Vulnerability Report: WebOS Remote Root via ThinQ Login App

Reported by David Buchanan, 16th February 2021

Hardware Tested:

OLED55B9PLA TV, WebOS Version 04.71.04, GB/UK region.

I believe this also affects the latest releases of WebOS, across all products - although I have only been able to test on my own device, for now.

Description:

The WebOS system application "`com.webos.app.iot-thirdparty-login.app`" (referred to herein as the ThinQ Login App) has permission to access the Luna "private" bus. Because the application is web-based, it is possible to navigate to any website, by clicking links on webpages. Once the web view is navigated to an attacker-controlled domain, arbitrary JavaScript code can be run in the context of the app, including the ability to perform Luna IPC calls.

The arbitrary-file-write vulnerability in DownloadManager, publicly described here, can still be triggered via the Luna "private" bus. An attacker can overwrite/create system files, such as "`/var/luna/preferences/devmode_enabled`" (to enable developer mode), and then "`/media/cryptofs/apps/usr/palm/services/com.palmdts.devmode.service/start-devmode.sh`" - to execute arbitrary code as Root, at boot time.

This works even if the developer mode app is not yet installed, because DownloadManager will create directories if they do not already exist.

Impact:

With only an Infrared TV remote, an in-range attacker can remotely trigger arbitrary code to run with root privileges, on a persistent basis.

Exploitation:

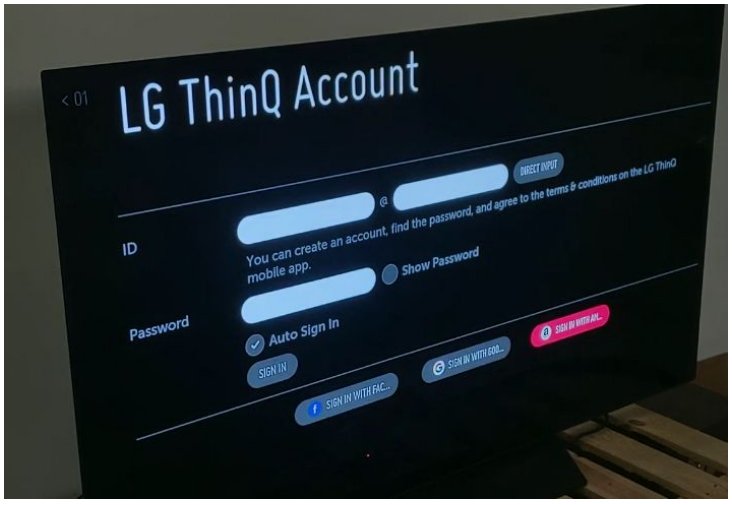
Please see the source files provided with this report - their function should be self-explanatory. "`index.html`" contains HTML and JavaScript code to perform Luna IPC commands. First, it force-enables developer mode, then it installs a custom "`start-devmode.sh`" script. Finally, it reboots the TV.

When the TV reboots, it will start executing the "`start-devmode.sh`" script as root, which will spawn a Toast notification as a Proof-of-Concept.

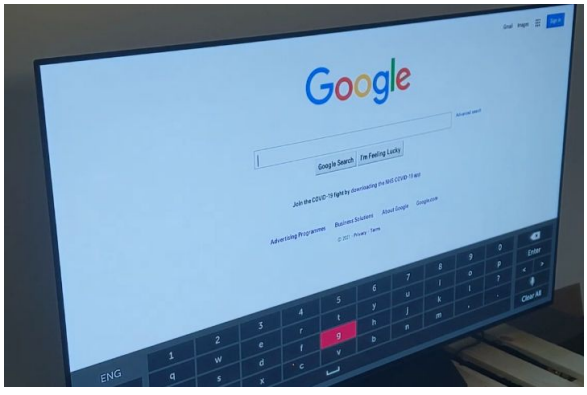
The exploit files must be hosted from a web server, so that the TV can access them.

Steps to Reproduce:

1. Start a web server, to host the exploit script. I used the python command "`python -m http.server`" to open a temporary web server on port 8000 of my local development machine.
2. Open the ThinQ Login App
    a. Navigate to the Home Dashboard.
    b. Click the Settings icon near the top right of the screen.
    c. Click "LG ThinQ Account"



3. Navigate to Google (NOTE: these sub-steps may be region-specific. I tested from a GB/UK region device)
    a. Click "Sign in with Amazon" near the lower right.
    b. Click "Conditions of Use", below the sign-in button.
    c. Click the country option for United Kingdom.
    d. Click "Mobile Apps" towards the lower right side of the navigation bar.
    e. Click the Google Play store logo.
    f. Click the menu icon (made of 9 small squares) in the top right corner.
    g. Click the Google Search logo in the menu that opens.



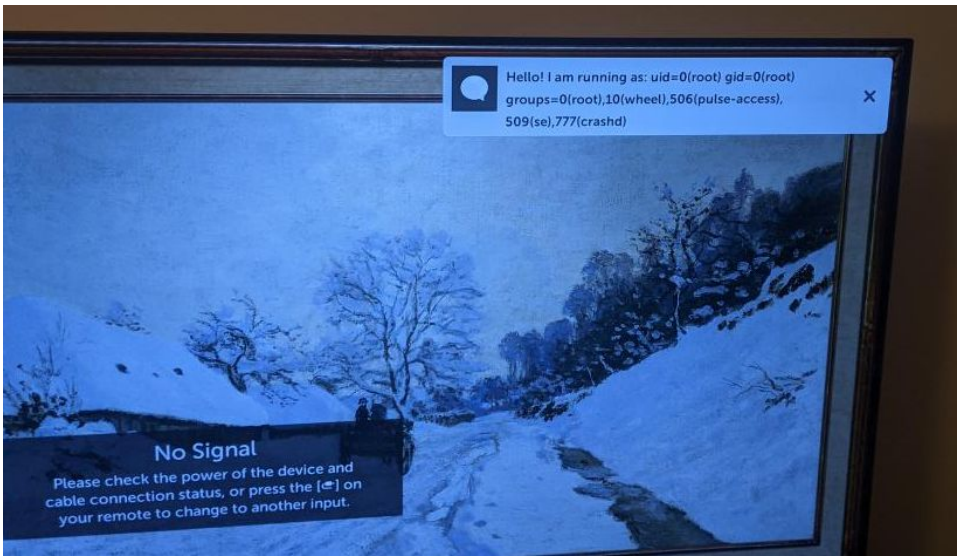
4. Open the PoC host
    a. Search "multilinkr" on Google

b. Click the result for https://multilinkr.com/
c. Type the URL of your PoC host web server into the main text box - In my case, http://192.168.0.26:8000/index.html
d. Click “Links” , then click on the newly created clickable link.

e. The TV should now be navigated to the PoC index.html page.

5. Trigger the exploit:
   a. Click the “Go!” button, and wait for the exploit to run

   b. Wait for the TV to reboot.
   c. After approximately 20 seconds, a toast notification should pop up, showing the output of the “`id`” command (demonstrating that code is running as root)



References:

1. https://blog.recurity-labs.com/2021-02-03/webOS_Pt1.html