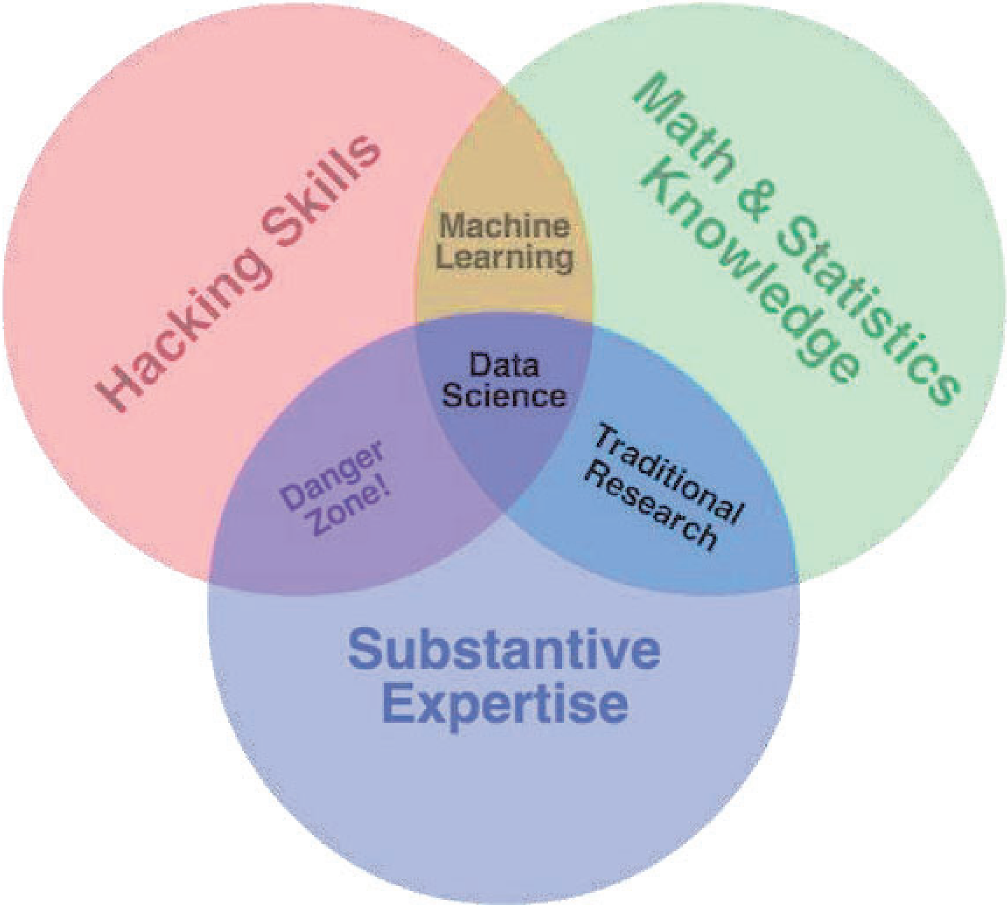
@PacSec 2013

Fighting advanced malware using machine learning

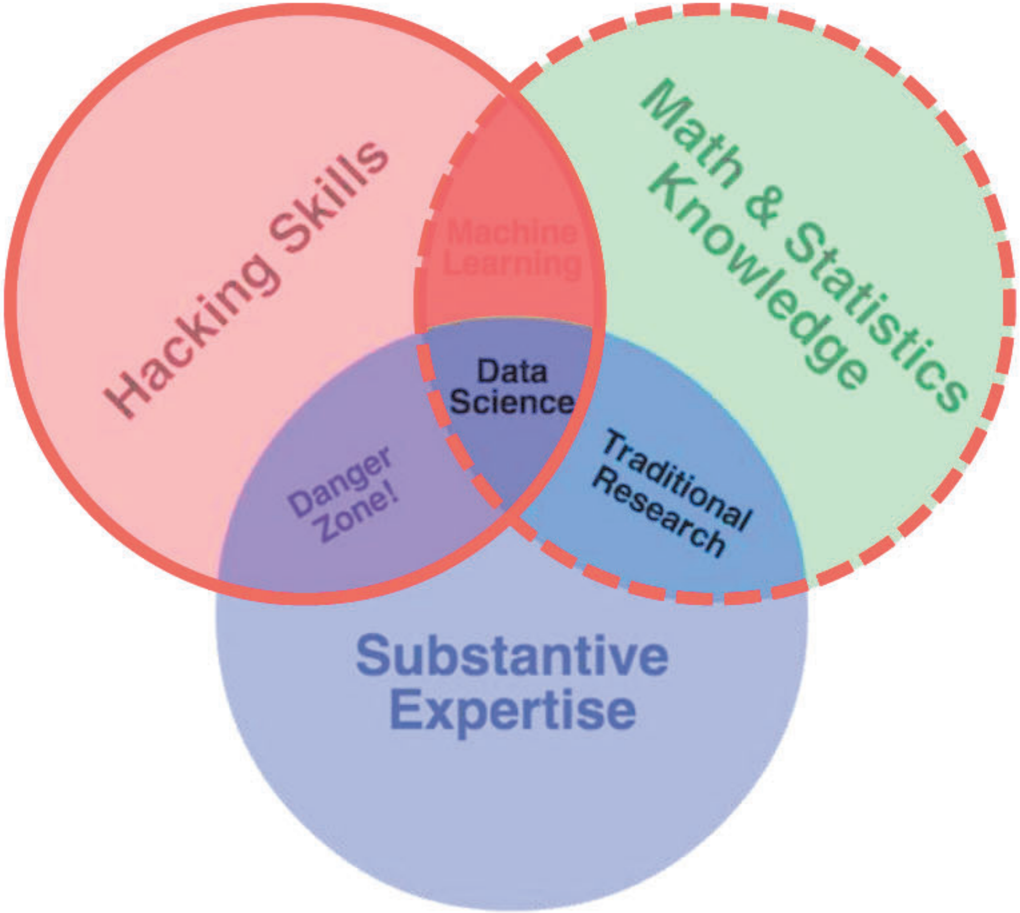
FFRI, Inc.
http://www.ffri.jp

The Data Science Venn Diagram (in security)



http://www.niemanlab.org/images/drew-conway-data-science-venn-diagram.jpg

I am, and this talk is

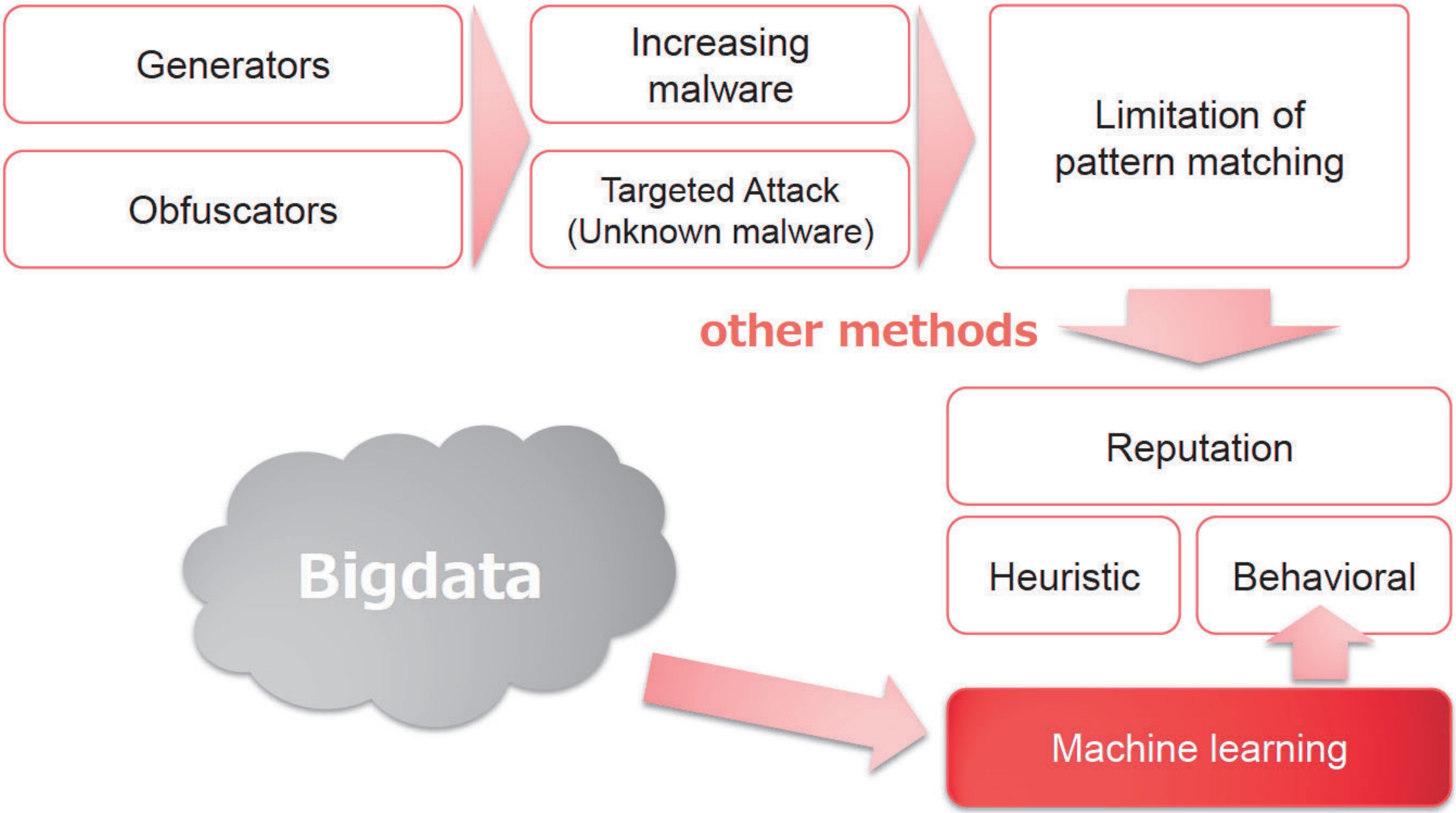


http://www.niemanlab.org/images/drew-conway-data-science-venn-diagram.jpg

Agenda

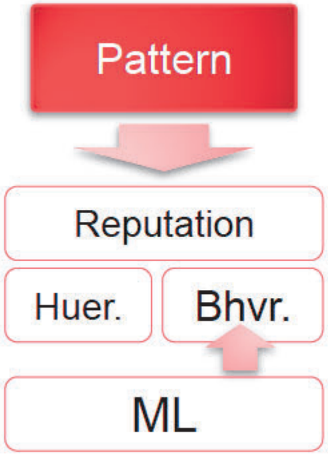
- Background
- Our approach
- Future works
  - Computers vs. Man
  - Applying to real time protection
- Conclusion

Background – malware and its detection

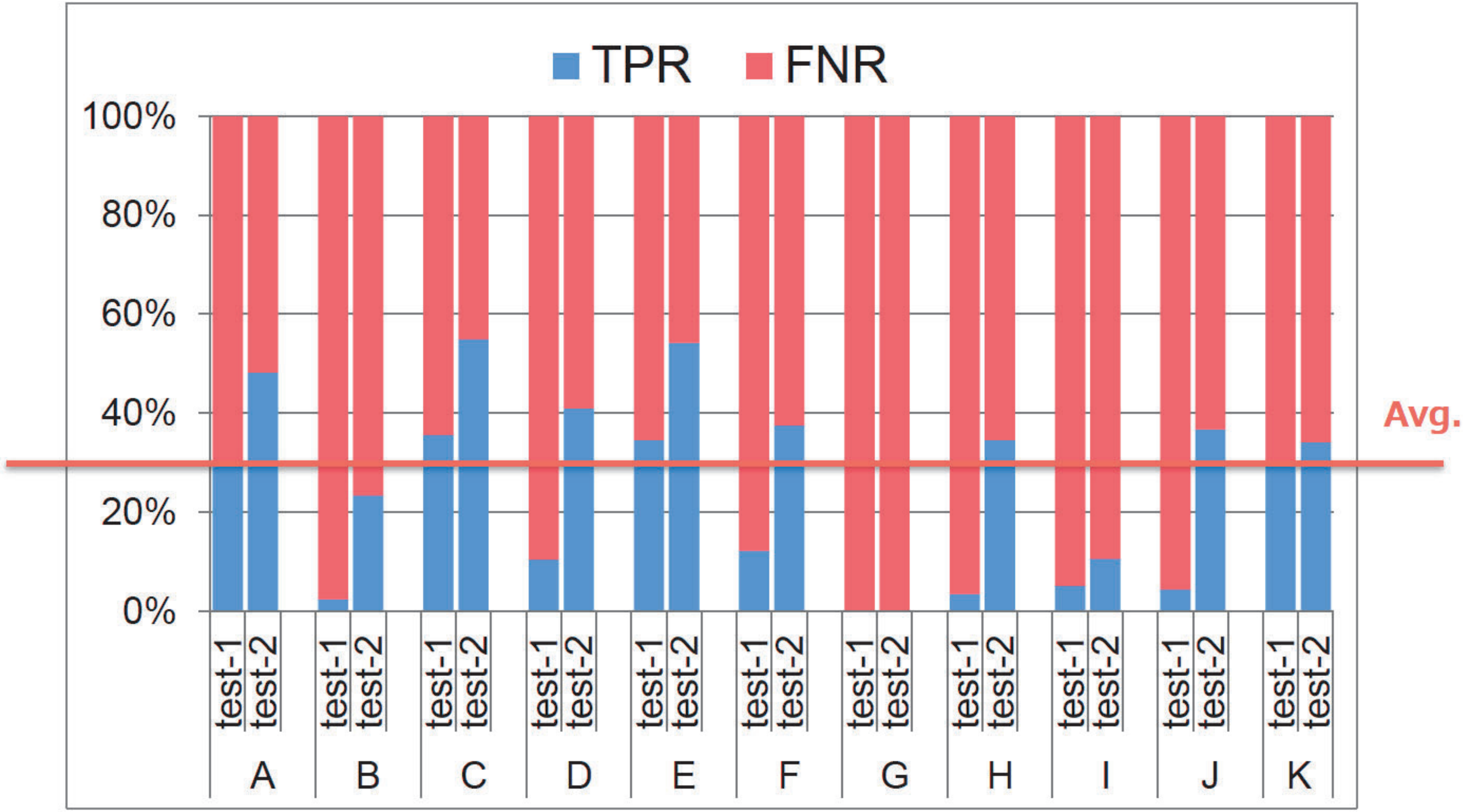


Limitation of signature matching

- Evaluated 11 AV-product's TRP using Metascan
- Used fresh malware (not wildlist malware)
- Prepared 2 test sets from different sources and period
  - test-1: 1,000 samples
  - test-2: 900 samples

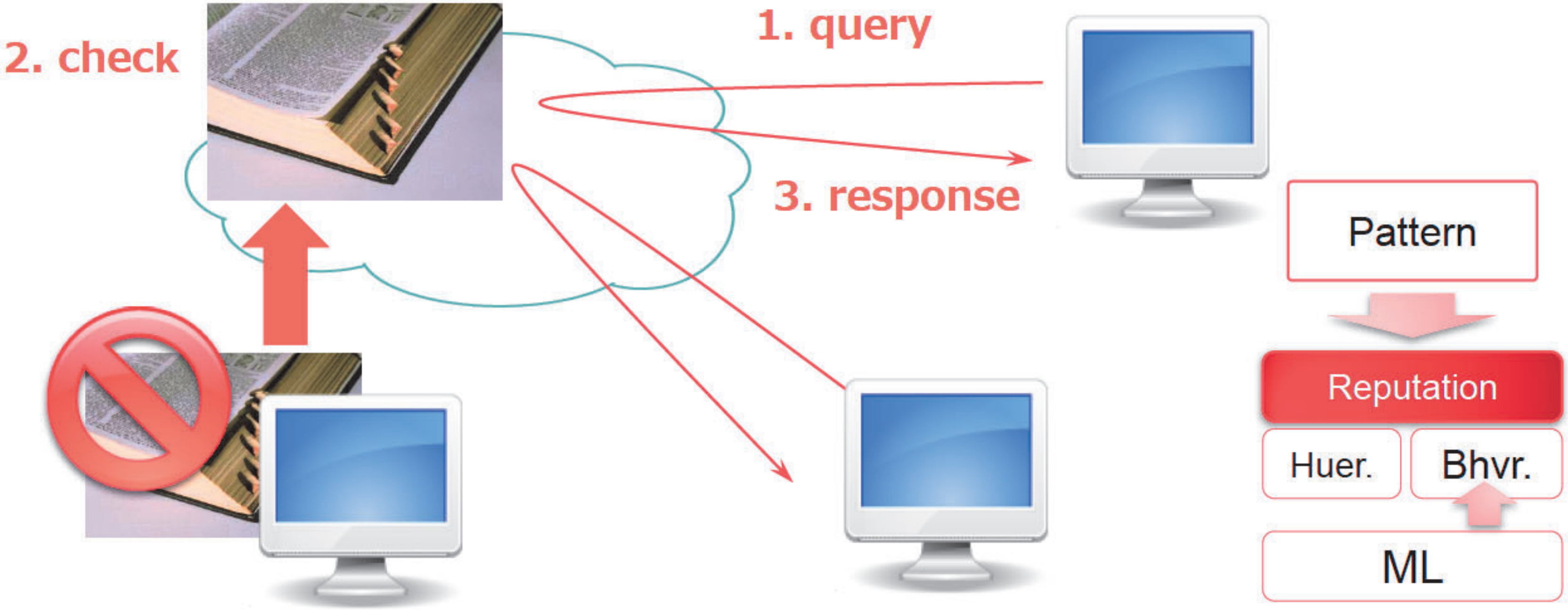


Limitation of signature matching



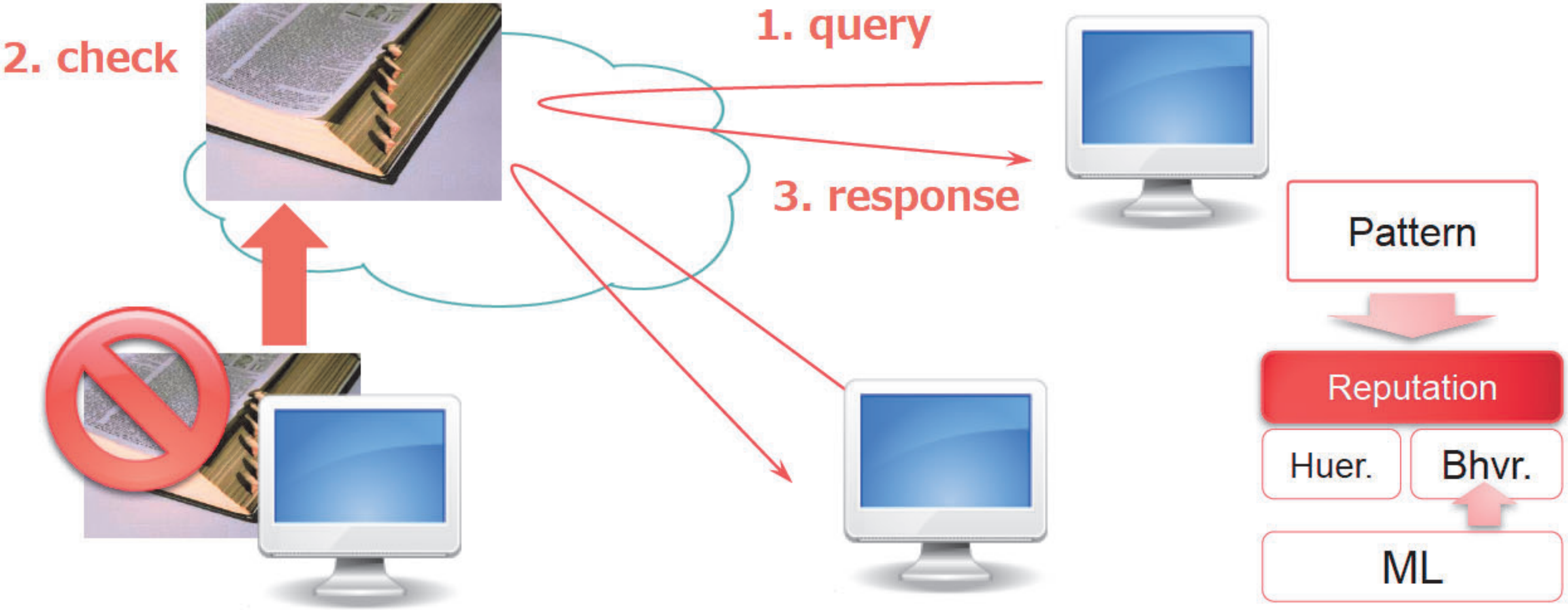
Advantage of (cloud) reputation

- Concept is the same with signature-matching(Blacklisting)
- Endpoints don’t have to keep **HEAVY** patterns anymore
- Easy to reflect a new pattern to the others



Disdvantage of (cloud) reputation

- “Detectable” means that someone is already attacked
- What if you are the first victim?
- How much effective against “Targeted Attack” it is?

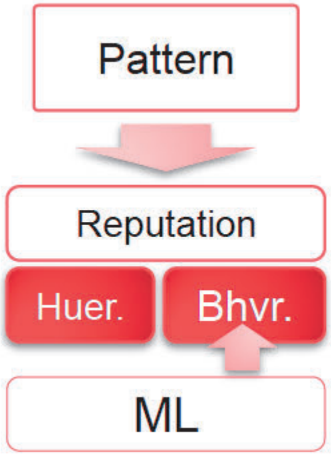


Advantage of heuristic/behavioral detection

- Based on pre-determined characteristics or behaviors
  - OpenProcess -> WriteProcessMemory -> CreateRemoteThread
  - Registering itself to auto start extensibility points
  - Disabling Windows Firewall, etc.

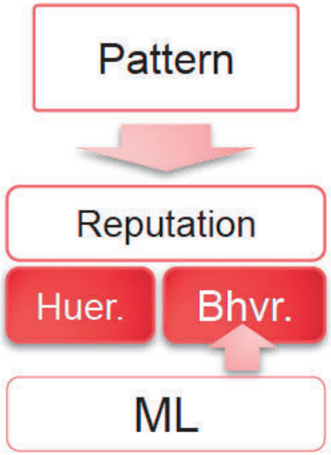
- Providing generic logics to detect malware

- Signature-free
  (eliminate regular scanning and update)



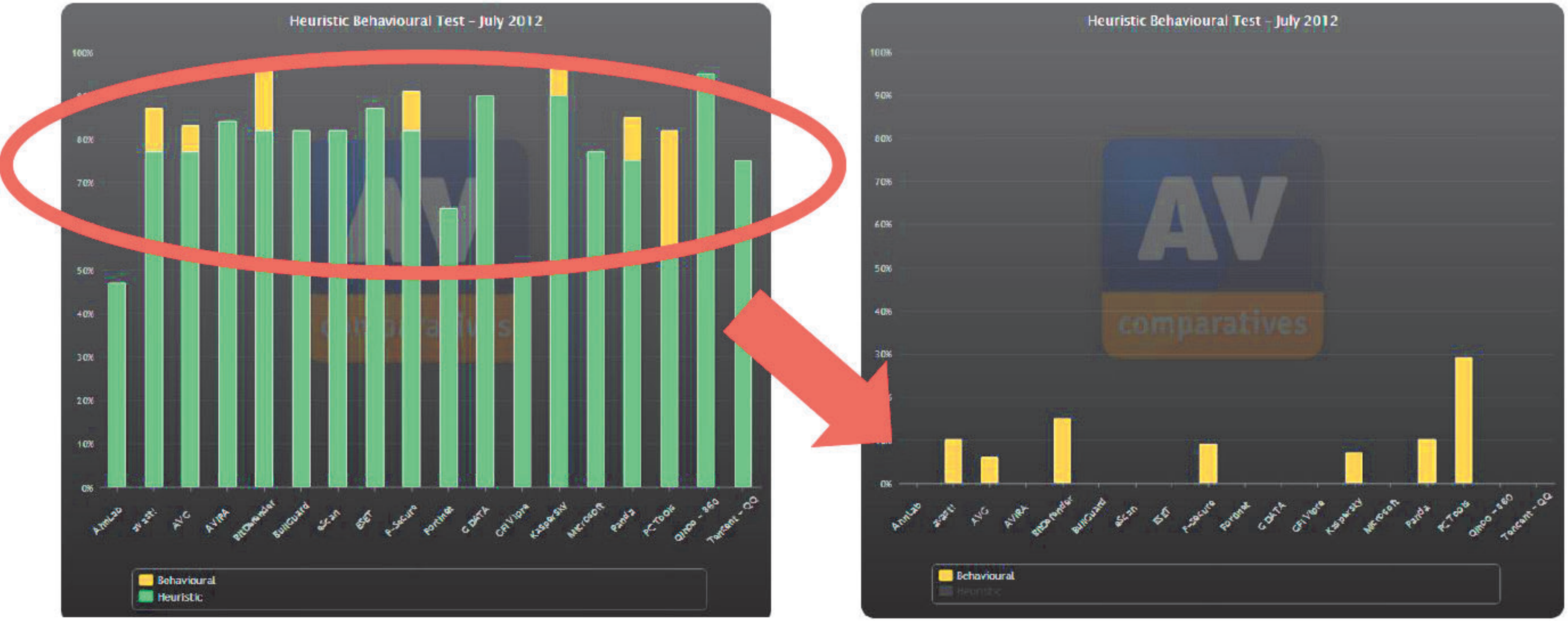
Disadvantage of heuristic/behavioral detection

- Difficult to avoid false positives completely
- Or ask user to determine if an action would be allow or not (User dependent)
- Have to analyze malware and update logics continuously (Not human task, more suitable for computers)



Heuristic Behavioral Test - July 2012

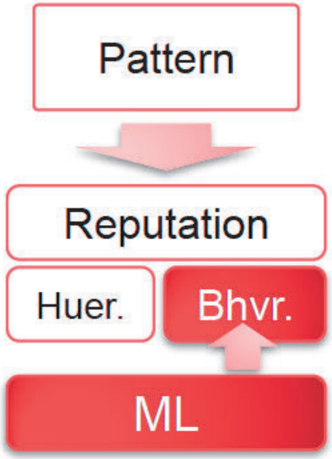
- AV-comparatives publishes H/B Test results since 2012
- The behavioral hardly contributes to detect (avg: 4.8%)



http://chart.av-comparatives.org/chart1.php

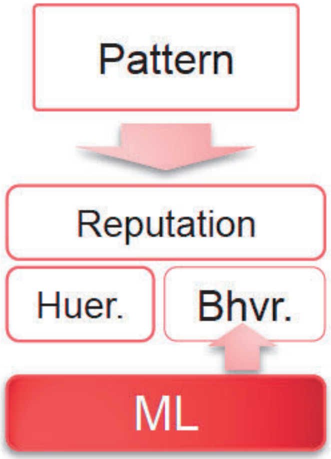
Our approach

- Behavioral-based detection powered by machine learning
- Not new but more practical for the industry
- Easy to try and automate using open source below
  - Cuckoo Sandbox
  - Jubatus



Machine learning-based detection

- Most research is doing in academic
- Basically, it is a classification problem (task)
- Mainly focusing on a combination of the factors below
- Some good results are reported



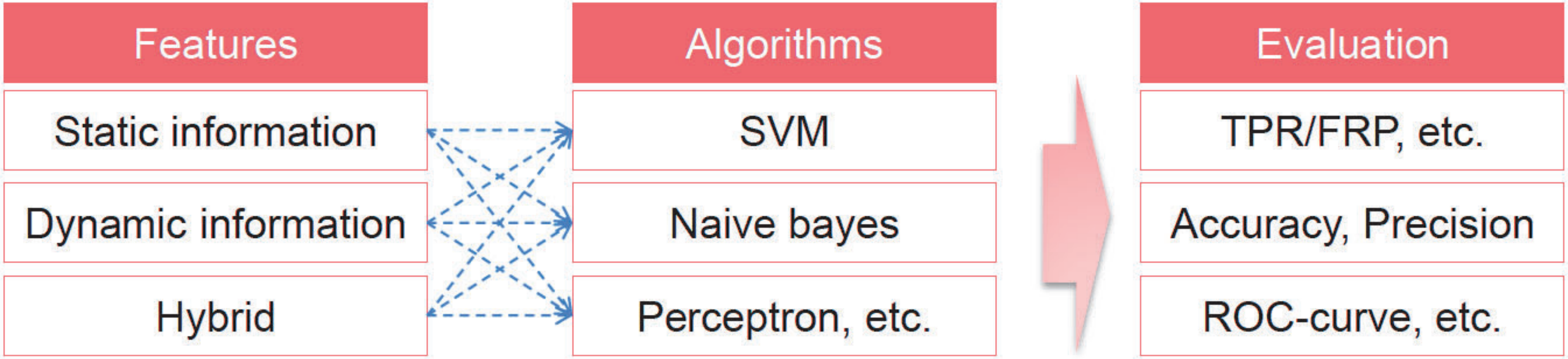
Pattern

Reputation

Huer.

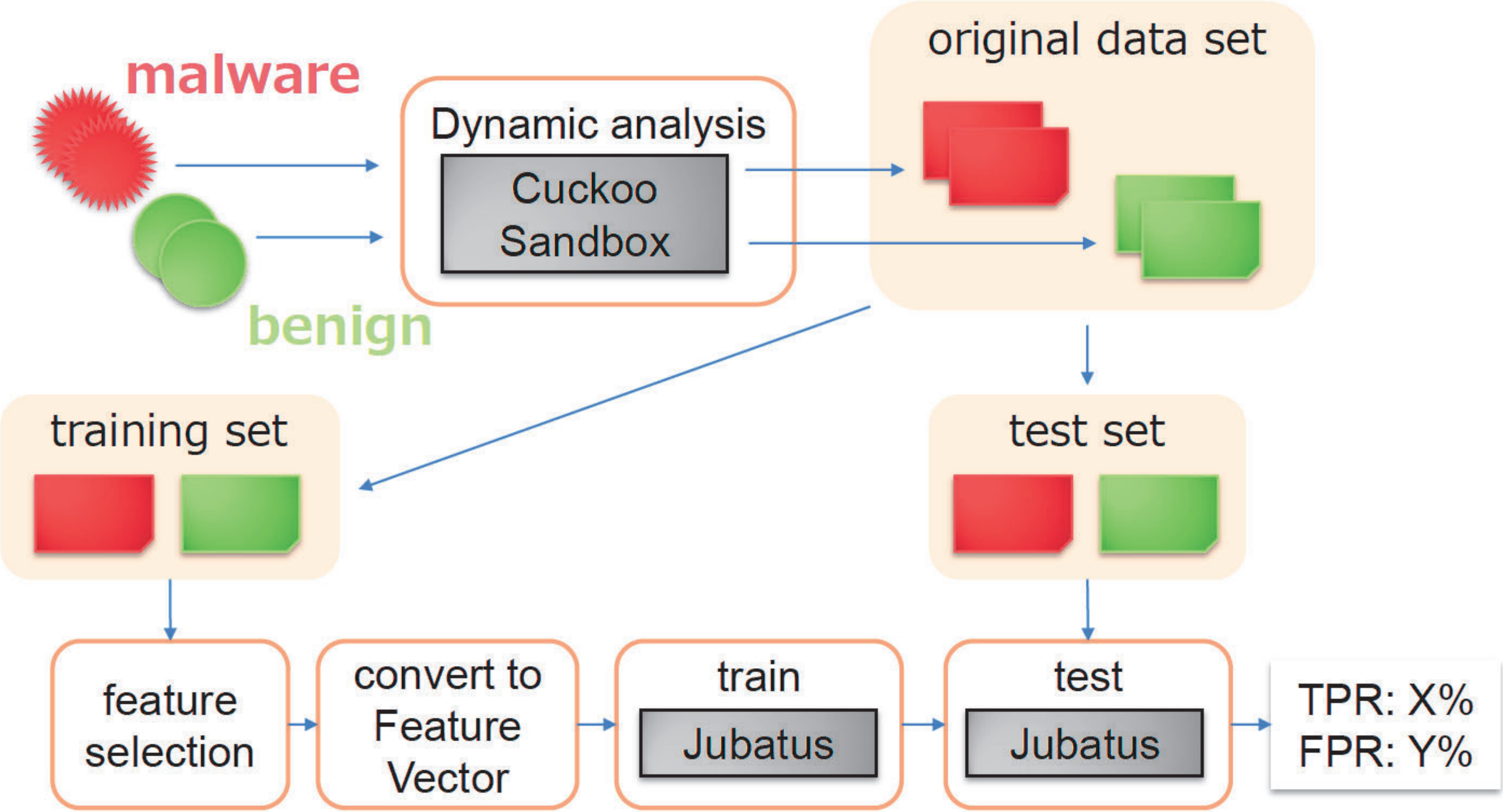
Bhvr.

ML



Features	Algorithms	Evaluation
Static information	SVM	TPR/FRP, etc.
Dynamic information	Naive bayes	Accuracy, Precision
Hybrid	Perceptron, etc.	ROC-curve, etc.

Overview

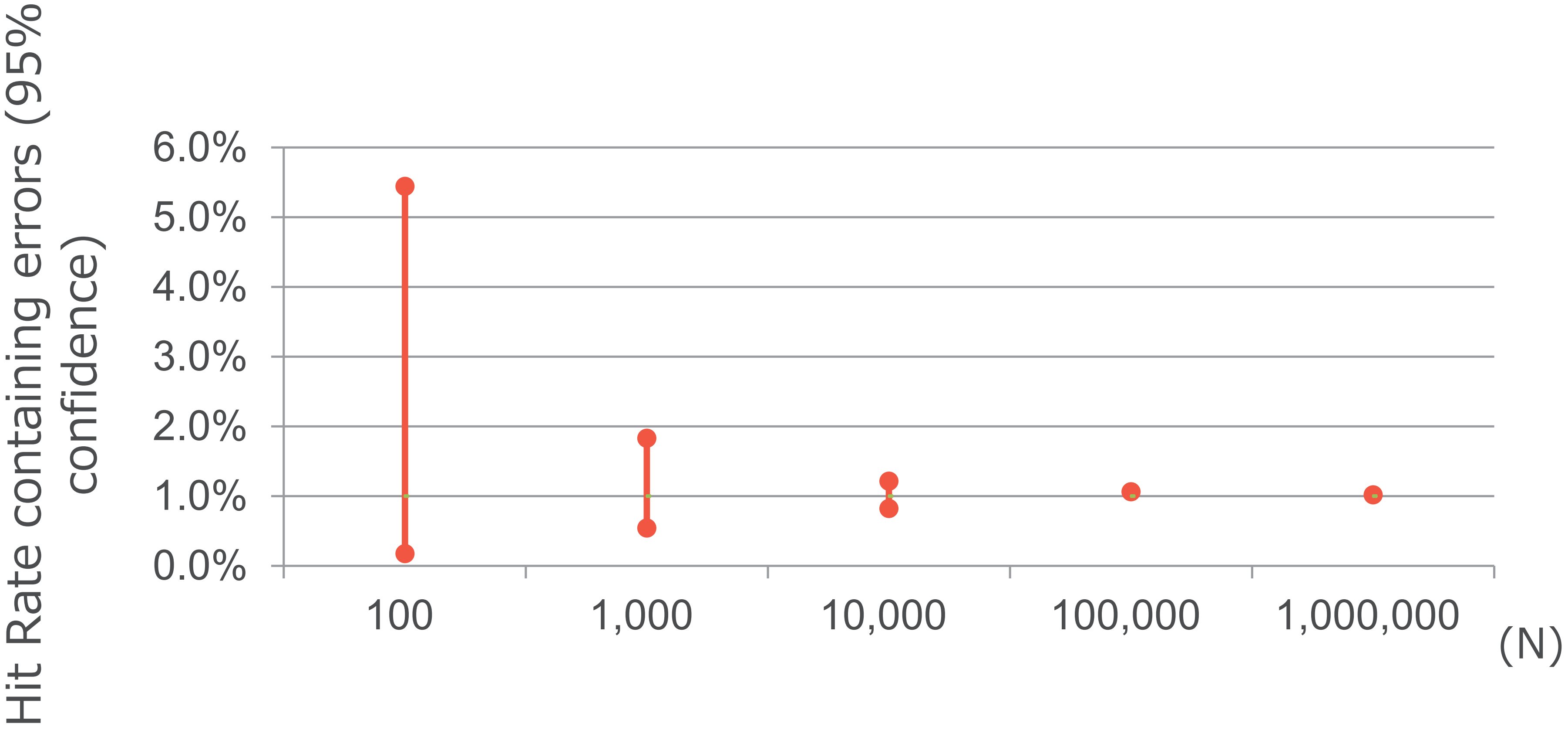

malware
benign
Dynamic analysis
Cuckoo Sandbox
original data set
training set
test set
feature selection
convert to Feature Vector
train
Jubatus
test
Jubatus
TPR: X%
FPR: Y%

How many samples should we use?

- It is a “Confidence Interval” theory
- It depends on how margin of error we accept
- All of these below hit rates are 1%
  - 1/100 (N=100)
  - 10/1,000 (N=1,000)
  - 100/10,000 (N=10,000)
- Each confidences for determining to “1%” are different
  - Each of them has different **error**

How many samples should we use?

- We can calculate estimation of margin of error based on N



Malware and benign files

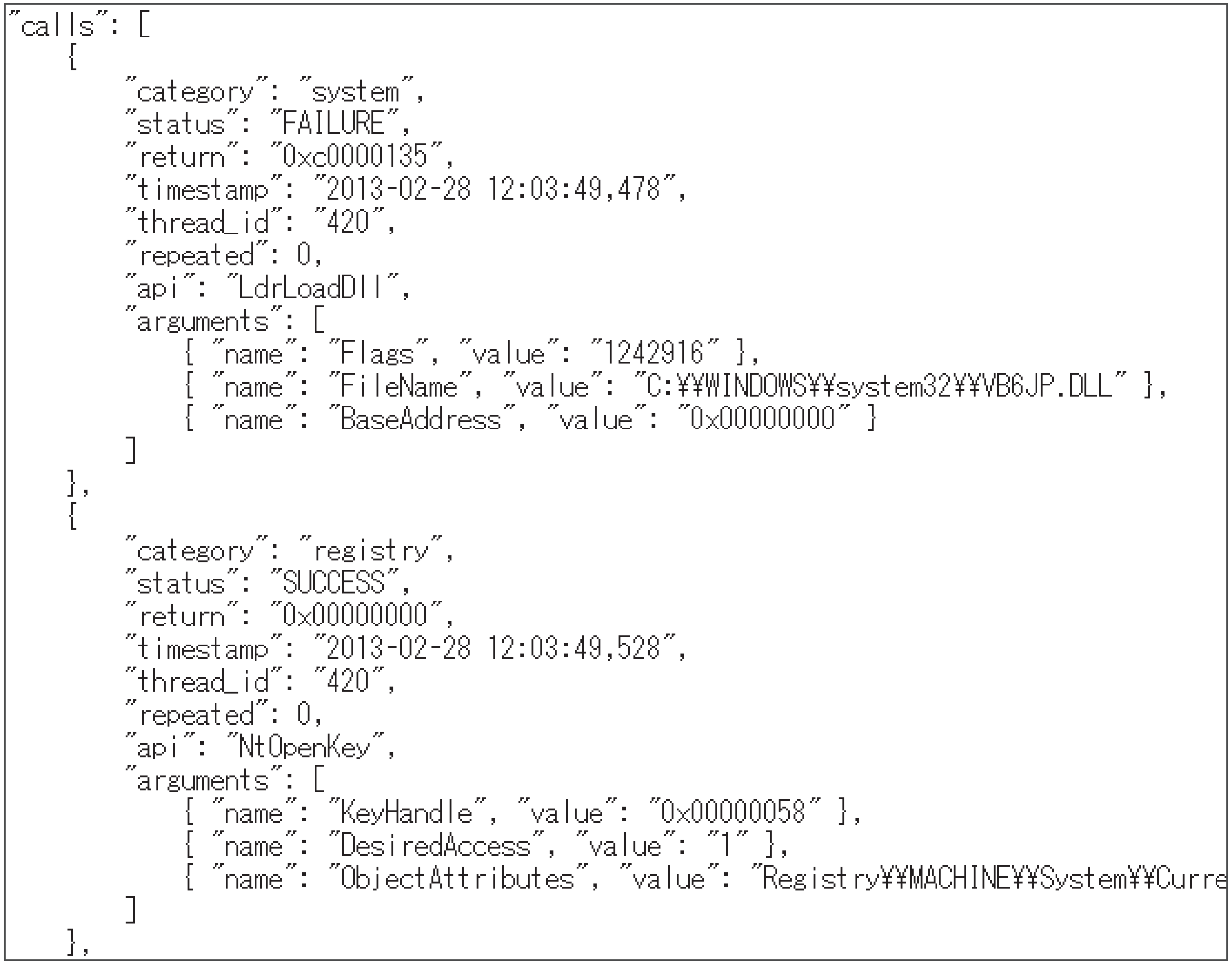
- Randomly sampling from files collected by ourselves
  - Malware: 15,000(5,000 = training, 10,000 = testing)
  - Benign: 15,000 (5,000 = training, 10,000 = testing)

- *Random* is very important
  - Different period (choose 15,000/N sample from every day)
  - Different sources
  - Never care about filetype or malware type

Cuckoo Sandbox - http://www.cuckoosandbox.org/

- Open source automated malware analysis system
  - Sending malware into a virtual machine from a host
  - Executing the malware inside the virtual machine
  - Monitoring and saving its behaviors at runtime
    - API calls, network activity, VT results, etc.

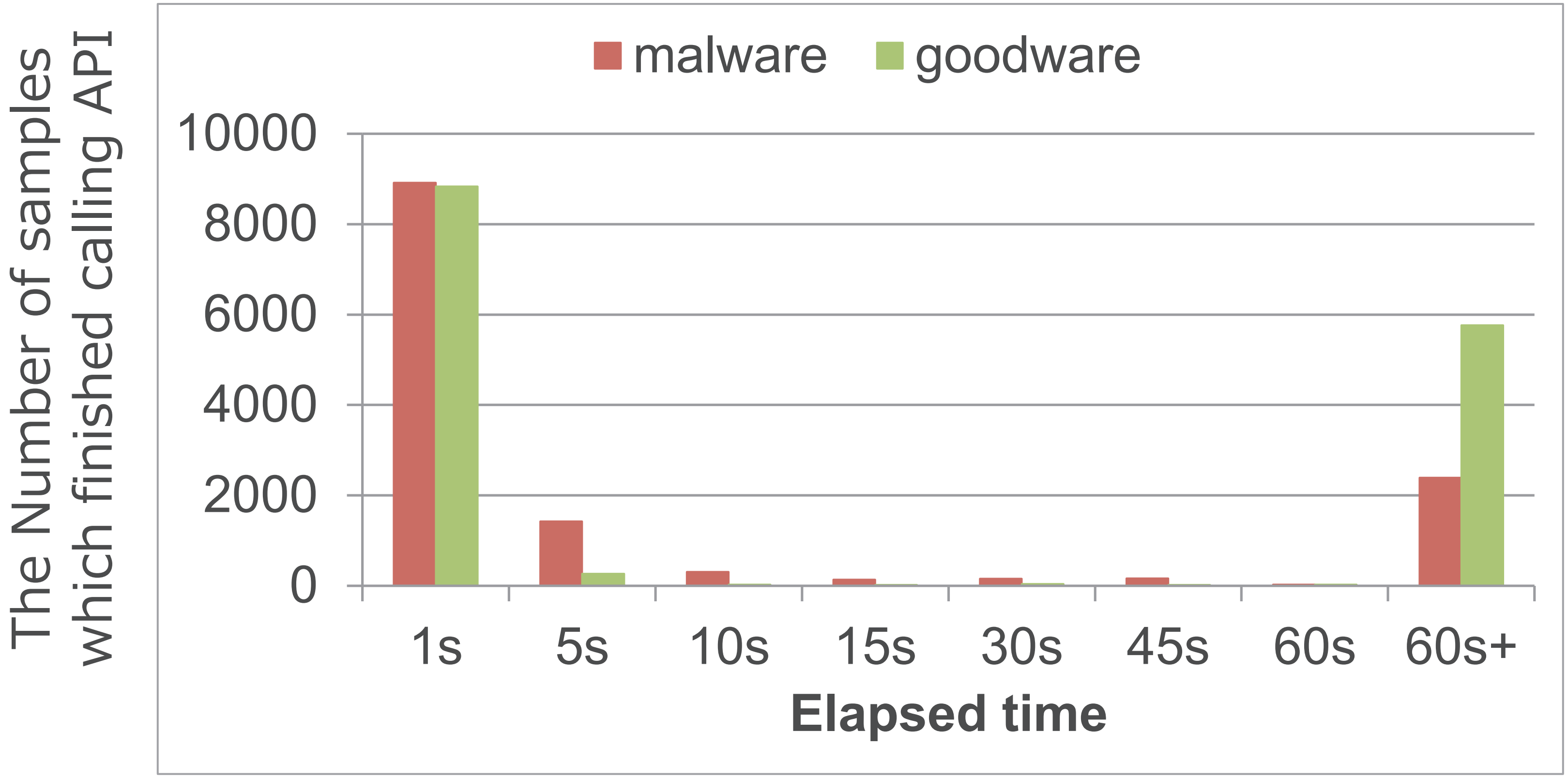
API calls



```
"calls": [
    {
        "category": "system",
        "status": "FAILURE",
        "return": "0xc0000135",
        "timestamp": "2013-02-28 12:03:49,478",
        "thread_id": "420",
        "repeated": 0,
        "api": "LdrLoadDll",
        "arguments": [
            { "name": "Flags", "value": "1242916" },
            { "name": "FileName", "value": "C:¥¥WINDOWS¥¥system32¥¥VB6JP.DLL" },
            { "name": "BaseAddress", "value": "0x00000000" }
        ]
    },
    {
        "category": "registry",
        "status": "SUCCESS",
        "return": "0x00000000",
        "timestamp": "2013-02-28 12:03:49,528",
        "thread_id": "420",
        "repeated": 0,
        "api": "NtOpenKey",
        "arguments": [
            { "name": "KeyHandle", "value": "0x00000058" },
            { "name": "DesiredAccess", "value": "1" },
            { "name": "ObjectAttributes", "value": "Registry¥¥MACHINE¥¥System¥¥Curre
        ]
    },
```

Trends of API calls

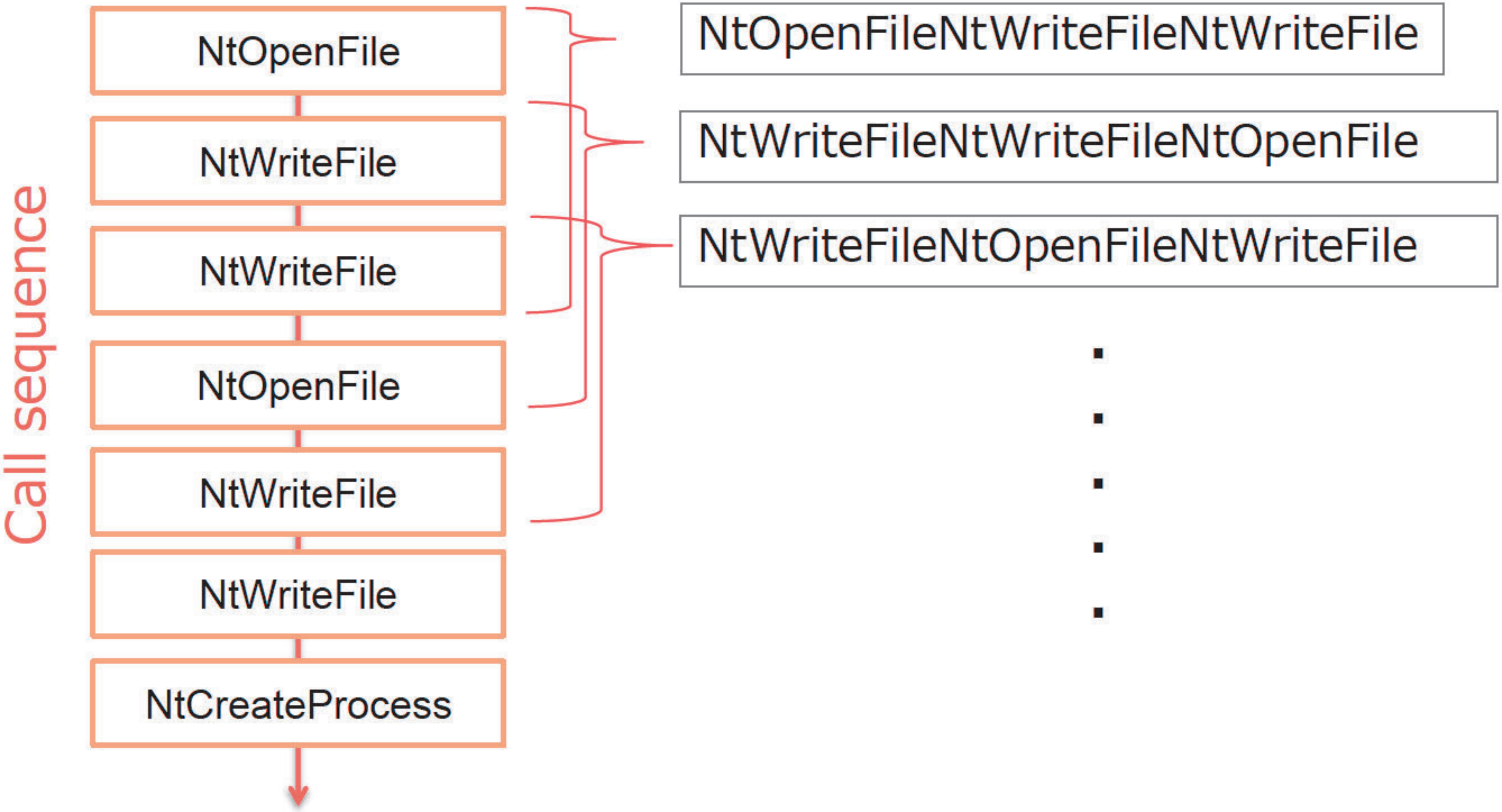
- Most of the samples finished calling API within 1s (or keep calling APIs) -> Used API only called within 5s



Jubatus - http://jubat.us/en/

- Distributed Online Machine Learning Framework
  - Distributed: Scalable
  - Online: Not batch, continuous learning
- Open source, LGPL v2.1. Latest release is 0.4.5(22/07/2013)
- Developed by Preferred Infrastructure, Inc. and NTT Software Innovation Center
- Support various machine learning
  - Classification, Regression, Recommendation, Anomaly Detection
- Easy to use (many language bindings, feature converter, etc)

Feature selection and convert to FV



Call sequence

NtOpenFile

NtWriteFile

NtWriteFile

NtOpenFile

NtWriteFile

NtWriteFile

NtCreateProcess

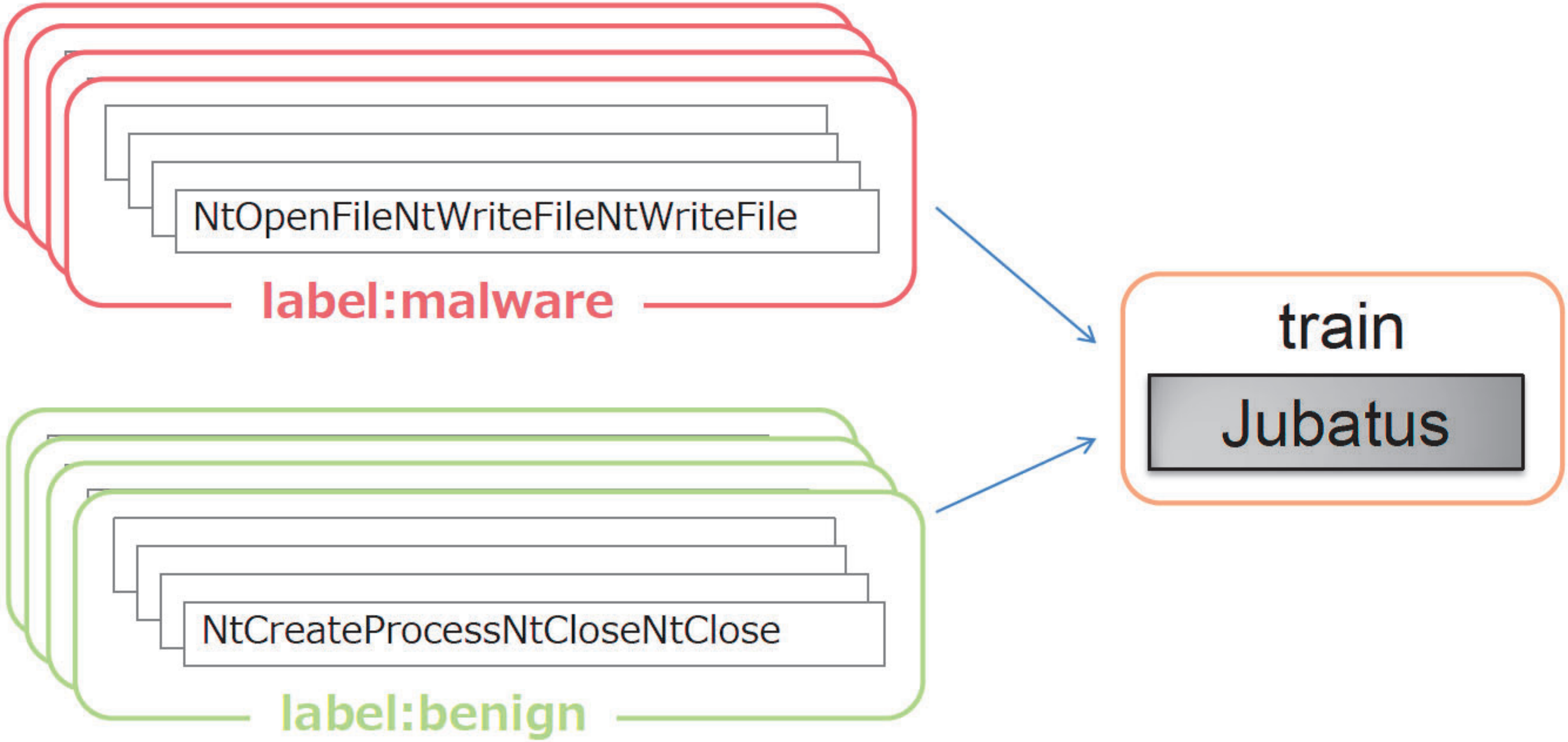
NtOpenFileNtWriteFileNtWriteFile

NtWriteFileNtWriteFileNtOpenFile

NtWriteFileNtOpenFileNtWriteFile

・
・
・
・
・

Feature selection and convert to FV



NtOpenFileNtWriteFileNtWriteFile

label:malware

NtCreateProcessNtCloseNtClose

label:benign

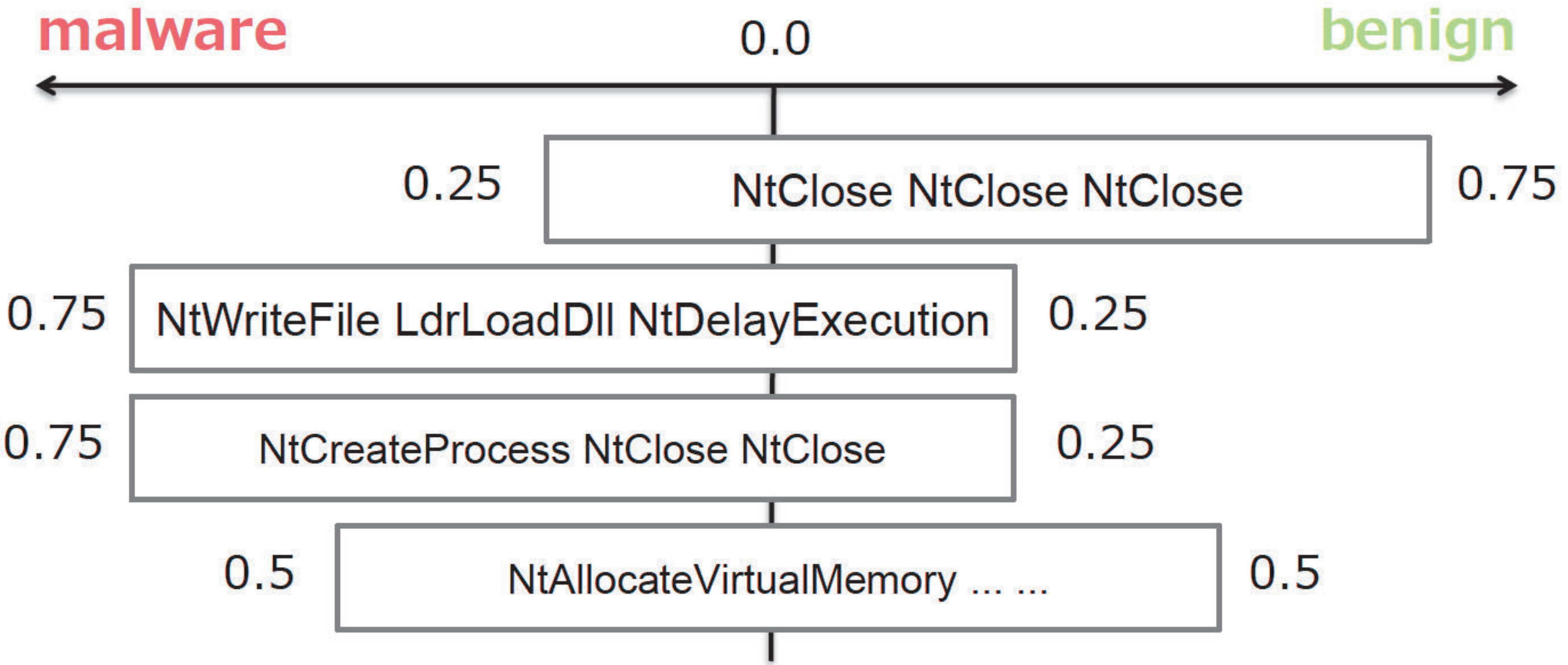
train

Jubatus

(Image) Training internal

Calc and update each FV's weight based on its freq. and label

(for the detail is dependent on the algorithm called **AROW**, don't ask me :-)


malware
0.0
benign
0.25
NtClose NtClose NtClose
0.75
0.75
NtWriteFile LdrLoadDll NtDelayExecution
0.25
0.75
NtCreateProcess NtClose NtClose
0.25
0.5
NtAllocateVirtualMemory ... ...
0.5

Testing results

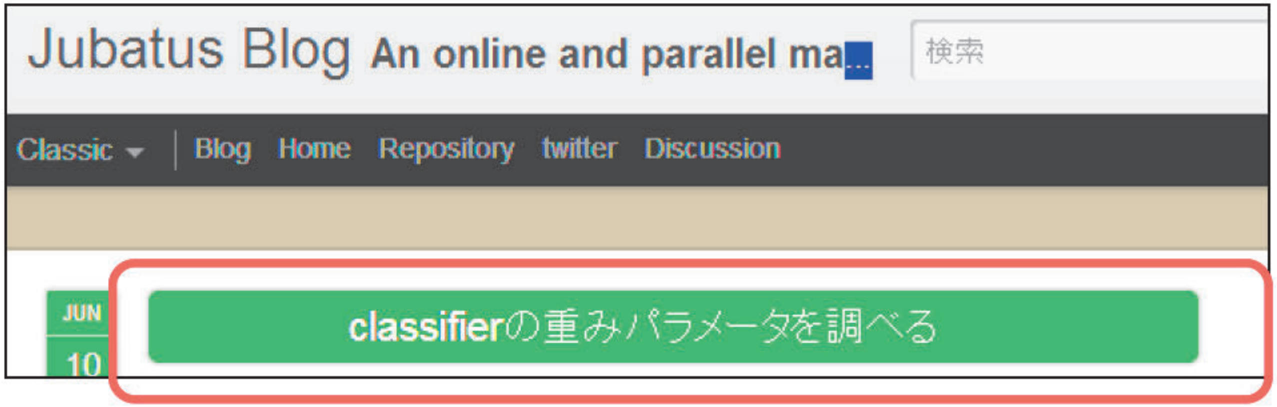
- N of N-gram
  - 3~5-gram > 2-gram = 6-gram
- Best: 3-gram
  - TRP: 72.33% [71.58 ~ 73.07 % (95% confidence)]
  - FPR: 0.77% [0.60 ~ 0.99% (95% confidence)]
- The result above is an example
  - A lot of combination of features are available (we used only “API-name” and its sequence)

Demo

Future Works

Dumping training model

- http://blog.jubat.us/2013/06/classifier.html (Japanese only)



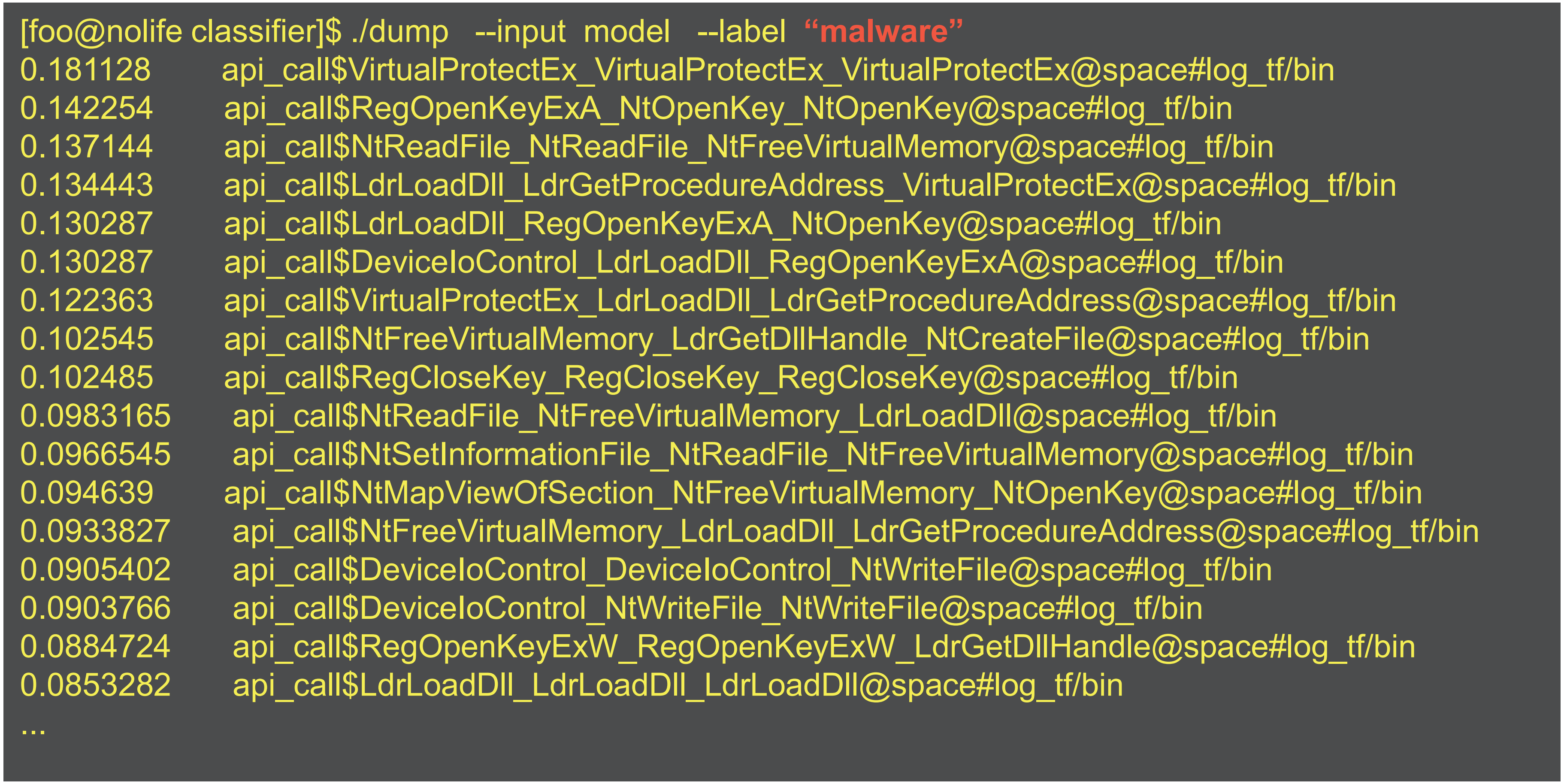
Jubatus Blog An online and parallel ma...

classifierの重みパラメータを調べる

Investigating weight parameters of classifier

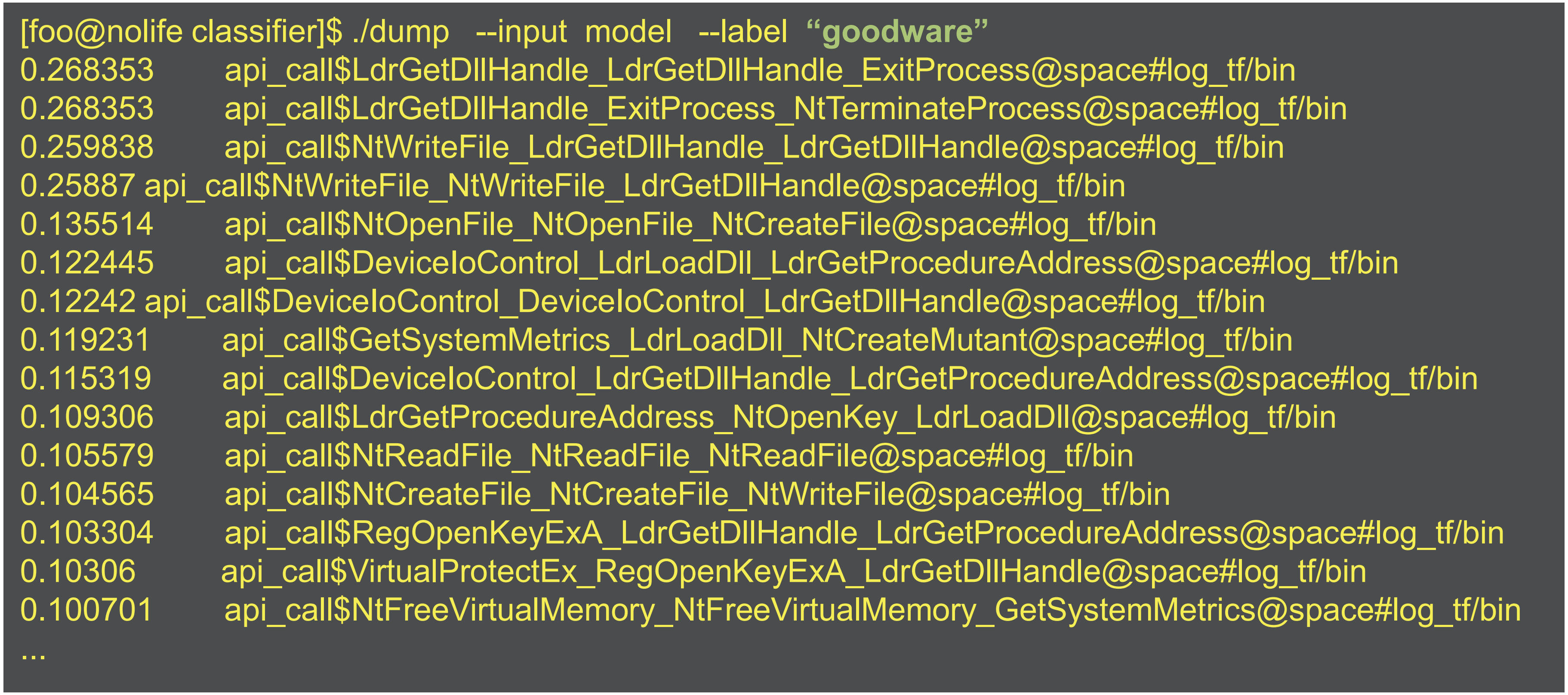
jubalocal_storage_dump.cpp
https://gist.github.com/t-abe/5746333

Indicators of malware likeness in API 3-gram



```
[foo@nolife classifier]$ ./dump   --input  model   --label  “malware”
0.181128       api_call$VirtualProtectEx_VirtualProtectEx_VirtualProtectEx@space#log_tf/bin
0.142254       api_call$RegOpenKeyExA_NtOpenKey_NtOpenKey@space#log_tf/bin
0.137144       api_call$NtReadFile_NtReadFile_NtFreeVirtualMemory@space#log_tf/bin
0.134443       api_call$LdrLoadDll_LdrGetProcedureAddress_VirtualProtectEx@space#log_tf/bin
0.130287       api_call$LdrLoadDll_RegOpenKeyExA_NtOpenKey@space#log_tf/bin
0.130287       api_call$DeviceIoControl_LdrLoadDll_RegOpenKeyExA@space#log_tf/bin
0.122363       api_call$VirtualProtectEx_LdrLoadDll_LdrGetProcedureAddress@space#log_tf/bin
0.102545       api_call$NtFreeVirtualMemory_LdrGetDllHandle_NtCreateFile@space#log_tf/bin
0.102485       api_call$RegCloseKey_RegCloseKey_RegCloseKey@space#log_tf/bin
0.0983165      api_call$NtReadFile_NtFreeVirtualMemory_LdrLoadDll@space#log_tf/bin
0.0966545      api_call$NtSetInformationFile_NtReadFile_NtFreeVirtualMemory@space#log_tf/bin
0.094639       api_call$NtMapViewOfSection_NtFreeVirtualMemory_NtOpenKey@space#log_tf/bin
0.0933827      api_call$NtFreeVirtualMemory_LdrLoadDll_LdrGetProcedureAddress@space#log_tf/bin
0.0905402      api_call$DeviceIoControl_DeviceIoControl_NtWriteFile@space#log_tf/bin
0.0903766      api_call$DeviceIoControl_NtWriteFile_NtWriteFile@space#log_tf/bin
0.0884724      api_call$RegOpenKeyExW_RegOpenKeyExW_LdrGetDllHandle@space#log_tf/bin
0.0853282      api_call$LdrLoadDll_LdrLoadDll_LdrLoadDll@space#log_tf/bin
...
```

Indicators of goodware likeness in API 3-gram



```
[foo@nolife classifier]$ ./dump   --input  model   --label  “goodware”
0.268353	api_call$LdrGetDllHandle_LdrGetDllHandle_ExitProcess@space#log_tf/bin
0.268353	api_call$LdrGetDllHandle_ExitProcess_NtTerminateProcess@space#log_tf/bin
0.259838	api_call$NtWriteFile_LdrGetDllHandle_LdrGetDllHandle@space#log_tf/bin
0.25887 api_call$NtWriteFile_NtWriteFile_LdrGetDllHandle@space#log_tf/bin
0.135514	api_call$NtOpenFile_NtOpenFile_NtCreateFile@space#log_tf/bin
0.122445	api_call$DeviceIoControl_LdrLoadDll_LdrGetProcedureAddress@space#log_tf/bin
0.12242 api_call$DeviceIoControl_DeviceIoControl_LdrGetDllHandle@space#log_tf/bin
0.119231	api_call$GetSystemMetrics_LdrLoadDll_NtCreateMutant@space#log_tf/bin
0.115319	api_call$DeviceIoControl_LdrGetDllHandle_LdrGetProcedureAddress@space#log_tf/bin
0.109306	api_call$LdrGetProcedureAddress_NtOpenKey_LdrLoadDll@space#log_tf/bin
0.105579	api_call$NtReadFile_NtReadFile_NtReadFile@space#log_tf/bin
0.104565	api_call$NtCreateFile_NtCreateFile_NtWriteFile@space#log_tf/bin
0.103304	api_call$RegOpenKeyExA_LdrGetDllHandle_LdrGetProcedureAddress@space#log_tf/bin
0.10306 	api_call$VirtualProtectEx_RegOpenKeyExA_LdrGetDllHandle@space#log_tf/bin
0.100701	api_call$NtFreeVirtualMemory_NtFreeVirtualMemory_GetSystemMetrics@space#log_tf/bin
...
```

Computer vs. Man

- “VirtualProtectEx_VirtualProtectEx_VirtualProtectEx” looks like to related to malware
- How about “RegOpenKeyExA_NtOpenKey_NtOpenKey”?
- Computers might recognize indicators which human can’t (Extremely strong left-brain player)
- Why don’t we cooperate with machine?

Using computers

- Generating models using computers
- Checking them and guessing new logics by human (Using our right-brain)
- ML-based detection is sometimes difficult to control
  - Cannot specify strict conditions to detect

    " It is detected because ML said so ! "
- Hybrid of ML-based and Logic-based would be powerful

Applying to real time protection

- Using static information as feature
  - We can check a file before its execution
  - The performance is dependent on features

- Using dynamic information as feature
  - Malware is already executed
  - Sometimes, detections would be too late
  - The hybrid detection above might be also useful in this perspective

Conclusion

- Traditional pattern-matching reaches its limit

- Current behavioral detections hardly contributes to detect

- By applying ML to behavioral detections
  - TPR is improved
  - Computers recognize new features which human can’t
  - We should make use of them

Thank you!

Contact: research-feedback@ffri.jp
Twitter: @FFRI_Research