Joel Koehler, John Martin, Daniel Rao

Week 10 Lab

Most Impactful Embedded OS Vulnerabilities and Breaches

Android has a vulnerability in the way it processes images. Images store RGB information and Android uses sRGB format. Prior to Android 11 there was no check for if an image followed the sRGB convention or not, and this is where the crux of the issue lies. If formatted correctly, the sum of R, G and B values should equal 255. However, if the sum is greater than 255 then there is a fatal error and the OS will crash. This happens because of a buffer overflow since the expected sum of RGB values *shouldn't* be greater than 255. Worse, if a corrupted image is set as the wallpaper, the phone will go into an infinite bootloop since the image will be processed upon booting. This vulnerability was exploited in a small-scale attack where a corrupted image was sent around to android devices as a free wallpaper. When the image was set as the wallpaper, the device would be sent into a fatal bootloop. While this vulnerability was not exploited to the fullest potential, it was still detrimental to many users (the exact number is not known).
https://www.androidauthority.com/android-wallpaper-crash-1124577/

Another Embedded OS vulnerability was with the 2015 Jeep Cherokee. There was a zero-day vulnerability with Jeeps that gave an attacker "wireless control, via the Internet, to any of thousands of vehicles. Their code is an automaker's nightmare: software that lets hackers send commands through the Jeep's entertainment system to its dashboard functions, steering, brakes, and transmission". (https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/) While this also

had a daily low impact, the implications of vehicles being remotely commandeered by attackers while on the road are not.

Another Embedded OS vulnerability is the iOS 1970 bug. In 2016 a video was uploaded to youtube showing a user setting the phone's year to 1970 and the phone getting stuck in bootloop upon rebooting. This bug was induced manually through the system UI and, while this bug has since been patched by Apple via the system UI, the bug persists. This is a problem since iOS devices check NTP to sync their clocks with the local time, and malicious wifi networks were set up to change the date remotely. This has since been patched by Apple. (https://www.smh.com.au/technology/apple-1970-glitch-has-been-automated-can-brick-devices-through-wifi-20160413-go4xl8.html)