

# A look back at the SecGen associate team



Collaboration Inria-CISPA from 2022 to 2025

Pierre-François Gimenez, Inria researcher

---

# Context

# Context

## Who am I?

- ▶ PhD in machine learning in 2018
- ▶ Now a researcher at Inria in Rennes
- ▶ Research keywords: anomaly-based intrusion detection, synthetic data generation

## Collaboration with CISPA

- ▶ I visited Mario Fritz's research group for 3 months in 2022
- ▶ Mario and I coordinated an "associate team" between 2023 and 2025
- ▶ SecGen focuses on synthetic data generation and anomaly detection
- ▶ Website <https://files.inria.fr/secgen/>



# Associate Team program

## What is it?

- ▶ A way to quick-start collaborations between Inria and labs abroad
- ▶ A mobility funding from Inria's international relations management
- ▶ ~ 10000€ per year for three years
- ▶ A lightweight application file
- ▶ The perfect tool to explore collaboration with minimal risk

How much can we do with a "small" funding?

# Scientific output

## Scientific stays

- ▶ Adrien Schoen: 2 months at CISPA
- ▶ Joscha Cüppers: 2 months at Inria
- ▶ Lénaïg Cornanguer: 1 week at Inria

## SecGen's achievements

- ▶ PF Gimenez, S Sivaprasad, M Fritz, *Certifiably robust malware detectors by design*, at IFIP SEC (B-rank), 2025
- ▶ J Cüppers, A Schoen, G Blanc, PF Gimenez, *Flowchronicle: synthetic network flow generation through pattern set mining*, at CoNEXT (A-rank), 2024, best paper
- ▶ L Cornanguer, PF Gimenez, *TADAM: Learning Timed Automata from Noisy Observations*, at SDM (A-rank), 2025

## The team





# The team

## From Inria

- ▶ **Pierre-François Gimenez** (PI, tenured researcher)
- ▶ Frédéric Majorczyk (PhD, engineer)
- ▶ Adrien Schoen (PhD student)

## From CISPA

- ▶ **Mario Fritz** (PI, faculty)
- ▶ Jilles Vreeken (faculty)
- ▶ Lénaïg Cornanguer (post-doc)
- ▶ Sarath Sivaprasad (PhD student)
- ▶ Joscha Cüppers (PhD student)

# Scientific Contributions



# Information system security

## Information system security

- ▶ Prevent the attack, detect it, and react
- ▶ Detection with **IDS**: *Intrusion Detection System*

## Detection relies on observation

- ▶ **System** : OS and applications logs
- ▶ **Network** : network communications

## Constraints

- ▶ Partial and heterogeneous observations
- ▶ Adversarial context: the attacker hides!

```
2024-05-06T23:24:16.806598+02:00
stellar-sheep sshd[16039]: Failed
password for pfg from 192.168.1.36
port 48650 ssh2
```

```
"ts": 1591367999.305988,
"id.orig_h": "192.168.4.76",
"id.resp_h": "192.168.4.1",
"id.resp_p": 53, "proto": "udp",
"service": "dns", "duration":
0.066851, "orig_bytes": 62,
"resp_bytes": 141, "conn_state":
"SF", "orig_pkts": 2,
"orig_ip_bytes": 118, "resp_pkts":
2, "resp_ip_bytes": 197
```



# The issue of data in security

## 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 intrusion detection systems because of this dubious quality

The goal of SecGen: **use AI to generate security data**

# Network data example

## 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	0.706049029	193.51.196.138	131.254.252.23	DNS	126	Standard query response 0x170d AAAA pfginenez.fr SOA dns12.ovh.net
18	0.708149061	131.254.252.23	185.199.109.153	TCP	74	42578 → 443 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM TS=1731066143
19	0.718482667	185.199.109.153	131.254.252.23	TCP	74	443 → 42578 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1444 SACK_PERM TS=1731066143
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=2597043326
21	0.718615194	131.254.252.23	185.199.109.153	TLSv1.3	599	Client Hello (SMI=pfginenez.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=2597043326
23	0.742171740	185.199.109.153	131.254.252.23	TLSv1.3	519	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
24	0.742187989	131.254.252.23	185.199.109.153	TCP	66	42578 → 443 [ACK] Seq=534 Ack=454 Win=63872 Len=0 TSval=1731066692 TSecr=2597043326
25	0.742191103	131.254.252.23	185.199.109.153	TLSv1.3	100	Change Cipher Spec, Application Data
26	0.743855851	131.254.252.23	185.199.109.153	TLSv1.3	158	Application Data
27	0.747936849	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=2597043326
29	0.765612735	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=454 Ack=699 Win=143872 Len=0 TSval=2597043226 TSecr=2597043326
30	0.765612978	185.199.109.153	131.254.252.23	TLSv1.3	131	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=2597043236 TSecr=2597043326
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=2597043326
34	0.851093286	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	0.857904663	131.254.252.23	185.199.109.153	TLSv1.3	206	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	0.864607066	131.254.252.23	185.199.109.153	TLSv1.3	192	Application Data
40	0.867657307	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1256 Win=145408 Len=0 TSval=2597043330 TSecr=2597043326
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=2597043326
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=2597043326
43	0.879109357	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1719 Win=147456 Len=0 TSval=2597043342 TSecr=2597043326
44	0.883252568	185.199.109.153	131.254.252.23	TCP	66	443 → 42578 [ACK] Seq=777 Ack=1755 Win=147456 Len=0 TSval=2597043346 TSecr=2597043326
45	0.959652163	185.199.109.153	131.254.252.23	TLSv1.3	178	Application Data
46	0.959652475	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=1000 Win=64128 Len=0 TSval=1731066999 TSecr=2597043326
48	0.960032125	131.254.252.23	185.199.109.153	TLSv1.3	101	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

```

Frame 25: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bytes) on interface eth0
Ethernet II, Src: Intel_E8:9e:8c:cd (28:a0:60:9e:8c:cd), Dst: Intel_E8:9e:8c:cd (28:a0:60:9e:8c:cd)
Internet Protocol Version 4, Src: 131.254.252.23, Dst: 185.199.109.153
Transmission Control Protocol, Src Port: 42578, Dst Port: 443
Transport Layer Security
  TLSv1.3 Record Layer: Change Cipher Spec Protocol
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  TLSv1.3 Record Layer: Application Data Protocol;
  
```

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



# Synthetic network traffic generators

## What is synthetic data?

Data generated without simulation or emulation

## Why synthetic data?

Some advantages:

- ▶ Much faster generation
- ▶ Allows to quickly iterate over dataset quality
- ▶ Ensure generation replicability
- ▶ Can have better scalability

But generating high-quality data is difficult



## Related work

### What generation techniques are used?

Deep learning techniques:

- ▶ Many, many GAN (generative adversarial network) and variations
- ▶ VAE (variational auto-encoder) and variations
- ▶ Diffusion models
- ▶ LLM (of course)

Other techniques:

- ▶ SMOTE
- ▶ Bayesian networks

In our approach, we focus on **statistical, explainable** AI techniques for **fast, trustworthy** generated data

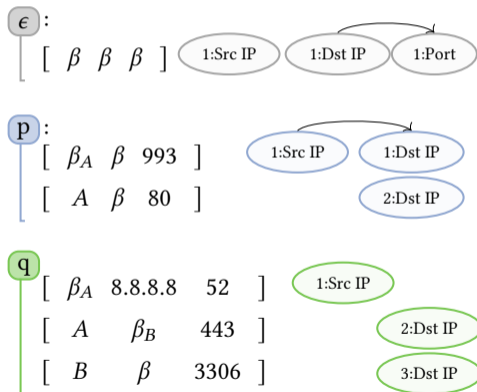


# FlowChronicle (CoNEXT'24, best paper)

## General idea

- ▶ Joint work with Joscha Cüppers
- ▶ General approach: find patterns in the data and use them to generate new data
- ▶ We focus on temporal patterns of flows
  - DNS query then HTTP(S)
  - IMAP request then HTTP(S)
- ▶ These patterns are self-explanatory:
  - they can be verified by an expert
  - they can also be added manually

## 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

# Pattern Description

## Pattern language

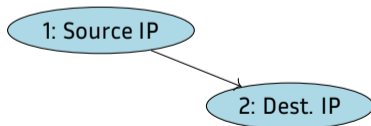
Each pattern has two part: a partially defined flow, and a Bayesian network

- ▶ **Fixed** values are defined in the partial flow
- ▶ the distribution of **Free** variables is defined in the Bayesian network
- ▶ **Reused** variables are always equal to some **Free** variable

Partial flows

Source IP	Dest. IP	Dest. Port
$\beta_A$	8.8.8.8	53
$A$	$\beta$	80

Bayesian Network



In reality there are more columns!

# Data quality evaluation

## Hard to evaluate

- ▶ No standard metrics
- ▶ Evaluation often partial

## Proposition

A set of evaluating metrics:

**Realism** : Are the generated data part of the target distribution?

**Diversity** : can any point in the target distribution be generated?

**Novelty** : can the generator create data absent from the training set?

**Compliance** : do the generated data comply with the technical specifications?

We do not consider privacy yet

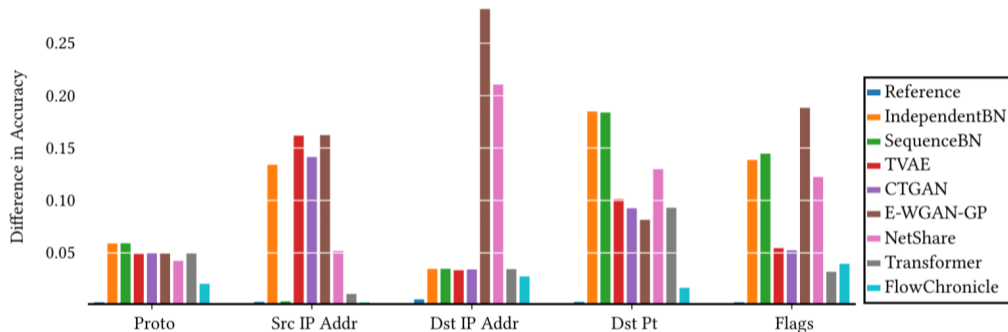


# FlowChronicle: generation quality

	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>
	↑	↓	↓	↓	↓	↑	↓	=	
Reference	<b>0.69</b>	<b>0.06</b>	<b>1.38</b>	<b>0.00</b>	<b>0.15</b>	<b>0.59</b>	<b>0.00</b>	<b>6.71</b>	-
IndependentBN	0.24	0.22	2.74	<b>0.11</b>	0.27	0.38	0.05	5.47	5.25
SequenceBN	0.30	<b>0.13</b>	2.18	0.08	0.21	0.44	<b>0.02</b>	5.51	3.875
TVAE	0.49	0.18	1.84	<b>0.01</b>	0.30	0.33	0.07	5.17	4.125
CTGAN	<b>0.56</b>	0.15	<b>1.60</b>	0.01	<b>0.15</b>	<b>0.51</b>	<b>0.11</b>	<b>5.70</b>	<b>3.0</b>
E-WGAN-GP	<b>0.02</b>	0.34	<b>3.63</b>	0.02	0.38	<b>0.02</b>	0.07	4.66	7.0
NetShare	0.32	0.28	<b>1.47</b>	0.03	0.36	0.22	0.05	3.82	5.25
Transformer	<b>0.62</b>	<b>0.78</b>	3.62	<b>0.00</b>	<b>0.55</b>	0.03	0.05	<b>3.75</b>	<b>5.375</b>
FlowChronicle	0.41	<b>0.03</b>	2.06	0.02	<b>0.10</b>	<b>0.59</b>	<b>0.02</b>	<b>5.87</b>	<b>2.125</b>



## FlowChronicle: temporal generation quality





# Data generated with FlowChronicle

## Output of FlowChronicle

- ▶ FlowChronicle outputs network flow records, e.g:

```
ts,proto,src_ip,dst_ip,dst_port,fwd_pkts,bwd_pkts,fwd_bytes,bwd_bytes  
1730800143,TCP,131.254.252.23,216.58.213.78,443,33,41,5988,1950
```

- ▶ But in the end, we want to generate packets!

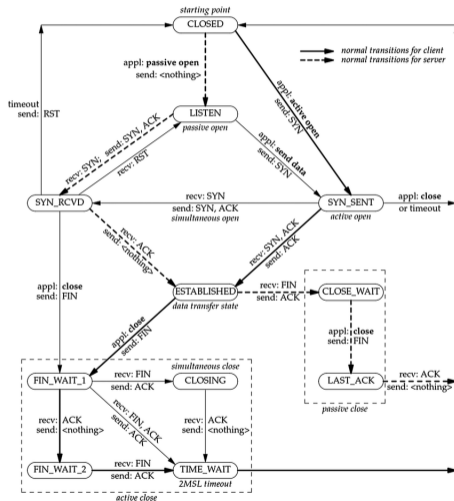
## Next step

- ▶ Before generating complete packets, we propose to first generate an intermediate representation
- ▶ More precisely, we generate for each packet a tuple with:
  - the direction (forward or backward)
  - the TCP flags
  - the size of the payload
  - the time since the last packet (i.e., the inter-arrival time)

# Automata learning

## Why not use the specification?

- ▶ Network protocols typically rely on finite state automata
- ▶ Official automata from RFC do not tell about the actual behavior
- ▶ Not all protocols are documented with a finite state automaton
- ▶ It is very easy to sample new sequences from a probabilistic automata





## State of the art

### Automaton learning

- ▶ Active learning: requires a "teacher" to verify the automaton during learning
- ▶ Passing learning: requires observations
  - Learning from positive and negative examples
  - Learning from positive examples only (TAG, RTI+)

### Limitations of TAG and RTI+

- ▶ By design, all observations will be included in the automaton
- ▶ For example, if a client cannot connect and send many "SYN" packets to the server, this trace will be included in the protocol automaton
- ▶ But network data can be noisy! So it leads to bloated automata

**We introduce TADAM, an automata learning that is robust to noise**

# Learning process

## Learning process

- ▶ We start with an automaton initialized like a Markov chain
- ▶ We apply elementary operations (split a node or an edge, merge two nodes, delete an edge)
- ▶ We remove the noise of the data so it fits the model
- ▶ We evaluate the new automata with a **score** and keep the best

# Learning process

## Learning process

- ▶ We start with an automaton initialized like a Markov chain
- ▶ We apply elementary operations (split a node or an edge, merge two nodes, delete an edge)
- ▶ We remove the noise of the data so it fits the model
- ▶ We evaluate the new automata with a **score** and keep the best

## The score (intuition)

This an MDL score based on compression:  $L(D, A) = L(D | A) + L(A)$

- ▶  $L(D, A)$  is the score of automaton  $A$  for some data  $D$
- ▶  $L(D | A)$  is the length of describing the data encoded with the automaton
- ▶  $L(A)$  is the length of describing the automaton

## The full score

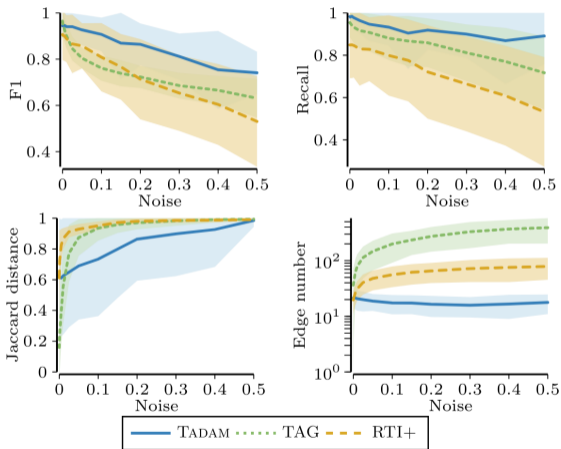
$$L(A) = L_{\mathbb{N}}(|\mathcal{Q}|) + L_{\mathbb{N}}(|\Sigma|) + \sum_{e \in \mathcal{E}} \left( 2 \log_2(|\mathcal{Q}|) + \log_2(|\Sigma|) + L_{\mathbb{N}}(\lfloor \mu_e \rfloor) + L_{\mathbb{N}}(\lfloor \sigma_e^2 \rfloor) \right) + 2 \log_2(|\mathcal{Q}|)$$

$$\begin{aligned} L(D|A) = & \sum_{(o,e,d) \in R} -\log_2(p(o)) + \sum_{(o,e,d) \in R_{\{\text{tr,fo,ad}\}}} -\log_2 p(e | q_s(e)) \\ & + \sum_{(o,e,d) \in R_{\{\text{tr,fo,de}\}}} -\log_2 p(d | e) + \sum_{(o,e,d) \in R_{\{\text{sk}\}}} (L_{\mathbb{N}}(d) + \log_2(|\Sigma|)) \end{aligned}$$

The originality is that we explicitly model the noise. Train set can have:

- ▶ missing packets (due to concurrent routes)
- ▶ swapped packets (due to system latency in probes)
- ▶ duplicated packets (due to TCP retransmission)

## TADAM: experiments



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.

# Conclusion



# Conclusion

## 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
- ▶ We achieved very promising results

## A look back at SecGen

- ▶ Overall, a very positive experience
- ▶ High-quality scientific articles, a "best paper" award
- ▶ SecGen reached a natural conclusion with these works
- ▶ While SecGen ended, I still build upon these works