FlowChronicle

Synthetic Network Flow Generation Through Pattern Set Mining

Joscha Cüppers[1]; Adrien Schoen[2]; Gregory Blanc[3]; **Pierre-François Gimenez**[2]

[1]CISPA, Germany; [2]Inria, France; [3]Telecom SudParis, France

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

THCon25
April 11, 2025

Inria

Information system security

How to protect information system?

- 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

Main issues

- Detect APT attacks on long period of time
- Limit false positives
- Good quality data?


```
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

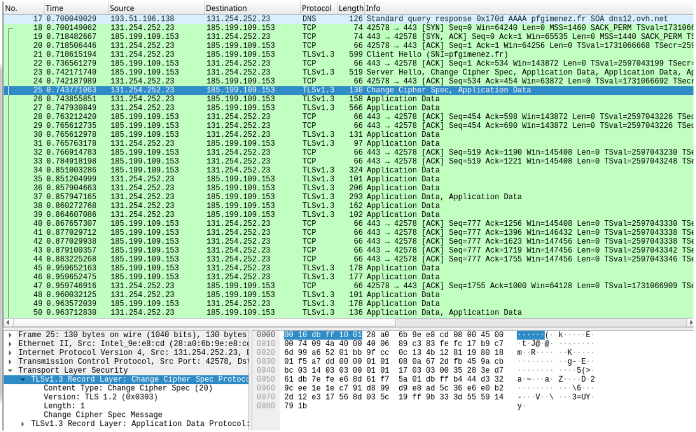
$\Rightarrow$ we cannot confidently evaluate intrusion detection systems because of this dubious quality

Our goal: **to use AI to generate synthetic network 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
- This is the kind of data we want to generate




```
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
```

Just use an LLM!

State of the part

- Several approaches have been tried to generate network flows or pcap: 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
  - The models are not explainable
  - More generally, the dependencies are not well replicated

Dependencies

- Intra-flow dependency
  - the port depends on the destination IP
  - the number of packets depends on the application protocol
- Inter-flow dependency:
  - DNS query then HTTP(S)
  - IMAP request then HTTP(S)

Contribution: FlowChronicle

FlowChronicle: A Novel Approach

- **Pattern Language**
  - Captures intra-flow and inter-flow dependencies
  - Summarizes data with non-redundant patterns
- **Data Generation**
  - Produces realistic traffic respecting protocols
  - Preserves temporal dependencies
- **Interpretability**
  - Patterns are interpretable and auditable

FlowChronicle

Intro

What is a pattern?

Frequently occurring substructure in data

Pattern Mining

- Define the set of possible patterns, named the "pattern language"
- Find a small set of patterns that best describes the data
- More precisely, we use the patterns to compress the data: higher the compression, better the patterns

Pattern description

Pattern language

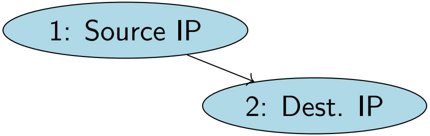
Each pattern has two part: a partially defined flow, and a Bayesian network

- 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
β_A	8.8.8.8	53
A	β	80

Bayesian Network



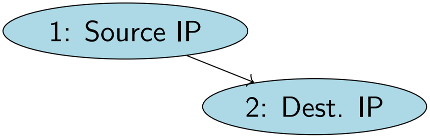
In reality there are more columns!

Pattern description

Partial flows

Source IP	Dest. IP	Dest. Port
β_A	8.8.8.8	53
A	β	80

Bayesian Network



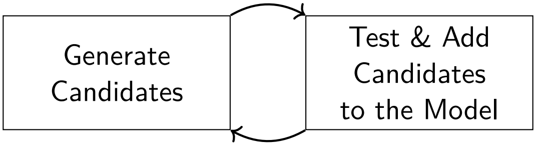
Example

- Here, there are two flows
- The first flow is contacting 8.8.8.8 on port 53 (DNS). The source IP is random
- The second flow has the same source IP as the first flow, and is contacting a destination IP that is random and depends on the first source IP, on port 80 (HTTP)

Our goal is to learn ("mine") such patterns

Mining process

Basic Idea - Two Steps:



Candidate generation

Extending existing pattern with attribute:

Existing Pattern:

Flow	Src IP	Dst IP	Port
1	β_A	8.8.8.8	53
2	A		443

New Pattern Candidate:

Flow	Src IP	Dst IP	Port
1	β_A	8.8.8.8	53
2	A		443
3			3306

Candidate generation

Extending existing pattern with attribute:

Existing Pattern:

Flow	Src IP	Dst IP	Port
1	β_A	8.8.8.8	53
2	A		443

New Pattern Candidate:

Flow	Src IP	Dst IP	Port
1	β_A	8.8.8.8	53
2	A		443
3			3306

Merging existing patterns:

Existing Patterns:

Flow	Src IP	Dst IP	Port
1	β_A	8.8.8.8	53
2	A		443

Flow	Src IP	Dst IP	Port
1		8.8.8.8	53

New Pattern Candidate:

Flow	Src IP	Dst IP	Port
1	β_A	8.8.8.8	53
2	A		443
3		8.8.8.8	53

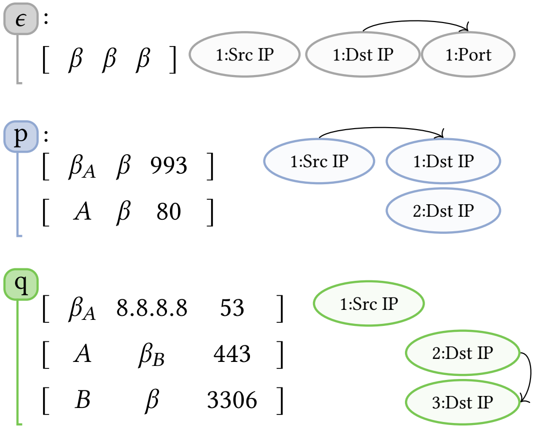
Pattern mining algorithm

Pattern Search:

1. Initialize Model with an empty pattern
2. Generate Pattern Candidates from existing patterns $p \in M$.
   - By extending with an attribute
   - By merging existing patterns
3. Test candidates for addition:
   - Cover the datasets with the patterns
   - Add patterns when it reduces MDL score: $L(D \mid M) + L(M)$

Dataset cover

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	53
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

Loss function

Length of data given the model:

$$L(D \mid M) = \sum_{p \in M} (L_{\mathbb{N}}(|W_p|) + L(W_p))$$

where:

$$L(W_p) = \sum_{i=1}^{|W_p|} \left(L(t_1 \text{ of } w_i) + \sum_{k=2}^{|p|} L(t_k \text{of } w_i \mid t_{i-1}) \right) - \log(Pr(w_i | BN_p, \{w_j | j < i\}))$$

Length of Model:

$$L(M) = L_{\mathbb{N}}(|M|) + \sum_{p \in M} L(p)$$

Length of one pattern:

$$L(p) = L_{\mathbb{N}}(|p|) + \left(\sum_{j=1}^{|p|} L(X[j]|p) \right) + L(BN_p)$$

Generating network flows from a model

Key Steps

1. Select Patterns: Sample patterns from the model.
2. Generate Timestamp of the First Flow: sample a timestamp from the timestamp distribution.
3. Generate Delays Between the Flows: sample a delay from the delay distribution.
4. Fill Values:
   - Fixed cells: Predefined values.
   - Free cells: Sampled from the Bayesian Network (BN).
   - Reuse cells: Context-based values.

Experiments

Data quality evaluation

Hard to evaluate

- No standard metrics
- Evaluation often partial

Proposition

A set of evaluating metrics:

Realism : could the data actually exist?

Diversity : do we generate the diversity of behavior from the training set?

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

Experimental protocol

Training data

We use the CIDDS 001 dataset: train on one week of traffic and generate one week of traffic

Baselines

We compare FlowChronicle with:

- Bayesian networks
- Variational autoencoders
- GAN
- Transformers
- "Reference": actual data from the same dataset to simulate the best generative method

Non-temporal Evaluation

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

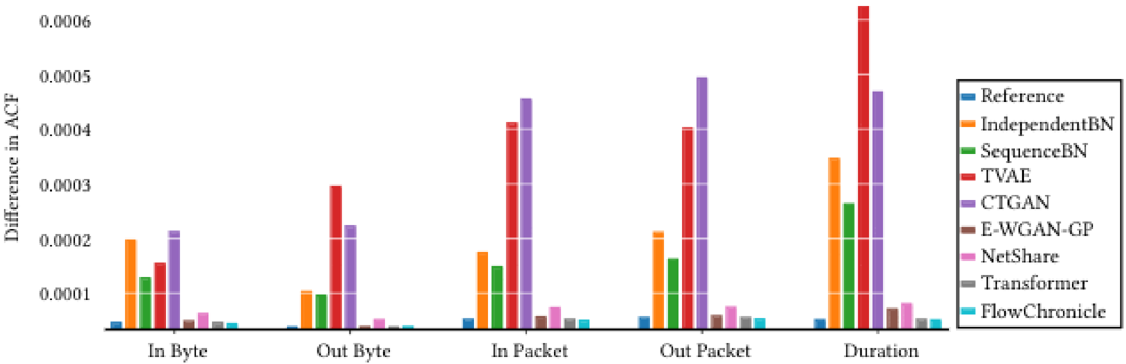
FlowChronicle produces overall the best traffic among the generative methods

Temporal Dependencies: Numerical Features

Difference in Autocorrelation Functions

- Autocorrelation function: correlation between the value of a feature and the value of this feature to other timestamps
- Evaluation : difference between autocorrelation of training data and synthetic data for each feature
- Lower is better

Temporal Dependencies: Numerical Features

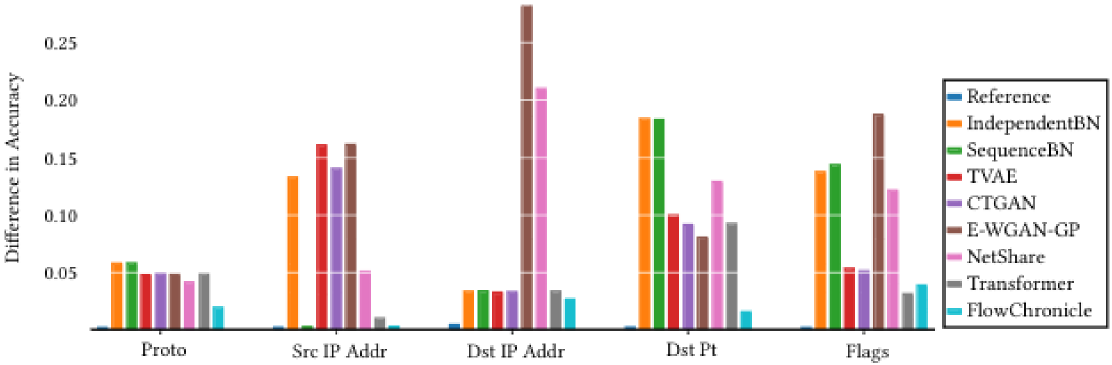


Temporal Dependencies: Categorical Features

Difference in the accuracy of LSTM autoregressive models

- Train an LSTM to predict the value of a feature
  - Input: Previous value of the feature $\rightarrow$ autoregressive task
- Difference of accuracy between two LSTMs on real data:
  - First LSTM trained on the Training Dataset
  - Second LSTM trained on the Synthetic Dataset
- Lower is better

Temporal Dependencies: Categorical Features



Beyond FlowChronicle

Data generated with FlowChronicle

Output of FlowChronicle

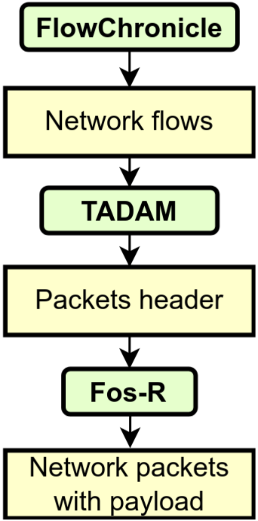
- FlowChronicle outputs network flow records, e.g:

```
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
```

- How to generate packets from that?

Next intermediary 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)



TADAM

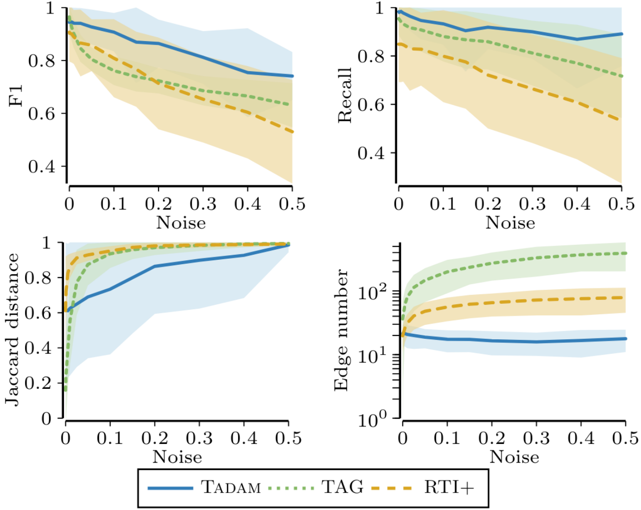
Learning

- 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

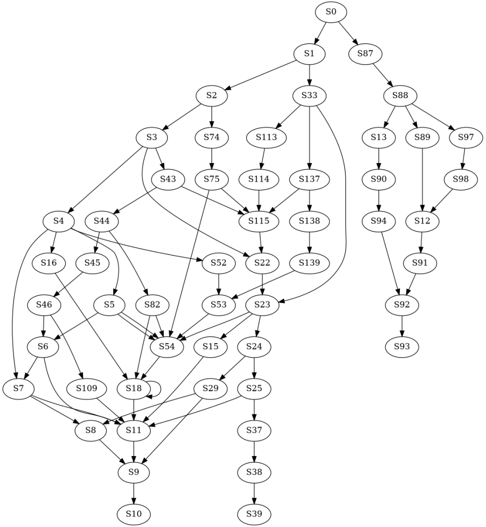
TADAM: experiments



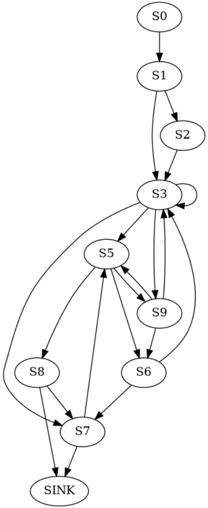
Learner	AU-ROC	TPR	FPR	F1
TADAM	**0.982**	0.998	**0.025**	**0.705**
TAG	0.891	**1**	0.142	0.298
RTI+	0.790	**1**	0.292	0.171
HMM	0.608	0.640	0.085	0.288

Table 3: *Anomaly detection performance on HDFS_v1 dataset.* We report the TPR, FPR and F1-score for the threshold maximizing TPR-FPR.

Example: Kerberos protocol



TAG, state of the art



TADAM, our method

Data generated with TADAM

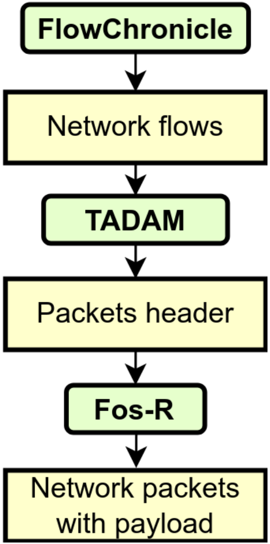
Output of TADAM

TADAM outputs tuples, e.g: (FWD, SYN, 0, 0), (BWD, SYN/ACK, 0, 2), (FWD, ACK, 0 3), (FWD, PUSH, 123, 10), ...

Fos-R: bridging the gaps

Fos-R creates the full packets:

- The rest of the header is creating according to some rules (window size, checksum, etc.)
- For now, the payload is replayed or random ⇒ payload generation is a difficult problem



Fos-R

Two modes of generation

- Static pcap creation
- Network injection: the flow are played on the network without communication overhead, for honeynet and cyber range

Maturity

- Fos-R has been deployed BreizhCTF2025 (biggest in-person CTF in France). It generated 20,000h of data in total
- The software will be publicly available in Winter 2025

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

Contributions of FlowChronicle

- Innovative pattern set mining approach for synthetic network traffic generation
- Maintains both flow quality and temporal dependencies
- Top performance: outperformes other generative models.
- Auditable Patterns: enables explainable and adaptable generation.

We built upon FlowChronicle for pcap generation

Future works

Next step: joint pcap/system logs generation