DDIFT: Decentralized Dynamic Information Flow Tracking for IoT Privacy and Security

N. Sapountzis, R. Sun, D. Oliveira,
University of Florida,
February 2019
Warming Up

Happiness is a fast internet connection

☺
Overview

1. *Intro*: IoT era and DIFT
2. **DDIFT 1**\textsuperscript{st} step: Mobile phone running DIFT [fast timescale]
3. **DDIFT 2**\textsuperscript{nd} step: Cloud running forensics [slow timescale]
4. **DDIFT**: Overview
5. *Simulation* of DDIFT
6. *Conclusions*
2019: The dominance of IoT in human life is a reality

- Wearables
- Smart homes
- Healthcare
- Greening ecosystem
… usually *applications*, or *applets*, running at the mobile phone, is the interface to manage these devices

- Wearables --> *applet to upload to Doogle Dr. all new pics*
- Smart homes --> *open thermostat when I approach home*
- Healthcare --> *notify my doctor if my heart rate has improper impulses*
- Greening ecosystem --> *provide PV array analytics*
these apps ( = applications + applets ), and most popular platforms to develop them, open up for various privacy, security, availability and integrity concerns.

1. Intro
1. Intro: traditional DIFT

DIFT: works with 2 processes

- PROCESS A: Tag insertion: insert tags to variables (or memory bytes)
- PROCESS B: Tag propagation: propagate tags during system execution

- Direct Flow Propagations (DFP)
- Indirect Flow Propagations (IFP)

1. int base = 5; //the base of the triangle
2. int height = 4; //the base of the triangle
3. int area = base * height; //the area

1. int base = 5; //the base of the triangle
2. int height = 4; //the base of the triangle
3. int area = base * height; //the area

1. int base = 5; //the base of the triangle
2. int height = 4; //the base of the triangle
3. int shape = “triangle”; //type of shape
4. (If shape == “triangle”)
5. area = base * height //the area
2. DDIFT Algorithm at device

What about DIFT in IoT?
2. Device: running DIFT at the app level

Challenges while applying DIFT in IoT [Tag Insertion]:

1. Different system activities add heterogeneous knowledge in the information flow and should add different level of security concerns
   - We consider tag differentiation (many colors), e.g., network, file, RV tags
   - Each variable has a list that accommodates up to N #of tags

2. Ability to reverse engineer back to the inputs
   - We keep provenance information (e.g., network tag coming from IP 13.2.3.5)
2. Device: running DIFT at the app level

Challenges while applying DIFT in IoT [Tag Propagation]: (cont’)

3. Limited (memory, battery resources).
   • Algorithm 1 (next slide): How to optimally allocate the limited resources to the vast #tags that attempt to propagate?

4. Minimize false alarms that DIFT usually bring.
   • Algorithm 1 (next slide): Should all indirect flows be propagated?
Algorithm 1 Optimal resource allocation and indirect flow decisioning.

**Input.** $C$: the objective metric we attempt to optimize.

**Output.** $\Delta(n_{t,i})$: drop or schedule the tag.

1: Define the objective metric we attempt to optimize based on the per-tag metric value $c_{t,i}$, the tag type weights $\lambda$, and the tag weights $\mu$.

$$C = \sum_t \lambda_t \sum_i \mu_{t,i} \cdot c(n_{t,i})$$  \hspace{1cm} (1)

2: In order to decide about the potential propagation of a tag, the DDIFF system should consider which decision offers the best gain for $C$. To that end, we differentiate $C$ with respect to $n_{t,i}$ (number of copies of the $i$-th tag belonging at the $t$-th type), we discretize and obtain:

$$\Delta(C) = \sum_t \lambda_t \sum_i \mu_{t,i} \frac{\partial c(n_{t,i})}{\partial n_{t,i}} \Delta(n_{t,i}) =$$

$$= \sum_t \sum_i U_{t,i} \Delta(n_{t,i}),$$ \hspace{1cm} (2)

where:

$$U_{t,i} = \lambda_t \cdot \mu_{t,i} \cdot \frac{\partial c_{t,i}}{\partial n_{t,i}}$$ is the utility of tag $\{t,i\}$ \hspace{1cm} (3)

$$\Delta(n_{t,i}) = \begin{cases} 
-1, & \text{if the tag } \{t,i\} \text{ is dropped} \\
0, & \text{if no action for the tag } \{t,i\} \text{ is taken} \\
+1, & \text{if the tag } \{t,i\} \text{ is scheduled} 
\end{cases}$$ \hspace{1cm} (4)

3: For resource allocation: the DDIFF should (i) schedule (i.e., keep or propagate) the tags in the order of decreasing $U_{t,i}$, and (ii) drop (i.e., delete or not propagate) the tags with the lowest $U_{t,i}$ (i.e., to ensure that the tags “carrying” more information are prioritized).

4: For indirect flow propagation decisioning: the DDIFF should propagate a tag if this tag has $U_{t,i} > 0$ (i.e., the indirect flow propagation brings information to the DDIFF).
3. Cloud: running forensics

- Cloud performs heavy forensics analysis in a slow timescale, relying on a continuous analysis of a large volume of tags.
- Cloud’s main objective is to:
  - dictate the best values for the weighting parameters $\lambda$, $\mu$, so e.g. the devices boost the important tags:
    $$\lambda = \lambda_{prev} + \zeta \cdot \sum_i \mu_{t,i} \cdot c(n_{t,i}).$$
  - develop privacy and security policies,
    - e.g. URL tag + String tag + netflow tag = URL Attack
    - learn a priory what strings, network connections etc. are suspicious
4. DDIFT: Decentralized Dynamic Information Flow Tracking
5. Simulations (N=5): traditional DIFT
(propagate all IFs 😞)

1. name = GoogleContacts.newContactAdded.Name
   //retrieve the name of the new contact
2. num = GoogleContacts.newContactAdded.PhoneNumber
   //retrieve the number
3. digits = Math.random() * 10 + ''
   // returns [0,10] - # of digits that the attack will change
4. for (int i = 0; i <= digits; i++)
   // parse all "digit" first digits
5. digits = do things
   //(e.g., connect to diff. files/ports)
   to congest provenance lists
6. value = Math.floor(Math.random() * 10) + ''
   // flip a coin to find the new digit
7. num = num.replace(num.charAt(i), value) // MALICIOUS ACTION
   → change digit
8. exit = GoogleSheets.appendToGoogleSpreadsheet(name + num)
   //upload to Google drive spreadsheet
5. Simulations (N=5): traditional DIFT (propagate all IFs 😞)

- All IFs propagated
  =>
  Over-tainting (+ memory pollution)

- No one of the IF is propagated
  =>
  Under-tainting

Other over-tainting cases (when more space is available)
  =>
  Aggressively propagate tags
Simulated different provenance-list sizes scenarios:

Detection efficiency improvement 43%
Memory usage improvement 71%
6. Conclusions

- DDIFT: dynamically tracks the information flow at the mobile device level, and adapts to the IoT challenges:
  - **Large #devices**: Scalable scheme through the synergy of cloud (slow timescale) and device (fast timescale)
  - **IoT limited resources**: optimally prioritizing tags
  - [open problem] **Indirect Flow Propagation**: tackled with optimization theory
  - Extendable to software and hardware
  - Able to design malware signatures through the tag confluences and further detect them
Thank you!