Adriana Donkers

Dr. Johnson

CMSI 401: Software Engineering Lab

7 October 2020

Assignment 1

The WIRED article "A Cyberattack on Garmin Disrupted More Than Workouts," written by Lily Hay Newman, highlights the ransomware attack that hit Garmin earlier this year and temporarily caused a disruption in numerous services across the company. Garmin is a technology company that offers products for fitness tracking and services for navigating airplanes for pilots. When the attack hit the company, services that were affected included the cloud platform called Garmin Connect which syncs user activity data and parts of the Garmin website. As a result of the ransomware, athletes were unable to record their workouts and runs, and pilots were unable to use the position, navigation, and timing services for their airplanes. Corporate email systems, customer call centers, and many other portions of Garmin were affected by the attack, disrupting the usual workday for Garmin employees and preventing Garmins users who rely on Garmin products from continuing with their usual routines. For example, outages in the Gamin pilot apps and hardware affected flight-planning mechanisms and the ability to update mandatory Federal Aviation Administration aeronautical databases. Pilots were unable to download database updates, which is a requirement from the FAA in order to fly. So, many airplanes were temporarily grounded because of this issue. Some other difficulties pilots had were when planning and scheduling flights through the Garmin apps. Garmin Pilot makes it easier to file a flight plan from an iPad, however the outage caused pilots to go through a more troublesome approach through the FAA website or by making a phone call. Thus, these caused major problems across Garmin and really affected the aviation industry that were using Garmin for their flights. Ransomware is something which affects not only the aviation industry or Garmin apps, but any system which will cause major failures when the services go down because these targets are more likely to pay up a larger sum of money quickly to restore them.

Ransomware attacks could happen to any company and they will likely happen to any application where it causes a lot of loss to the owners when it goes down. When building software applications, it is important to build strong security measures in order to protect the systems from such dangerous attacks. In software, one way we can protect against such attacks is by testing the systems for all edge cases. In the software development lifecycle, there is a phase where developers are supposed to test their products. Now, there are different approaches to testing. Some software developers believe that the best way to prevent vulnerabilities in one's code is to practice what is called Test Driven Development (TDD). In TDD, the developer is writing tests that will check for the functionality of the developer's code before or as the developer writes the code. I think this will help prevent breaking the system or application if there are tests written and thought of throughout the process. It might not necessarily prevent ransomware because tests are typically functionality-focused and not security-focused. However, it does eliminate a way in which outages can occur and disrupt use from the users.

An important takeaway from the Garmin attack is that it is important to always have a fallback mechanism or a way in which the products can continue running safely even if there is a disruption. When a developer considers ways in which the software can fail from user interaction, then it protects the system and ensures that at least systematically, the service will always be available for the user. One way in which this connects with my portion of the senior capstone project is that when I build the chatbot for the new version of Coco, I want to make sure that the chatbot has a fallback mechanism. If the chatbot doesn't understand user input, it can still continue functioning and try to ask questions to the user to understand their intent. By developing this as a feature, I will be able to prevent a way in which the application could fail.

In addition to having a great testing practice in software development and considering fallback mechanisms for unexpected input, the attack on Garmin implies that an important part of making sure that the systems themselves are safe and secure from such vulnerabilities. Software developers often reference or use packages and dependencies in their code. It is important that the code that is written for all services and applications are able to adjust as needed when there are new updates. Using an older version of an operating system or of a package when an update is released could potentially leave the application susceptible to attacks or vulnerabilities that are fixed in the new updates. Being cautious about the versioning can prevent ransomware from happening either through someone's computer software or through the application.

Another security preventative idea for developers is thinking about where you are keeping your code for your products. If you are using a version control platform such as a GitHub repository, it's important that developers do not put any information on there publicly that could potentially cause someone to break into sensitive information. For my senior capstone project, our team has already discussed this idea so that we do not commit anything into our GitHub repository if it is private information, such as passwords or keys for the database. Especially since our repository is publicly accessible, we would not want anyone who isn't a contributor to the project to have access to that information and disrupt anything. Especially after reading about the Garmin attack, I recognize the implications of ignoring these security measures.

The Garmin ransomware attack in July presents many concerns around security and implications about thinking thoroughly when trying to build a stable platform for a user. In the case of software development, there are ways developers can build a stable platform by incorporating fallback mechanisms, encouraging the practice of writing tests, keeping software and packages updated to the newest releases, and protecting sensitive data when using a public platform for your code. These are all important to consider as a software developer in light of the lessons learned from the Garmin event.

Link to Article:

A Cyberattack on Garmin Disrupted More Than Workouts