AutoFeatures: Knowledge-Driven Automatic Feature Engineering for Detection Systems

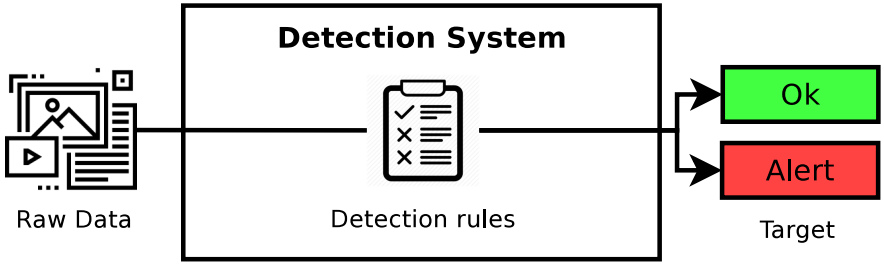
Pierre Collet✉ and Anaël Beaugnon

Agence nationale de la sécurité des systèmes d'information (ANSSI)

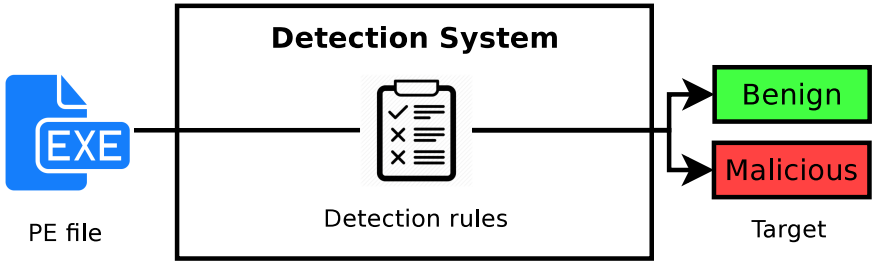
20/11/2019

Machine Learning for Detection Systems

Machine Learning for Detection Systems

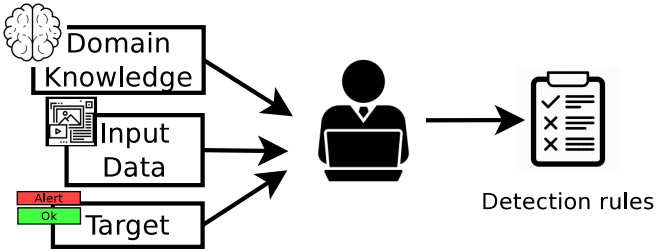


Machine Learning for Detection Systems



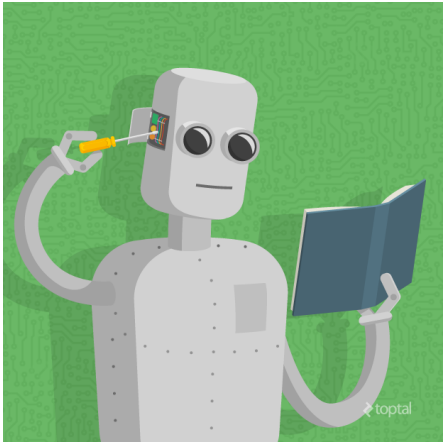
Machine Learning for Detection Systems

Creating detection rules is expensive

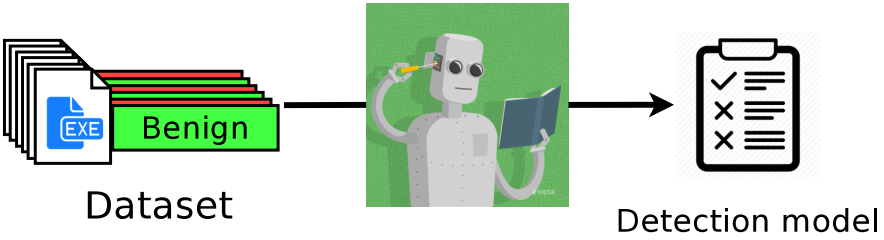


- Created & maintained by domain expert manually
- Depend on detection target
- Depend on the input data

Machine Learning for Detection Systems



Machine Learning for Detection Systems



Machine learning creates automatically detection rules from data

Machine Learning for Detection Systems

Feature Engineering

- Machine learning takes array of numerical values called features as input
- A feature is a numerical representation of a property or characteristic.
- A features must be informatives and discriminatives.

Machine Learning for Detection Systems

Feature Engineering

Machine Learning for Detection Systems

Feature Engineering

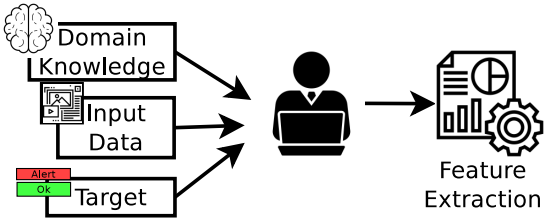
Machine Learning for Detection Systems

Feature Engineering

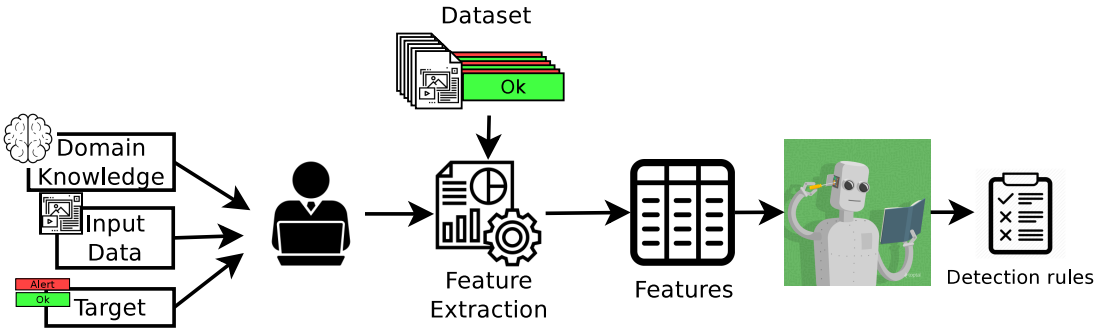
Machine Learning for Detection Systems

Feature Engineering is expensive:

- Done manually by domain expert
- Related the input data
- Related the target

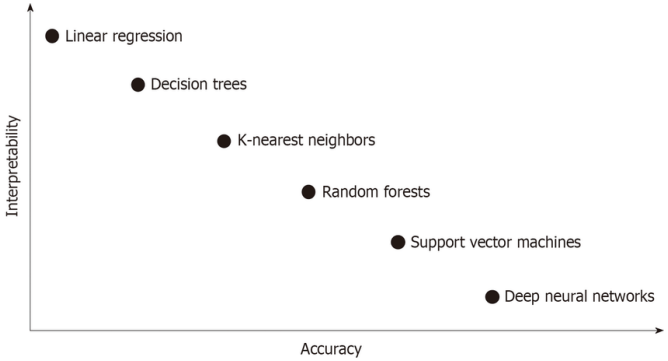


Machine Learning for Detection Systems



Machine learning creates automatically detection rules from data
still feature engineering is done manually

Machine Learning for Detection Systems



Intepretability imply feature engineering

Machine Learning for Detection Systems

Using interpretable detection models is important

- Computer Security experts require to analyze alerts

Machine Learning for Detection Systems

Using interpretable detection models is important

- Computer Security experts require to analyze alerts

Using expert knowledge is important

- Expert can help detection system with valuable information

Machine Learning for Detection Systems

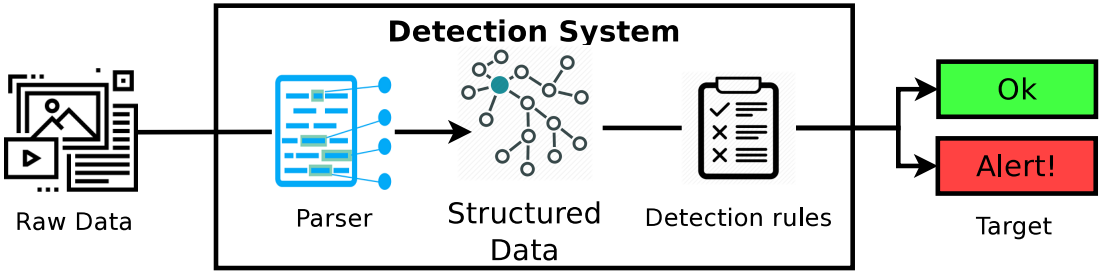
We can help lower the cost of feature engineering

AutoFeatures

AutoFeatures

What features can we generate if our data are already stored in a database

AutoFeatures



Detection Systems have parsers

AutoFeatures

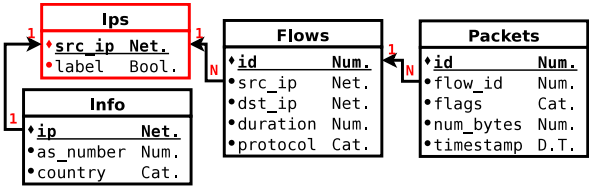
Parsers already contains domain knowledge

- **Entities** - Structures and fields
- **Types of values** - Numerical, Categorical, Strings, Boolean etc.
- **Relationships between entities** - A PE file has one PE header and multiple sections

AutoFeatures

This representation can easily be converted into an Entity Relationship Diagram (ERD)

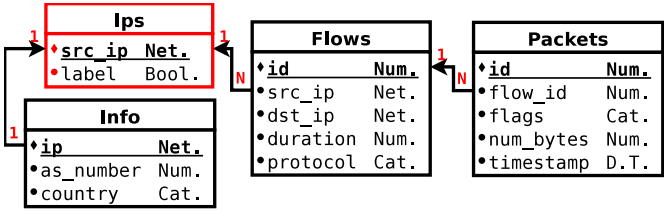
Structured Data



Entity Relationship Diagram (ERD)

AutoFeatures

ERD example - Network communcation analysis



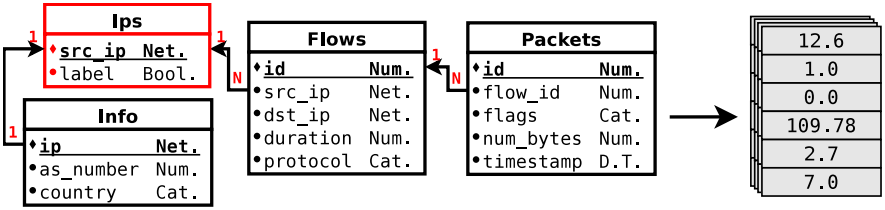
Entity Relationship Diagram (ERD)

- Root table is in red
- Each field in table has a type
- Arrows show relationships between entities

AutoFeatures / Feature Generation

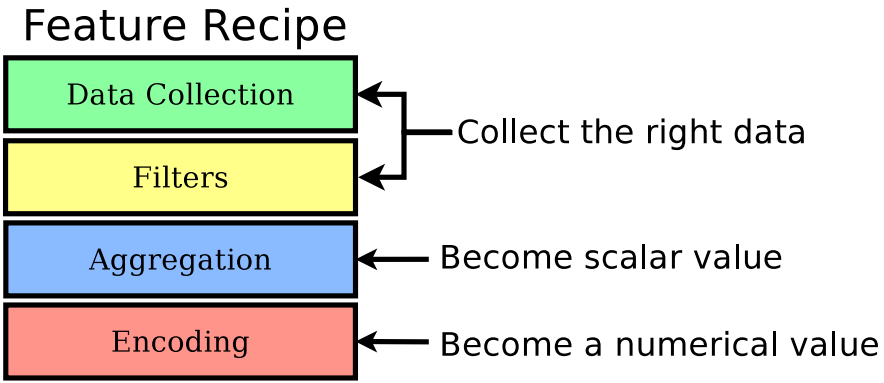
AutoFeatures

Goal: Generating features from an arbitrary Entity Relationship Diagram



Entity Relationship Diagram (ERD)

AutoFeatures - Feature Generation



AutoFeatures - Feature Generation

Data Collection - Joining path

Data Collection | Filters | Aggregation | Encoding

$$T_{root} \rightarrow T_1 \cdots \rightarrow T_i \rightarrow c$$

- Joining Path starting at table T_{root}
- It follows relationships
- to select the target column c belonging to table T_i.

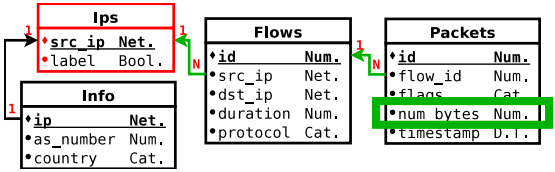
AutoFeatures - Feature Generation

Data Collection - Select

Data Collection | Filters | Aggregation | Encoding

$$T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \xrightarrow{\texttt{toN}} T_{\texttt{Packets}} \rightarrow \texttt{num_bytes}$$

The "size of each packet in bytes": list of numerical values

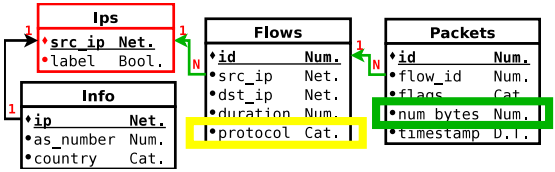


AutoFeatures - Feature Generation

Filters

$$T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \xrightarrow{\texttt{toN}} T_{\texttt{Packets}} \rightarrow \texttt{num_bytes} \mid \texttt{Flows.protocol} = \texttt{UDP}$$

Collects only the "sizes of packets involved in flows using the UDP protocol".



AutoFeatures - Feature Generation

Aggregation

Input Data	Aggregation Functions
numeric	MIN, MAX, SUM, MEAN, STD, VAR
boolean	AND, OR
categorical	MODE, NUM UNIQUE
primary key	COUNT, EXISTS

Table: AutoFeatures Default Aggregation Functions.

AutoFeatures - Feature Generation

Aggregation

$$\text{MEAN}\left(T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \xrightarrow{\texttt{toN}} T_{\texttt{Packets}} \rightarrow \texttt{num_bytes}\right)$$

The "average size (in bytes) of the packets".

AutoFeatures - Feature Generation

Multiple aggregations

$$\text{MEAN}\left(\text{COUNT}\left(T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \xrightarrow{\texttt{toN}} T_{\texttt{Packets}} \rightarrow \texttt{id}\right)\right.$$
$$\left.\text{GROUP BY}\,(\texttt{Flows.id})\right)$$

The "average number of packets by flow"

AutoFeatures - Feature Generation

Encoding

$$\text{SUM}\left(T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \rightarrow \texttt{duration}\right)$$

The "total duration of flows" : **is a numerical feature**

AutoFeatures - Feature Generation

Encoding

$$\text{EXISTS}\left(T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \rightarrow \texttt{id} \mid \texttt{Flows.protocol} = \texttt{UDP}\right)$$

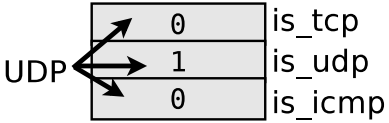
"Has connection through UDP" : **is a boolean feature**

AutoFeatures - Feature Generation

Encoding

$$\text{MODE}\left(T_{\texttt{Ips}} \xrightarrow{\texttt{toN}} T_{\texttt{Flows}} \rightarrow \texttt{protocol}\right)$$

The "most used protocol" : **is a categorical feature** (possible values `TCP`, `UDP` and `ICMP`)

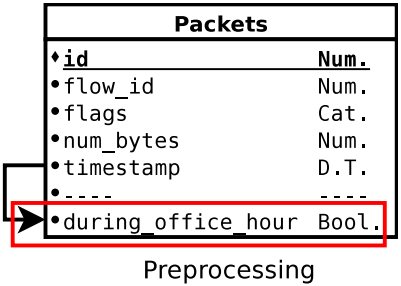


AutoFeatures / Expert Knowledge Injection

AutoFeatures - Expert Knowledge injection

Preprocessing

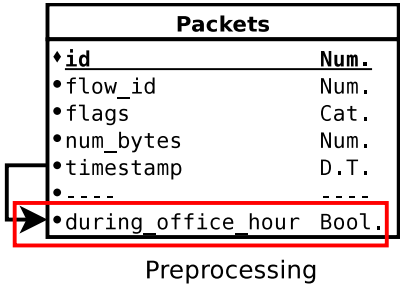
- Arbitrary transformation.
  - Computing ratios



AutoFeatures - Expert Knowledge injection

Preprocessing

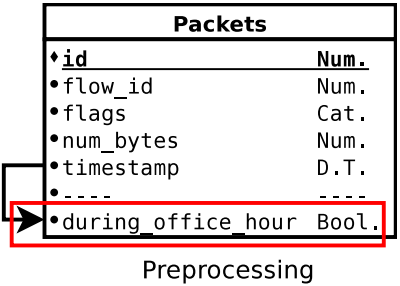
- Arbitrary transformation.
- Transform hardly exploitable values
  - Strings, ip adresses... $\rightarrow$ numerical values.



AutoFeatures - Expert Knowledge injection

Preprocessing

- Arbitrary transformation.
- Transform hardly exploitable values
- Values discretization
  - **Datetime** → **"during office hours" (Boolean)** or "the day of the week" (Categorical)
  - Network address → "local network address" (Boolean)



AutoFeatures - Expert Knowledge injection

Black and White listing

Blacklist

- Biased columns:
  `select:main.processes.pid, agg_funcs:`[SUM]

AutoFeatures - Expert Knowledge injection

Black and White listing

Blacklist

- Biased columns:
  `select:main.processes.pid, agg_funcs:`[SUM]
- Overly complex features:
  `agg_funcs:`[VAR, VAR]

AutoFeatures - Expert Knowledge injection

Black and White listing

Blacklist

- Biased columns:
  `select:main.processes.pid, agg_funcs:[SUM]`
- Overly complex features:
  `agg_funcs:[VAR, VAR]`

Whitelist

- Explore specific regions of the feature space
  `select:pe_files.sections.*`

Case Study: Generating Features for Windows PE Files

Case Study: Generating Features for Windows PE Files

Dataset: Ember [3] - Static analysis of PE files

- Various informations about Portable Executable files
- Informations extracted with Library to Instrument Executable Formats [1]

Domain knowledge:

- Windows PE Format Specification [2]
- Suspicious API functions (Zakeri et al. [4])

What features are generated?

Case Study: Generating Features for Windows PE Files

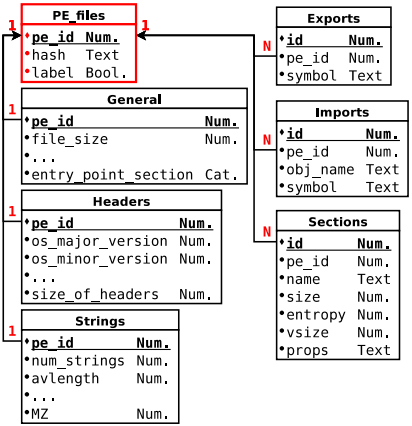
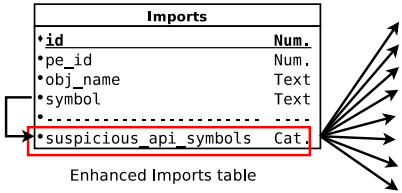


Figure: Simplified Entity Relationship Diagram (ERD) for the Ember Dataset.

Case Study: Generating Features for Windows PE Files

Preprocessing - Imports table

Zakeri et al. - A static heuristic approach to detecting malware targets [4] provides **16 suspicious win32 API call**.



Enhanced Imports table

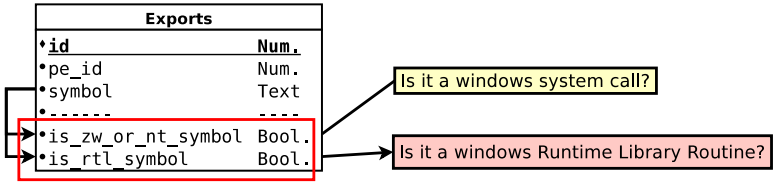
Table II. Application Program Interface calls frequency in malware and benign files.

API name	Frequency rate in malware	Frequency rate in benign file
GetProcAddress	0.74	0.45
LoadLibraryA	0.67	0.20
ExitProcess	0.66	0.11
GetModuleHandleA	0.57	0.24
VirtualAlloc	0.43	0.17
VirtualFree	0.37	0.15
RegCloseKey	0.35	0.33
GetModuleFileNameA	0.34	0.15

9 of the 16 suspicious API call listed by Zakeri. et al

Case Study: Generating Features for Windows PE Files

Preprocessing - Export table

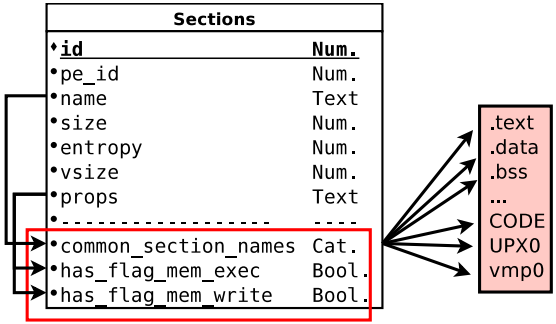


Enhanced Exports table

Case Study: Generating Features for Windows PE Files

Preprocessing - Export table

Windows PE Format Specification [2] provides **standard sections names**.



Enhanced Sections table

Case Study: Generating Features for Windows PE Files

Example of generated features

	Feature
1.	$T_{\texttt{PE_files}} \rightarrow T_{\texttt{General}} \rightarrow \texttt{entry_point_section}$
2.	$\text{SUM}\left(T_{\texttt{PE_files}} \xrightarrow{\text{toN}} T_{\texttt{Sections}} \rightarrow \texttt{size}\right)$
3.	$\text{EXISTS}\left(T_{\texttt{PE_files}} \xrightarrow{\text{toN}} T_{\texttt{Imports}} \rightarrow \texttt{id} \mid \texttt{Imports.suspicious_api_symbols} = \texttt{GetProcAddress}\right)$
4.	$\text{EXISTS}\left(T_{\texttt{PE_files}} \xrightarrow{\text{toN}} T_{\texttt{Imports}} \rightarrow \texttt{id} \mid \texttt{Imports.suspicious_api_symbols} = \texttt{LoadLibraryA}\right)$
5.	$\text{AND}\left(T_{\texttt{PE_files}} \xrightarrow{\text{toN}} T_{\texttt{Exports}} \rightarrow \texttt{id} \mid \texttt{Exports.is_zw_or_nt_symbol} = \texttt{True}\right)$
6.	$\text{MEAN}\left(T_{\texttt{PE_files}} \xrightarrow{\text{toN}} T_{\texttt{Sections}} \rightarrow \texttt{entropy} \mid \texttt{Sections.flag_mem_exec} = \texttt{True}\right)$
7.	$\text{MAX}\left(T_{\texttt{PE_files}} \xrightarrow{\text{toN}} T_{\texttt{Sections}} \rightarrow \texttt{entropy} \mid \texttt{Sections.flag_mem_exec} = \texttt{False}\right)$

Table: Examples of Generated Features

Conclusion

Conclusion

AutoFeatures

An automatic feature generation method designed to meet computer security experts' needs:

- Lowers the cost of feature engineering
- Generates interpretables features
- Gathers valuable expert knowledge

Conclusion

Future work

- Broader experiments on more datasets
- Feature space exploration strategies

Any questions?

Contact:

✉ **pierre.collet@ssi.gouv.fr**

Références

https://github.com/lief-project/LIEF.

https://docs.microsoft.com/en-us/windows/win32/debug/pe-format.

Ember: an open dataset for training static pe malware machine learning models.
arXiv preprint arXiv:1804.04637, 2018.

A static heuristic approach to detecting malware targets.
Security and Communication Networks, 8(17):3015–3027, 2015.