

# Generative AI for assessing network intrusion detection systems

---

Pierre-François Gimenez  
Researcher at Inria

Inria/AISI Workshop  
November 6th, 2024

## Why do we need data?

- For evaluating security measures, most notably detection
- For using machine learning in cybersecurity

## Current state of datasets

- Public datasets are typically run in testbed with no real users
- They can suffer from mislabelling, network and attack configurations errors, etc.
- We cannot access private data due to confidentiality and privacy reasons

⇒ we cannot confidently evaluate anomaly-based detection because of the dubious quality and the lack of realistic users

My research project: **use AI to generate security data**

## Goals

- Generation of network data (pcap files) and system data (logs)
- Temporal consistency and between network and system
- In-depth data quality evaluation
- Minimal expert's input

## Ongoing work: pipeline prototype

- We focus on benign network data
- Input data: pcap file
- Output data: a pcap file statistically similar to the input data

## Network data

- Raw data consist of packets, regrouped in conversation
- Cybersecurity analysis typically rely on network flow records that describe conversations statistically

No.	Time	Source	Destination	Protocol	Length	Info
17	0.709049029	193.51.196.138	131.254.252.23	DNS	126	Standard query response 0x170d AAAA pfgimenez.fr SOA dns12.ovh.net
18	0.708149062	131.254.252.23	185.199.109.153	TCP	74	42578 → 443 [SYN] Seq=0 Min=64240 Len=0 MSS=1460 SACK_PERM TSval=1731061
19	0.718482657	185.199.109.153	131.254.252.23	TCP	74	443 → 42578 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1448 SACK_PERM TS
20	0.718506446	131.254.252.23	185.199.109.153	TCP	66	42578 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1731066668 TSecr=25
21	0.718615194	131.254.252.23	185.199.109.153	TLsv1.3	599	Client Hello (SWIpfGimenez.fr)
22	0.736561279	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=1 Ack=534 Win=143872 Len=0 TSval=2597043199 TSecr=
23	0.742117140	185.199.109.153	131.254.252.23	TLsv1.3	519	Server Hello, Change Cipher Spec, Application Data, Application Data, Ap
24	0.742187989	131.254.252.23	185.199.109.153	TCP	66	42578 → 443 [ACK] Seq=534 Ack=454 Min=63872 Len=0 TSval=1731066692 TSecr=
25	0.743711093	131.254.252.23	185.199.109.153	TLsv1.3	139	Change Cipher Spec, Application Data
26	0.743855851	131.254.252.23	185.199.109.153	TLsv1.3	158	Application Data
27	0.747939089	131.254.252.23	185.199.109.153	TLsv1.3	566	Application Data
28	0.763212420	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=454 Ack=598 Win=143872 Len=0 TSval=2597043226 TSecr=
29	0.765612735	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=454 Ack=698 Win=143872 Len=0 TSval=2597043226 TSecr=
30	0.765612978	185.199.109.153	131.254.252.23	TLsv1.3	121	Application Data
31	0.765763178	131.254.252.23	185.199.109.153	TLsv1.3	97	Application Data
32	0.766914783	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=519 Ack=1190 Win=145408 Len=0 TSval=2597043230 TSecr=
33	0.784918198	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=519 Ack=1221 Win=145408 Len=0 TSval=2597043248 TSecr=
34	0.851063286	185.199.109.153	131.254.252.23	TLsv1.3	324	Application Data
35	0.851204999	131.254.252.23	185.199.109.153	TLsv1.3	101	Application Data
36	0.857984663	131.254.252.23	185.199.109.153	TLsv1.3	296	Application Data
37	0.857947165	131.254.252.23	185.199.109.153	TLsv1.3	293	Application Data, Application Data
38	0.860272768	131.254.252.23	185.199.109.153	TLsv1.3	162	Application Data
39	0.864607086	131.254.252.23	185.199.109.153	TLsv1.3	192	Application Data
40	0.867657367	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1256 Win=145408 Len=0 TSval=2597043330 TSecr=
41	0.877029712	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1396 Win=146432 Len=0 TSval=2597043338 TSecr=
42	0.877029938	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1623 Win=147456 Len=0 TSval=2597043338 TSecr=
43	0.878190357	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1719 Win=147456 Len=0 TSval=2597043342 TSecr=
44	0.883225268	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1755 Win=147456 Len=0 TSval=2597043346 TSecr=
45	0.950652163	185.199.109.153	131.254.252.23	TLsv1.3	178	Application Data
46	0.950652475	185.199.109.153	131.254.252.23	TLsv1.3	177	Application Data
47	0.959746916	131.254.252.23	185.199.109.153	TCP	66	42578 → 443 [ACK] Seq=1755 Ack=1800 Win=64128 Len=0 TSval=1731066909 TSecr=
48	0.968032125	131.254.252.23	185.199.109.153	TLsv1.3	191	Application Data
49	0.963572039	185.199.109.153	131.254.252.23	TLsv1.3	178	Application Data
50	0.963712830	131.254.252.23	185.199.109.153	TLsv1.3	136	Application Data, Application Data

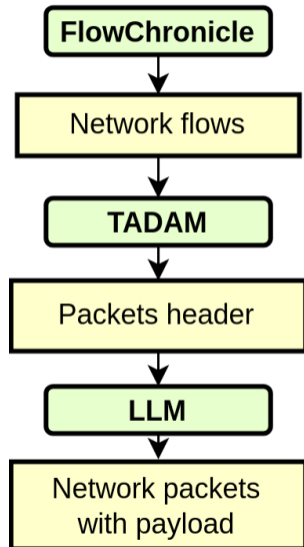
ts,proto,src\_ip,dst\_ip,dst\_port, fwd\_packets, bwd\_packets, fwd\_bytes, bwd\_bytes  
 1730800143,TCP,131.254.252.23,216.58.213.78,443,33,41,5988,1950

### State of the part

- Several approaches have been tried to generate network flow records or pcap files: VAE, GAN, LLMs
- The results are not very good:
  - A significant portion of generated data do not comply with network protocols
  - Generated data do not reflect the diversity of the original data

### Our approach: a three-step generation

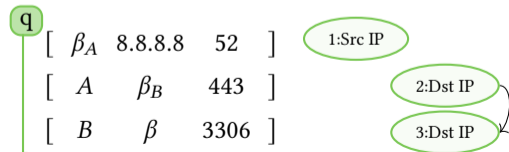
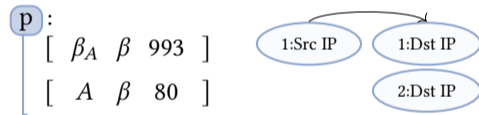
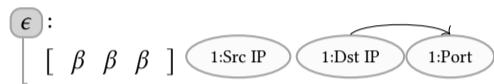
- FlowChronicle (published): a network flow generator
- TADAM (under review): a packet header generator
- Starting work with LLMs: full packet generator



## Pattern language

- Hybrid approach: pattern detection and statistical modeling
- Pattern detection: find temporal patterns of flows
  - DNS query then HTTP(S)
  - IMAP request then HTTP(S)
- Some values are fixed in the patterns
- The values that are not fixed are modeled with a Bayesian network
- These patterns are self-explanatory:
  - they can be verified by an expert
  - they can also be added manually
- This work was published recently

## Model – Pattern and Bayesian Network:



## Data and Pattern Windows:

Time	Src IP	Dst IP	Port
12	134.96.235.78	142.251.36.5	993
56	134.96.235.129	8.8.8.8	52
89	134.96.235.78	212.21.165.114	80
113	134.96.235.129	198.95.26.96	443
145	198.95.26.96	198.95.28.30	3306
156	134.96.235.78	134.96.234.5	21
178	134.96.235.36	185.15.59.224	993
206	134.96.235.36	128.93.162.83	80

	Density	CMD	PCD	EMD	JSD	Coverage	DKC	MD	Rank
	<i>Real.</i> ↑	<i>Real.</i> ↓	<i>Real.</i> ↓	<i>Real./Div.</i> ↓	<i>Real./Div.</i> ↓	<i>Div.</i> ↑	<i>Comp.</i> ↓	<i>Nov.</i> =	<i>Average Ranking</i>
<b>Reference</b>	(0.69)	(0.06)	(1.38)	(0.00)	(0.15)	(0.59)	(0.00)	(6.71)	-
<b>IndependentBN</b>	7 (0.24)	5 (0.22)	6 (2.74)	8 (0.11)	4 (0.27)	4 (0.38)	4 (0.05)	4 (5.47)	5.25
<b>SequenceBN</b>	6 (0.30)	2 (0.13)	5 (2.18)	7 (0.08)	3 (0.21)	3 (0.44)	2 (0.02)	3 (5.51)	3.875
<b>TVAE</b>	3 (0.49)	4 (0.18)	3 (1.84)	2 (0.01)	5 (0.30)	5 (0.33)	6 (0.07)	5 (5.17)	4.125
<b>CTGAN</b>	2 (0.56)	3 (0.15)	2 (1.60)	3 (0.01)	2 (0.15)	2 (0.51)	8 (0.11)	2 (5.70)	3.0
<b>E-WGAN-GP</b>	8 (0.02)	7 (0.34)	8 (3.63)	5 (0.02)	7 (0.38)	8 (0.02)	7 (0.07)	6 (4.66)	7.0
<b>NetShare</b>	5 (0.32)	6 (0.28)	<b>1 (1.47)</b>	6 (0.03)	6 (0.36)	6 (0.22)	5 (0.05)	7 (3.82)	5.25
<b>Transformer</b>	<b>1 (0.62)</b>	8 (0.78)	7 (3.62)	<b>1 (0.00)</b>	8 (0.55)	7 (0.03)	3 (0.05)	8 (3.75)	5.375
<b>FlowChronicle</b>	4 (0.41)	<b>1 (0.03)</b>	4 (2.06)	4 (0.02)	<b>1 (0.10)</b>	<b>1 (0.59)</b>	<b>1 (0.02)</b>	<b>1 (5.87)</b>	<b>2.125</b>

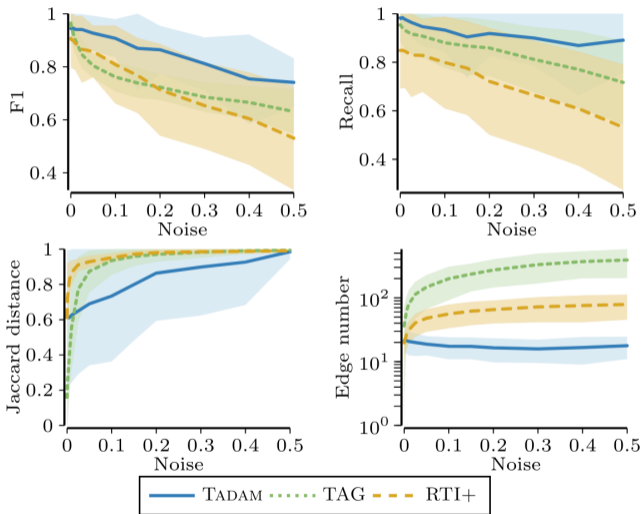


## Learning

- Network protocols typically rely on finite state automata
- We propose to learn probabilistic timed automata to capture packet header sequences
- Existing automata learners from observations cannot handle noisy data
- We propose TADAM: a robust timed automata learner
- Two main contributions:
  - A compression-based score to avoid overfitting
  - An explicit modelization of the noise

## Experimental results

- TADAM is far more robust to noise
- TADAM learns smaller models
- TADAM has better performance on real-world classification and anomaly detection tasks



Learner	AU-ROC	TPR	FPR	F1
TADAM	<b>0.982</b>	0.998	<b>0.025</b>	<b>0.705</b>
TAG	0.891	<b>1</b>	0.142	0.298
RTI+	0.790	<b>1</b>	0.292	0.171
HMM	0.608	0.640	0.085	0.288

Table 3: Anomaly detection performance on *HDFS\_v1* dataset. We report the TPR, FPR and F1-score for the threshold maximizing TPR-FPR.

## Generation from automata

- With a probabilistic automata, we can easily sample packet headers sequences
- But generation must be parameterized according to a network flow record!
- For example: total size = 5200 bytes, 5 forward packets, 8 backward packets
- This can be done easily by representing the constraints by an automaton and computing the intersection between the protocol automaton and the constraints automaton

## From headers sequence to packets

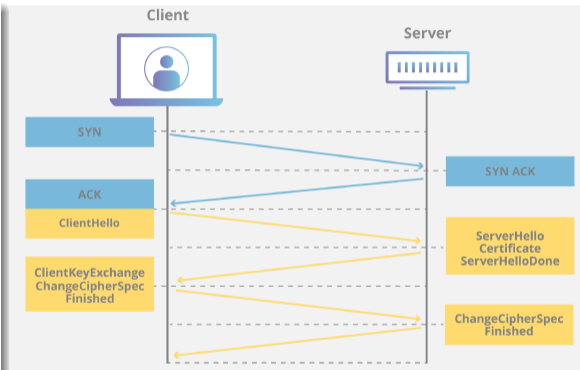
- Most data can be filled automatically (ACK number, checksum, etc.)
- Some payloads are encrypted, so we can generate random data that are indistinguishable
- For plain-text payloads, we propose to replay them or to use LLMs
- We did some preliminary experiments with GPT-4 to generate realistic payloads, but conditioning the generation is not reliable and it is slow

## Example: TLS handshake generation

It must be:

- Consistent with the packet size generated by TADAM: the length of packet is highly influenced by the signature length of the cipher suite
- Consistent with the protocol:
  - The server name should be consistent in ClientHello and ServerHello
  - The cipher used in ServerHello should be available in ClientHello
  - Different OS use different ciphers

Not an easy task for LLMs!



The four packets (in yellow) of a TLS handshake

## System data generation

- Our next goal will be to generate system data, i.e., logs
- We propose to proceed with a two-step approach:
  - generate a provenance graph (graph of interactions between system entities and resources)
  - generate logs from such interactions

## Log parsing and generation

- Log parsing is notoriously complex
- Each application has its own semi-structured format, and it tends to change
- Log parsing and generation could be a perfect application for LLMs
- On top of well-known formats that could be directly generated, more obscure formats could be learned with few-shot learning or fine tuning

## The need of data

- Good quality data is of utmost importance for security system evaluation
- One way to achieve such quality is through generative AI

## Current and future work

- "Classical" AI can yield better quality generation for low-dimension feature spaces, on top of being explainable
  - adapted to intermediate data structure generation
- LLMs is certainly a key to generating actual data, i.e., packet payload and logs
  - conditioning their generation remains a challenge