RÉPUBLIQUE
FRANÇAISE
Liberté
Égalité
Fraternité

Learning timed automata for synthetic network traffic generation

TADAM: Learning Timed Automata from Noisy Observations

Lénaïg Cornanguer[1]; **Pierre-François Gimenez**[2]

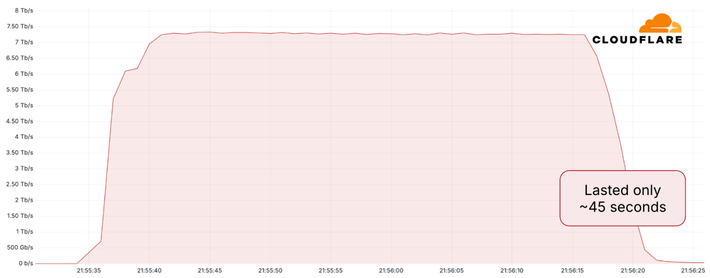
[1]CISPA, Germany; [2]Inria, France

Introduction

Systems are under attack

- Many untargeted, opportunistic attacks like password bruteforce
- Some targeted attacks with a huge power (e.g., DDoS attacks)
- Some very sophisticated attacks months or years in the making (SolarWinds, Stuxnet, xz...)

Cloudflare defenses autonomously block a 7.3 Tbps DDoS attack



In May 2025, an attack delivered 37.4 terabytes in 45 seconds

Information system security

Information system security

- Prevent the attack, detect it, and react
- Detection with **Intrusion Detection Systems**

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 tools, most notably detection
- For using machine learning in cybersecurity

The issue of data in security

Why do we need data?

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

Current state of datasets

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

The issue of data in security

Why do we need data?

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

Current state of datasets

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

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

The issue of data in security

Why do we need data?

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

Current state of datasets

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

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

A solution is to rely on **synthetic data**

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

But generating high-quality data is difficult

Generated network data

- Flow, like NetFlow. Commonly used in cybersecurity. Very adapted to ML (tabular data). The vast majority of work generate flows.
- Raw packets. Much less studied, but necessary for some applications (such as background traffic generation)

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

Other techniques:

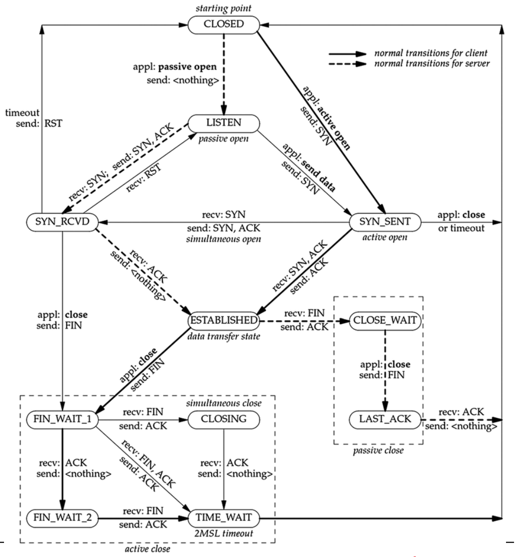
- SMOTE
- Bayesian networks

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

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

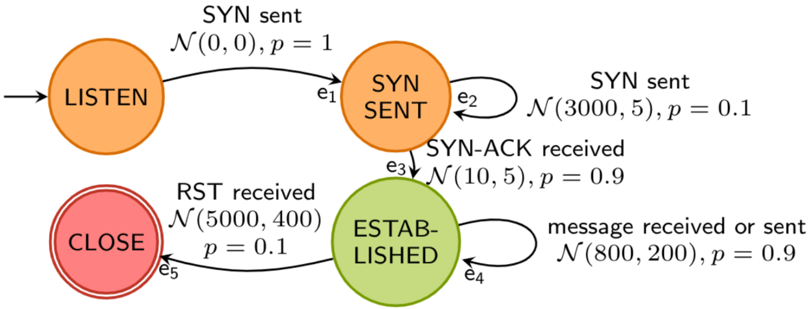
We introduce TADAM, an automata learning that is robust to noise

Modelization

Probabilistic timed automaton

Each edge has:

- a symbol
- a probability
- a inter-packet time distribution



Learning process

Learning process

- We start with an automaton initialized like a Markov chain
- We apply elementary operations:
  - split a node
  - split 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
  - split 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: $L(D, A) = L(D \mid A) + L(A)$

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

The full score

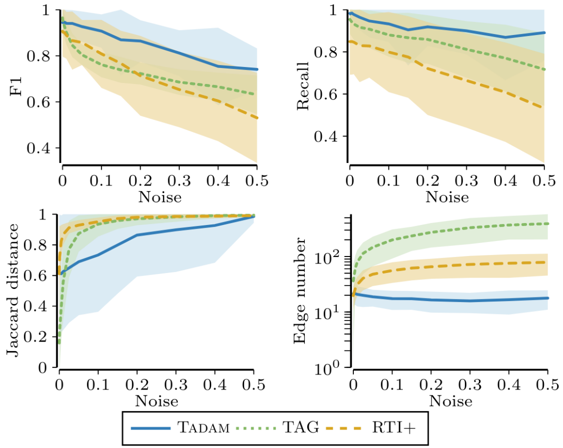
$$L(A) = L_{\mathbb{N}}(|\mathcal{Q}|) + L_{\mathbb{N}}(|\Sigma|) + \sum_{e \in \mathcal{E}} \Big(2\log_2(|\mathcal{Q}|) + \log_2(|\Sigma|) + L_{\mathbb{N}}(\lfloor \mu_e \rfloor) + L_{\mathbb{N}}(\lfloor \sigma_e^2 \rfloor) \Big) + 2\log_2(|\mathcal{Q}|)$$

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

The originality is that we explicitely 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	**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.

TADAM: beyond SDM’25

And for network?

We modelize a packet as a symbol with:

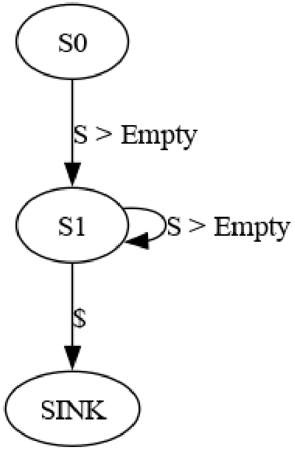
- TCP flags (for TCP protocols)
- The packet direction
- The (inferred) class of the payload:
  - Empty (not payload)
  - Random (indistinguishable from random, probably encrypted)
  - Text (clear-text), possibly with a query identifier
  - Replay (binary protocol, should be replayed)

Besides, the edges contain an inter-arrival time (IAT) and a payload size joint distribution

Datasets

The following experiments were performed on the CICIDS17 and CUPID datasets

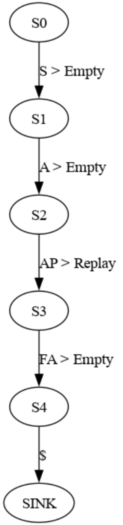
Example: HTTP (server not responding)



We observe only SYN (S) packets from the client (>)

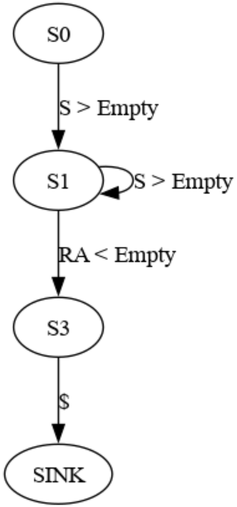
The $ is an end-of-connection marker and does not correspond to a packet

Example: Kerberos (answers not observed)



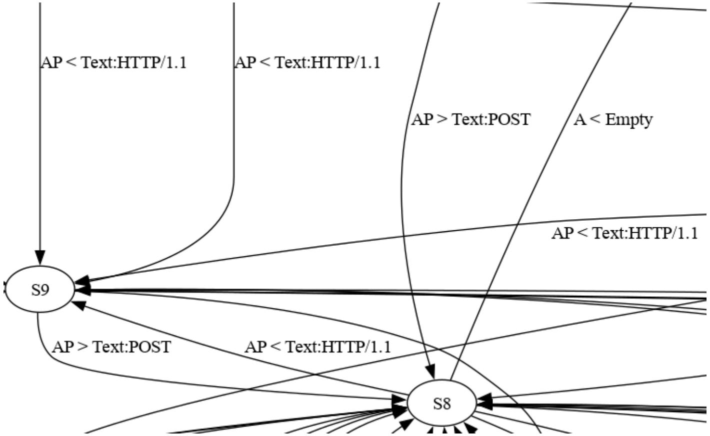
We observe only the packets emitted by the client: the packets emitted by the server have not been captured

Example: SSL (rejected connection)



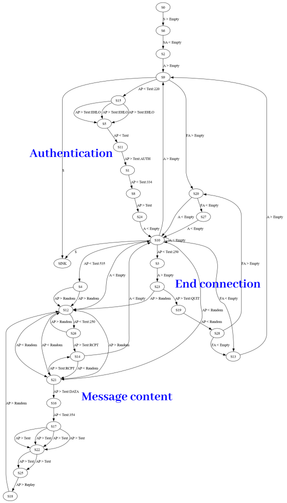
The connection is rejected by the server

Example: HTTP (extract)



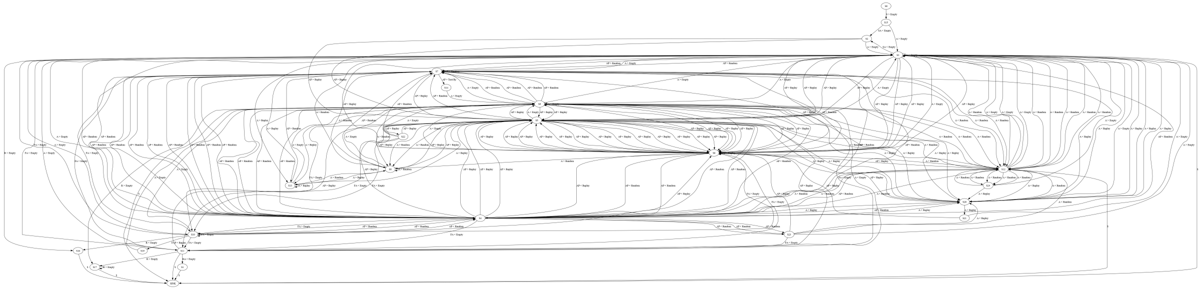
We use a simple heuristic to extract some information from the payloads in plain-text protocols

Example: SMTP



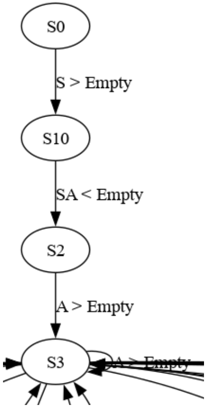
With some time, we can identify parts of the protocol.

Example: SSL

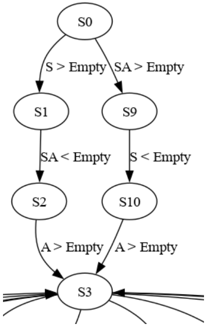


Complex protocol can have many edges (200+ here), making the manual review difficult

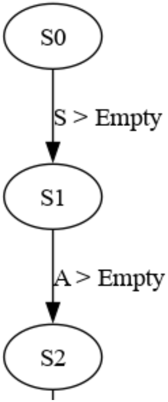
Example: TCP handshake



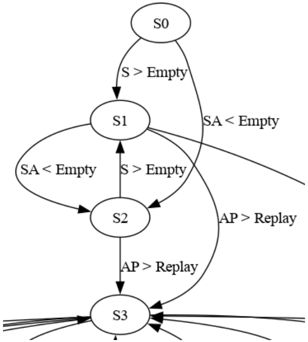
The most common case, where data is not too noisy



TADAM can identify when packet are reversed, but here the directions are modified too



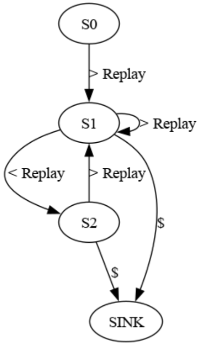
Sometimes we can miss one side of the conversation due to routing



Some protocol like GSSAPI put data in the last ACK packet

Limitations

- UDP protocol are more difficult to learn because TADAM has less information (no TCP flags)
- Plain-text protocols are easier to learn
- Some automaton are huge (200+ edges for SSL), making them difficult to review
- It is difficult to modelize the noise properly (what should be learned?)



Automaton learned for the DNS protocol

Packet generation

Generation from automata

- With a probabilistic automata, we can easily sample packet headers sequences
- Intrusion detection system typically do not inspect the payload, so its realism is not our highest priority
- 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 generation is slow

Conclusion

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

Application to network protocols

- Qualitative result are good: the learned automata are consistent with our knowledge of the protocols
- Preliminary quantitative results with Zeek's protocol analyzers are good, e.g., actual data have 1.0% ”weird” connections while synthetic data has 2.6% ”weird” connections
- Difficulty of running other generation methods, no common evaluation metrics nor usual datasets
- High generation rate: 4 GB/s on a laptop