance function matching method



Labeled Cluster Centroids

Experimental Results

- ~9000 clusters. Each cluster is represented as a sparse vector
- 84% detection rate on ~750K malicious URLs
  - Some malicious URLs are indistinguishable from benign URLs
- Malware family coverage
  - More than 3000 families of malware PCAPs
  - About half have data for clustering
  - Cluster model covers more than 95% of the 1500 families
- New malware variant detected
  - Malware variant detected in deployment before our partner reported data to us

Example of Detection Result

RiskWare[Downloader]/Win32.Donex

Detection Time: 2016/07/13 11:42:34

Domain: api.down.72zx.com

URI:

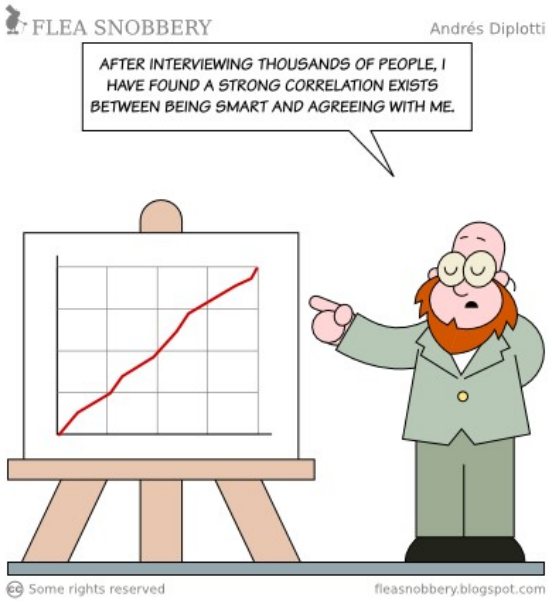
/xml/18?winver=6.2&sdsoft=4&webid=18&softid=29621&ver=1.3.1.14&usesnum=1&mac=vYXbi0zPoms%253D&filename=office2015%25BC%25A4%25BB%25EE%25B9%25A4%25BE%25DF(KMSpico)+_18@29621.exe&errcode=0&userev=0&rnd=4124

Known URL:

http://api.down.72zx.com/xml/34?winver=5.1&sdsoft=8&webid=34&softid=0&ver=1.3.1.14&usesnum=1&mac=zPKt%252BjWzpGo%253D&filename=C6F5BAD88E23D89DE798C4D6FFCFA789.BC492044&errcode=0&userev=0&rnd=9324

Correlations: Higher Confidence

- Local correlations
  - Within HTTP connection: HTTP Method, Headers, User-Agent, Path, host domain name
  - Spatial correlation. Within the network, number of the same incidents in a period of time
    - E.g. more than 20 hosts made the same connection in one hour
- Higher level contextual correlations
  - Short term (within minutes):
    - DNS (number of queries, NXDOMAIN)
  - Long term (hours and days):
    - Suspicious file download
    - Suspicious Email behavior
    - Large amount of file transfers



Work together: Comprehensive Security Arch.

- Multiple defender architecture
  - Traditional: AV, IPS
  - Next-Generation: AppID,
  - Intelligence: Behavioral ML, AI, cloud

Limitations and Future Work

- The Dark Side
  - Encryption
  - Lack large collection of clean benign traffics
  - JSON/RESTful API
  - Trend in malware families (less HTTP use?)
- Malware samples
  - Label accuracy
  - Sandbox execution
  - Evasive technologies

Black Hat Sound Bytes

- Network level behavioral analysis based on HTTP traffic has benefit in detecting compromised hosts.
- A novel way to extract URL parameter feature dimensions, and an unique method to transform infinite dimensions into limited feature space.
- Semi-supervised clustering method with filtering and post-processing, demonstrates strength in precision, model size, variance coverage.

Thank you!