DiDe

build a pattern-based detection module from scratch

About us

- Zhanglin He (@zhanglin2022), a principal researcher in Palo Alto Networks, focuses on web security, sandboxing, and cloud security. Speaker in KCON.
- Royce Lu (@RoyceLu), a distinguished research engineer in Palo Alto Networks, focuses on kernel security, system vulnerability, machine learning, and cloud security. Speaker in BlackHat, Virus Bulletin, and KCON.

Index

- Background
    - What is pattern-based detection module
    - How it works
    - What’s the limitation
- DiDe
    - DiDe idea
    - Understand VirusTotal report
    - Before rule generation
    - After rule generation
    - Build a your own detection module
- Issues and workaround

Background

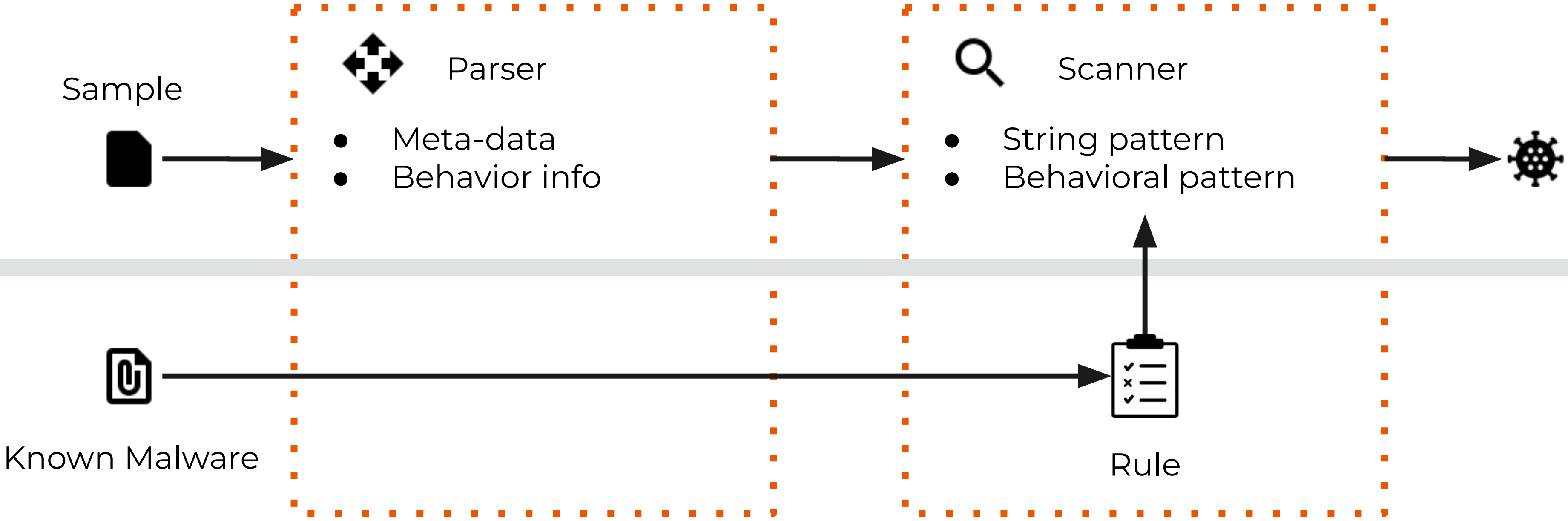
Malware Detection

- Target
  - Executable files
  - Documents
  - HTML/javascript
  - etc.
- Detection method
  - Pattern-based
  - Machine Learning

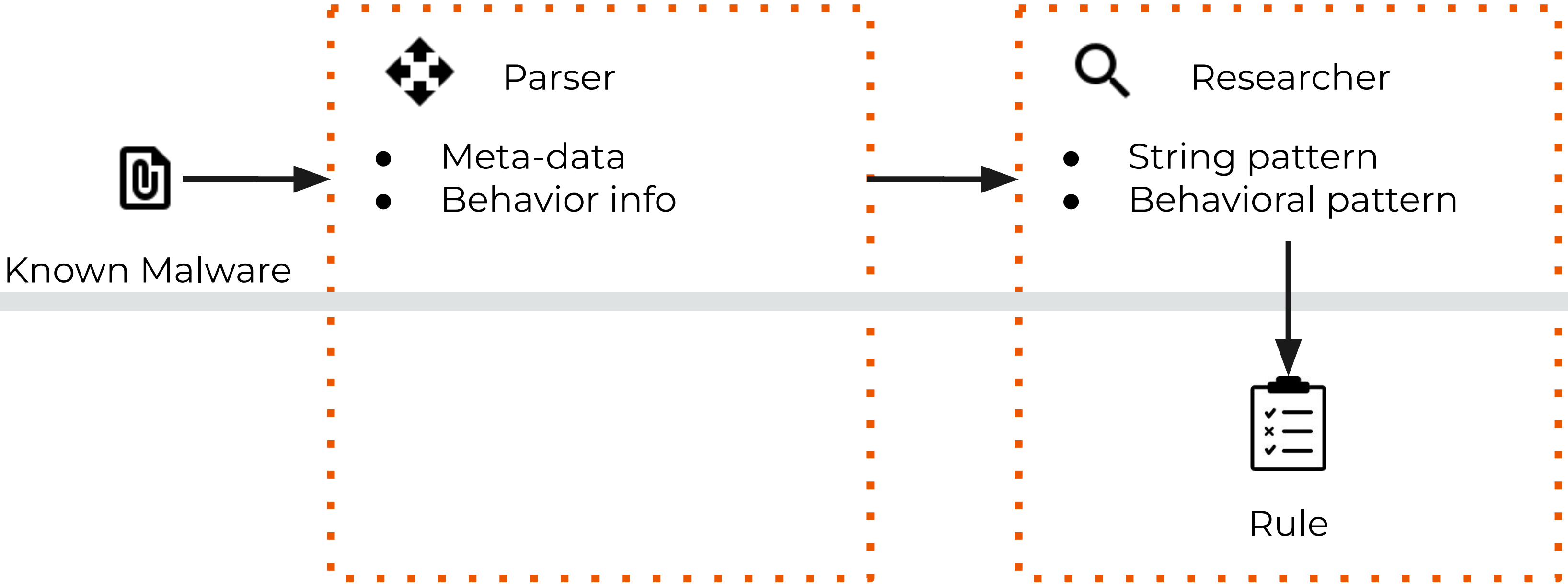
Pattern-based Detection

- Parse sample to get as much as possible information
  - Static analysis
  - Dynamic analysis
- Scanner
  - Label sample with rules (known patterns)
  - Final verdict is based on the rules triggered.

How does a Typical Detection Module Work



Rule Generation



Limitation

- Manual rule generation cannot scale
    - In order to processing at large scale, automation is a must.
- Hard to verify rule quality with traditional approaches
    - Usually need a big set of benign files to run against.
    - Especially for rule with text pattern (e.g. script)

Previous work

- Static clustering approaches
  - Manual static features, N-gram based, Fuzzing hash based ...etc
  - Some approaches researchers need to came out reliable static features manually for clustering.
  - Packer challenge
- Dynamic clustering approaches
  - API sequence (DNA?), Observed Behaviors
  - Resource challenge (time for VM, hardware ...etc)
  - Challenge to clustering based on the log. (Tokenize, peta data processing ...etc)

DiDe

Purpose

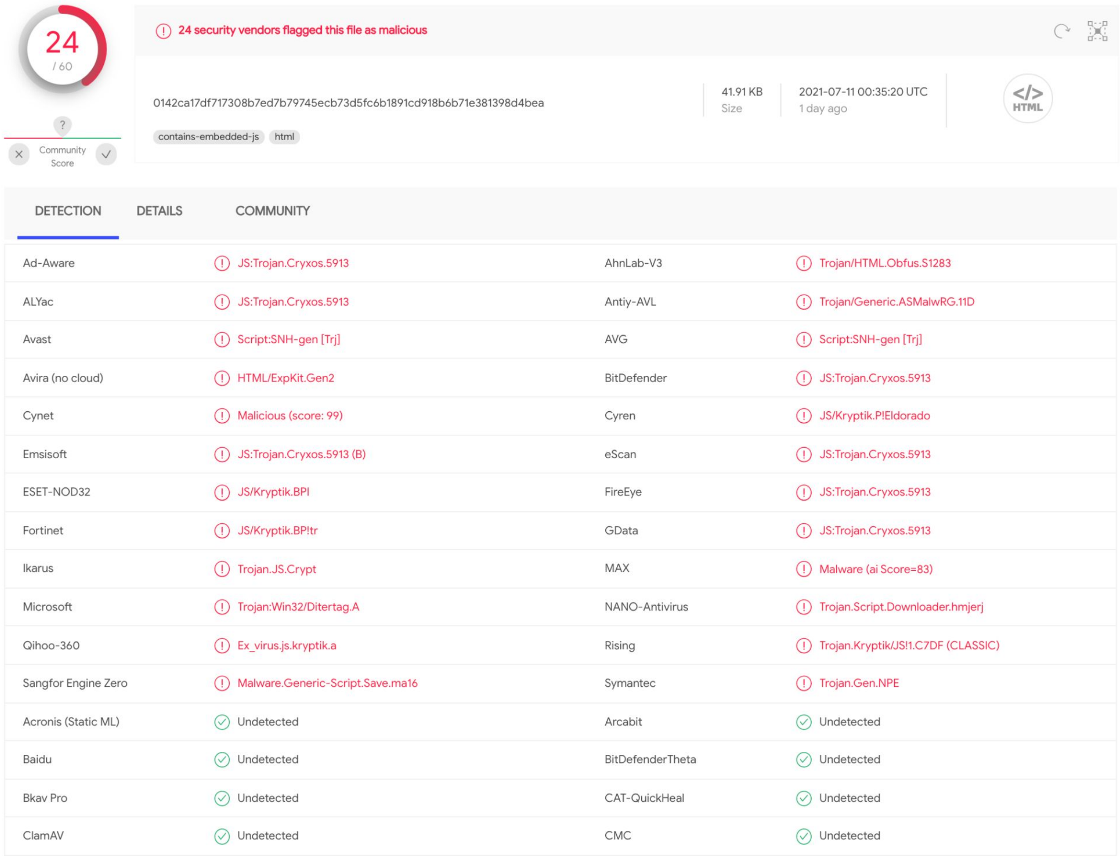
- Malware is evolving everyday, we want to catch up
- Millions of new samples everyday, we want to verify each detection
- Resource is limited, we want to make it automatic
  - Automatic rule generation
  - Automatic rule evaluation

How to automatically generate a rule

- Usually, security researcher will pick some unique text or behavior patterns for one malware sample and use them as a rule.
  - Automatic find unique patterns
  - Automatic identify malicious patterns
- If we can find a group of samples from the same malware family or somehow similar to each other
  - Automatic find common patterns
  - Automatic remove if not efficacy

VirusTotal Report

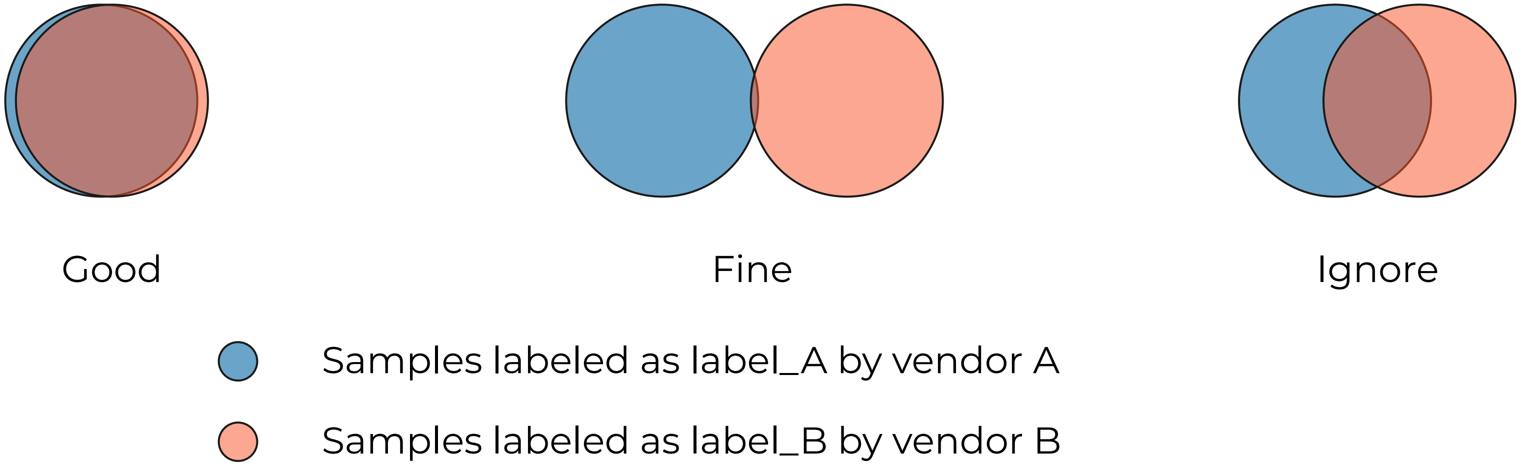
- VirusTotal is a good source
  - Variety on source samples
  - Offers results from many popular vendors
  - Offers labels
- Detection module may not be the latest from each vendor
- Detection module may produce FP/FN



Ad-Aware	JS:Trojan.Cryxos.5913	AhnLab-V3	Trojan/HTML.Obfus.S1283
ALYac	JS:Trojan.Cryxos.5913	Antiy-AVL	Trojan/Generic.ASMalwRG.11D
Avast	Script:SNH-gen [Trj]	AVG	Script:SNH-gen [Trj]
Avira (no cloud)	HTML/ExpKit.Gen2	BitDefender	JS:Trojan.Cryxos.5913
Cynet	Malicious (score: 99)	Cyren	JS/Kryptik.P!Eldorado
Emsisoft	JS:Trojan.Cryxos.5913 (B)	eScan	JS:Trojan.Cryxos.5913
ESET-NOD32	JS/Kryptik.BPI	FireEye	JS:Trojan.Cryxos.5913
Fortinet	JS/Kryptik.BP!tr	GData	JS:Trojan.Cryxos.5913
Ikarus	Trojan.JS.Crypt	MAX	Malware (ai Score=83)
Microsoft	Trojan:Win32/Ditertag.A	NANO-Antivirus	Trojan.Script.Downloader.hmjerj
Qihoo-360	Ex_virus.js.kryptik.a	Rising	Trojan.Kryptik/JS!1.C7DF (CLASSIC)
Sangfor Engine Zero	Malware.Generic-Script.Save.ma16	Symantec	Trojan.Gen.NPE
Acronis (Static ML)	Undetected	Arcabit	Undetected
Baidu	Undetected	BitDefenderTheta	Undetected
Bkav Pro	Undetected	CAT-QuickHeal	Undetected
ClamAV	Undetected	CMC	Undetected

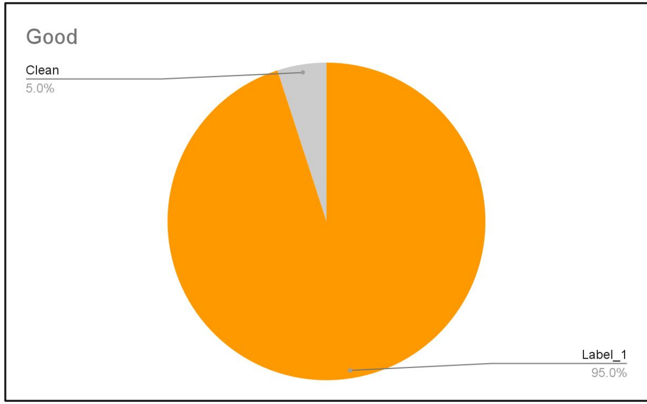
Insights

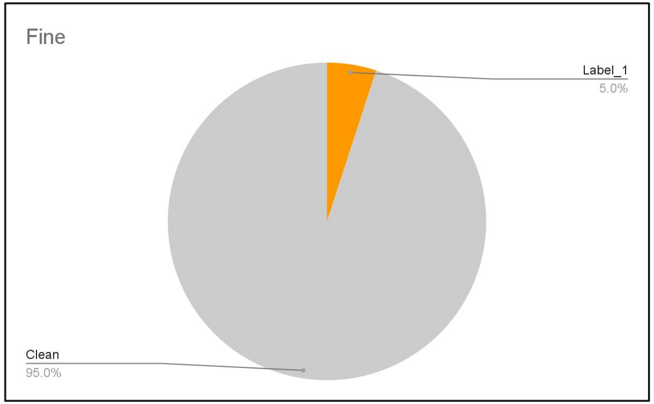
- A labeled sample means it contain a pattern from known malware
- A group of samples with same label contains the same pattern
- Majorly overlapped groups prones to be the same malware

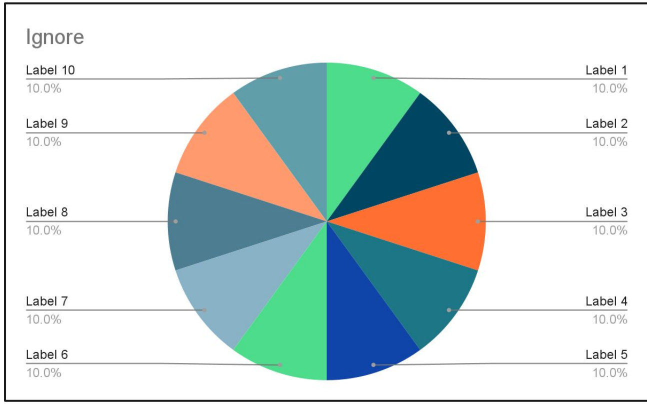


Insights cont.

- Labels could be cross-validated by labels (from different vendors)
- A group with a combination of labels (from different vendors) are prone to a malware family







◯ Samples labeled as label_A by vendor A

How to find a good malware group

Looking for a sample group which is good for automatic rule generation, a group with a **reasonable amount** of **true malwares** sharing some special **patterns**.

- How many labels in the combination
- How many samples in the group
- Total number of labels in the group
- Average number of labels
- Variance of label count
- *Strict sample percentage

**Strict sample, sample has and only has labels the group has.*

Example Group

Sample group 300dc896c810f1ce25834475028d76adcadb2dfeb6a9484f6a89b0636874b6a7

<AVG>:<JS:Facelike-B [PUP]>

<Avast>:<JS:Facelike-B [PUP]>

<CAT_QuickHeal>:<JS.Trojan.Agent.42292>

<Ikarus>:<Trojan.Script>

<Microsoft>:<Trojan:Script/Sabsik.FL.B!ml>

<Rising>:<Trojan.FaceLiker/JS!1.BAA9 (CLASSIC)>

<Zillya>:<Trojan.Iframe.JS.3>

- Based on all VT reports (HTML and JavaScript) from 20220101
- There are 14,219 unique samples
- There are 54 detection modules (vendors)
- There are 16,430 different labels
- There are 9525 groups based on all combinations of labels
- Sample group with a combination of 7 labels
  - There are 80 samples falls into this group
  - There are 73 strict samples among them
  - The average label count for this group 7.19
  - The variance of label count for all samples in this group is 1.05

Build your sample cluster

- Parse report and get labels align with each vendor
- Construct a sample group with any combination of labels
- Recommendation:
  - Start with group with combination of more labels
  - Start with group with low variance
  - Start with group with most samples are strict sample

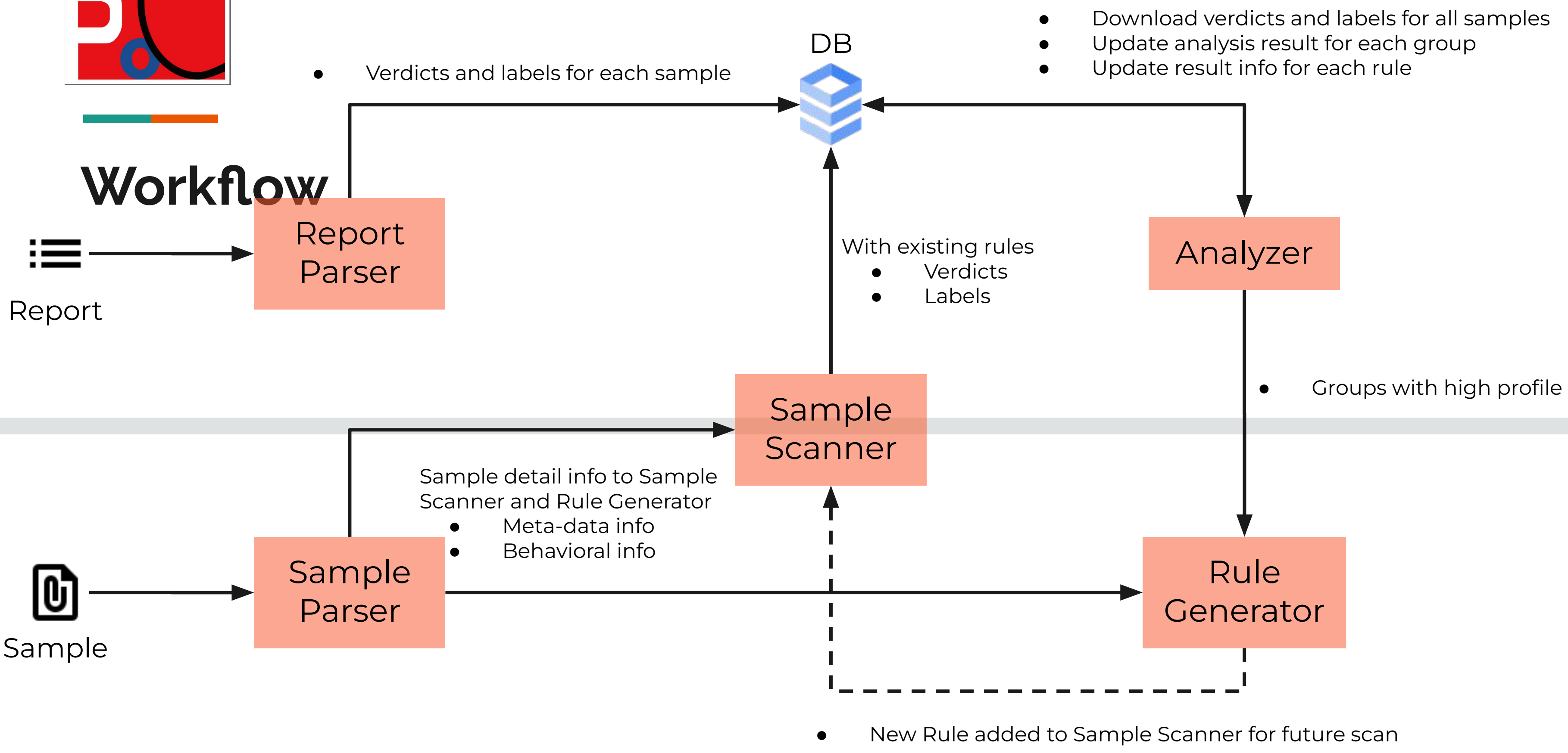
Build your rule generator

- Start with the most simple solution, such as common string
- Improve it based on your observations
  - Weight different pattern
  - Add relationship between different patterns
- Recorded information about the original group

Build your rule verifier

- The group which used to generate a rule is also the target group of the rule
- Evaluate triggered sample
  - Triggered label count VS expected label count
  - Triggered labels VS expected labels
  - Strict match sample
  - Loose match sample
  - Unmatched sample
- Evaluate triggered sample group
  - Triggered label count VS expected label count
  - Triggered label count variance VS expected label variance
  - Triggered labels vs expected labels
  - Strict sample rate
  - Loose sample rate
  - Unmatched sample rate

Workflow



Verdicts and labels for each sample
DB
Download verdicts and labels for all samples
Update analysis result for each group
Update result info for each rule
Report
Report Parser
With existing rules
Verdicts
Labels
Analyzer
Groups with high profile
Sample Scanner
Sample detail info to Sample Scanner and Rule Generator
Meta-data info
Behavioral info
Sample
Sample Parser
Rule Generator
New Rule added to Sample Scanner for future scan

Build your sample parser

- Get as much as possible information from one sample
- Choose a target file type. For example: HTML
- Statical analysis
  - Normalize
  - Deobfuscate
- Dynamical analysis
  - Resource downloaded during rendering the file: request/response
  - Script behavior: Information collection, exploit
  - Browser behavior: Subprocess
  - System behavior: registry change, file change

Build your sample scanner

- Balance the feature and performance
- Regex: simple and useful
- Yara: more powerful and meet most situation
- Customize: could be much more powerful when you design a scanner match the data your get from parser

Issues and workaround

How to handle label generated by ML?

- You may get some labels like: Malicious (score: 92)
- Given the machine learning algorithm is unknown to us, it may impact our system.
- Solution 1:
    - Ignore any label directly from ML module
- Solution 2:
    - Ignore a group if most labels are from ML modules

How to handle label rotation?

- Some vendors are keep updating the label name for the same malware family
- You may get some labels like: JS.Trojan.202208ABC and JS.Trojan.202209ABC
- Solution 1
  - Understand the format of label name so that you can recognize them automatically
- Solution 2
  - Treat them as independent labels
  - Reconsider about mismatched labels

How to handle a group with “same” samples?

- If samples in one group are almost the same, any pattern could be a common pattern.
- Solution 1:
  - Build pattern database, so that we check the popularity of one pattern among other groups.
- Solution 2:
  - Start with some random common patterns.
  - If the rule triggered one FP, them remove these common patterns and randomly pick new ones.

Thank you