N-BaIoT
Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders

Yair Meidan, et. Al.

Ben-Gurion University of the Negev

Singapore university of Technology and Design

IEEE Pervasive Computing 2018

Presenter: Junghwan Song

2021. 04. 15.

Outline

- Introduction
- N-BaIoT detection method
- Evaluation
- Conclusion

Introduction

- The number of Internet of Things (IoT) devices deployed dramatically increases
- The traffic volume of IoT-based DDoS attacks reaches unprecedented levels

➔ The need for timely detection of such attacks (bots) has become imperative

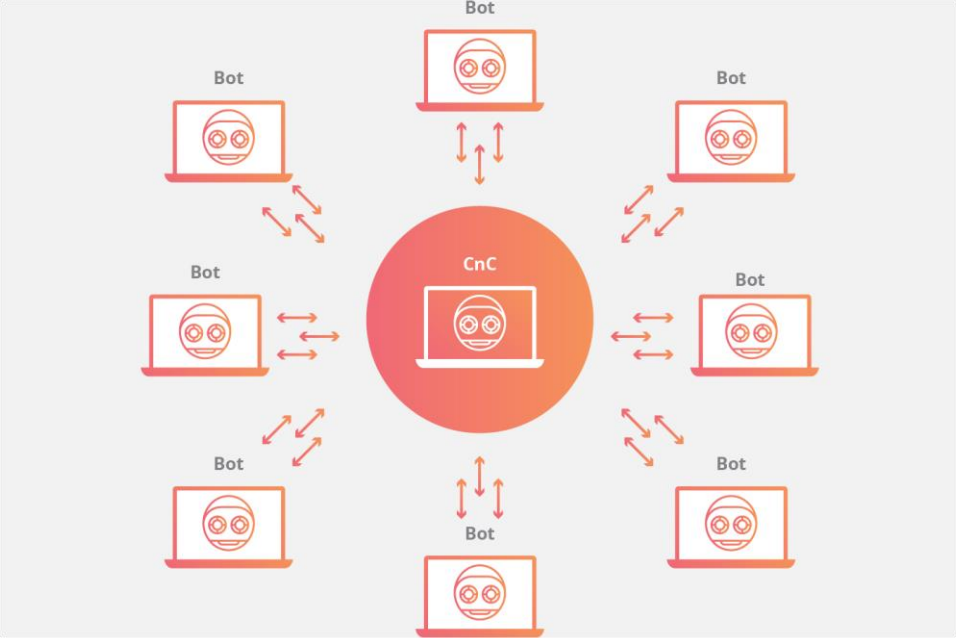
Main focus of this work

- Supposition: A large number of **heterogeneous IoT devices** connected to an organizational network

- Goal: A centralized, automated method that is highly effective and accurate in **detecting compromised IoT devices**

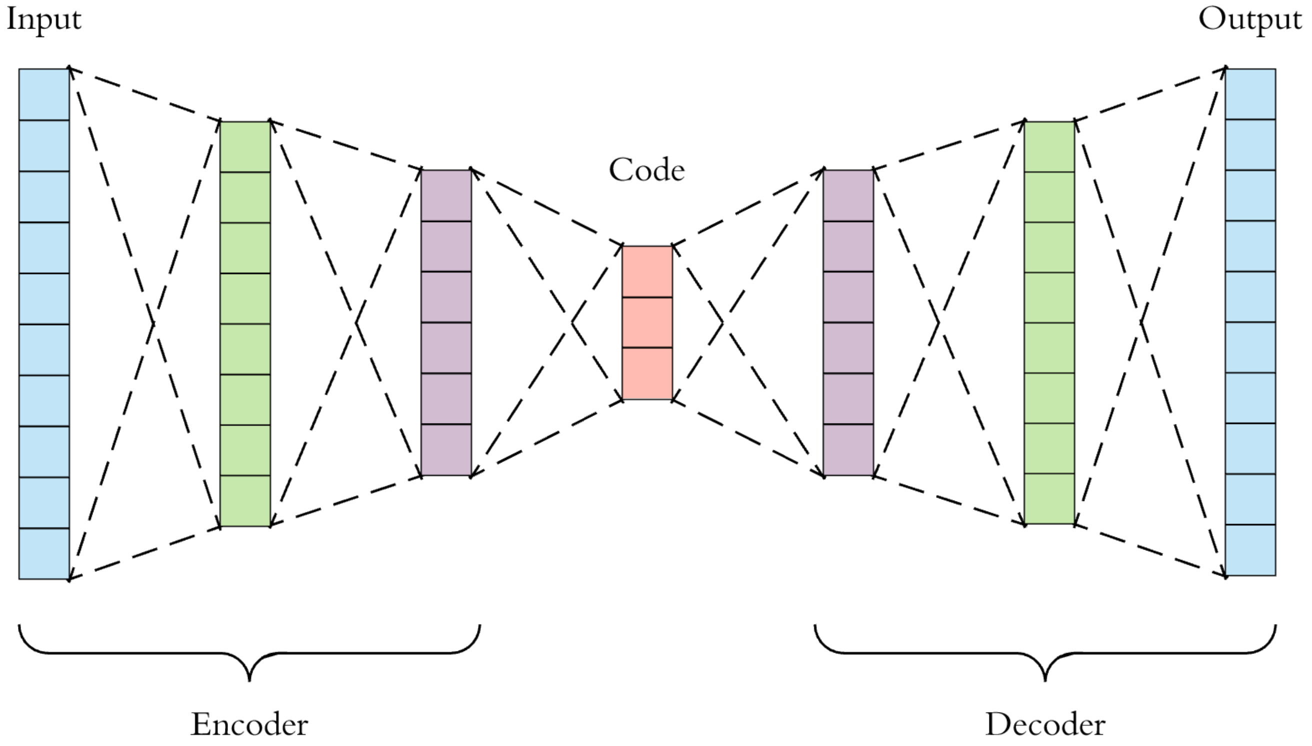
- Method: **Network-based**, last step of botnet operation, **autoencoder**

Botnet attacks



- Botnet: Collection of internet-connected devices infected by malware
- Botnet attack: Hackers control botnet to operate malicious activities such as DDoS attacks
- Botnet operational step
  - Propagation
  - Infection
  - **Command-and-Control (C&C) communication**
  - **Execution of attacks**

Autoencoder



- A neural network used to learn efficient data codings in an unsupervised manner
- An encoder compress inputs to code
- A decoder regenerate outputs using code
- Inputs and outputs have same dimension

Benefits of N-BaIoT

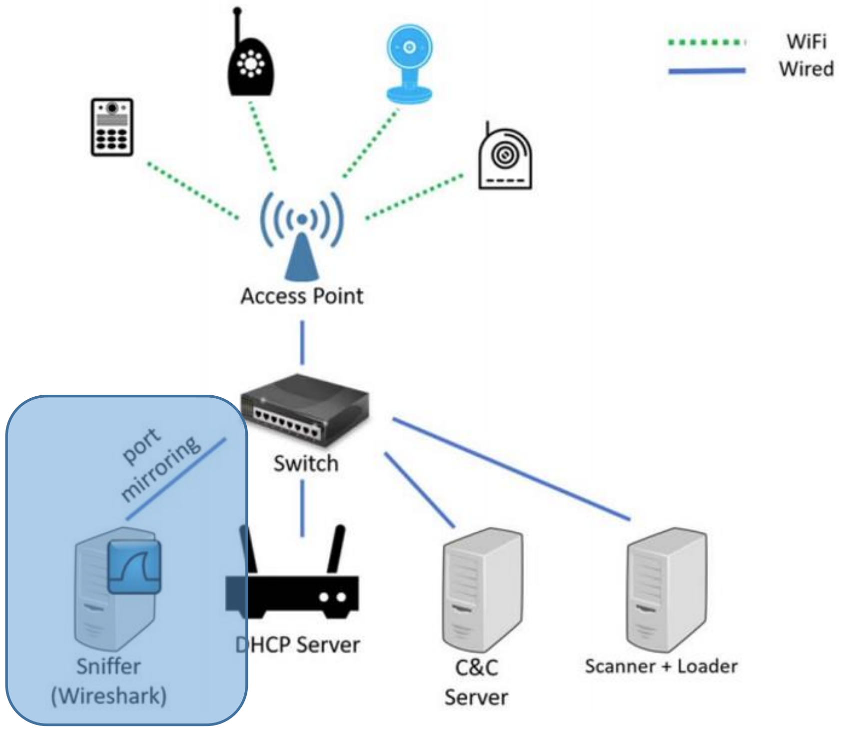
- Heterogeneity tolerance
  - Profiling each device with a separate autoencoder
- Open world
  - No need for both datasets (benign or malicious) for learning
- Efficiency
  - Semi-online training, network-based detection

N-BaIoT dectection method

Four steps of detection method

- 1. Data collection
- 2. Feature extraction
- 3. Training an anomaly detector
- 4. Continuous monitoring

1. Data collection



- Using the raw network traffic data (in pcap format)
  - By port mirroring on the switch
  - Organizational traffic typically flows

- IoT network's normal traffic is collected immediately following the device's installation in the network
  - Ensuring that the training data is clean of malicious behaviors

2. Feature extraction (1/2)

- Whenever a packet arrives, we take a behavioral snapshot of the hosts and protocols that communicated this packet

- Snapshot obtains the packet's context by extracting 115 traffic statistics
  - Aggregated by source IP, source MAC-IP, channel (src IP-dst IP), socket (src-dst sockets)
  - 5 time windows: recent 100ms, 500ms, 1.5s, 10s, 1m

2. Feature extraction (2/2)

Table 2. Extracted features.

Value	Statistic	Aggregated by	Total Num. of Features
Packet size (of outbound packets only)	Mean, Variance	Source IP,* Source MAC-IP,** Channel, Socket***	8
Packet count	Number	Source IP, Source MAC-IP, Channel, Socket	4
Packet jitter (the amount of time between packet arrivals)	Mean, Variance, Number	Channel	3
Packet size (of both inbound and outbound together)	Magnitude, Radius, Covariance, Correlation coefficient	Channel, Socket	8

* The source IP is used to track the host as a whole.

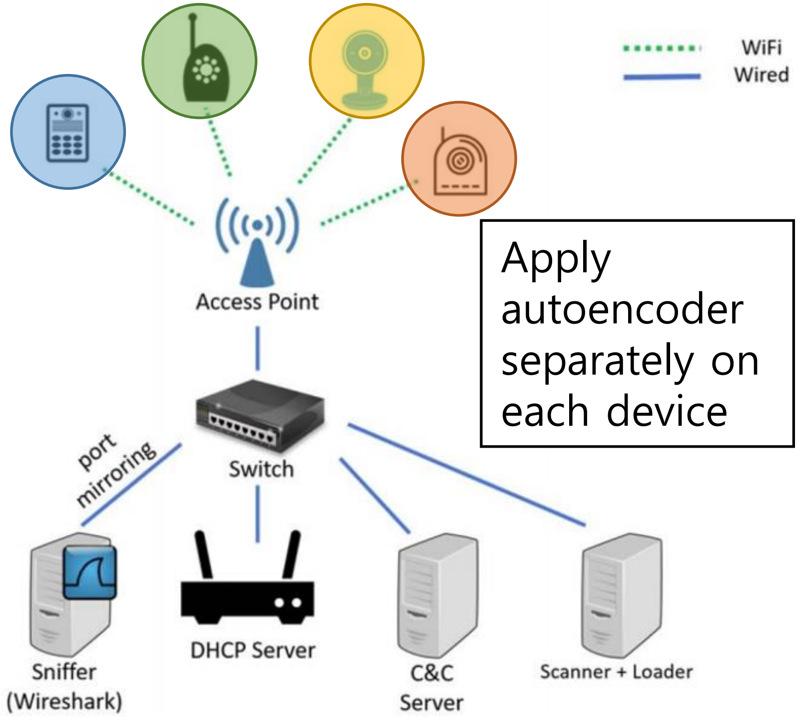
** The source MAC-IP adds the capability to distinguish between traffic originating from different gateways and spoofed IP addresses.

*** The sockets are determined by the source and destination TCP or UDP port numbers. For example, all of the traffic sent from 192.168.1.12:1234 to 192.168.1.50:80 (traffic flowing from one socket to another).

Further details and the datasets themselves are publicly available at http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

3. Training an anomaly detector (1/3)

- Using deep autoencoders and maintain a model for each IoT device separately
  - Autoencoder: a neural network trained to reconstruct its inputs after some compression
  - Compression ensures that the network learns the meaningful concepts and the relation among its input features
- An autoencoder is trained on benign instances only
  - Succeed at reconstructing normal observations
  - Fail at reconstructing abnormal observations → anomalous



Apply autoencoder separately on each device

3. Training an anomaly detector (2/3)

- Goal of optimizing parameters and hyperparameters
  - Maximizing the true positive rate (TPR, detecting attacks once they occur)
  - Minimizing the false positive rate (FPR, wrongly marking benign data as malicious)

- Two datasets are used
  - Training set (DStrn) is used for training the autoencoder, given
    - Learning rate (η, the size of the gradient descent step)
    - Number of epochs (complete passes through the entire DStrn)
  - Optimization set (DSopt) is used to optimize η, epochs, and tr
    - Threshold (tr, discriminates between benign and malicious observations)

3. Training an anomaly detector (3/3)

- After model training and optimization, anomaly threshold (tr*) is set

$$tr^* = \overline{MSE}_{DS_{opt}} + s(MSE_{DS_{opt}})$$

The sum of the sample mean and standard deviation of mean square error over DSopt

4. Continuous monitoring

- Applying optimized model to feature vectors extracted from continuously observed packets

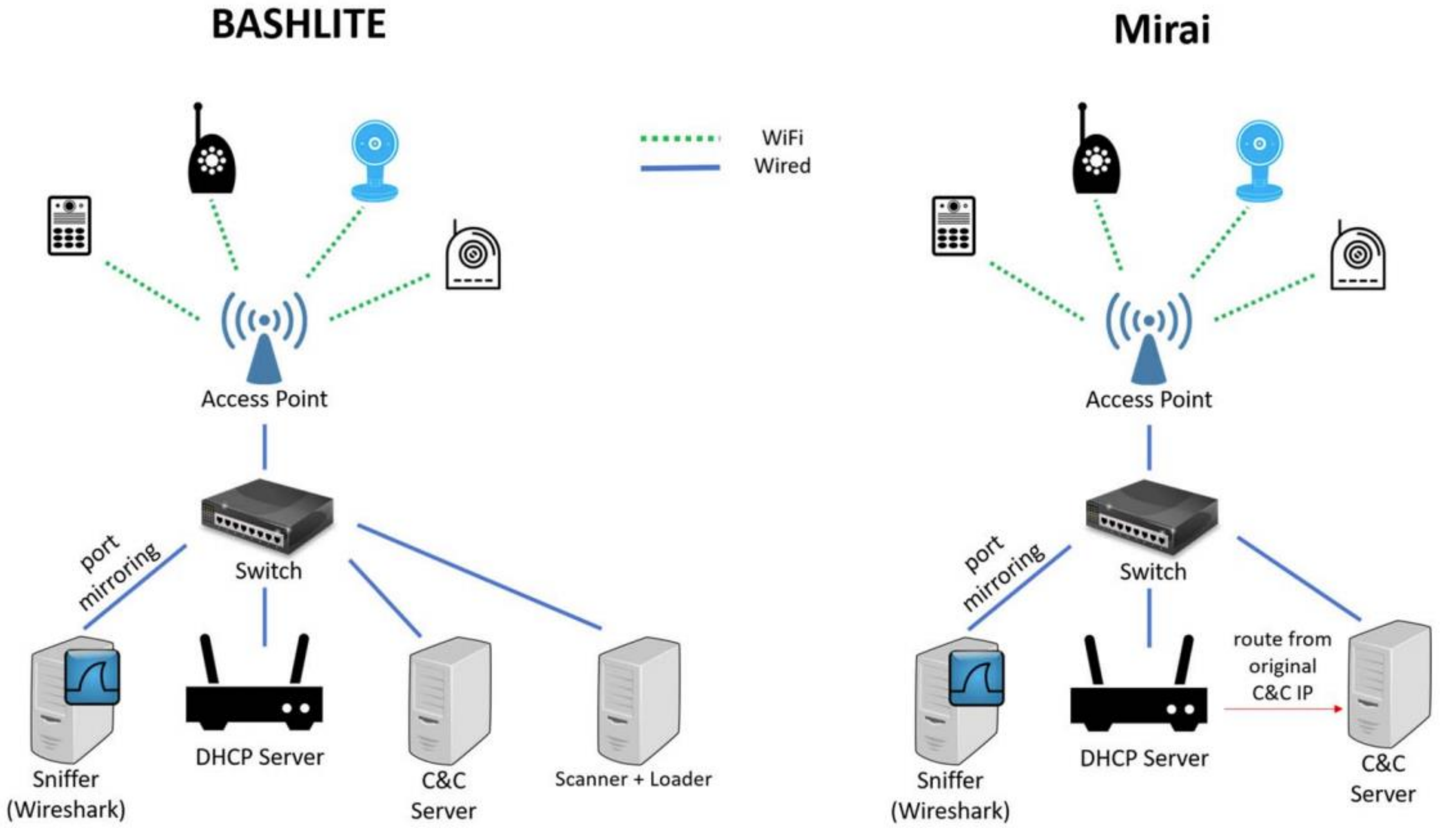
→ Deciding whether each instance as benign or anomalous

- Majority vote on a sequence (the length of ws) of marked instances

→ Deciding whether entire stream is benign or anomalous

Evaluation

Testbed setup



Attacks executed

BASHLITE Attacks

1. Scan: Scanning the network for vulnerable devices
2. Junk: Sending spam data
3. UDP: UDP flooding
4. TCP: TCP flooding
5. COMBO: Sending spam data and opening a connection to a specified IP address and port

Mirai Attacks

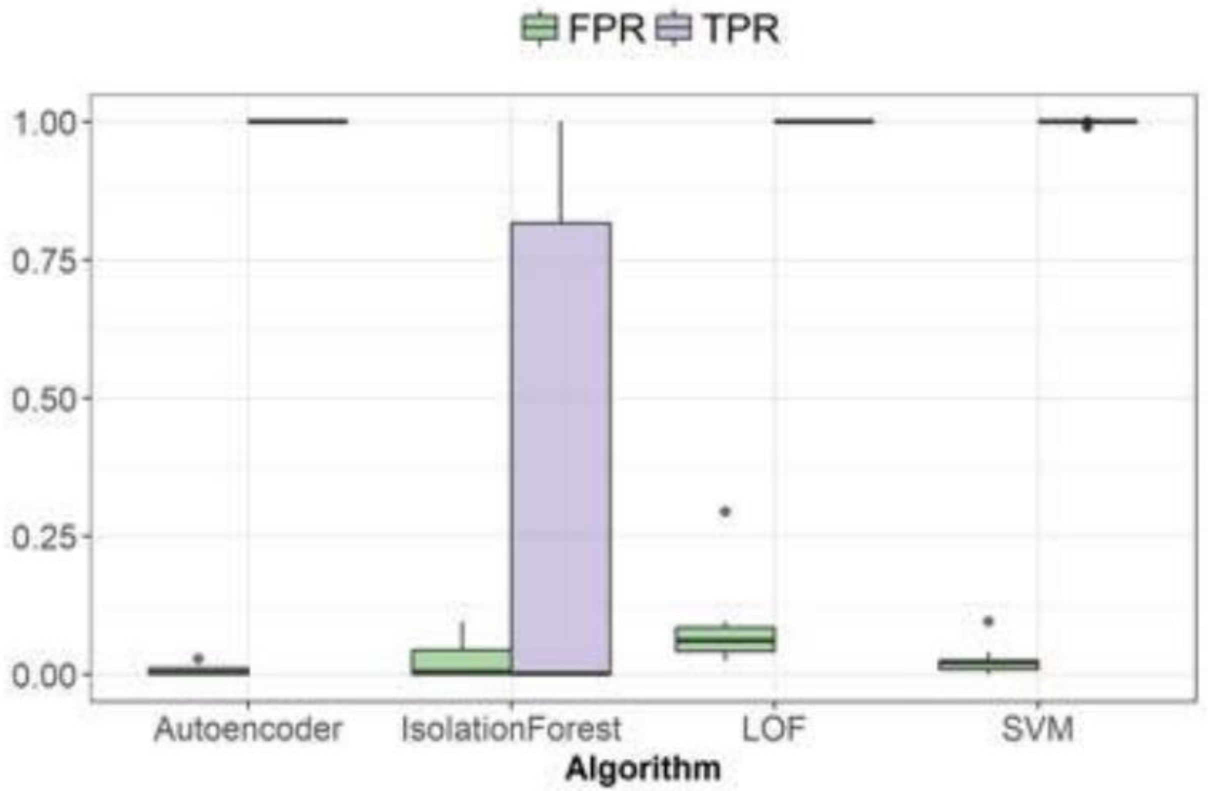
1. Scan: Automatic scanning for vulnerable devices
2. Ack: Ack flooding
3. Syn: Syn flooding
4. UDP: UDP flooding
5. UDPplain: UDP flooding with fewer options, optimized for higher packets per second

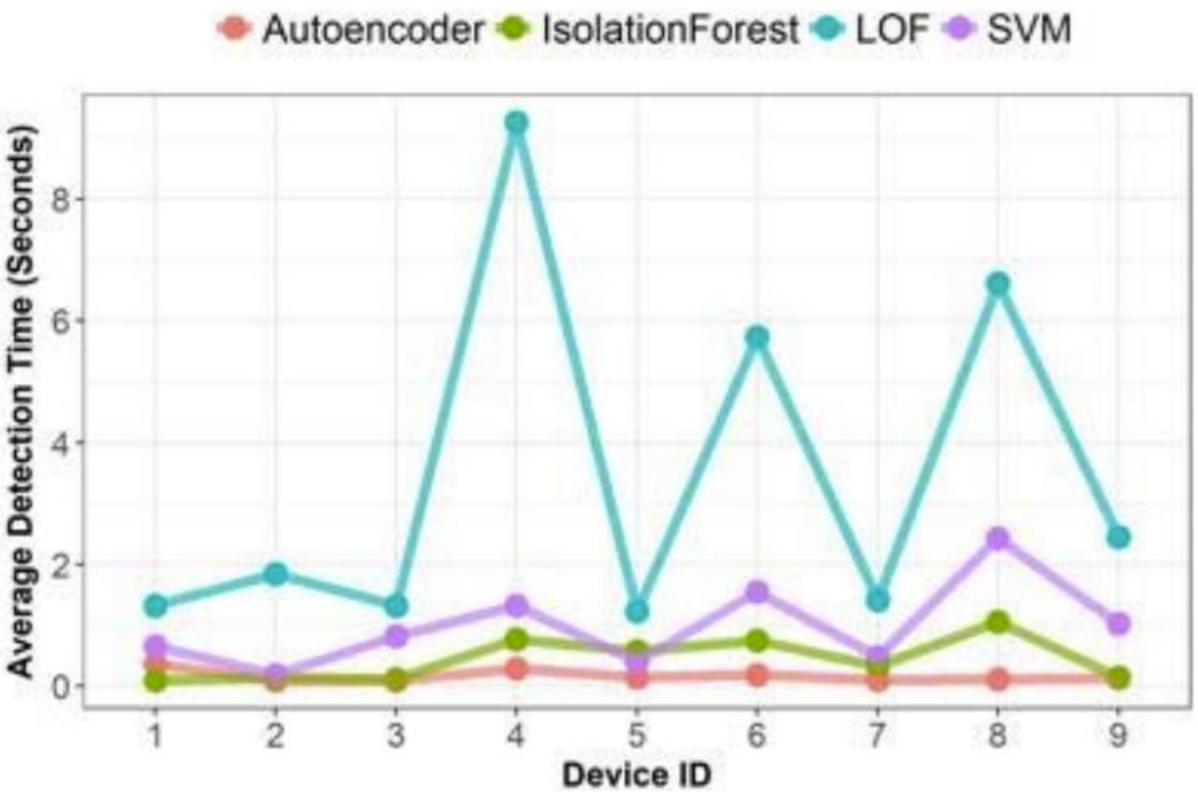
Training overview

Table 3. Overview of the training stage.

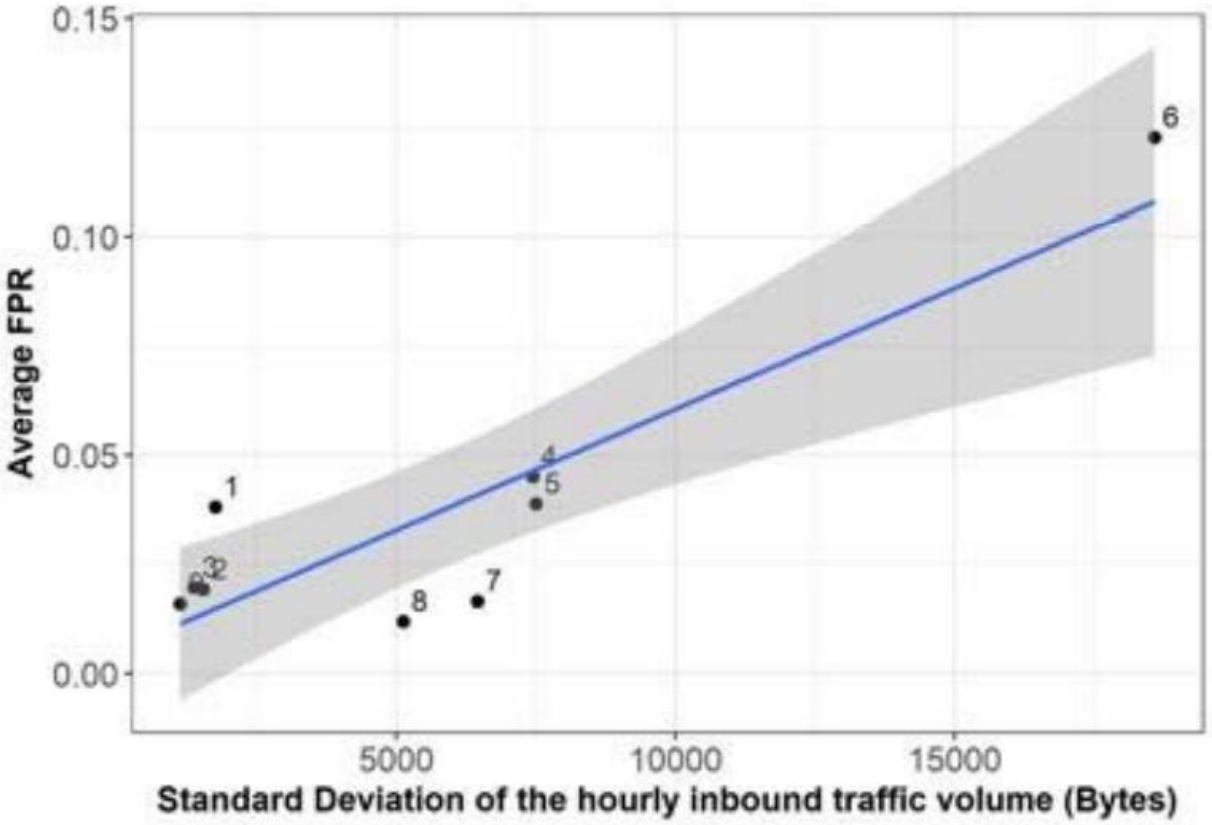
Device ID	Dataset properties and training summary					Optimized hyperparameters of autoencoders				Botnet infections	
	Device make and model	**Device type**	**NUmber of benign instances**	**Training time (*seconds*)**	**Object size (*kB*)**	**Learning rate (η)**	**Number of epochs (*epochs*)**	**Anomaly threshold (tr^*)**	**Window size (ws^*)**	**Mirai**	**BASHLITE**
1	Danmini	Doorbell	49,548	555	172	0.012	800	0.042	82	✓	✓
2	Ennio	Doorbell	39,100	215	172	0.003	350	0.011	22	-	✓
3	Ecobee	Thermostat	13,113	54	172	0.028	250	0.011	20	✓	✓
4	Philips B120N/10	Baby monitor	175,240	292	172	0.016	100	0.030	65	✓	✓
5	Provision PT-737E	Security camera	62,154	275	172	0.026	300	0.035	32	✓	✓
6	Provision PT-838	Security camera	98,514	795	172	0.008	450	0.038	43	✓	✓
7	SimpleHome XCS7-1002-WHT	Security camera	46,585	220	172	0.017	230	0.056	23	✓	✓
8	SimpleHome XCS7-1003-WHT	Security camera	19,528	190	172	0.006	500	0.004	25	✓	✓
9	Samsung SNH 1011 N	Webcam	52,150	150	172	0.013	150	0.074	32	-	✓

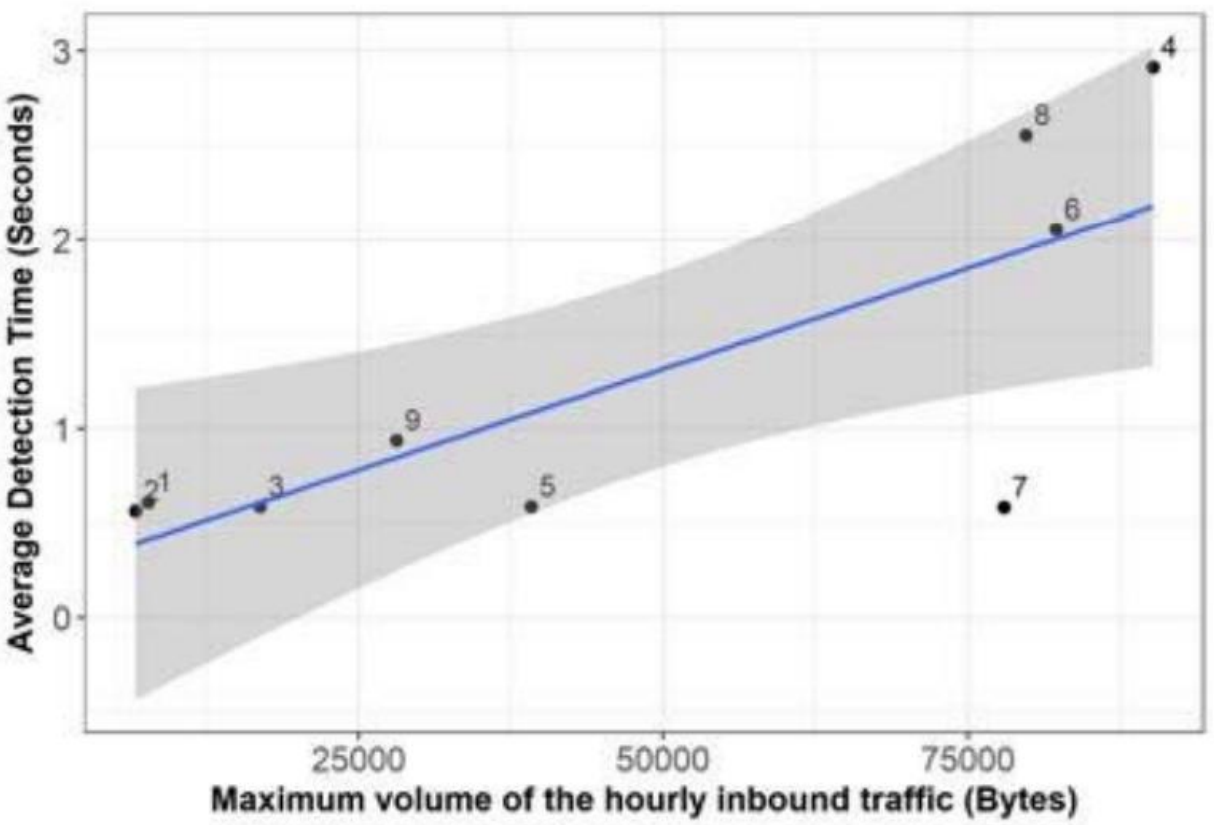
Evaluation results (1/2)





Evaluation results (2/2)





Conclusion

- The traffic volume of IoT-based DDoS attacks reaches unprecedented levels

- The paper suggests N-BaIot, a network-based automated bot detection method using autoencoders

- Evaluation results show N-BaIot can be effective solution in heterogeneous IoT networks