F1: Insufficient State Guard.

In the state machine of an IoT cloud (Figure 2a), when the cloud is working in state 4 (running), ideally it should only accept status upload requests (edge 6) or device unbinding requests (edge 3). Unfortunately, we found that the IoT cloud also accepts other requests. Depending on which request is accepted incorrectly when the IoT cloud is in state 4, we break down flaw F1 into three sub-flaws.

F1.1: An attacker, having all the device identify information, can use a phantom device to send a registration request to the cloud, which is fooled to return the corresponding device ID to the attacker (Figure 3F1.1).

F1.2: An attacker can use a phantom device to send a binding request that links the device (identified by device ID) with the attacker's account (Figure 3F1.2). the binding request is sent from the device and the cloud unconditionally accepts the binding request (see Section 2.3). As a result, the phantom device can bind the attacker's account to the victim device.

F1.3: The IoT cloud accepts device login requests even if it is in state 4. This flaw is a pre-condition of flaw F3 which we will describe shortly

F2:

Illegal State Combination. IoT cloud is in state 1, if the attacker remotely issues a request to register this device, the request is allowed and the cloud transfers to state 2. For the same reason, if the attacker continues to send a request to bind the device, the cloud accepts the request and transfers to state 4 (state 3 is skipped because a connection is still maintained with the victim device). At this time, if the attacker sends a control command to the device, the cloud will mistakenly forward the command to the retired device. This essentially causes a device hijacking attack.

F3:

Unauthorized Device Login. A connection is maintained between the device and the IoT cloud after device login. Ideally, the cloud should only allow a login request if the request is issued from the device that is bound with the owner account. However, the IoT cloud does not perform any account-based authorization check during device login. In other words, the connection is decoupled from the user account. Consequently, when the attacker uses a phantom device to login with the device ID of the victim device, the cloud is fooled into establishing a connection with the phantom device (Figure 3F3). As a necessary condition of this flaw, the cloud must accept device login requests in state 4, which is exactly what Flaw f1.3 states.

F4:

Unauthorized Device Unbinding. Ideally, only the user who holds an account currently bound to a device has the privilege to unbind the device. This is true if unbinding operations are conducted on mobile apps, which indeed include user credentials in unbinding messages.

Building Phantom Devices.

The behavior knowledge is obtained by reverseengineering the firmware extracted from real devices (XiaoMi and TP-LINK). In total, we implemented five kinds of phantom devices for Samsung, Joylink, Alink, XiaoMi and TP-LINK, respectively. They were implemented by 22, 17, 14, 61 and 72 lines of code (excluding SDK functions if any), respectively.

Network Configuration.

We placed the target devices and the phantom devices in two separate LANs behind NATenabled routers. As a result, the target devices and the phantom devices cannot communicate with each other directly. This setting resembles real-world scenarios where a remote attacker does not have access to the LAN of the victim.

Remote Device Substitution

In this subsection, how an attacker can remotely replace the victim's device with a phantom device under his control. To simplify the presentation, in the following, Alice to denote the victim/legitimate user and Trudy to denote the attacker. use sequence diagrams to represent the interactions among the three entities. An attack happens when Trudy interferes with the normal interaction diagrams.

Attack Workflow (Type I).

On the top of Figure 5 (above the highest dashed red line), the normal workflow of how Alice uses her IoT device on a Type I platform. After Alice provisions the device with a WiFi credential, the device sends its legitimacy credential and device identity information to the cloud to get registered (Step A.1). Based on the device identity information, the cloud registers the device with a device ID A and binds it to Alice's account (Step A.2). After the device logs in (Step A.3), Alice can control the device with her account. Then, the attacker, Trudy, kicks in as shown in the middle of the figure (between the two dashed red lines). She first lets the phantom device send the same device registration request as used in step A.2 to the cloud (Step T.1). Due to F1.1, the cloud accepts this request and registers the phantom device with the same device ID A, but still keeps device ID A bound with Alice. At this moment, Alice actually binds the phantom device and real device at the same time. Then Trudy could leverage the flaw F1.3 and F3 to log in a phantom device without Alice's account information (Step T.2). Since the phantom device has the same device ID as the real device, the cloud disconnects the original connection with the real device and establishes a new connection with the phantom device. However, when the real device does not receive the heartbeat message for a while, it automatically logs into the cloud again and puts the phantom device offline. Now, the real device and the phantom device are in fact competing for connection with the cloud. To win the competition, Trudy configures the phantom device to login very frequently. As a result, the phantom device always wins. Alice now still appears to "control" a device through her mobile app, although this device has actually been replaced by the phantom device under the control of Trudy.

Attack Workflow (Type II).

Similarly, the top part of Figure 6 (above the highest dashed red line) shows the normal workflow of how Alice uses her device on a Type II platform. After her mobile app sends her account information to the device (Step A.1), the device sends the binding request with device

ID and legitimacy information, as well as the account information to the cloud (Step A.2). The cloud binds the device ID A to Alice's account. After the device logs in the cloud (Step A.3), Alice can control/monitor the device with her mobile app. In the middle of the figure (between the two dashed red lines), Trudy launches the remote device substitution attack. Enabled by flaws F1.3 and F3, she lets the phantom device successfully log into the cloud with the same device ID (Step T.1). At this time, the device ID A is still bound to Alice's account. Like in the Type I platform, the phantom device maintains a connection with the cloud by periodically logging in. In this way, the attacker secretly substitutes Alice's device with a phantom device under her own control.

Remote Device Hijacking Trudy can further remotely control Alice's device by exploiting more flaws. We call this remote device hijacking attack.

Attack Workflow (Type I).

Continuing from Step T.2, device hijacking attack is depicted at the bottom of Figure 5 (below the lowest dashed red line). At this moment, the phantom device has already logged in the cloud. Due to flaw 4, Trudy is able to send a device-side unbinding request via her phantom device to the cloud (Step T.3), which puts the real device in dangling status (due to F2). Finally, Trudy binds the device with her account (Step T.4). As a consequence, Alice's device is connected with the cloud whereas Trudy is able to control the device on her smartphone**.**

Attack Workflow (Type II).

Unlike Type I platforms, to carry out a hijacking attack against Type II platforms, Trudy does not need to continue from the success of a remote substitution attack. Instead, she starts attacking from scratch, as depicted at the bottom part of Figure 6 (below the lowest dashed red line). Trudy starts by using her mobile app to send her account information to the phantom device (Step R.1). Next, the phantom device, which has Alice's device and other legitimacy information, will send a binding request with Trudy's account information (Step R.2) to the cloud. Due to F1.2, the cloud is fooled into accepting the binding request. Now, the ownership of the device has changed to Trudy from the view point of the cloud. The cloud then terminates the connection from the real device. However, the real device has re-connecting mechanism that continuously restores lost connection to the cloud. Due to F3, the cloud does not verify whether the account information (Alice) matches the current device owner (Trudy) or not. Therefore, the re-connecting request from the real device can be successful. At this point, Trudy could completely control Alice's device (Step R.3).