



Security & Privacy for IoT Devices

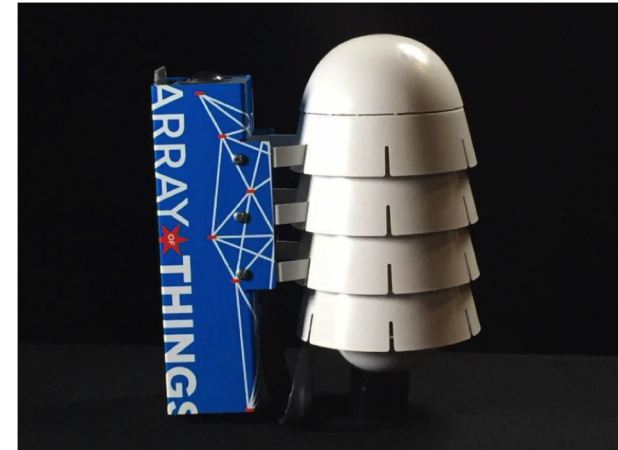
For CMSC 23200/33250 Introduction to Security

Presented by Weijia He on March 9, 2020



THE UNIVERSITY OF
CHICAGO

Examples of Internet of Things (IoT) Device?

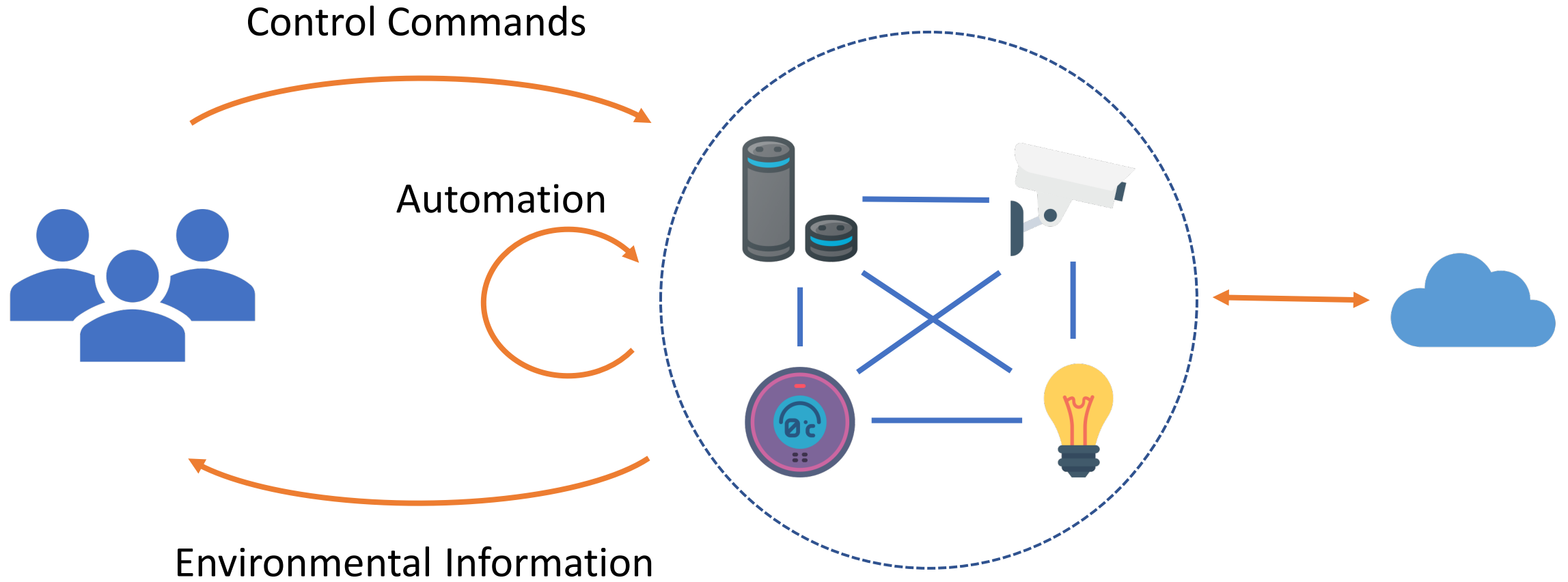


What is an Internet of Things (IoT) Device?

“These devices, or *things*, connect to the network to provide information they gather from the environment through sensors, or to allow other systems to reach out and act on the world through actuators.”

Source: <https://cloud.google.com/solutions/iot-overview>

Smart Home Architecture



Threat Model

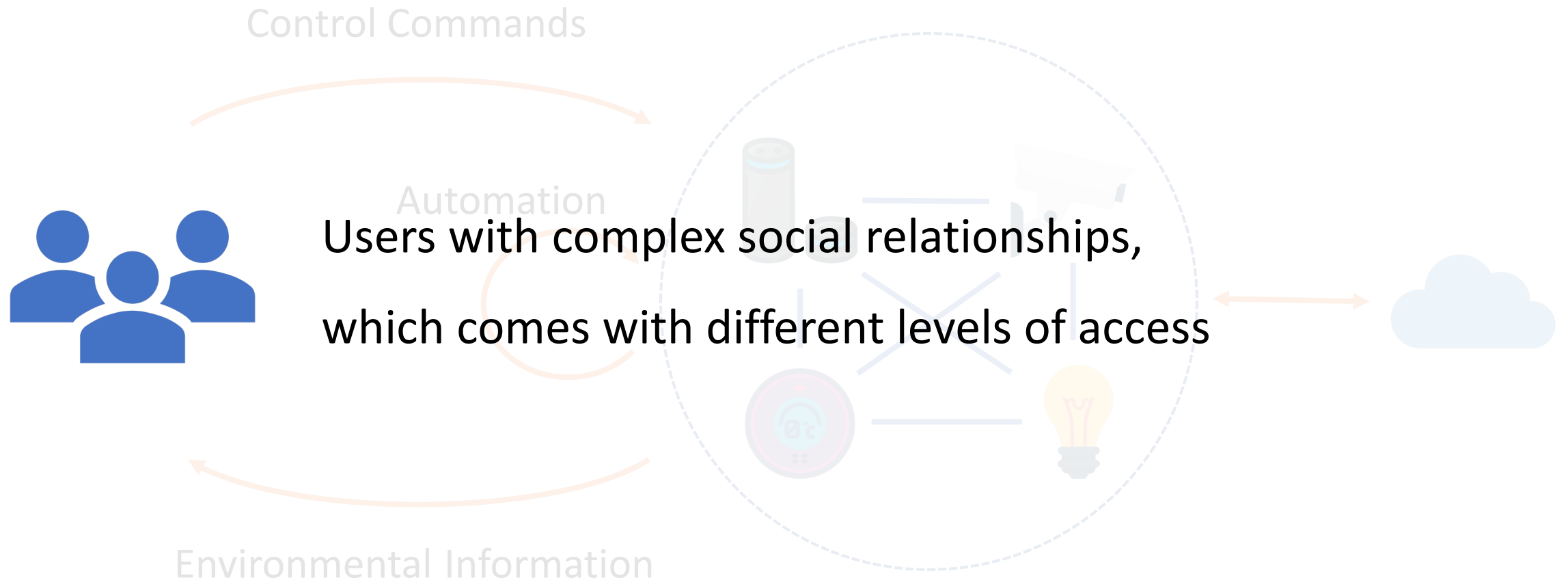


- DDoS
- Network Spoofing
- ...

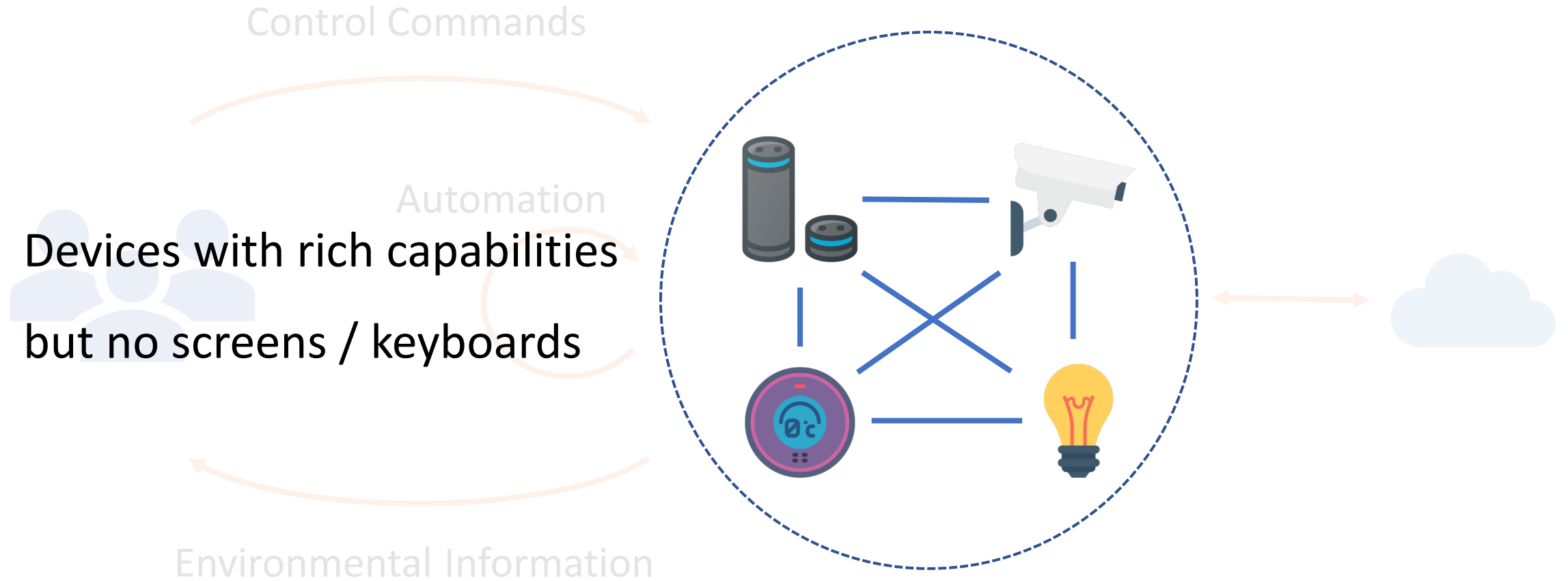
} Network Attacks

We have already studied all of them! Anything New?

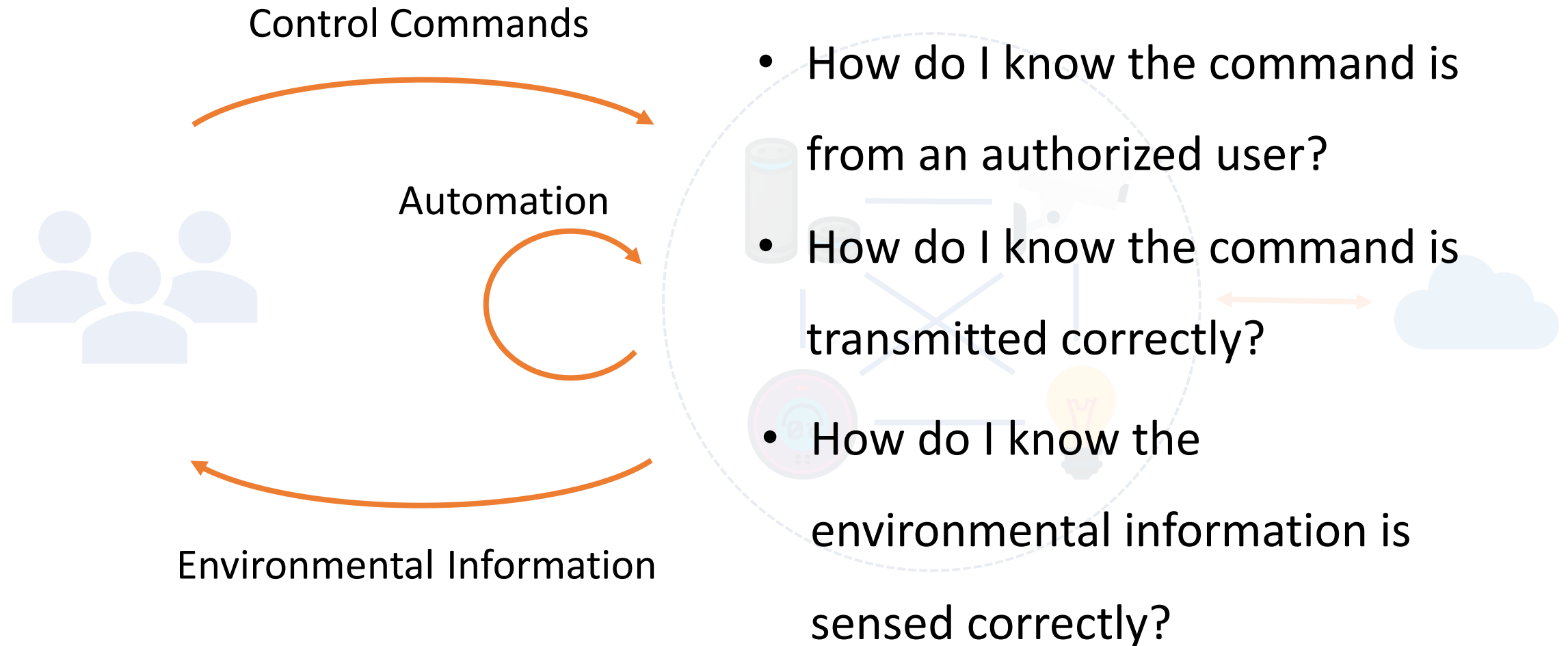
New Challenges in a Smart Home



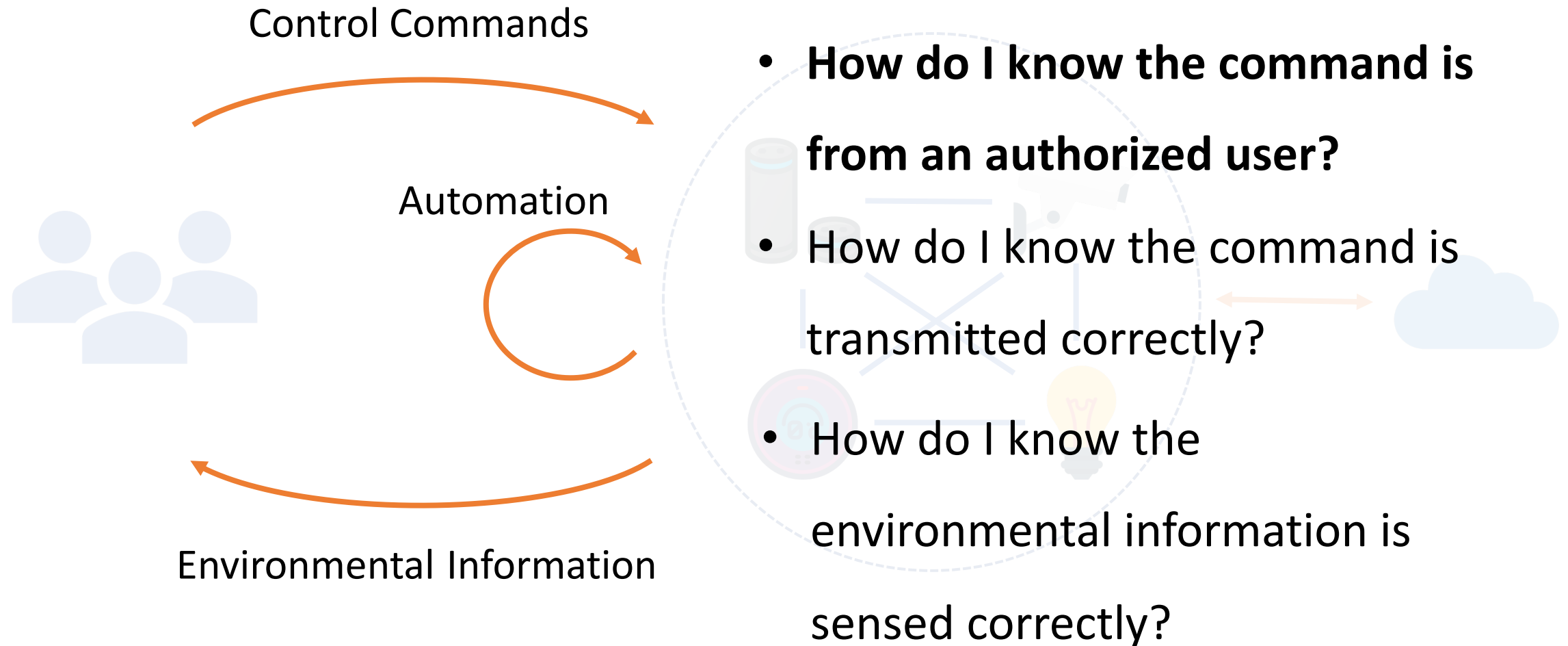
New Challenges in a Smart Home



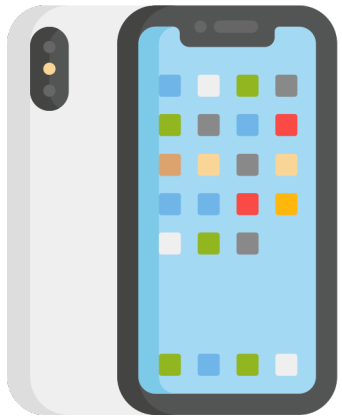
New Challenges in a Smart Home



New Challenges in a Smart Home



Current Authentication in Smart Homes



- Account Logins
- PIN

Nothing at all

Various Modalities in a Smart Home



- Voice



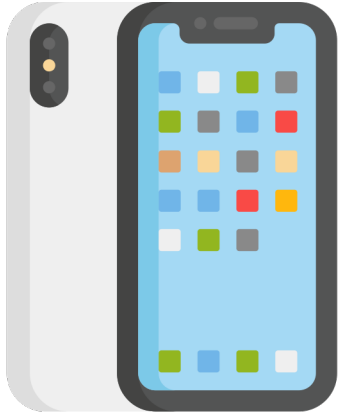
- Physical Control

- All of the modalities should be protected.

Smartphone \neq Identity

- Device / account sharing
- Temporarily taking away someone's phone without being noticed

Authentication in Smart Homes



- Account Logins
- PIN

Smartphone is not enough for authentication in smart homes!

Complementation

- Voice authentication (for voice assistants)
 - Speaker Verification (usually text-dependent)
 - Problem: easy to be recorded and replayed (replay attack)
 - Solution: liveness detection (e.g. oral airflow, body vibration, etc.)
 - Speaker Identification (usually text-independent)
 - Problem: more false positives, requires more training

Complementation

- Facial recognition
 - Pros: mature, easy to deploy
 - Cons: privacy concerns
 - Attacks:
 1. Presentation Attacks (e.g. holding a photo to the camera)
Solution: adding secondary information (e.g. infrared camera, depth camera, etc.)
 2. Adversarial Examples

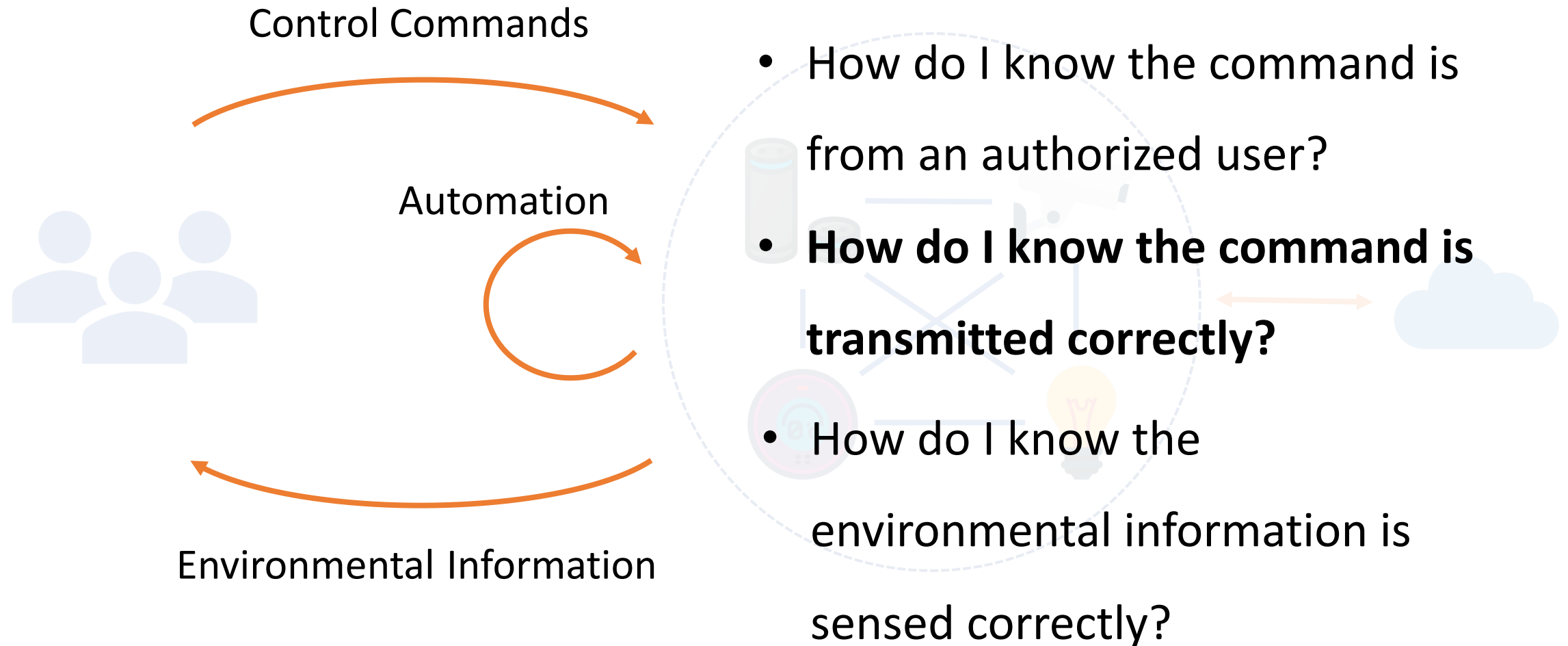
Other Potential Solutions

- Continuous Authentication
 - Wearable devices
 - Assumption: People will carry the device around.

Other Potential Solutions

- Contextual Access Control
 - Assumption 1: Authentication is for access control.
 - Assumption 2: Not all access control policies require to know the user's identity.
 - Example: The cat monitor should be on only when nobody is at home. (GPS)

New Challenges in a Smart Home



Attacks on Voice Assistants

- Skill Squatting Attacks
 - Alexa may be confused by some words that are phonetically similar
- Potential Defense:
 - Amazon should restrict registration on skill names that may be confusing to Alexa (similar to domain registration)

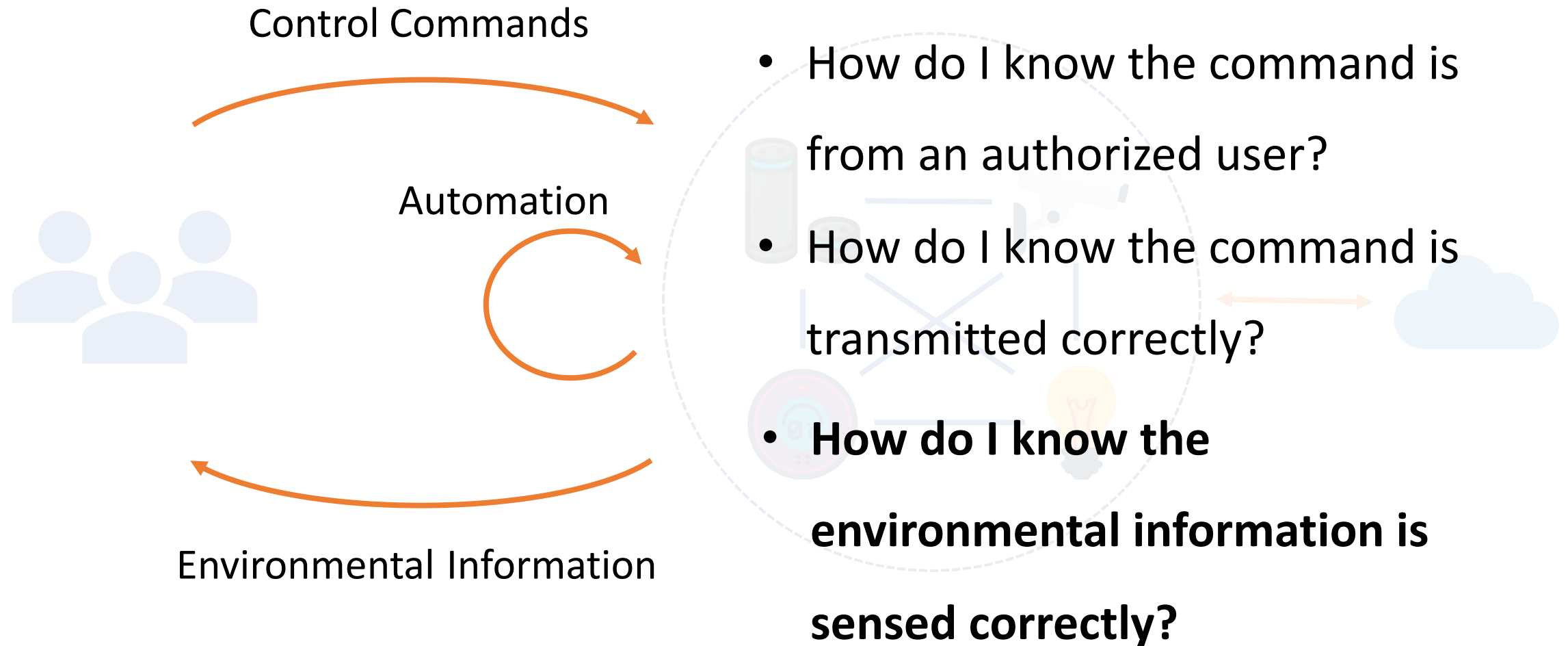
Attacks on Voice Assistants

- Inaudible voice commands
 - Voice commands can be issued at ultrasonic frequencies, which can be picked up by microphones, but not by human beings.
- Potential Defense
 - Using human voice structure as a detection method

Attacks on Voice Assistants

- Hidden voice commands
 - Commands that are unintelligible to human beings, but can be understood by voice assistant.
- Potential Defense:
 - Decrease the fidelity of the input audio, because normal commands can still be understood with degrading of quality, while hidden commands might be misclassified, considering they are at the border of classification.

New Challenges in a Smart Home



Attacks on Context Sensing

- Replay attacks, imitation attacks, adversarial examples
- Physical DoS
 - Example 1: Blocking passive IR motion sensors
 - Example 2: Flooding the sensors with noise
 - Problem: Hard to perform detection

Privacy in a Smart Home

- Between users and companies
- Between legitimate users and unauthorized users
- Between owners and visitors/bystanders
- Among legitimate users

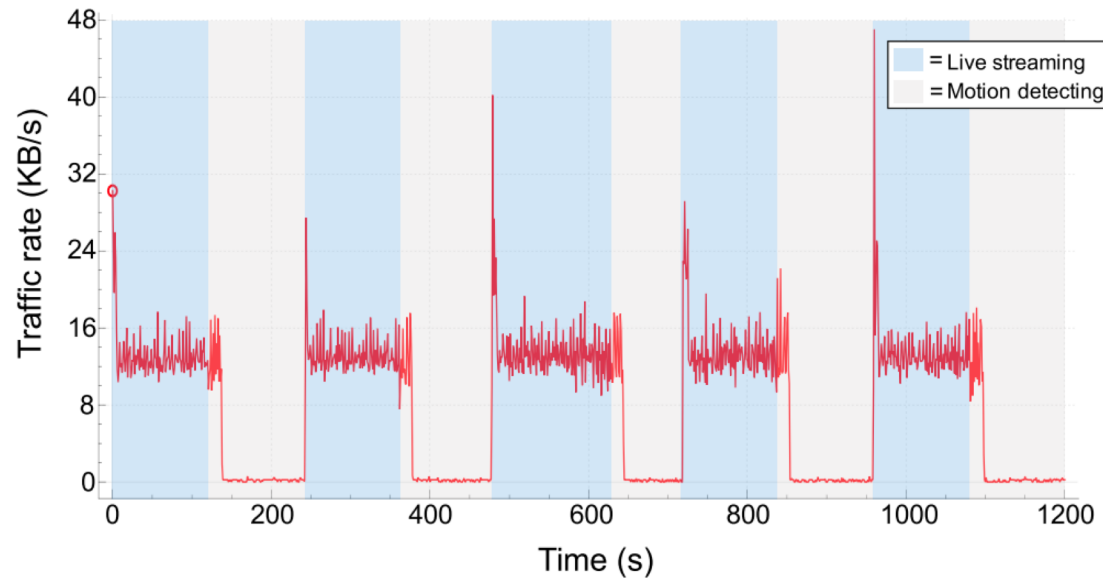
For companies...

- Data collection
 - Functionality vs. data collection
- Data storage
 - Local vs. cloud
- Data retention
 - Temporary data vs. persistent data

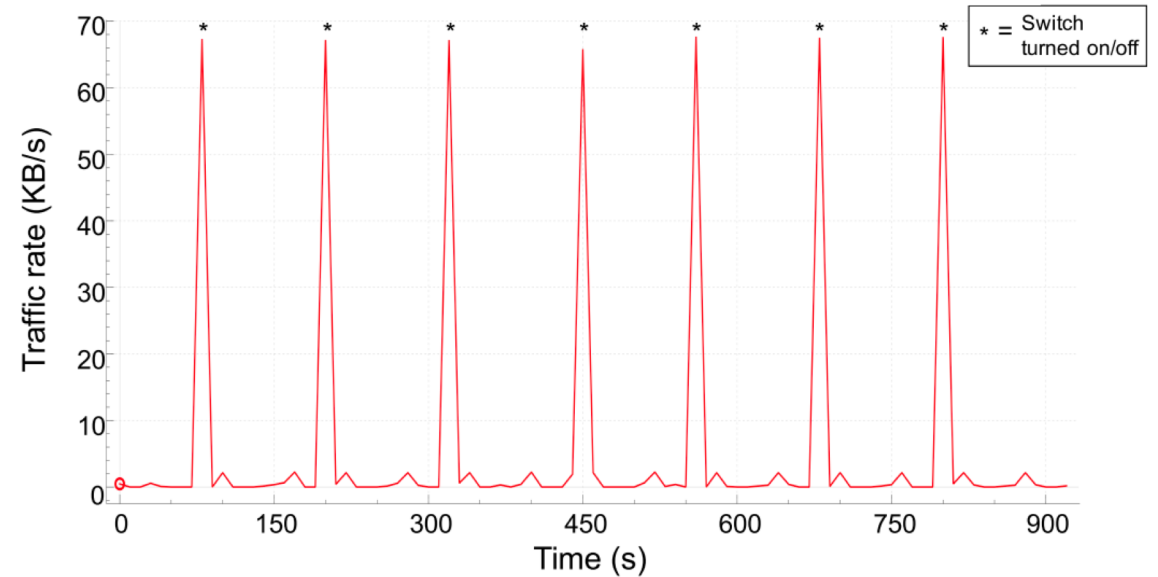
For unauthorized users...

- Network traffic analysis
 - Activity identification

(a) Nest security camera – Video monitoring

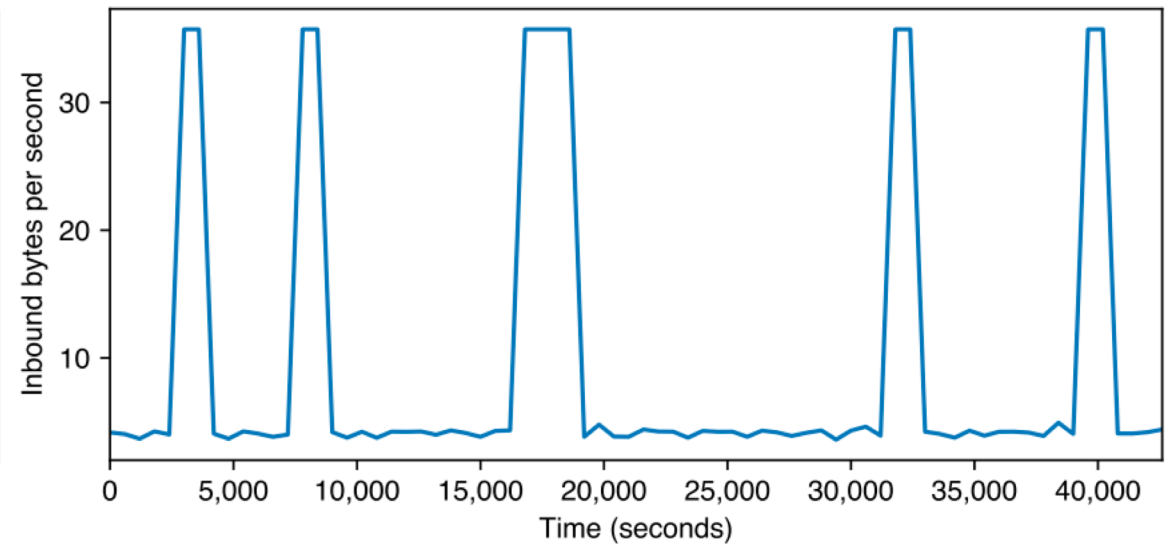
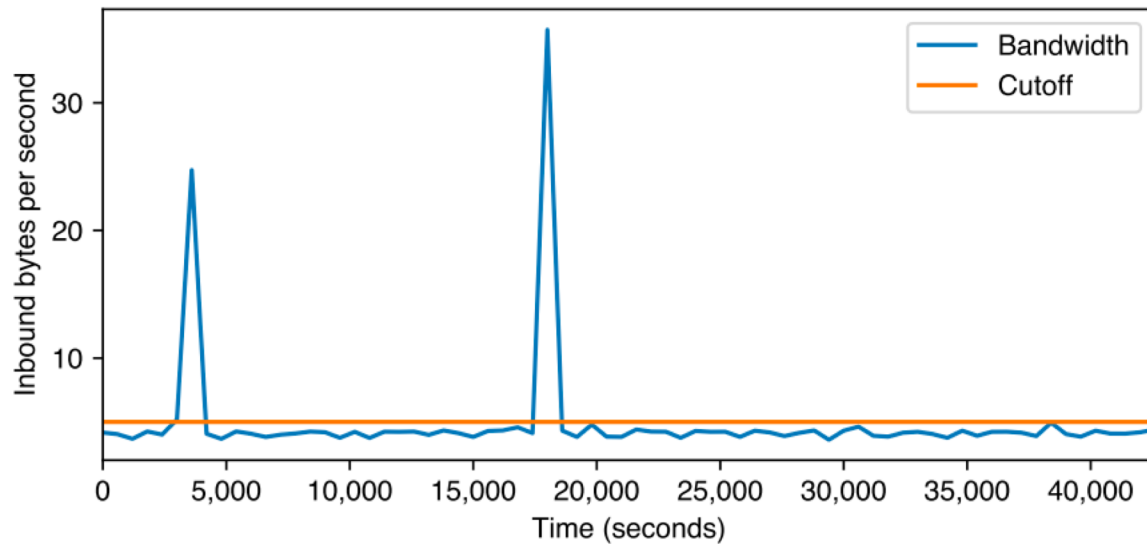


(d) Belkin Wemo switch – Appliance power cycle



For unauthorized users...

- Network traffic analysis
 - Activity identification
 - Solution: Padding




For bystanders...

- Different privacy preferences from owner's
- Sometimes it's hard to tell what is smart by the visitors themselves
 - A smart scale
- Hard to inform beforehand unless the owner is being careful

For other legitimate users...

- Between children and parents
- Between landlords and renters
- Between roommates
- Between spouse



No one-fit-all privacy preferences and concerns

Intimate Partner Violence (IPV)

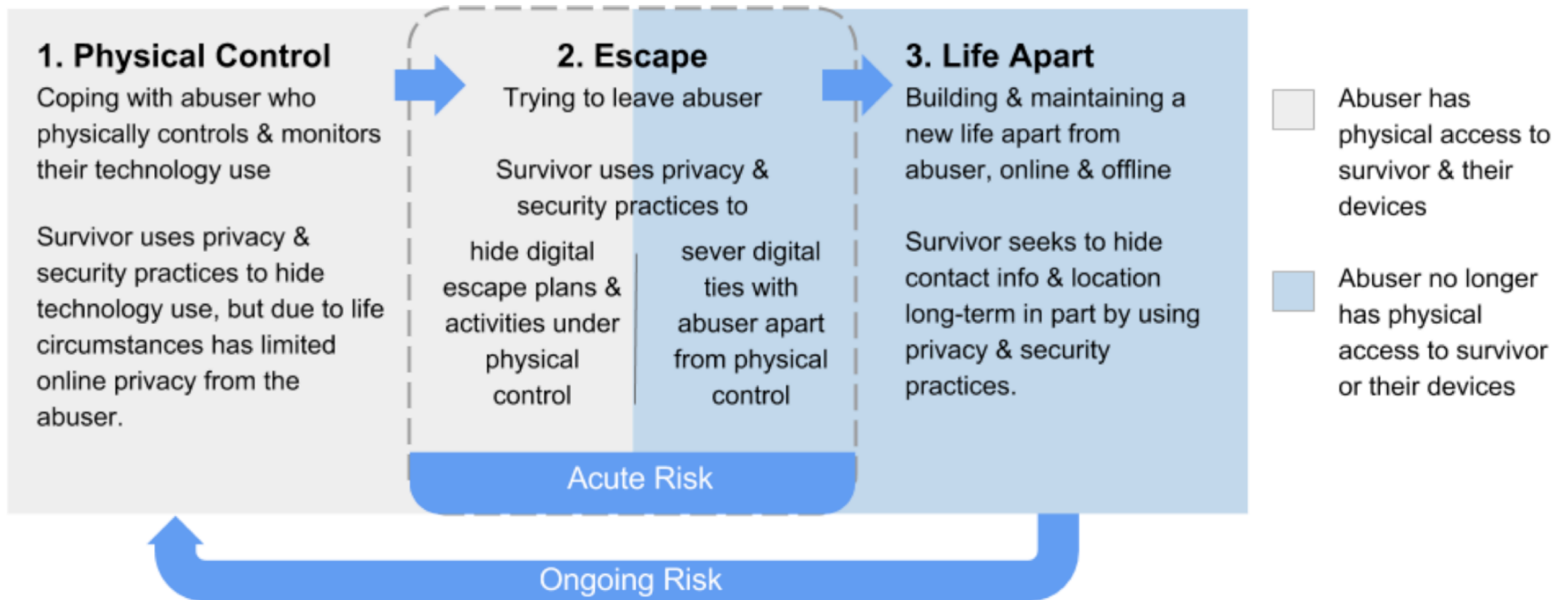


Figure 1. Three phases of IPA that affected technology use, focusing on privacy & security practices.

Src: Matthews, Tara, et al. "Stories from survivors: Privacy & security practices when coping with intimate partner abuse." *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017.

Intimate Partner Violence (IPV)

Abuser Attacks Experienced		#Part.
<i>Physical control</i>	(a) Device/account controlled & monitored - Physical means	10
	(b) Device destroyed	4
	(c) Spyware installed	3
Cross-phase digital attacks	(d) Harassed online	8
	(e) Account hijacked - Impersonated	5
	(f) Account hijacked - Locked out	4
	(g) Account monitored - Remote or unknown means	2

Src: Matthews, Tara, et al. "Stories from survivors: Privacy & security practices when coping with intimate partner abuse." *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017.

IPV in a Smart Home

- Device & account control:
 - Smartphone -> all the smart devices in one's home (more pervasive)
- Spying
 - Installing spyware -> naturally provided by all smart devices (more constant)

Challenges

- An abuser is likely to be the one who install all the smart devices, which means they have the highest privilege in the system.
- Limiting victim's access to the system is possible.
- Hard to tell the difference from the perspective of the system and may cause severe problems if there is false positives.

Formal modeling and dynamic analysis

March 9, 2020



Overview

Dynamic taint tracking: determine sensitive data leaks or insecure data flows

- + Useful for knowing where information is going
- Can be hard to track

Fuzzing: find bugs randomly

- + Low cost
- Not guaranteed to find all bugs

Model checking: verify whether the model of a system satisfies certain requirements

- + Verifies expected behavior across all scenarios & state space
- Expensive

Static vs. dynamic analysis

Static: analysis when the code isn't running

- + Checks all code paths
- Checks code paths that don't get executed
- Unsure about runtime properties

Dynamic: analysis as the code runs

- + Reason about real executions
- Results can change based on input
- Slows down the code

```
void foo(int a) {  
    int x, y;  
    x = 2;  
    if (a > 10) {  
        x = 1;  
    }  
    y = 10;  
    print(x);  
    print(y);  
}
```

Taint tracking (dynamic)

Assign **security labels** to data and computations

- **Taint sources:** sensitive data or untrusted input
 - Files, network protocols (HTTP, UDP, etc.), keyboard / mouse / touchscreen input messages, webcam, USB, VM images, etc.
- **Taint sinks:** where tainted data exits the system
 - Network output, disk writes, etc.
- **Taint markings:** assigned to data affected by taint sources

Taint tracking - applications

Buffer overflow

→ Tainted data being used as target of a jump / format string / syscall argument

SQL injections

→ Taint source = user input

Tracking info leakage

→ Hayes, USENIX ATC '17: tracking extracted DNN features on GPU?

Debugging, testing

→ Isolating code that pertains to insecure flows

Taint tracking - explicit flows

```
1  int a, b, w, x, y, z;  
2  (a) = 11;  
3  (b) = 5;  
4  (w) = (a) * 2;  
5  (x) = (b) + 1;  
6  (y) = (w) + 1;  
7  (z) = (x) + (y);
```

Taint tracking - implicit flows

```
1  void foo(int a) {  
2      int x, y;  
3      if (a > 10) {  
4          x = 1;  
5      }  
6      else {  
7          x = 2;  
8      }  
9      y = 10;  
10     print(x);  
11     print(y);  
12 }
```

(a)

```
1  void foo(int a) {  
2      int x, y;  
3      x = 2;  
4      if (a > 10) {  
5          x = 1;  
6      }  
7      y = 10;  
8      print(x);  
9      print(y);  
10 }
```

(b)

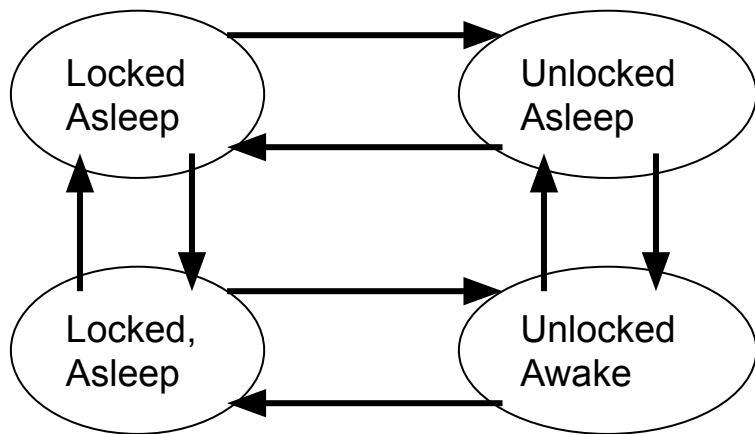
Fuzzing

Throw random inputs at the program until it crashes, fails an assertion, or leaks memory

```
void foo(int a) {  
    assert(a != ■); // new  
    int x, y;  
    x = 2;  
    if (a > 10) {  
        x = 1;  
    }  
    y = 10;  
    print(x);  
    print(y);  
}
```

Model Checking

“Model checking is an automated technique that, given a **finite-state model** of a system and a **formal property**, systematically checks whether this property **holds** for (a given state in) that model.”



(Safety) property:

The door should never be unlocked while I am asleep.

Atomic propositions:

- Whether the door is locked
- Whether I'm asleep

Application to IoT

Information flow analysis

- **Surbatovich, WWW '17**: security & integrity lattices for TAP rules
- **Bastys, CCS '18**: FlowIT - dynamic information flow tracker for JavaScript
- **Celik, USENIX '18**: SAINT - static taint analysis tool (language-independent, platform-aware)

Formal verification

- **Zhang, ICSE '18**: AutoTap - model checking on user specifications
- **Wang, CCS '19**: iRuler - discover inter-rule vulnerabilities with model checking