ADROIT: Detecting Spatio-Temporal Correlated Attack-Stages in IoT Networks

NUS-Singtel Cyber Security R&D Corp. Lab

Dinil Mon Divakaran, Rhishi Pratap Singh, Kalupahana Liyanage Kushan Sudheera, Mohan Gurusamy, Vinay Sachidananda

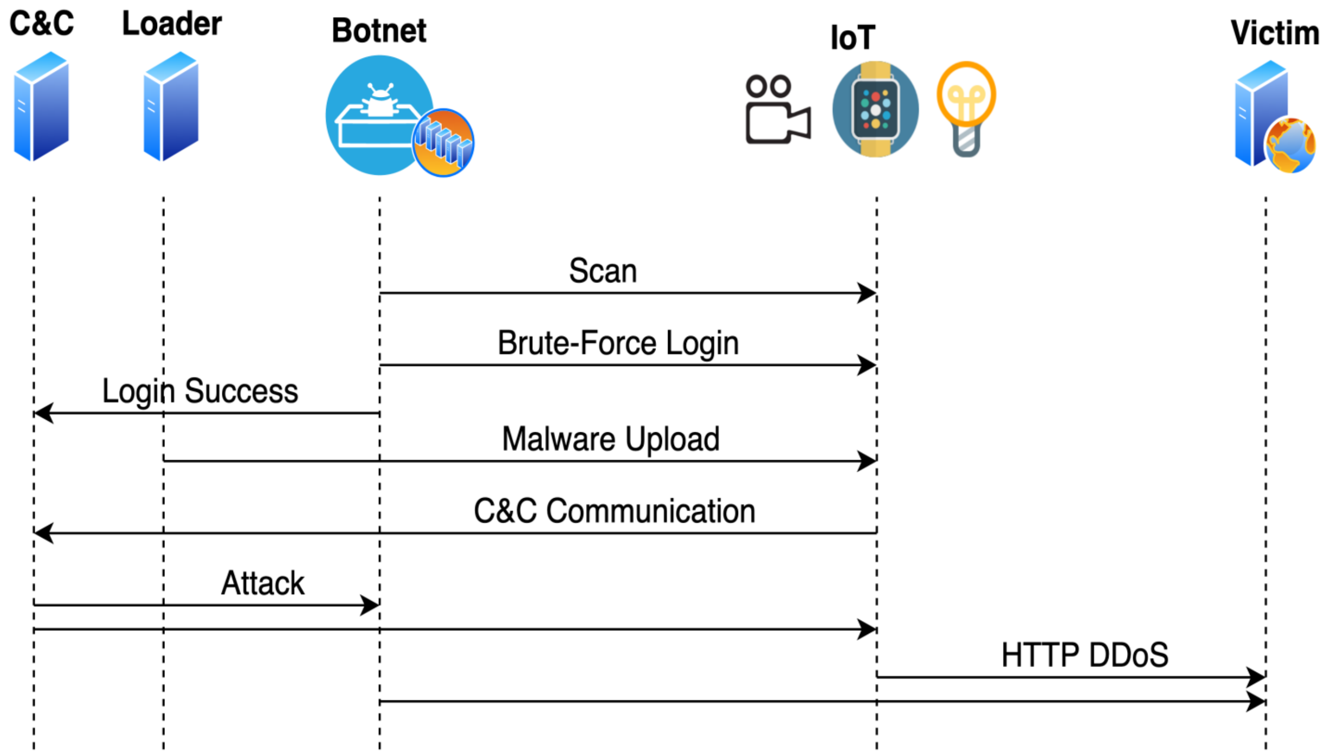
Trustwave®
a Singtel company

Context

- IoT increasing in numbers, types, applications and deployments
- Mostly unattended by humans
- Vulnerable and easily exploited
- Question: at a network level (e.g., ISPs), how can we detect and prevent attacks on and due to the *things*?

Problem

- Can we detect stages of a coordinated large-scale cyber attack?
- For example
  - Scan
  - Brute-force login attempts
  - Malware downloads
  - C&C communications
  - Launch of specific and targeted attack (DDoS, RDDoS)



Challenges - I

I. Activities might be spread across different network premises

- Analyzing just one network might not show any significant activity
- E.g., a low-rate DDoS or brute-force login attempts at different n/ws might be related

Spatial dispersion

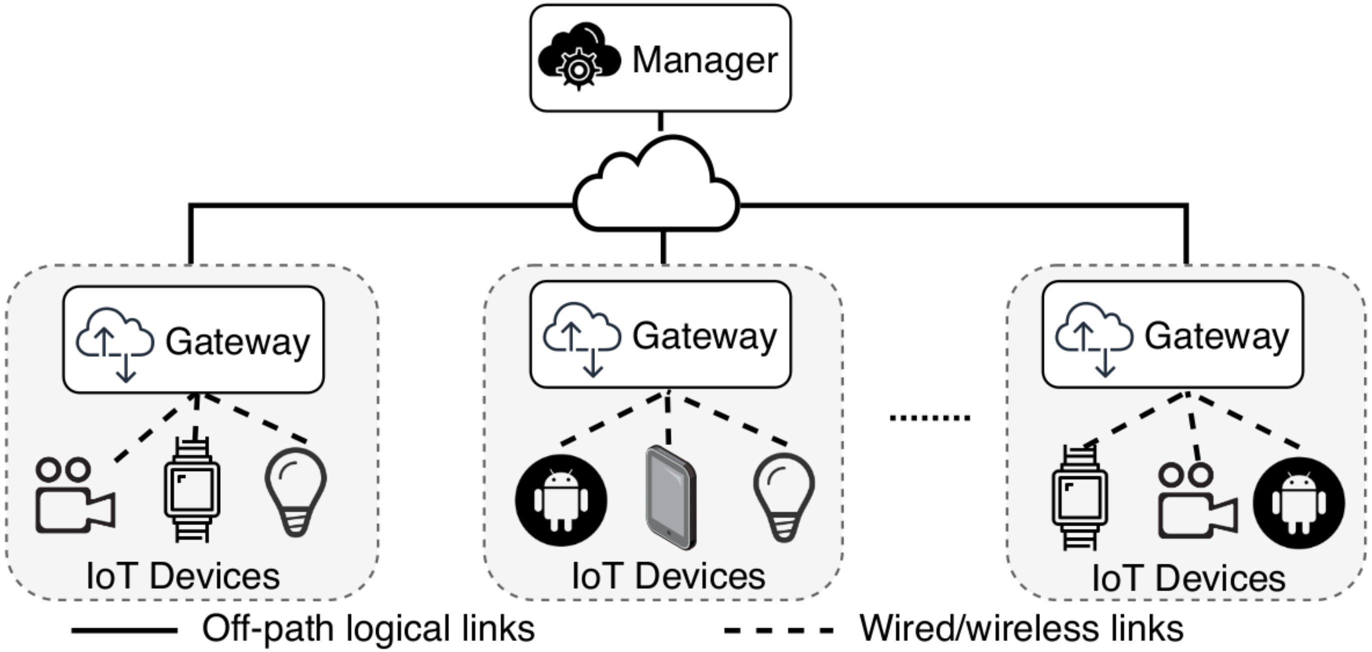
Challenges - II

II. One or multiple stages of an attack might happen at different times

- Bot may be infected for a long time, during which it may engage in malicious activities
- C&C communication establishment often involves multiple connection attempts

Temporal dispersion

ADROIT: network architecture



- Each premise (smart home/building) has a gateway, connected to devices in it’s network
- All gateways connected to a manager in the Cloud or ISP datacenter

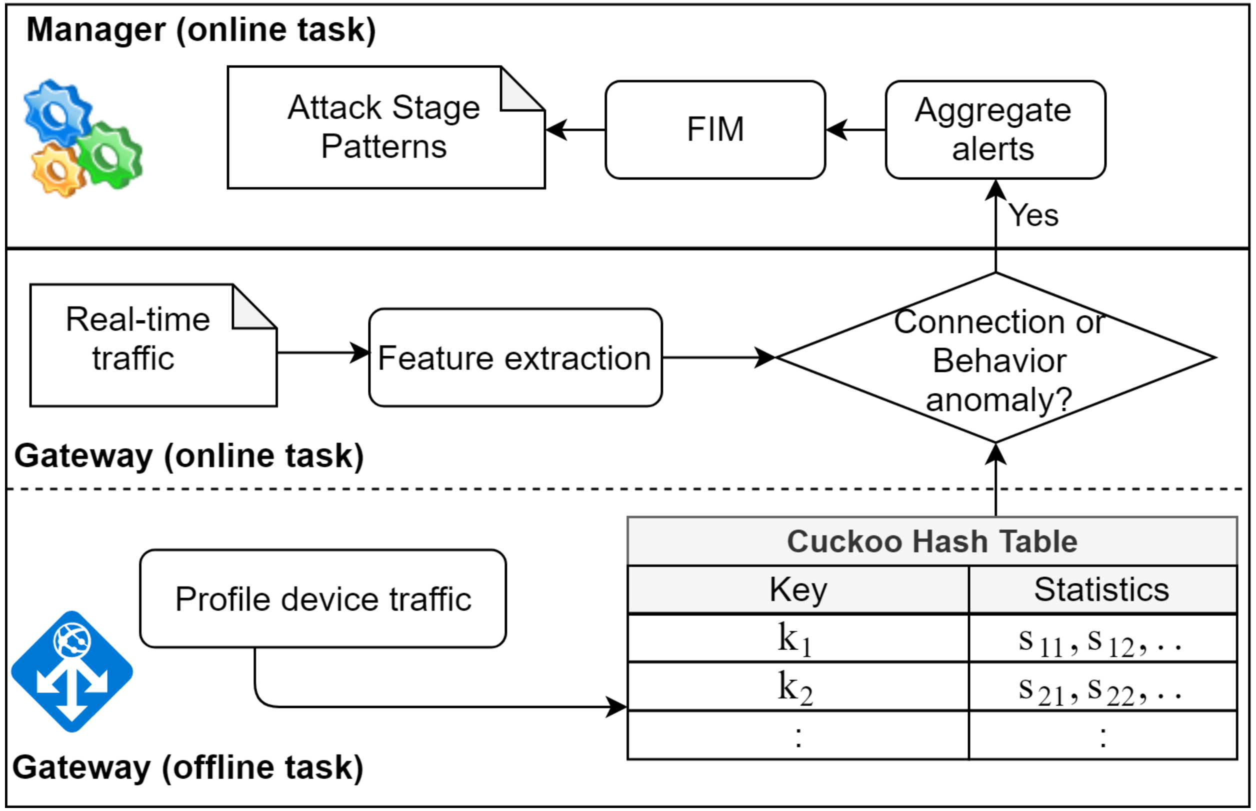
ADROIT

Properties

- ✓ Traffic processed locally, at the gateways
- ✓ Only alerts anomalies sent to Manager
  - Privacy of normal application not compromised
  - Minimal leak of info → even for anomalous traffic, only meta info shared with Manager
  - Bandwidth consumed is reduced by orders of magnitude
- ✓ Unsupervised approach in detecting attack-patterns
  - No reliance on labeled data for training models
  - Potentially detect new attacks

Overview of ADROIT

1. [Device profiling] Done for the connected devices at the gateway in an offline manner
2. [Anomaly detection] At deployment, the anomalies are detected when the packet features are extracted & compared with IoT profiles
3. [Pattern mining] These alerts are sent to the manager for detecting attack-stages



Device profiling

- IoT devices connect to limited number of destinations
  - Exceptions include hubs and changes in servers or server to IP address mapping
- A baseline profile (hash table) can be built from packets and connections
- Each gateway can profile their devices independently, and in an offline manner
  - Some compute and storage resources required
- Once profile table built → (local) anomaly detection requires only lookups based on the keys

				Count		Size	
External IP	**Port**	**Proto**	**Dir**	**mean**	**std**	**mean**	**std**
dns.google.	53	UDP	Out	2	0	219.8	4.3
api.dch.dlink.com.	80	TCP	Out	10	0	1227	0
api.dch.dlink.com.	443	TCP	Out	22.6	2.26	5792.4	955.4
ntp1.dlink.com.	123	UDP	Out	2	0	152	0
r0802.dch.dlink.com.	443	TCP	Out	124.4	9.39	5212.9	974.8
tzinfo.dch.dlink.com.	80	TCP	Out	10	0	824	0
wrpd.dlink.com.	80	TCP	Out	10	0	1202	0

Example profile: D-Link socket

Key = hash(Internal IP | External IP | App Port | Direction)

Cuckoo hash table

Device profiling

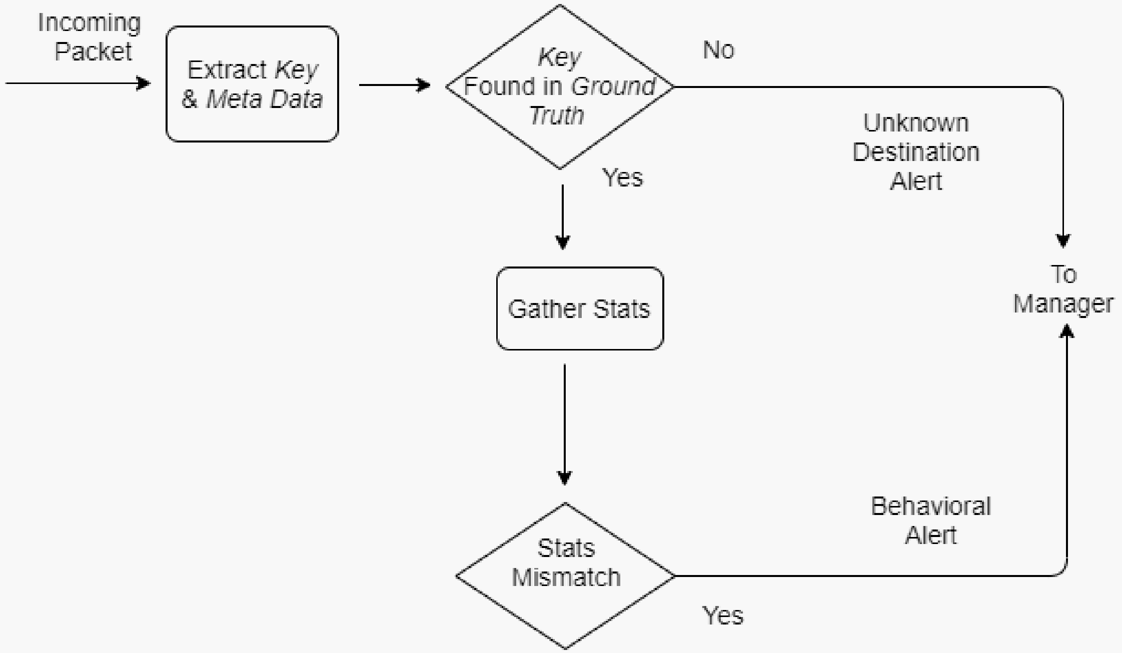
- Hash table operations of interest: insert(), update(), lookup()
- Insert() *and* update() required only during profile creation
- Real-time detection requires only lookup()
- Traditional hash table can incur linear lookup times in worst cases
- Alternative → Cuckoo hash table
  - ✓ lookup() has constant worst-case time; to be precise, just two, for two hash functions
  - ✓ Trade-off → insert()
  - ✓ But insert() is performed offline, where lookup() is required to performed online

Anomaly detection at a gateway

- ❖ Real-time operation: extract key from incoming packet

Two anomalies of interest:

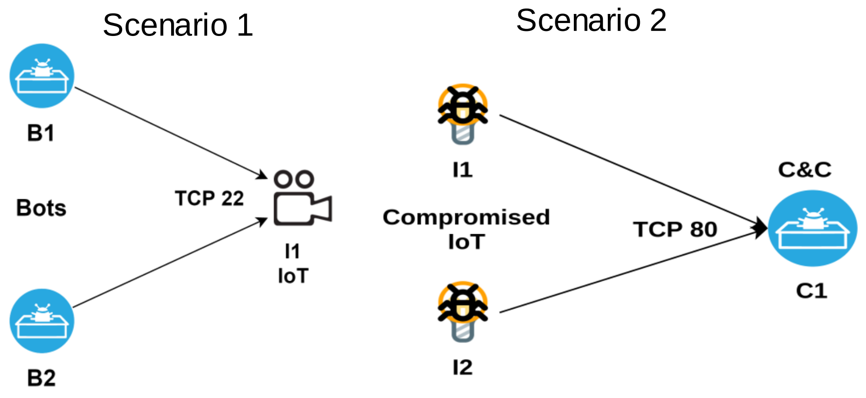
- ❖ Connection anomaly: If key not found in profile table
- ❖ Behavior anomaly: If is found in profile table, but if stats do not match
- ❖ In both cases, alert generated and sent to Manager
- ❖ Observe: only alerts, i.e., meta-information and of anomalies sent to Manager



- Key = (Internal IP, External IP, Port, Protocol, Direction)
- Meta data = (Packet & Payload Length, Number of sessions)

Alert analysis at the manager

- Manager analyzes the alerts
  - Attack-stages such as Scan, Login, C&C, RDDoS, DDoS could form dominant patterns
  - All alerts are not related to attack-stages
  - Noises are random and spurious. Even if the noises form patterns, would they be dominant in volume?
- How to capture patterns?



IoT	External IP	Protocol	Port	Direction
I1	B1	TCP	22	In
I1	B2	TCP	22	In
I1	C1	TCP	80	Out
I2	C1	TCP	80	Out

Alerts sent by Gateways

IoT	External IP	Protocol	Port	Possible Cause
I1	*	TCP	22	Scan-Attempt
*	C1	TCP	80	C&C

Manager Output

Pattern detection

At manager

- **Frequent Itemset Mining (FIM)**
  - Data mining approach to extract recurring patterns
  - Each field of an alert corresponds to an item, in FIM
  - A k-itemset is a set of k items
  - Given n alerts, an itemset/pattern is called frequent, if it appears in at least θ x n alerts, where θ is called minimum support
  - Goal: mine frequent itemsets in alert database
  - Parameters: itemset length (k), minimum support θ

Example

- Upper table: consider alerts arriving at Manager
- Some related to attacks, and,
- Some false positives
  - Can arise due to random scans, firmware updates, etc.
- Lower table: patterns extracted, using a small set of features

Incoming Alerts

#	srcIP	dstIP	Protocol	srcPort	dstPort	Dir	sizeBin
1	scanner1.com	10.6.1.12	TCP	45678	23	In	Small
2	scanner2.com	10.6.1.12	TCP	56897	23	In	Small
3	scanner3.com	10.6.2.2	TCP	55001	23	In	Medium
4	scanner3.com	10.6.5.173	TCP	45877	23	In	Medium
5	10.6.2.2	cnc.com	TCP	23669	48000	Out	Medium
6	10.6.5.173	cnc.com	TCP	56814	48000	Out	Medium
:	:	:	:	:	:	:	:
31	10.6.2.2	victim1.com	TCP	23456	80	Out	Medium
32	10.6.5.173	victim1.com	TCP	35689	80	Out	Medium
33	victim2.com	dns.server	UDP	13074	53	Out	Small
34	victim2.com	dns.server	UDP	18869	53	Out	Small
	:	;	:	:	:	:	:
101	10.6.2.13	firmware1.com	TCP	49225	80	Out	Large
102	10.6.13.144	random1.com	TCP	48369	443	Out	Medium
103	firmware2.com	10.6.19.66	UDP	23698	69	In	Large
:	:	:	:	:	:	:	:

Alerts related to attack stages (rows 1–34)

False alerts, not related to attack stages (rows 101 onward)

FIM

Extracted Itemsets

#	srcIP	dstIP	Protocol	srcPort	dstPort	Dir	sizeBin
1	*	10.6.1.12	TCP	*	23	In	Small
2	scanner3.com	*	TCP	*	23	In	Medium
3	*	cnc.com	TCP	*	48000	Out	Medium
4	*	victim1.com	TCP	*	80	Out	Medium
5	victim2.com	dns.server	UDP	*	53	Out	Small
:	:	:	:	:	:	:	:

FIM

Algorithms

- Algorithms like Apriori: mine frequent itemsets of all lengths
- Extracting all patterns exhaustively is neither useful nor efficient
  - Many patterns are closely related
  - Lower length itemsets are subsets of higher length itemsets
  - E.g., <<*,*,TCP,*,23,In,*>> and <<*,10.6.1.12,TCP,*,23,In,Small>>
- Alternative 1: Closed Frequent Itemset (CFI) mining
  - Itemsets do not have any superset with the same support
- Alternative 2: Maximal Frequent Itemset (CFI) mining
  - Itemsets do not have any superset which is frequent
- We use MFI
  - More information, and generally of higher length,
  - Number of patterns and complexity are lowest

Atttack-pattern mining algorithm with look-back

At Manager

- Correlation within one single window and across multiple windows
- Basically, to dynamically change minimum support
- Minimum support plays a critical role in extracting out attack patterns and leaving out false patterns
- Once a pattern is found, only mine on the alerts related to that pattern
- Not only in the current window, but also in a set of previous windows (looking back)

Algorithm 1 Pattern mining at time-slot τ with look-back

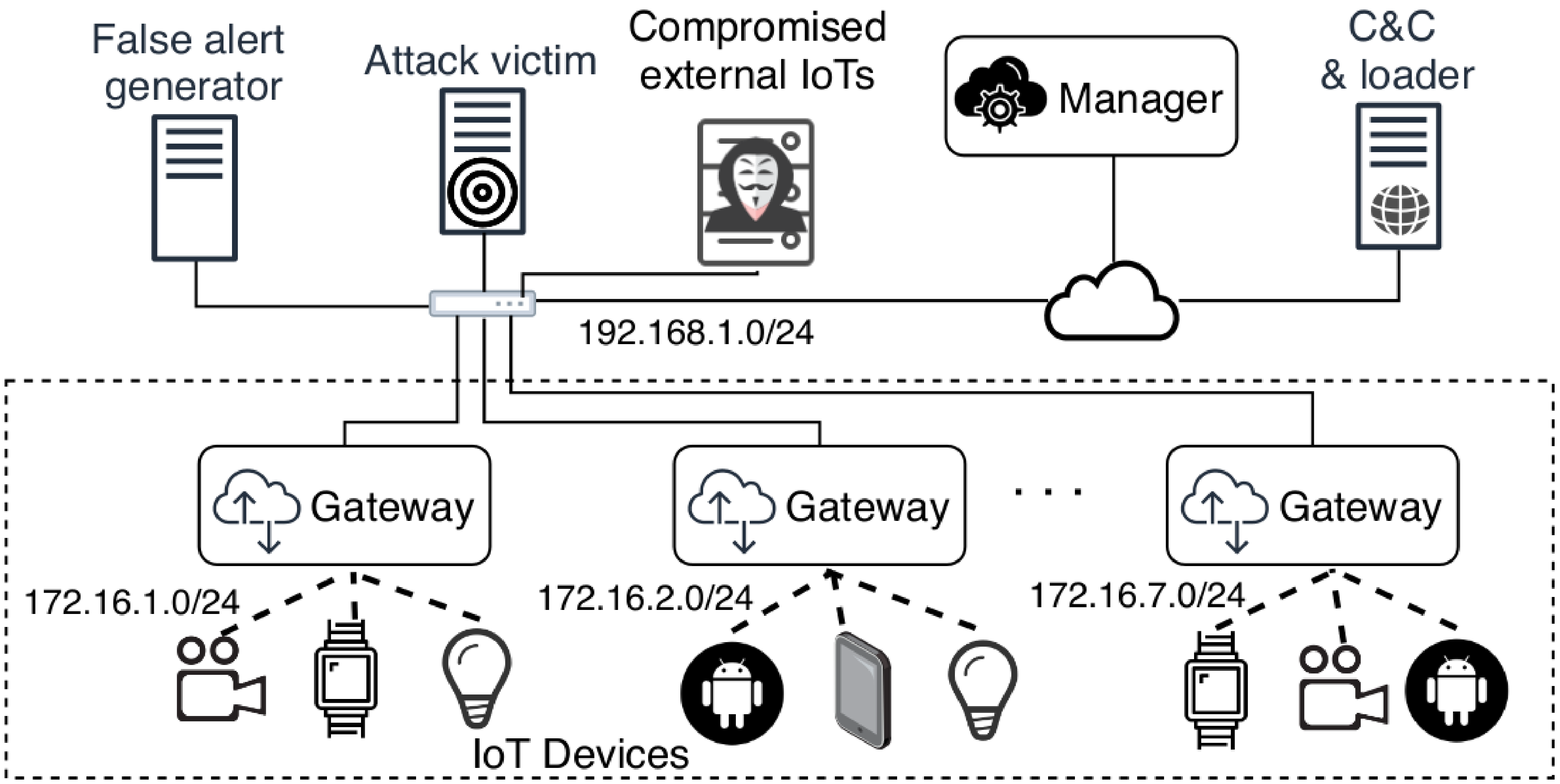
Input: $\mathcal{F}$: mined patterns (an array), $\mathcal{A}$: alerts, θ_l: lower bound of minimum support, Δ^-, Δ^+: decrement and increment step sizes of minimum support, T_w: look-back time-slots

$\mathcal{F}[\tau] \leftarrow$ `MFI_Iter`$(\texttt{any\_pattern}, \mathcal{F}, \mathcal{A}[\tau],\ \theta,\ \theta_l)$ $\triangleright$ mine for any maximal frequent itemset in alert database at time τ while reducing θ iteratively until θ_l
for each $t \in \{\tau, \ldots, \tau - T_w\}$ **do**
    for each $\mathbf{I} \in \mathcal{F}[\tau]$ **do**
        $\theta' \leftarrow (\theta - \Delta^-)$
        $\mathcal{A}' \leftarrow$ `filterAlerts`$(\mathbf{I}, \mathcal{A}[t])$; $\triangleright$ filter the alert database by pattern $\mathbf{I}$
        $\mathcal{F}' \leftarrow$ `MFI_Iter`$(\texttt{new\_pattern}, \mathcal{F}, \mathcal{A}', \theta', \theta_l)$ $\triangleright$ mine for any new pattern in filtered alert database $\mathcal{A}'$ while reducing θ' iteratively until θ_l
        $\mathcal{F}[t] \leftarrow \mathcal{F}[t] \cup \mathcal{F}'$ $\triangleright$ add new patterns
    end for
end for
$\theta \leftarrow (\theta + \Delta^+)$ $\triangleright$ increase for next time-slot

Performance evaluation

(preliminary)

Experiment setup



- OpenStack environment to emulate Mirai-like botnet
  → scans, brute force login attempts, m/w download, C&C comm., and specific DDoS attacks
- New IoT devices get infected during the experiment duration
- 7 gateways, 65 (emulated) IoT devices, 2 compromised devices, a victim, a C&C server and a loader
- VMs for generating false alerts (noises representing deviations from normal but not attacks)

Metrics for evaluation

$$\text{precision} = \frac{\#\text{True Positive}}{\#(\text{True Positive} + \text{False Positive})}$$

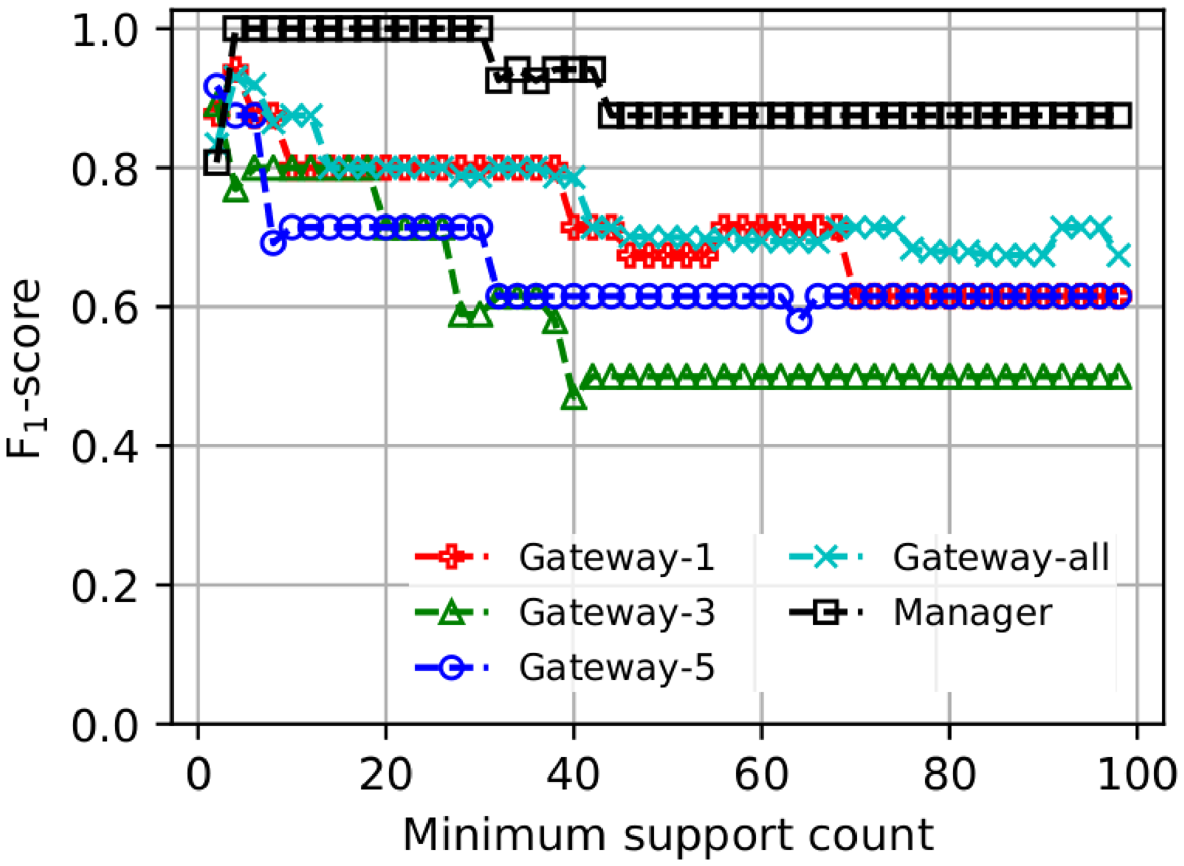
$$\text{recall} = \frac{\#\text{True Positive}}{\#(\text{True Positive} + \text{False Negative})}$$

$$F_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Experiment 1

Local v/s Global detection capabilities

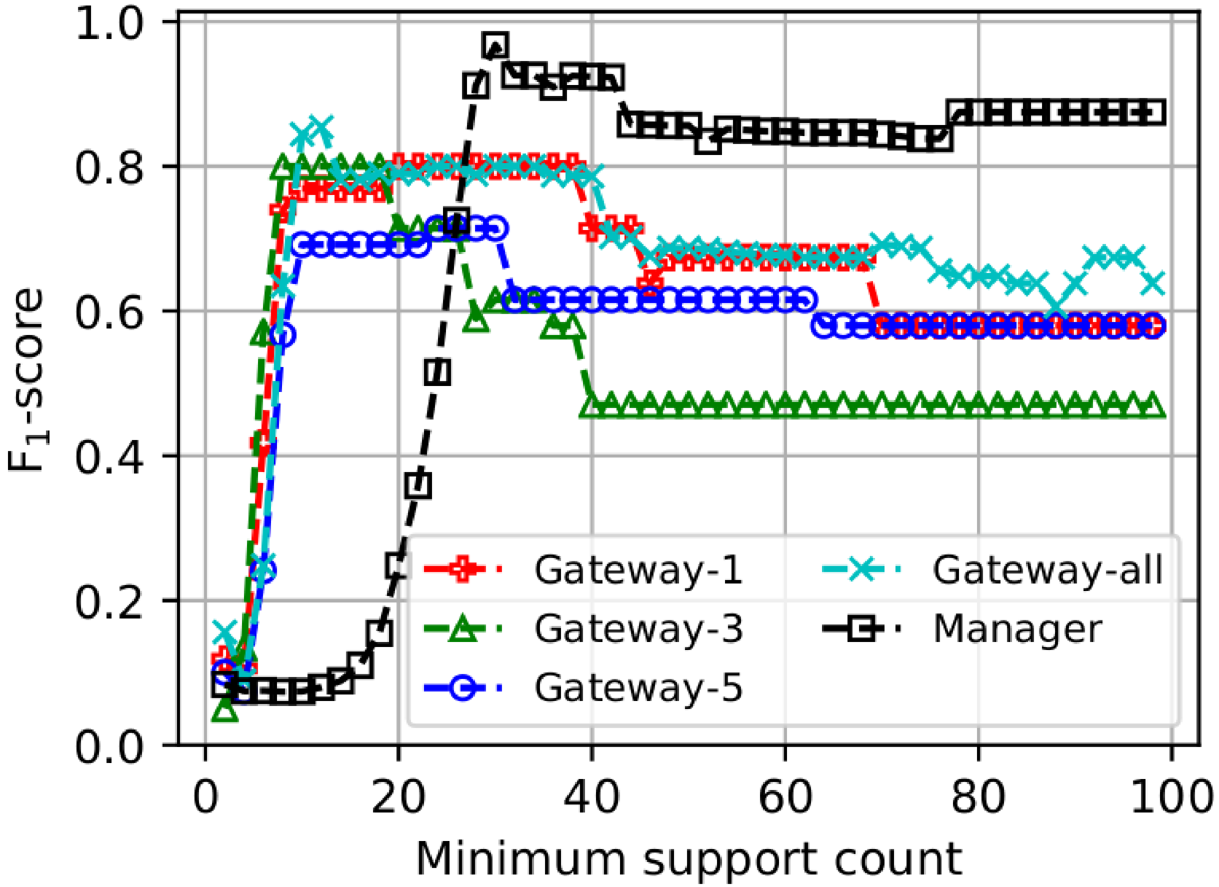
Goal: evaluate impact of spatial correlation at Manager, at different levels of false alerts



No false alerts

Experiment 1 (cont'd)

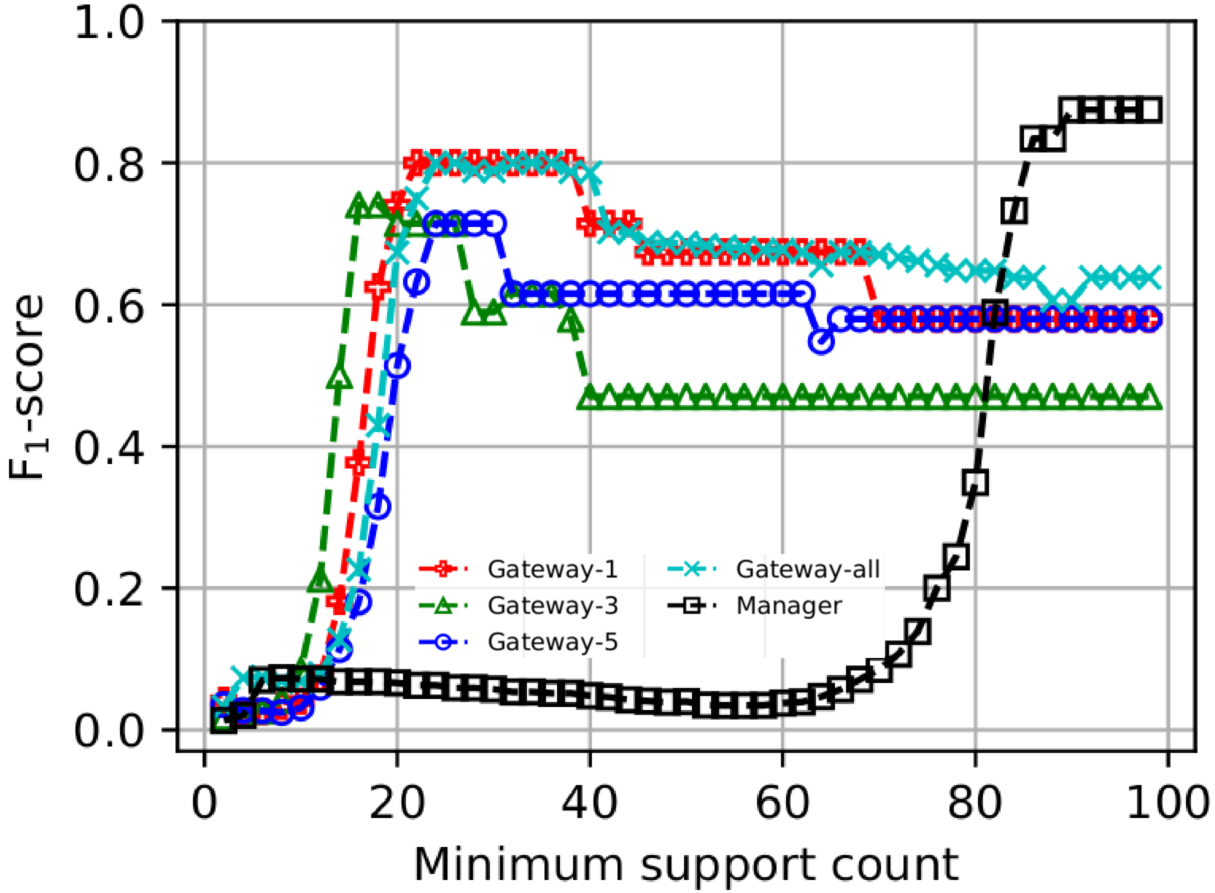
Local v/s Global detection capabilities



False alert level 1

Experiment 1 (cont'd)

Local v/s Global detection capabilities



False alert level 2

Takeaway from Experiment 1

- FIM helps in mining attack patterns
  - Both at gateways and at Manager
- Generally, Manager has higher detection capability with low false positives
- But depends on minimum support
  - Static minimum support is not a good idea

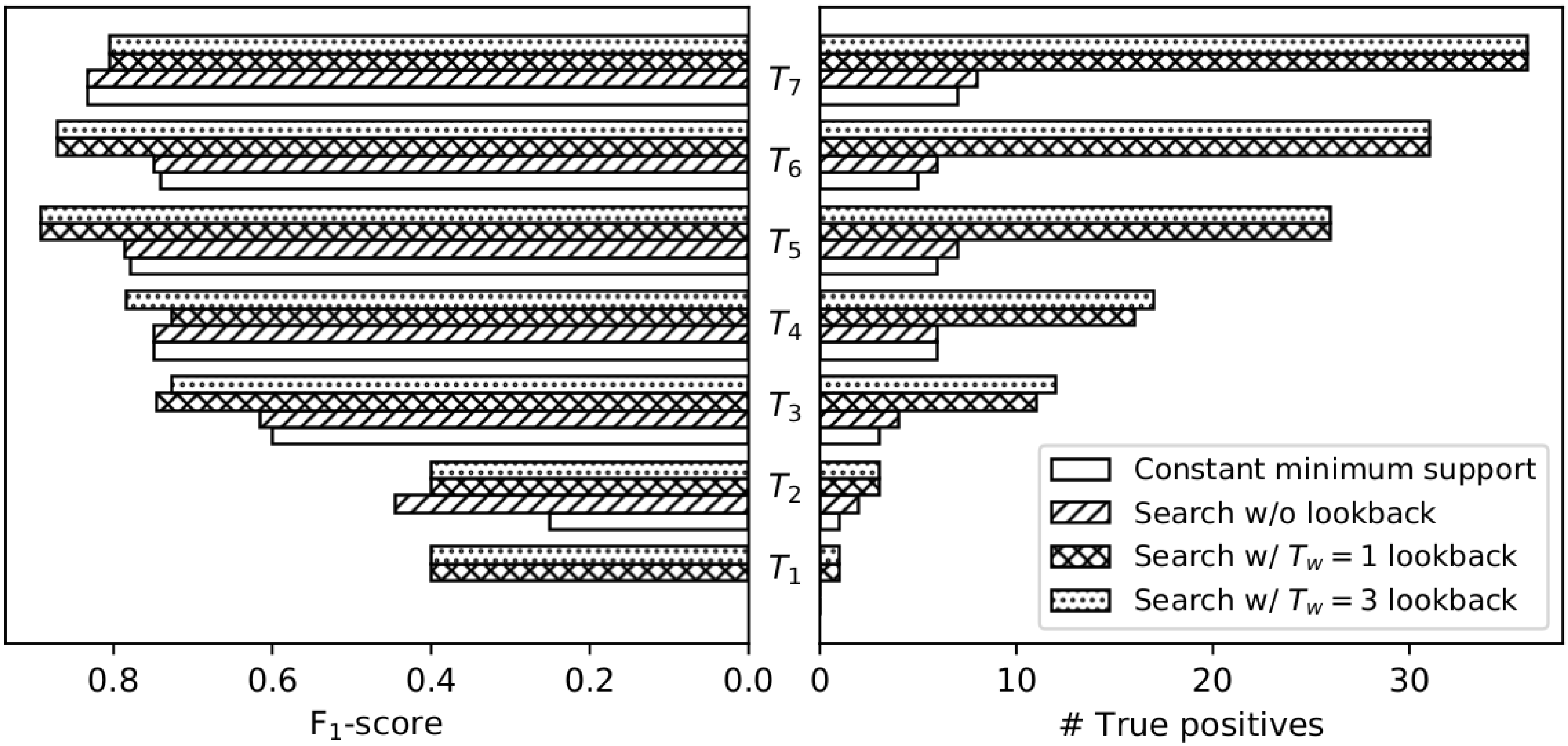
Experiment 2

Effectiveness of algorithm when attacks are temporally dispersed

- Different variants of mining algorithm at Manager
  - Constant minimum support
  - Search without lookback (vary support)
  - Search with lookback of one time-slot
  - Search with lookback of three time-slots

Experiment 2

Effectiveness of algorithm when attacks are temporally dispersed



Conclusions and plans

- ADROIT
  - A system for detecting anomalies and mining patterns related to attack-stages
  - Exploited the fact that, in comparison to end-hosts, IoT devices can be better profiled
  - The distributed architecture allows collapsing spatial dispersion, whereas proposed *look-back* algorithm helps to mine temporally dispersed alerts
- Next steps
  - Test of large-scale attack traffic, considering multiple botnets
  - Identify attack-stages automatically
  - Can we map to behaviors of specific botnets?

Thank You!