IoTReplay: Troubleshooting COTS IoT Devices with Record and Replay

Kaiming Fang and Guanhua Yan
Binghamton University
IoT security

1. wide spread of IoT devices
2. limited capabilities for security testing
3. existing attacks: Mirai against Dyn's DNS service
Challenges for troubleshooting IoT devices

1. blackboxness of IoT devices
2. real world use
3. IoT device interacts with physical environment
Contribution

1. we identify contextual events for record & replay
2. we design a scalable architecture for IoTReplay
3. we implement IoTReplay with static and dynamic instrumentation
4. we evaluate the effectiveness of IoTReplay
System architecture
Contextual events in record & replay

1. IoT App: UI events, geolocation, sensor data
2. IoT Device: human or physical
3. network traffic: partial
Record for IoT App

Static Analysis

1. IoT app
2. Amandroid tool
3. Data dependence graph
4. Backtracking
5. System APIs to be hooked
6. VirtualXposed

Dynamic Instrumentation

1. Instrumented IoT App
2. Event logging
   - UI events stored at /dev/input
3. Execution on user's mobile device
   - Geolocation information
4. System API hooking
   - Sensor data
Replay for IoT App and devices

- Geolocation information and sensor data: xposed hooking
  - Geolocation APIs: `WifiManager.getScanResults`, `LocationManager.getGpsStatus` ...
  - Sensor APIs: `onSensorChanged(SensorEvent se)`

- UI events: translation between different phones
  - `ABS_MT_TRACKING_ID -1 --> BTN_TOUCH`

- Exotic network traffic: replay the payloads
Experiment setup

COTS IoT devices:

- Google Nest camera
- D-Link smart plug
- TCL Roku TV
- Tycam smart camera

Android devices:

- Android virtual device
- Samsung Galaxy S5
- Nexus 5

Android version 7.1 AOSP

Workstation PC: i7-9700, 32GB
Execution trajectory and similarity

- **trajectory**: sequences of methods, that have been invoked for every message received by IoT App
  - example: ['a()Z', 'c(Lh/v/b$a;)V', 'a(Z)V', 'b()J', 'a(FF)V']
  - in smali format

- **similarity**: edit distance, normalized to (0, 1)
Results: execution similarities

\( \theta = 0 \): avg similarity > 0.90

\( \theta = 1 \): avg similarity > 0.95

\( \theta = 2 \): avg similarity > 0.97
Results: attack record and replay

<table>
<thead>
<tr>
<th>IoT Device</th>
<th>Nmap scanner</th>
<th>IoTSeeker</th>
<th>IoT Inspector</th>
<th>Reboot attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Nest camera</td>
<td>Yes/Yes</td>
<td>Yes/Yes</td>
<td>No/No</td>
<td>-</td>
</tr>
<tr>
<td>D-Link smart plug</td>
<td>Yes/Yes</td>
<td>Yes/Yes</td>
<td>No/No</td>
<td>-</td>
</tr>
<tr>
<td>Roku TV</td>
<td>Yes/Yes</td>
<td>Yes/Yes</td>
<td>No/No</td>
<td>-</td>
</tr>
<tr>
<td>Ty cam LTE camera</td>
<td>Yes/Yes</td>
<td>Yes/Yes</td>
<td>No/No</td>
<td>Yes/Yes</td>
</tr>
</tbody>
</table>

**Table 3.** Reproducibility of external attacks. In each entry “a/b”, a and b give the test result of online and offline modes, respectively.

*online: record & replay in real-time*

*offline: record & replay in different time*
Results: performance

<table>
<thead>
<tr>
<th>IoT App</th>
<th>Average frame latency (w/o VirtualXposed)</th>
<th>Average frame latency (with VirtualXposed)</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Nest camera</td>
<td>11.43 ms</td>
<td>11.62 ms</td>
<td>1.66%</td>
</tr>
<tr>
<td>D-Link smart plug</td>
<td>10.67 ms</td>
<td>10.83 ms</td>
<td>1.50%</td>
</tr>
<tr>
<td>Roku TV</td>
<td>10.25 ms</td>
<td>10.31 ms</td>
<td>0.59%</td>
</tr>
<tr>
<td>Tycam LTE camera</td>
<td>12.58 ms</td>
<td>13.21 ms</td>
<td>5.01%</td>
</tr>
</tbody>
</table>

**Table 4.** Average frame generation time

performance overhead is negligible
Results: Scalability

One-to-multiple record & replay:

- one operational world
- multiple shadow world
Conclusion

1. We present the design, and implementation details of IoTReplay.
2. We perform extensive experiments using four types of COTS IoT devices.
3. IoTReplay results in good similarities and incurs negligible performance.
Thanks for watching