Journal C - Lyell Read

2/15/2020

We have been contemplating our final project and specifically the implementation that we will have to fo soon and we are thinking about CI. CI (Continuous Integration) is where you check your code at each predefined point in the development cycle to ensure that it is functioning correctly and that it will work. This is important because when you merge changes into the master branch (or whatever is set to be the default branch) then the changes need to work before they make it to 'production' if that is how the master branch is being used. To do this, we use CI, which checks the code before a merge request.

2/16/2020

Cont'd from yesterday:

How will this process of using CI evolve? It will have to become more complex and more abstract. Right now, and certainly in its infancy, it was centered around small checks, maybe a script that a dev wrote a while ago, maybe something off-hand. As project codebases get larger, and the complexity of these things increases, these CI checks will need to be more all-encompassing. Further, these will have to start taking into account the use cases. Where they would have just (previously) checked that something runs on one architecture, it will be critical to test these things in context of other parts, other pieces of the final system. Further, it will be necessary to develop more comprehensive checks of the software - checking for the things that get typically left into code accidentally (API Keys, Passwords, Junk Scripts, ...).

2/18/2020

As an idea that I just had on this topic, I propose a codebase limiting fuzzer. This is run as CI. What it does is it fuzzes the software to determine exactly what parts of the codebase (once compiled, which it also does) are needed for functioning, and retains those otherwise, it ditches the rest.

This could be paired with a fuzzer that checks for vulnerabilities (typical fuzzer usage) and this could improve the condition of the final product. These two could run at the same time.

There might also need to be a location where users can install a tool that reports what applications are installed, and what libraries and versions these are all running. This can be used to generate a map of what things need to be able to (a) cooperatively interact, (b) simultaneously but separately run on a system. This can be broken down by system and frequency. In this way, developers can check their code for compatibility with their user base, rather than general CI tests.

2/22/2020

I have noticed a trend for hardware to replace software. This trend has both upsides and downsides. Over time, I see this becoming more widespread as people rarely care about performance, and in terms of performance, I see this as an absolute win. However, when it comes to security these features are hard to make 100% secure, and when there is a vulnerability in such things, it is often impossible to repair, instead the chip is permanently insecure. Therefore, as a security guy, I see this as a dangerous tradeoff in the pursuit of processor or MCU speed. However as a consumer that knows that laptops and desktops need to be continually faster to cope with the trends, I see this as something that is inevitable. Next, we will only have software performing certain cutting edge functions before that function gets offloaded to hardware in the next generation of processors or MCUs.

2/24/2020

I have an interesting view on cybersecurity and privacy that I have recently learned more about. I went to a talk about cryptography and the government. As cryptography becomes stronger, the government is increasingly seeking ways to install backdoors, or all out get around the new encryption, as they cannot efficiently crack it. This leads to a divisive state of privacy, where the individual is free to do as they please, while the government cannot wiretap or listen to the user. This makes the government upset, as they claim they use citizen’s data to ‘prevent terrorism’ and the likes. However, I much prefer it with good encryption - that is the spirit of freedom that this nation is founded on, and if you want pervasive surveillance, I’d recommend moving to China.

How does this rant relate to the future of software, though? This is closely tied with how the world will decide on whether to go for privacy or big brother surveillance. I think that in light of the features provided by lacking cryptography, or by the usage of apps that sell analytics to the government, we will make this cutting edge crypto obsolete, as the information that the government is after will come from other sources such as through apps like FaceBook, who will sell this to the government, and completely bypass the need for cracking encryption. This is not only bad for the people who are being spied on, but it is also bad for the goal of tracking down terrorists - they will use services that do not sell data, those with encryption. Therefore, this is a lose-lose, and it is also the situation I see us ending up in as users keep joining social media.

3/3/2020

I would like to touch back on our project now that we have started it. We have been putting together code and my specific task (in pair programming, of course) was to design the server of our server/client model. This basic HTTP server is notable as it is only HTTP (no SSL) and it notes that this server should not be used as a production server when it is started. This means that it is neither safe, nor is it ready to handle the requests of thousands of users.

As small startups begin to bec0me more and more prevalent, I suspect that we will see many MVP (minimum viable products) that are built in this way, as there is no time to spare for the frilly bits at the start (“time is money”). This will make it so that, if nobody notices (or the lead engineer moves companies after the MVP is created), then the final product might still (for example) be in debug mode, or be vulnerable, because that’s how their product has always been. Therefore, I think that with the MVP ideas of most startups, we might see code in the future that is less clean, less formulated, more sketched out.