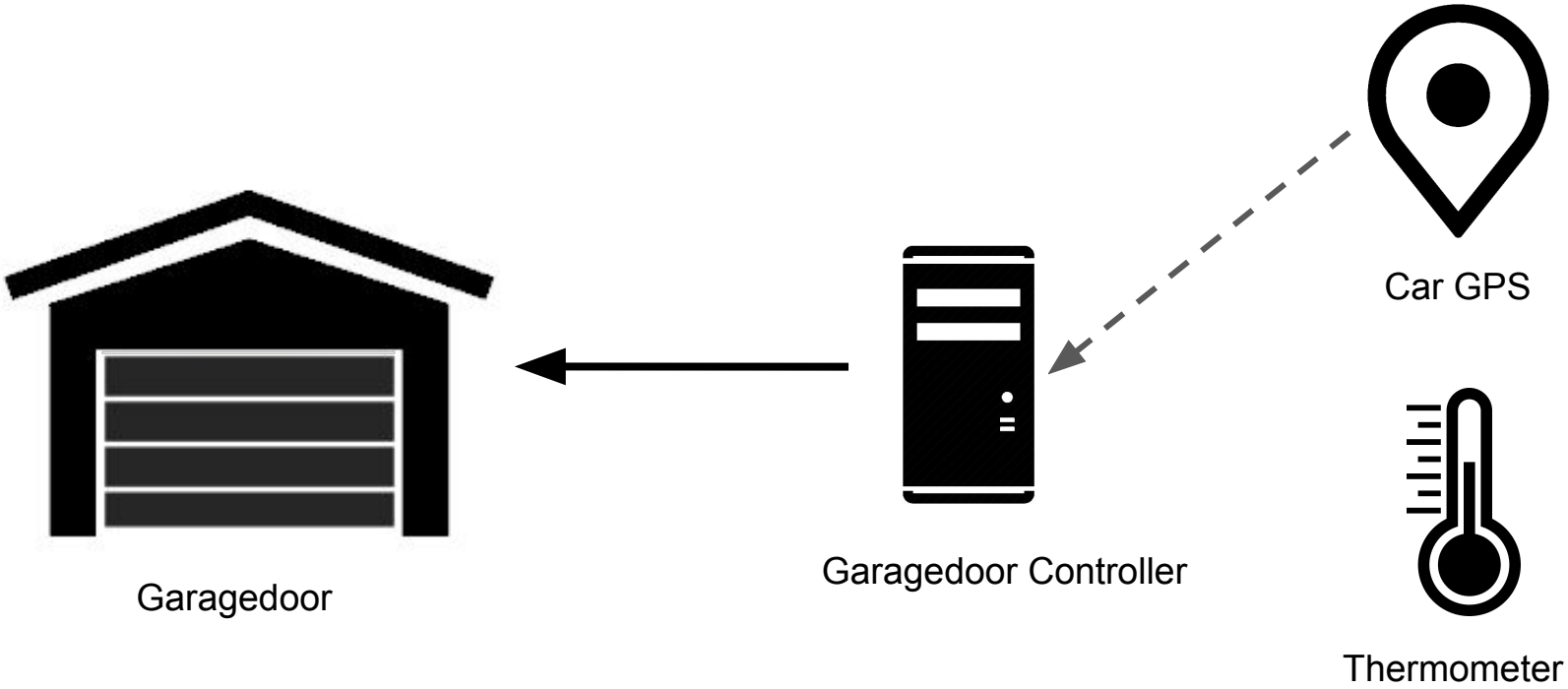
Verifying Security for Smart Homes

CSE 564 project

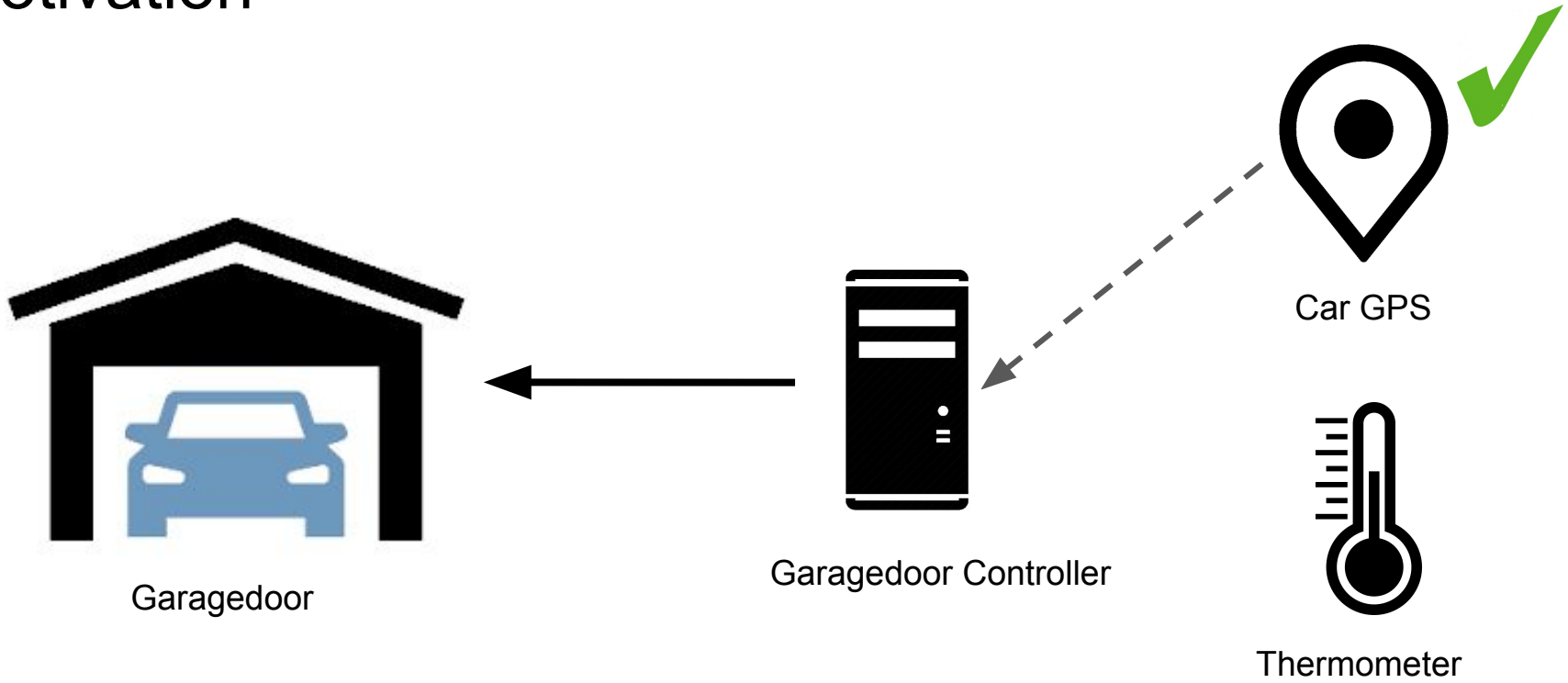
03-14-2016

Chandrakana Nandi, Jeanette Daum
Collaborator: Michael Ernst

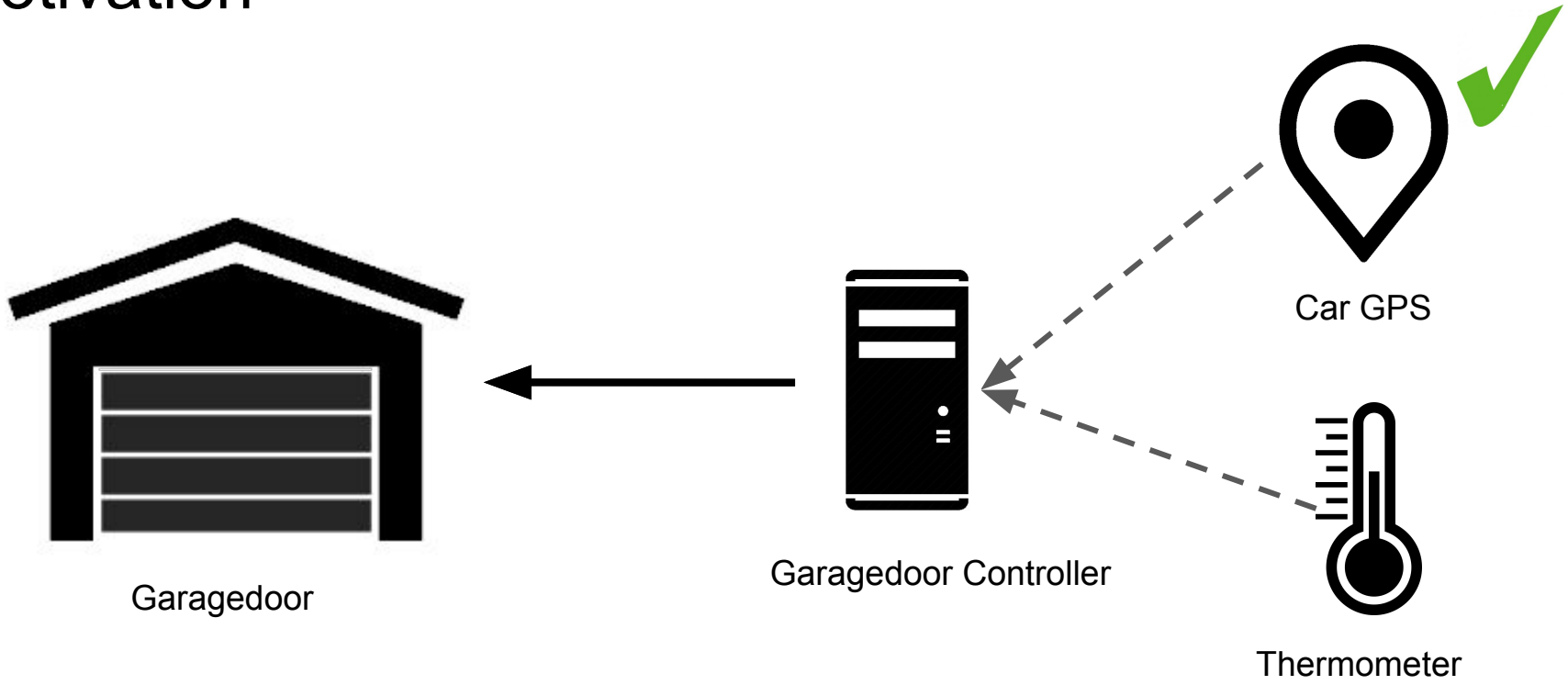
Motivation



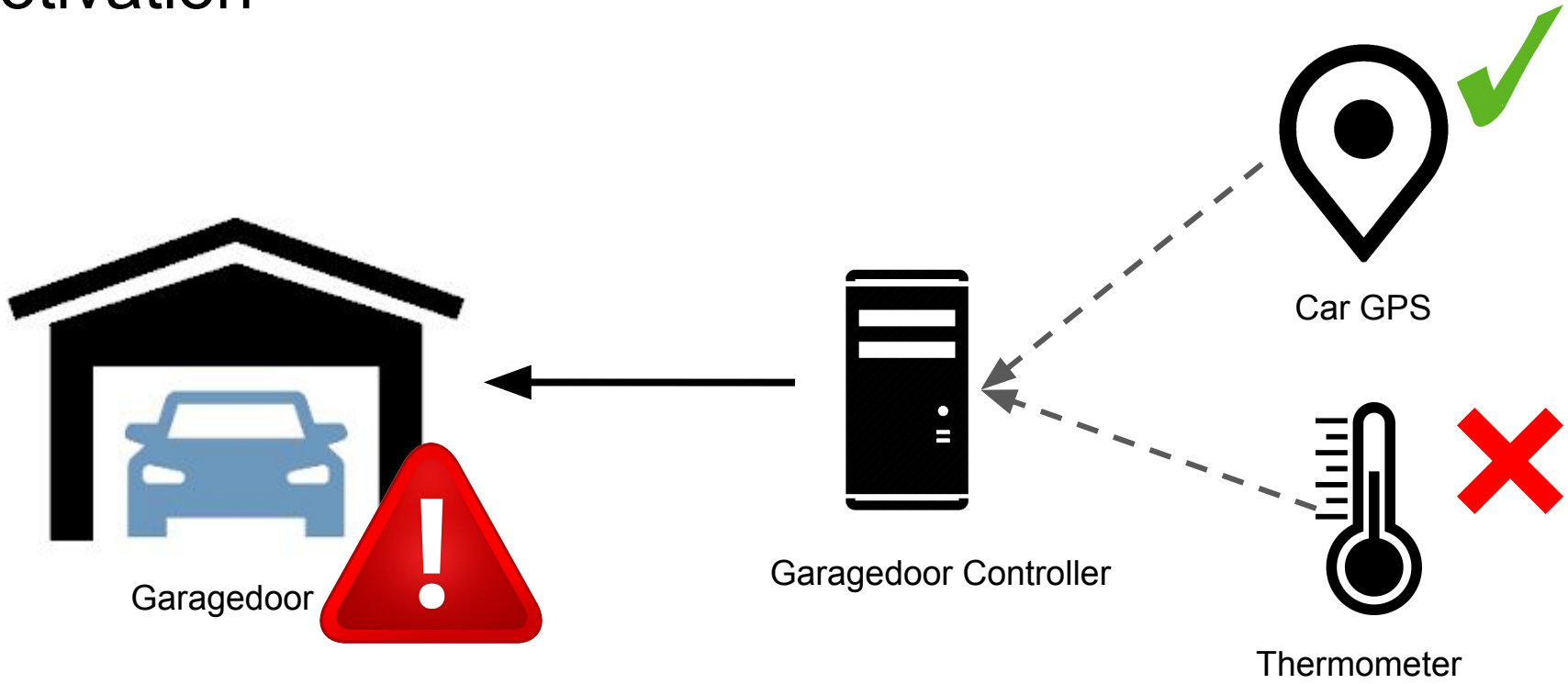
Motivation



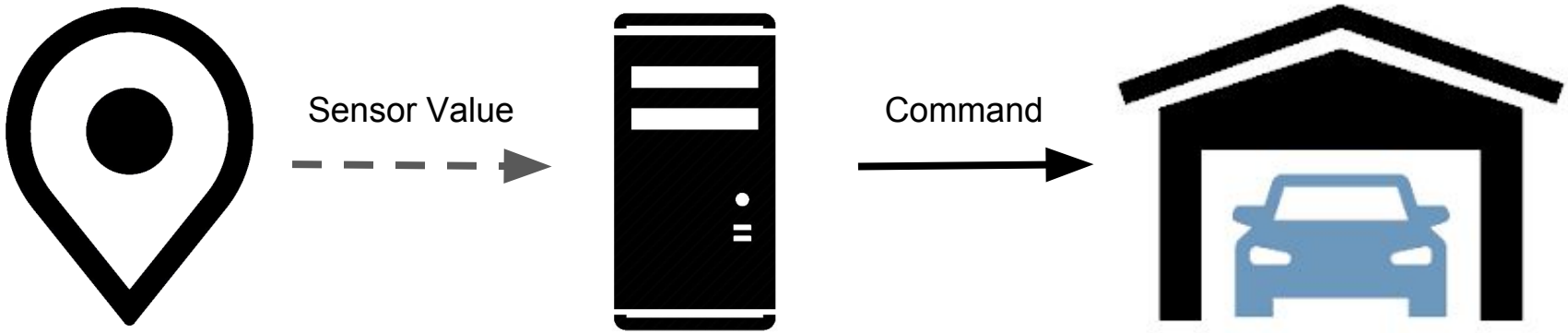
Motivation



Motivation



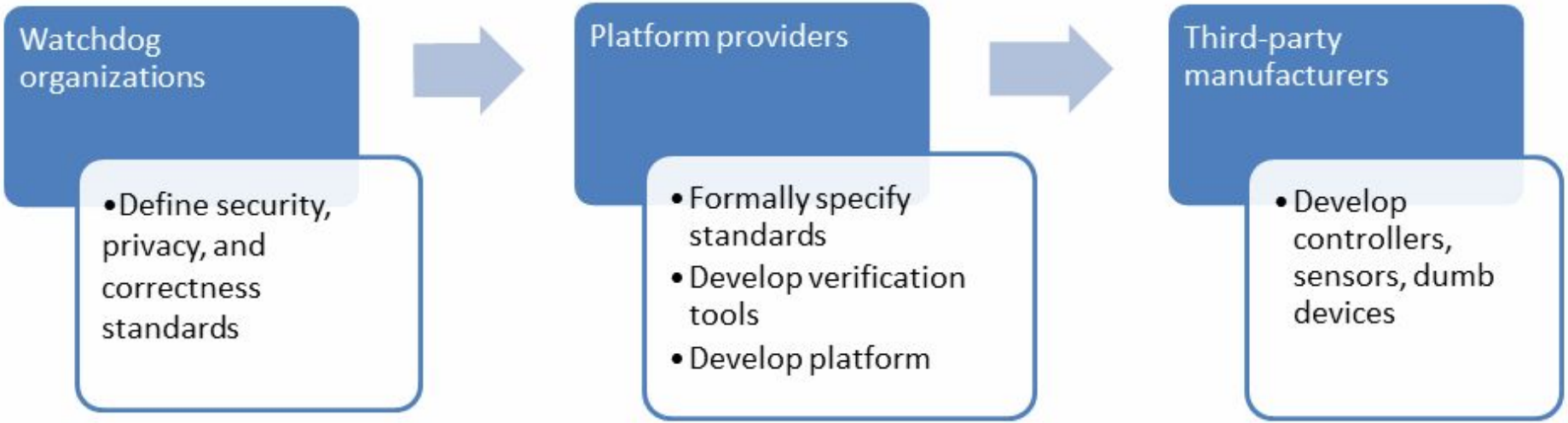
Architecture



Our approach

- We implemented a simulation of a smart home
- Specified security policies for the different smart devices in first order logic
- Verified the code of the devices using static analysis
  - Provide compile time guarantees about security

Business model



Threat model

- Controllers developed by third-party manufacturers
  - Send commands under incorrect sensor conditions
  - Misuse sensor values
  - Send wrong commands to right device
  - Send right/wrong commands to wrong device

- Watchdog organizations
- Specification and verification tools
- End users
- Dumb devices
- Communication protocols

Security policies

- Dependency policy
  - Commands are sent by controllers under correct sensor values
- Control policy
  - Controllers send right command to only those dumb-devices they can control
- Information flow policy
  - Sensitive information is not sent to the cloud without user permission and/or anonymization
- Temporal policy
  - Events respect the order in which they are supposed to happen: air-conditioner should turn on after windows are closed

Dependency policies

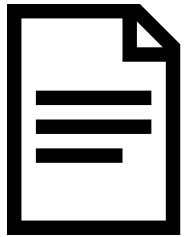
- **The garage door opens if and only if it is closed, the owner is inside the car, and the car is either approaching nearby or running within the garage.**
  - GARAGEDOOR_CONTROLLER **sends** *open_garagedoor* ⇔ (¬IS_GARAGE_OPEN) ∧ ((¬IS_CAR_INSIDE_GARAGE ∧ CAR_DISTANCE ≤ “50m” ∧ CAR_SPEED > 0) ∨ (IS_CAR_INSIDE_GARAGE ∧ IS_CAR_RUNNING)) ∧ (IS_OWNER_INSIDE_CAR)

- **The laundry machine may start when the doors are closed, the machine is not empty, the clothes are not already clean, and the machine is not already running.**
  - LAUNDRYMACHINE_CONTROLLER **sends** *start_washer* ⇒ (IS_DOOR_CLOSED) ∧ (¬IS_EMPTY) ∧ (¬IS_CLEAN) ∧ (¬IS_WASHER_ON)

Verification

We implemented a static analysis tool based on Google's error-prone framework

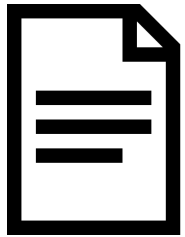
```
public void update() {
        if (Platform.washerCleanSensor.getEntities().get("IS_CLEAN").equals(false)
        && Platform.washerDoorSensor.getEntities().get("IS_DOOR_CLOSED").equals(true)
        && Platform.washerEmptySensor.getEntities().get("IS_EMPTY").equals(false)
        && Platform.washerSensor.getEntities().get("IS_WASHER_ON").equals(false)) {
                checkExecutabilityAndSend(new Command("start_washer"));
        }
}
```



Policy

Verification: Command sent inside update()

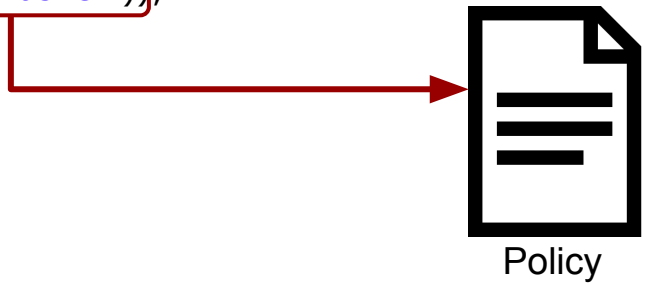
```
public void update() {
        if (Platform.washerCleanSensor.getEntities().get("IS_CLEAN").equals(false)
        && Platform.washerDoorSensor.getEntities().get("IS_DOOR_CLOSED").equals(true)
        && Platform.washerEmptySensor.getEntities().get("IS_EMPTY").equals(false)
        && Platform.washerSensor.getEntities().get("IS_WASHER_ON").equals(false)) {
                checkExecutabilityAndSend(new Command("start_washer"));
        }
}
```



Policy

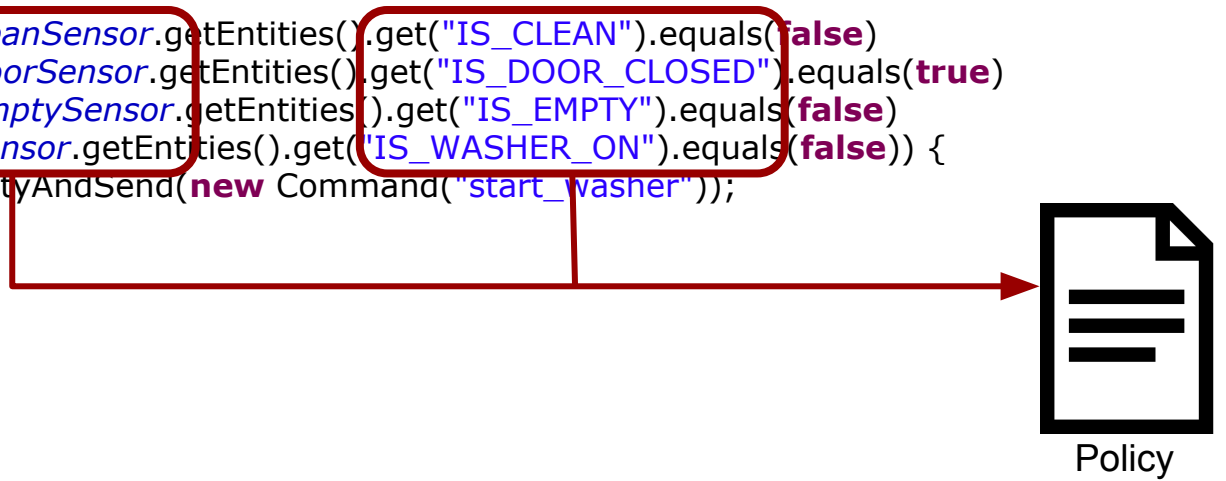
Verification: Command exists in policy

```
public void update() {
        if (Platform.washerCleanSensor.getEntities().get("IS_CLEAN").equals(false)
        && Platform.washerDoorSensor.getEntities().get("IS_DOOR_CLOSED").equals(true)
        && Platform.washerEmptySensor.getEntities().get("IS_EMPTY").equals(false)
        && Platform.washerSensor.getEntities().get("IS_WASHER_ON").equals(false)) {
                checkExecutabilityAndSend(new Command("start_washer"));
        }
}
```



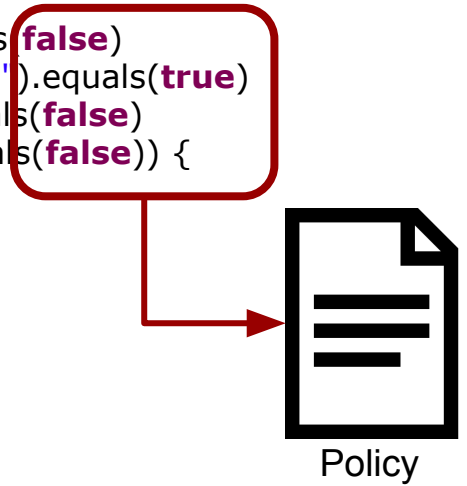
Verification: Sensors correspond to policy

```
public void update() {
        if (Platform.washerCleanSensor.getEntities().get("IS_CLEAN").equals(false)
        && Platform.washerDoorSensor.getEntities().get("IS_DOOR_CLOSED").equals(true)
        && Platform.washerEmptySensor.getEntities().get("IS_EMPTY").equals(false)
        && Platform.washerSensor.getEntities().get("IS_WASHER_ON").equals(false)) {
                checkExecutabilityAndSend(new Command("start_washer"));
        }
}
```



Verification: Correct sensor values

```
public void update() {
        if (Platform.washerCleanSensor.getEntities().get("IS_CLEAN").equals(false)
        && Platform.washerDoorSensor.getEntities().get("IS_DOOR_CLOSED").equals(true)
        && Platform.washerEmptySensor.getEntities().get("IS_EMPTY").equals(false)
        && Platform.washerSensor.getEntities().get("IS_WASHER_ON").equals(false)) {
                checkExecutabilityAndSend(new Command("start_washer"));
        }
}
```



Results

Based on three simulated smart systems:

- ❑ HVAC
- ❑ Garage door
- ❑ Laundry machine

14 evil and 3 good controllers

Type of errors	Yes/No
Command inside wrong method with/without correct policy check	✓
Command inside right method without any policy check	✓
Command inside right method with some/wrong policy check	✓
Sensor values changed in the clauses	✓
Controller checks values of wrong sensors	✓
Controller sends commands not in specification	✓
Commands sent in proper temporal order	X
Commands sent to only those dumb devices that can execute them	X
Wrong sensor values indirectly set in the code	X

Related work on security for smart homes

- Trivial security solutions such as strong passwords [1]
- Dynamic policy enforcement [2] causing system to stop if attack happens
- Crowdsourcing based solutions to create awareness about attacks [3]
  - Depends on the attacks published by individual organizations, no formal guarantee

References

[1] M. B. Barcena and C. Wueest. Insecurity in the internet of things. 2015

[2] J. Al-Muhtadi, M. Anand, M. D. Mickunas, and R. Campbell. Secure smart homes using jini and uiuc sesame. In Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference, pages 77–85. IEEE, 2000

[3] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion de-vices: Rethinking network security for the internet-of-things. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, HotNets-XIV, pages 5:1–5:7, New York, NY, USA, 2015. ACM

Conclusions & Future work

Verified security policies for a simulated smart home

- ❏ Designed and implemented a smart home simulation
- ❏ Defined a business model to assign roles to different organizations
- ❏ Defined and specified several policies for a smart home
- ❏ Implemented a static analysis to detect violations of the policies at compile time
- ❏ Evaluated our system on several malicious controllers

- ❏ Extend the verification to handle more complex properties
- ❏ Verify other security and correctness policies
- ❏ Evaluate our system on a real smart home

Example policy file

```
<controller name ="laundrymachinecontroller"
        <command name = "start_washer">
                <sensor name = "IS_DOOR_CLOSED" value ="true" class = "washerDoorSensor"/>
                <sensor name = "IS_EMPTY" value = "false" class = "washerEmptySensor"/>
                <sensor name = "IS_CLEAN" value = "false" class = "washerCleanSensor"/>
                <sensor name = "IS_WASHER_ON" value = "false" class = "washerSensor"/>
        </command>
        <command name = "stop_washer">
                <sensor name = "IS_CLEAN" value = "true" class = "washerCleanSensor"/>
                <sensor name = "IS_WASHER_ON" value = "true" class = "washerSensor"/>
        </command>
</controller>
```

References

[1] M. B. Barcena and C. Wueest. Insecurity in the internet of things. 2015

[2] J. Al-Muhtadi, M. Anand, M. D. Mickunas, and R. Campbell. Secure smart homes using jini and uiuc sesame. In Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference, pages 77–85. IEEE, 2000

[3] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, HotNets-XIV, pages 5:1–5:7, New York, NY, USA, 2015. ACM