Self Evolving Detection System

Q2 / 2017

Early AI Defined

Alan Turing called an infant's mind an '**unorganized machine**' in 1947

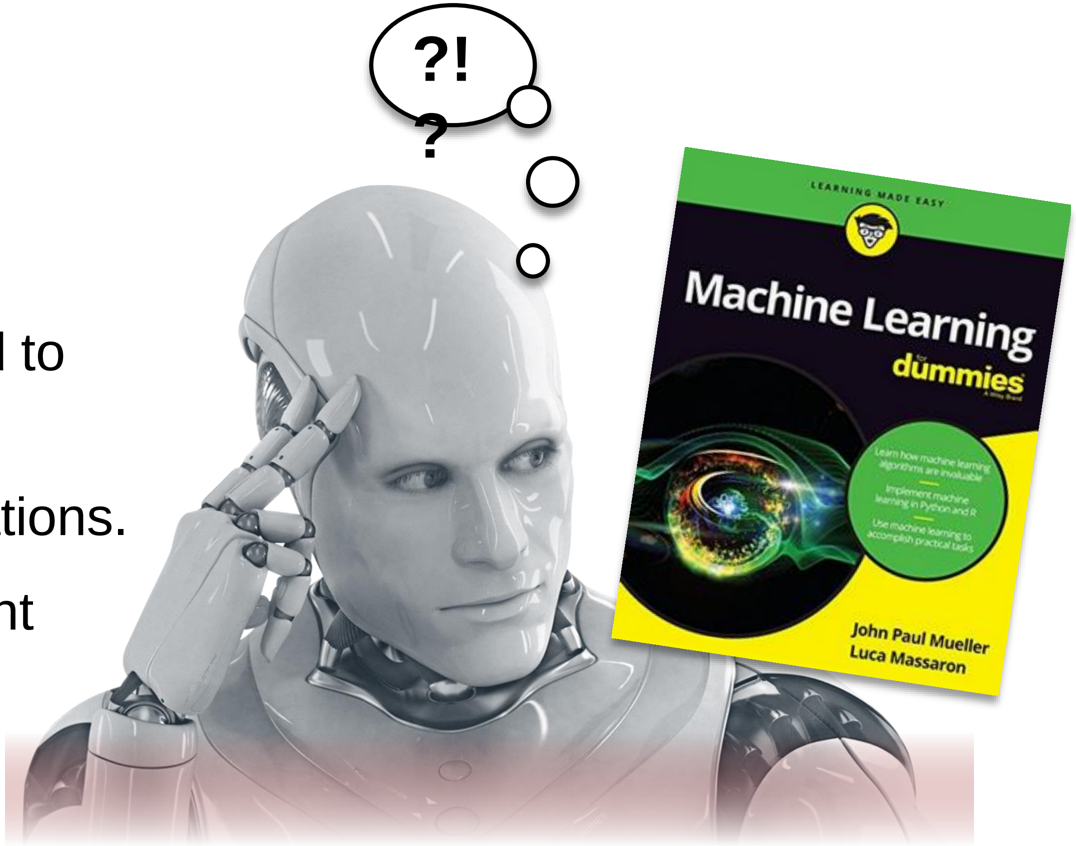
Created early definitions of machine learning

- » First type (A) consists of simple NAND (negative – AND gates
- » Second type (B) is combination of A types with modifiers added – results in weighted input/variable output method
- » Saw the need for:
  - Seeded solution set of accurate or known potential output
  - Population of variably weighted pieces or functions
  - A method for culling out the worst solutions while retaining the best

Major inhibitor of his research – was far ahead of available capabilities in terms of computing power.

Types of Problem Solving

- **Supervised Learning** – Using known solution sets to embed proper functions and create proper output.
- **Clustering** – group according to similarities.
- **Dimensionality Reduction** – deductive reasoning.
- **Structured Prediction** – random fields are analyzed to predict according to defined output probabilities.
- **Anomaly Detection** – input does not match expectations.
- **Reinforcement Learning** – action on an environment triggers an observation resulting in a defined state.

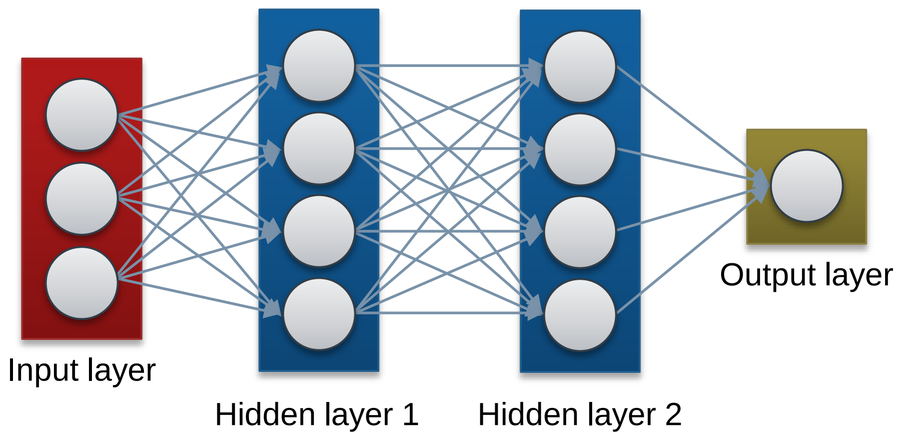


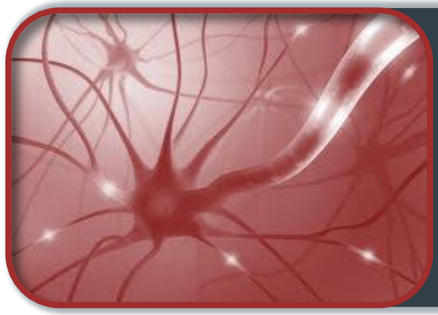
Artificial Neural Networks (ANNs)
Large collections of simple interconnected nodes (neurons), each with a weighted input and output value.

Type of ANN - Multilayer Perceptron

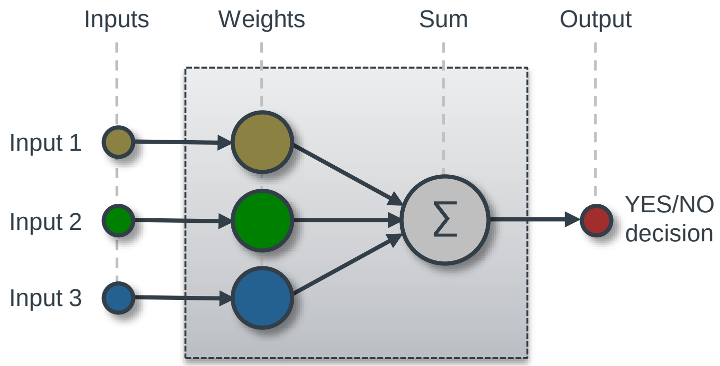
- Consists of three or more layers
  - » Input layer
  - » One or more hidden layers
  - » Output layer
- Layers are made of up nodes
  - » Connected to every node in the previous and subsequent layer
  - » Provide discrete processing of input information (files and features)
  - » Produces an output value based on inputs, function, and weighted valuation

The Multilayer Perceptron approach provides deep machine learning capabilities.





MP behavior is similar to human neurons - if input is strong enough, signal is passed according to weighted value



The Current Issue - Volume

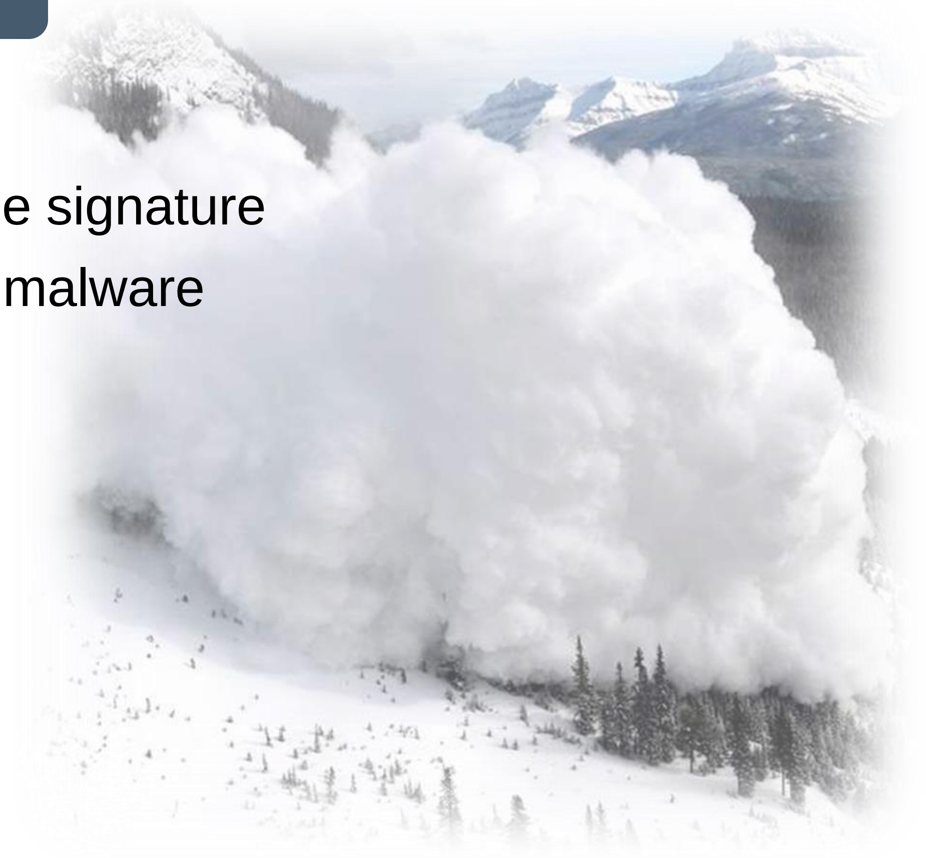
2M malware samples arrive daily & ingested for analysis

CPRL (Gen1)

- Patented algorithm used to identify malware variants from one signature
- Allowed for smaller antivirus database to detect polymorphic malware

Auto CPRL (Gen2)

- Creates signatures automatically, reduces analyst workload
- Integrated sandbox used for behavior analysis of samples
- Creates 500 to 2,000 unique signatures/day
- Still resulted in large levels of manual effort



Requires high levels of manual analysis, review (QA) of auto-generated signatures High value research labor input, potential signature provisioning delays

Fortinet Direction

- Create an advanced neural network structure
  - Allocate researchers to new and advanced areas of study
  - Expand current unknown threat management capabilities
- Create new techniques for searching beyond patterns in code and malware behavior
- Discover obtuse malware patterns
  - Hardware activity, electrical current allocation, memory usage anomalies
  - Malware insertion, operation, and malfeasance behaviors
  - Leverage supervised learning - multilayer perceptron

Going Beyond the Current and Common Uses of Machine Learning

Data Mining | Computational Statistics | Email Filtering | Optical Character Recognition (OCR)

Computer Vision | Breach Detection | Data Analytics | Predictive Behavior

Fortinet's Self Evolving Detection System - SEDS

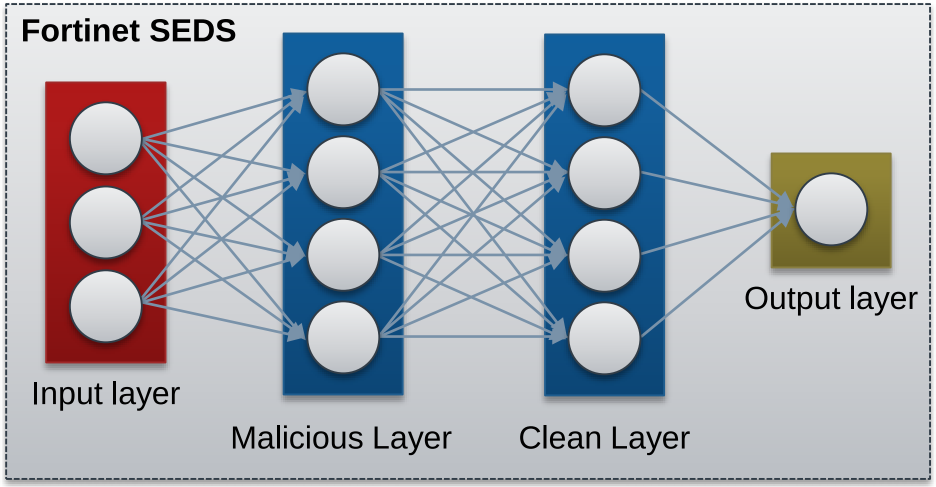
4 Layer Architecture

1 = process the input file

2 = 1.9 billion nodes analyzing potential malicious features

3 = 2.9 Billion nodes analyzing files for clean features

4 = output or decision layer (1 = malicious , 0 = clean)



Consists of separate layers for either malicious or clean feature processing
Mathematical models compare samples and features to decide output

Features and Input Effect

- Features = point observable characteristics
- Identified features are sent to the knowledgebase repository of each layer
- Quality is critical
  - Provides more accurate determination of file status
  - Fortinet leverages internal legacy samples (~.5PB) to create features from samples
- Each feature is weighted to assist in decisions
- Feature weighting can change over time
- Weighted features are processed within nodes
  - Output is weighted, based on presence of features
  - Weighted output passed to next layer for continued processing

Feature Weight Algorithm

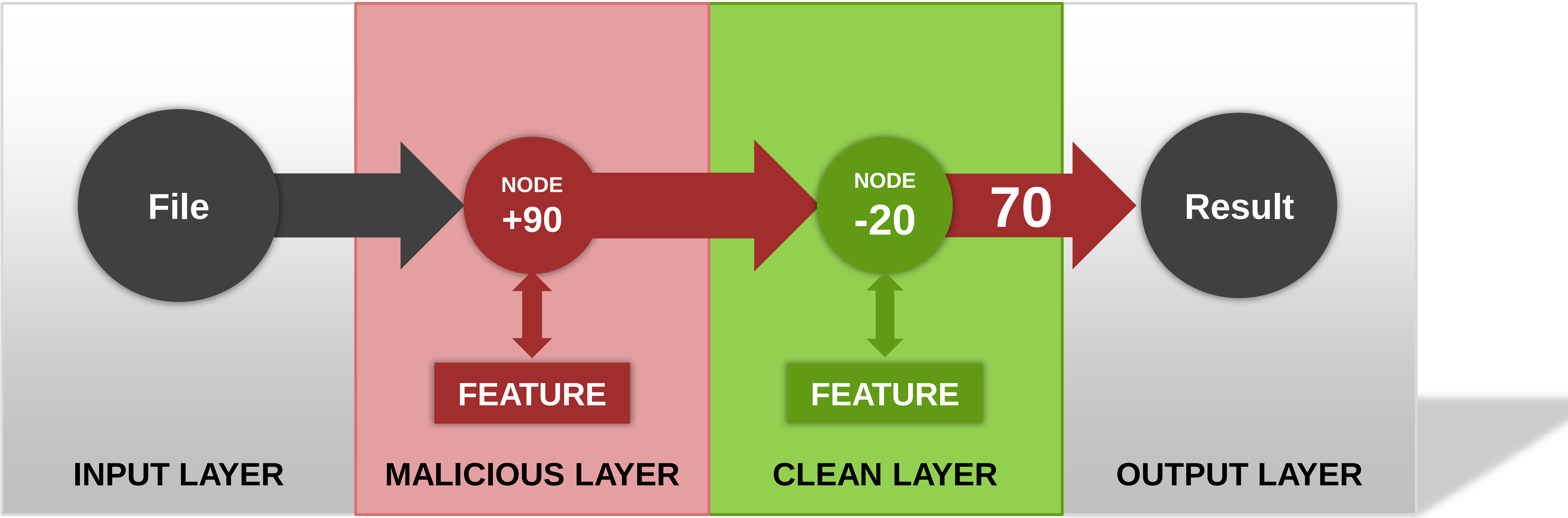
f - feature

w - weight

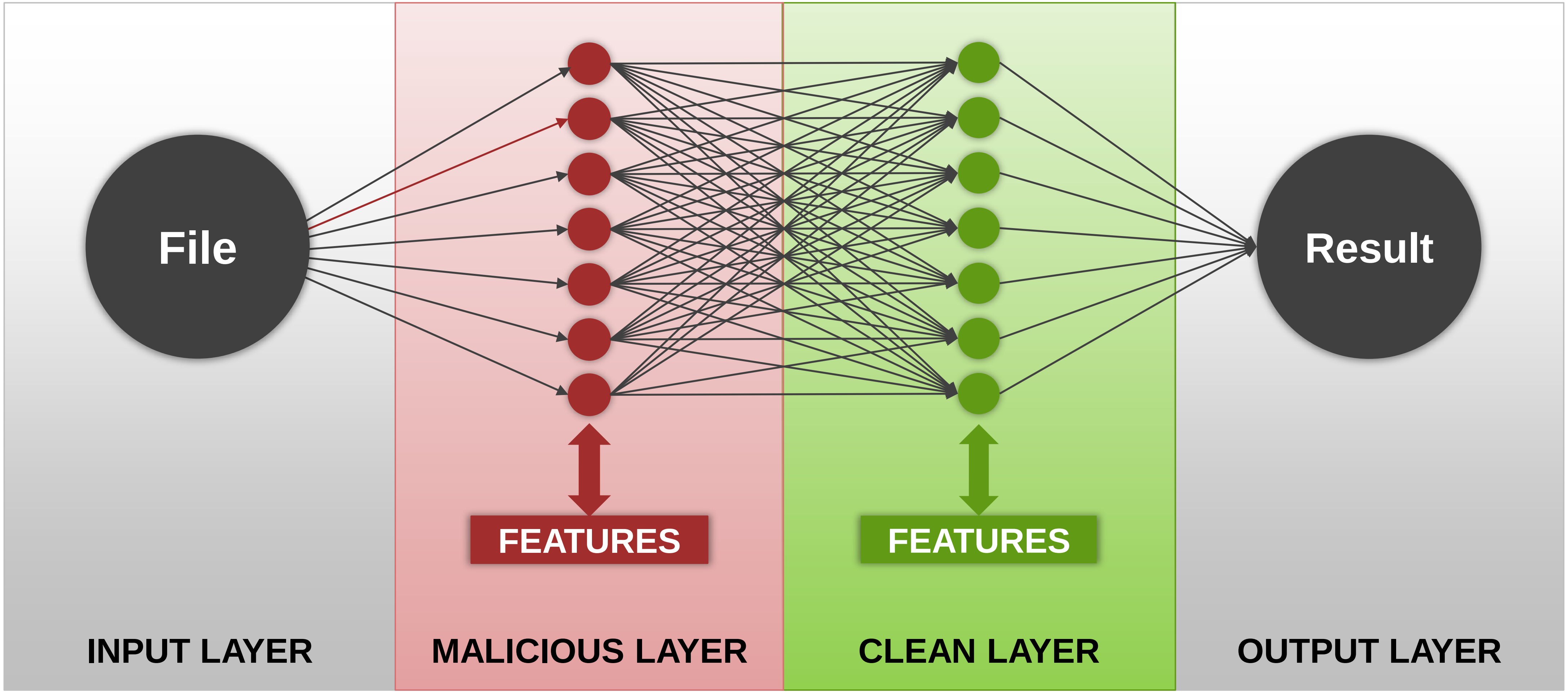
Func(f1*w1+f2*w2+...+fn*wn) -> {0,1}

Features, Nodes & Weights – Single Instance

1. We start with an input file – malicious or clean
2. Feature presence is calculated, weighted and passed forward to the next node
3. The analysis is repeated using the next layer feature, then passed to the next node
4. Result – the overall probability based on a score of feature presence



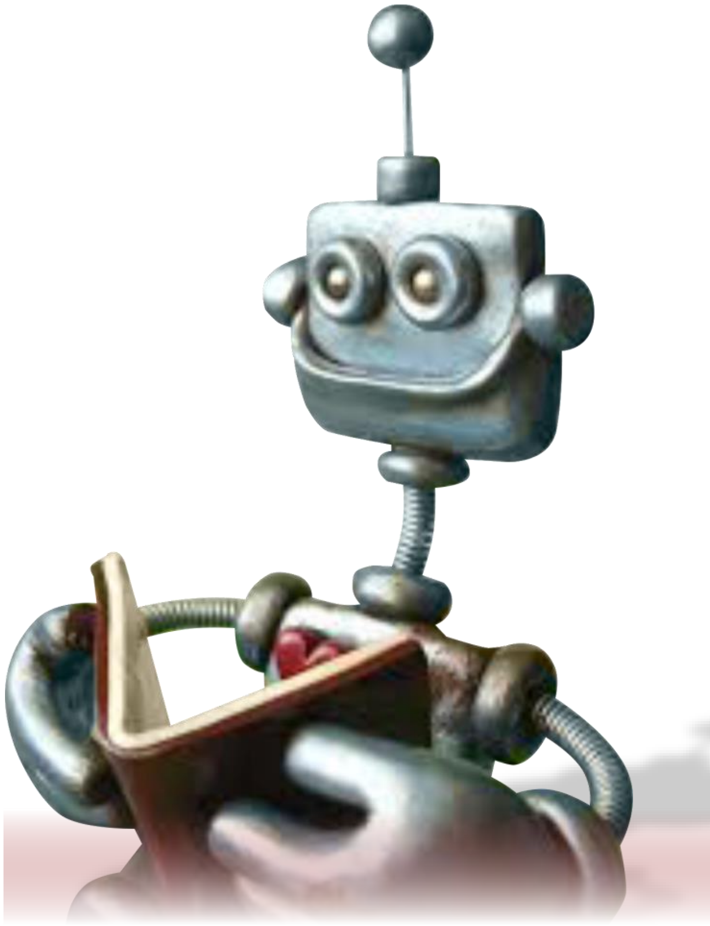
Features, Nodes & Weights – Multiple Instance



Output is a result of 1.9B x 2.9B individual node computations.

Layers + Nodes + Features = Learning

- System is fed initial data sets for analysis
  - » Supervised machine learning approach
- Information (features) extracted during the learning phase. Examples:
  - » Patterns of data present within the files
  - » Behavioral patterns during activation
- The system learns to weight features based on
  - » Surety of indicators ('tells')
  - » Frequency of observation



Source accuracy of the population input A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

Features, Databases and Potential Performance Issues

Active features can grow only so much

- Training set of features resulted in 10GB
- Can quickly fill the node capacity
- The data does not grow linearly
- Most seen features are weighted more heavily
- Features may get eliminated and new ones created

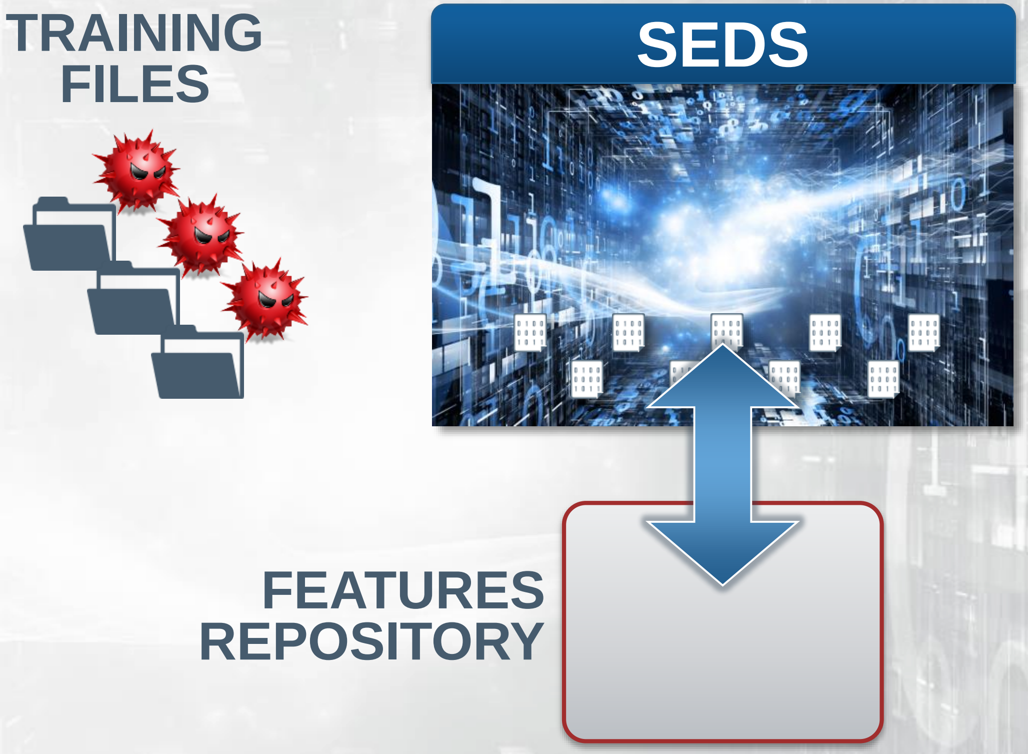
Current capabilities – 50 samples per second for feature analysis

Speed improves as the system learns to seek heavily weighted features

Continued learning - feature weighting changes as frequency and probability vectors readjust

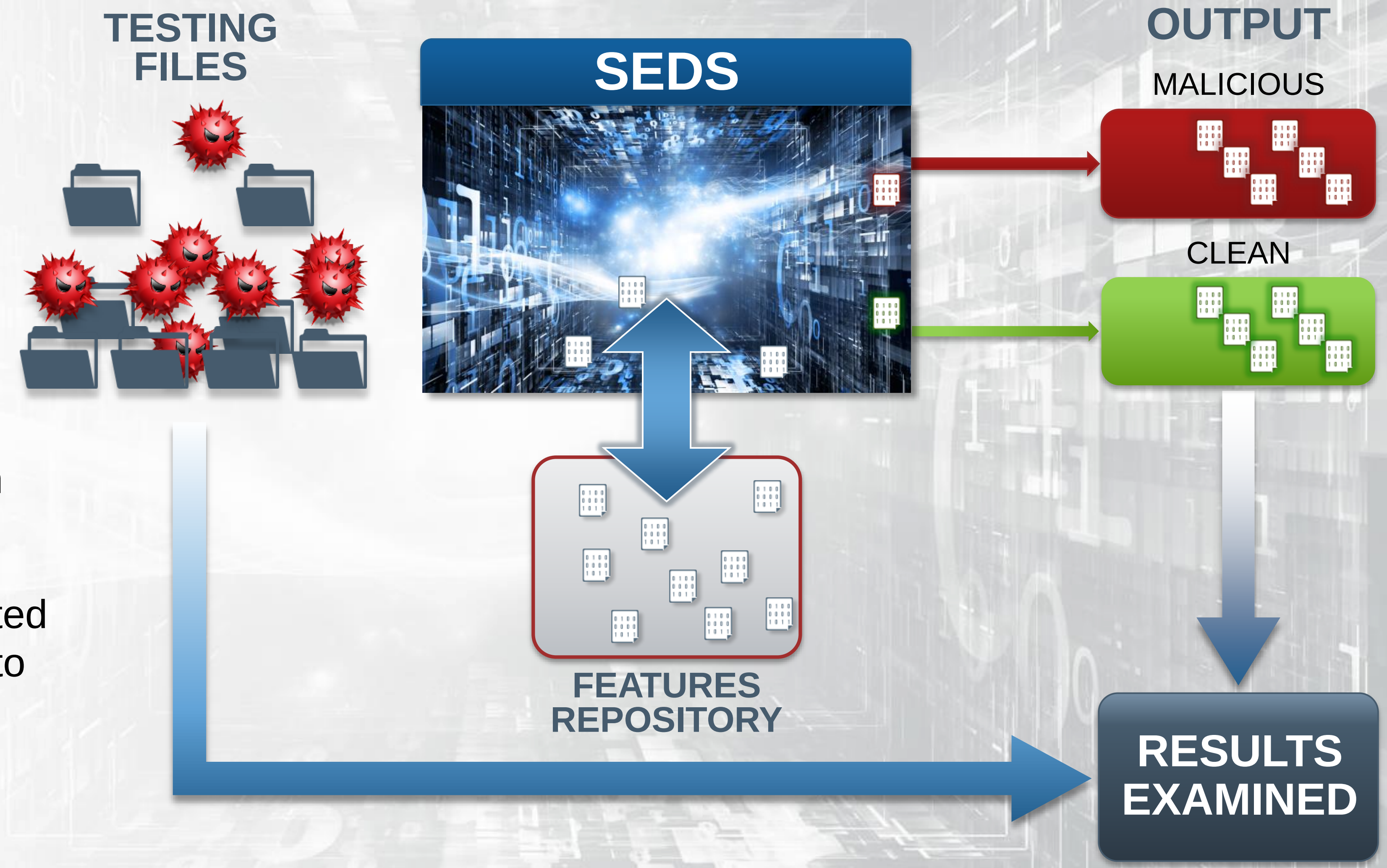
System Training

1. We start with SEDS and an empty feature repository
2. A training set of files are input, consisting of clean and malicious files. Files are labelled for initial training
3. SEDS logic determines commonality of files and builds a set of features.
4. Features are modified as the system learns (weighting values, next phase)

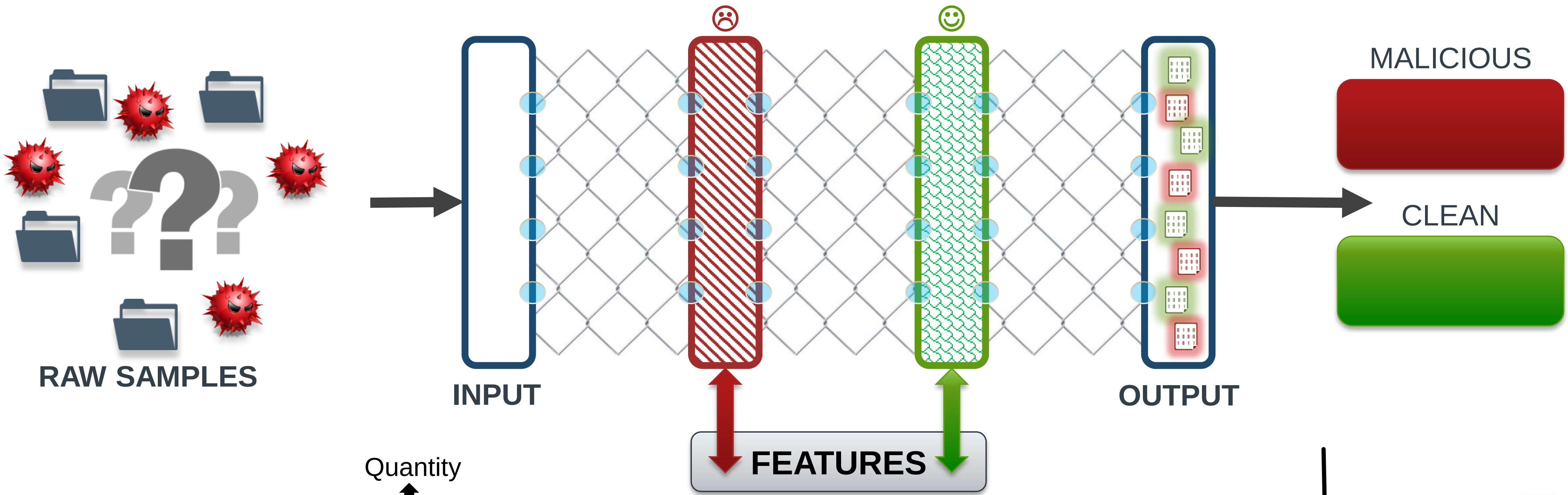


System Testing

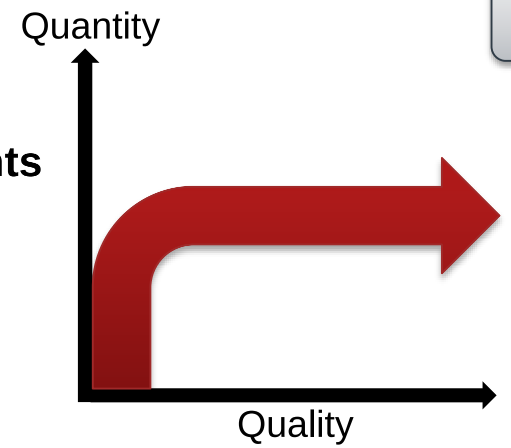
1. Test samples are selected and input to the system
2. Using the feature repository, samples are analyzed
3. As this occurs, existing features may get modified or others added
4. The system determines clean or malicious output
5. Output is compared to expected results. If not accurate, reset to known point and retrain.



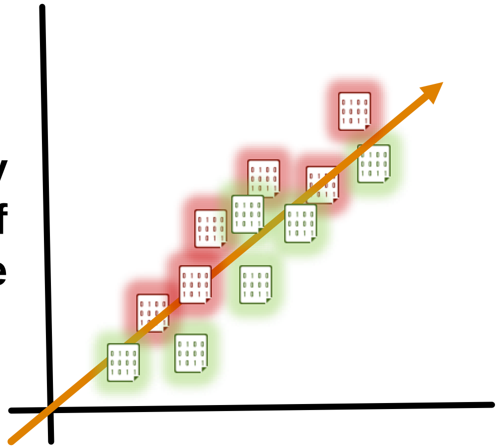
SEDS in Operation



Feature Set Improvements
- Quality
- Stabilized Number
- Weighting Confidence



Continued Accuracy to a High Degree of Confidence



Antivirus Evolution

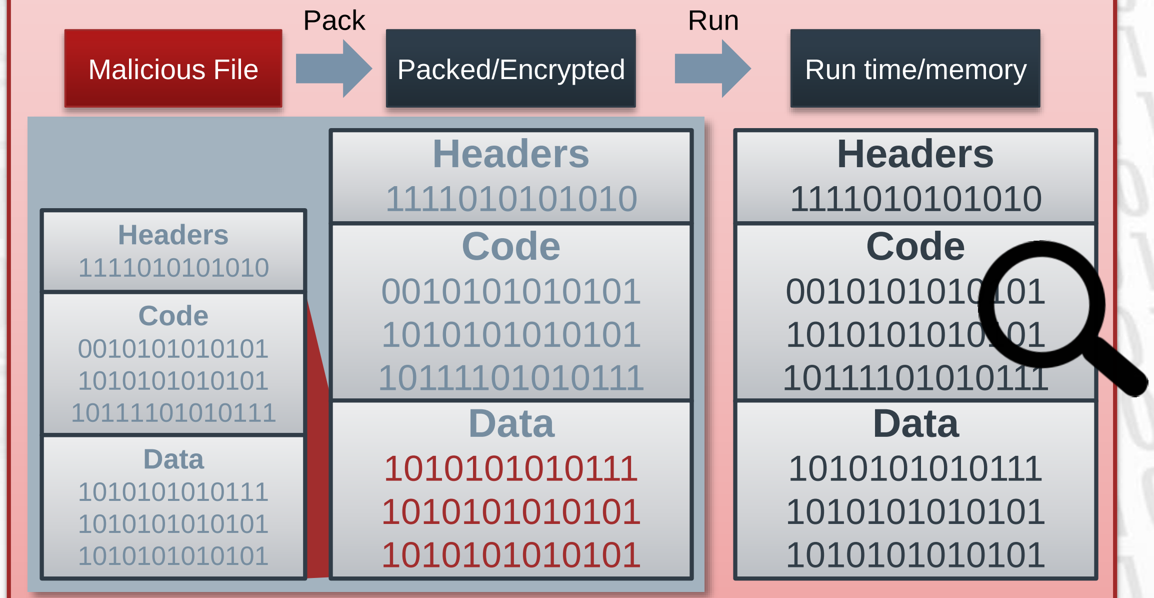
LEVEL I

- Simple MD5 / SHA 56 computations
- Resulted in large DBs for file comparisons
- One signature – one piece of malware
- Reactive and non-responsive to mutations

C:\Md5sum malware.exe
5e3830ee3282a53920e00784fec44cfd (malware.exe)

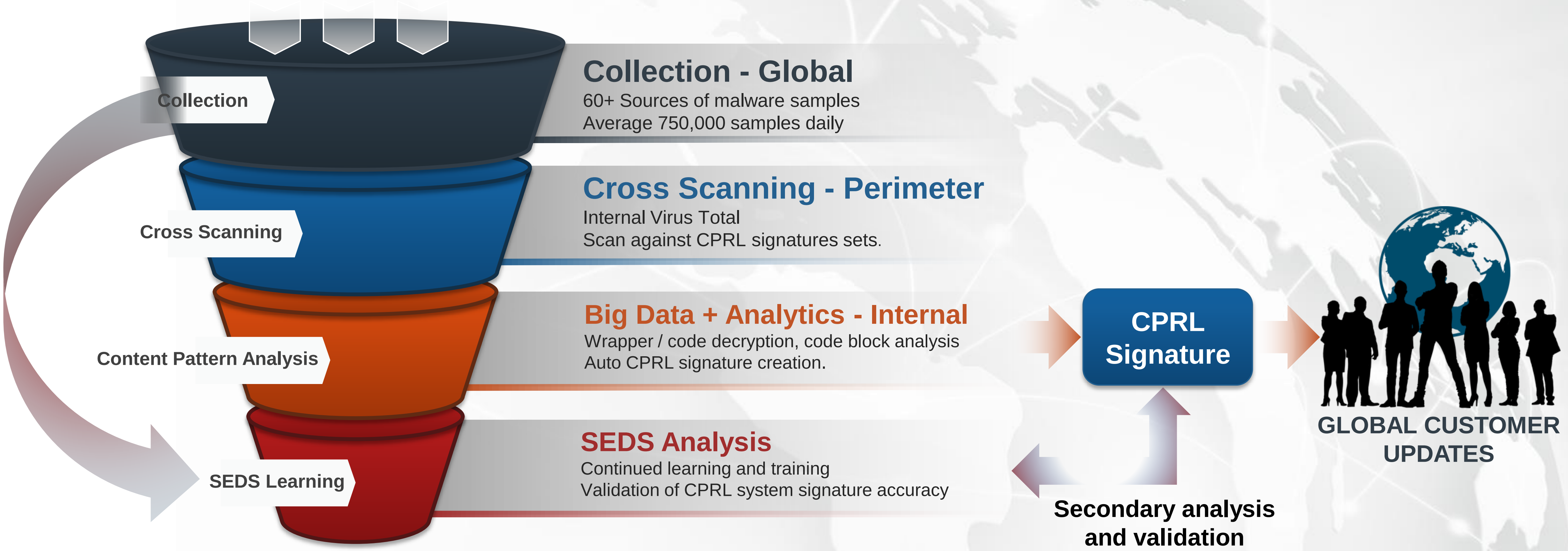
LEVEL II

- Content Pattern Recognition Language
- Looks at wrappers and payload for repeats
- Handles large volumes of permutations
- Proactive in nature



SEDS – CURRENT OPERATIONS

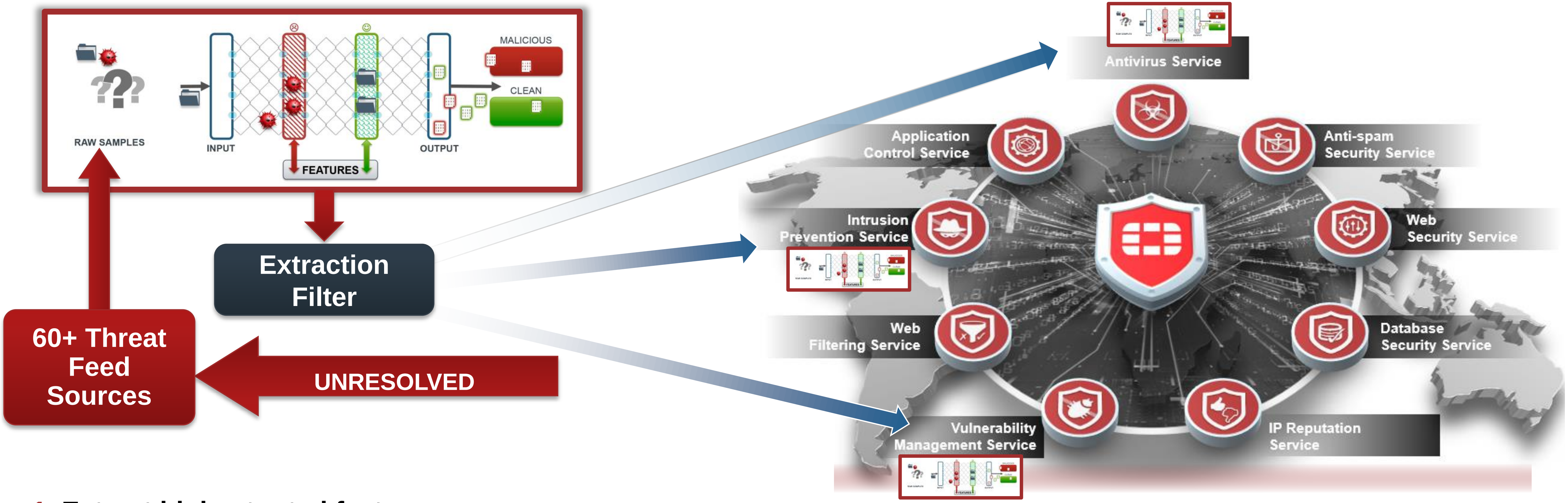
- **Augmenting pattern recognition and automatic signature creation technology**
- **Continued learning and feature improvement – higher accuracy of the system**



SEDS – Next Phase

Fortinet Customer Secure Fabric

UTM with right-sized SEDS implementations



1. Extract highest rated features
2. Customer Fortinet Secure Fabric with embedded SEDS
3. Distribute features to local SEDS: Client, FW, sandbox, IPS, etc.
4. Suspicious/unknowns sent for additional analysis

RESULT: Active, predictive intelligence at our customer sites

FORTINET®