Journal B - Lyell Read

1/27/2020

I read the RISKS forum Volume 31 issue 2, from Jan 11, 2019. From that, I read an article on “Alexa really is a spy”. Firstly, I noticed that the title indicates a tone of surprise, something that shocked me. The idea that someone would be surprised that a networked microphone in their house is listening and spying on you is laughable. This alone hints at a mentality that will be changing surrounding CS development in the future. In the past, it was a viable business strategy to assume that the customer is ignorant and will not assume that this network connected microphone (designed to recognize speech too) is listening to their every word. The same applies to all other smart devices, from Nest thermostats to fire alarms - they’re all using their sensors to collect information about you.

This paradigm is likely to change in the future as the consumer will slowly become more aware of the intrusive and privacy-violating methods these companies will use to collect data about you. Instead, either software will have to be developed in a manner that keeps these features more discrete (so that the end user and researchers cannot tell that the device is behaving maliciously), or exclude this monitoring technology from their software, something that will make these larger advertising companies take a large monetary hit. Either way, I expect we will see changes in this field.

1/29/2020

For this entry, I looked at the RISKS V31 article about escalating ug bounties for iOS. This marks a trend of whose future I am uncertain. Bug bounties are an interesting thing - they are meant to provide hackers a legal and positive alternative to the sometimes disastrous effects of selling 0-days (more generally exploits) on the black market). Their value would ideally be more than the black market can pay, but because the black market distributors can resell this exploit many times and make exorbitant sums on each transaction, this will never be the case. Instead they will always be lower than the black market. Interestingly, I believe that as the value of these on the white market gets higher, their value underground also increases.

When it comes to the applications to cybersecurity and software engineering, this hike in prices indicates that for each bug that someone can develop a POC exploit for, Apple loses $2 million. This all lies on one single developer who initially programmed that flaw. This could dissuade programmers from working on projects like iOS, as if the cost of this bounty were correlated with him, he could be viewed as a net loss for the company.

2/1/2020

The RISKS V31 article about “Hackers Target Chromecast Devices, Smart TVs With PewDiePie Message” indicates a trend about modern devices and modern campaigns where if someone wants a message out, they can get it out. In this case, the message was not politically loaded (more of a ‘meme’ or ‘prank’), but it still made it to reportedly 2700 TV’s. Whether legal or illegal, the more support that you have, the more likely someone will possess the knowledge to distribute your message in less-than-optimal ways like this.

2/5/2020

The Iowa Caucus just happened a couple days ago, and it was a total disaster. They used an app for tallying votes (very bad idea) and further issues with their opsec could have allowed someone with the app to enter data (as some pins were leaked on twitter). To add insult to injury, the same candidate that announced (prematurely) a win was also a benefactor of the company that built the app. This all leads me to consider the future of how code will be composed: unless we, as the consumer and the person impacted, speak up and make sure that code meets standards, these ‘epic fails’ will continue. We will never have election software that dates back many too many years, and infrastructure that is not properly programmed.

2/10/2020

There is no source for this entry, it is more a thought piece about code size as time goes on. It is already established that the codebases for projects like Word (Microsoft) and similar tools keep growing, and rarely shrink. Because of this, I guess that code sizes on projects will keep increasing, and so will the total codebases for those tools. The latter is true as (especially in languages without powerful builtin macros, such as JavaScript) more and more packages are required (each with their own dependency web). Each of these packages increases the code base size of the application, rendering it harder to understand and maintain. To be portable, such applications need to be packaged with sometimes thousands of dependencies. Therefore, over time, codebases will get larger and larger.