Prospectus

Designing an Admin Dashboard for an Anti-Money Laundering Platform

(Technical Project)

Technological Momentum of Computer Science Interviews

(STS Topic)

By

Quentin Bishop

23 November 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: *Quentin Bishop*

Approved: _____ Date _____

Benjamin Laugelli, Department of Engineering and Society

Approved: _____ Date _____

Daniel Graham, Department of Computer Science

Introduction

In recent years, an explosion of cyber crimes has taken the public by storm. From countless classified government documents being hacked and posted on Wikileaks (Barlow 2017) to millions of stolen credit card numbers being listed for sale on the dark web (Finklea, 2017), it seems cyber crimes have become omnipresent in our society. While a heightened emphasis on the importance of cybersecurity has accompanied the rise in cyber crimes, advances in protecting against cyber crimes have failed to keep up with cyber criminals. In many ways, simple probability favors cyber criminals - those working to keep a system secure must think of every possible vulnerability in advance and work to protect against them while those attempting to exploit a system only need to find one vulnerability that has not been protected against. This often results in a cat and mouse game in which cyber criminals continue finding new vulnerabilities and creative exploits while cybersecurity professionals rush to fix these issues ("Industries Lagging in Cybersecurity", 2017).

However, the problem is multi-faceted and involves a solution more complex than simply hiring more "good guys" to outnumber and outthink the "bad guys." Today's failure in preventing cyber crimes is in part due to both a lack of intelligent software to detect cyber crimes and a failure throughout the entire tech sector to hire enough competent cybersecurity specialists. Those designing systems need to be on the cutting edge of technology and ten steps ahead instead of playing catch up to criminals who continuously breach and go unnoticed. Robust automated software that utilizes artificial intelligence (AI) to detect all sorts of threats will be paramount in the ongoing war against all sorts of cyber crimes (Wolff, 2020). In order to help identify financial crimes, I propose a technical project in which I will describe my experience building an automated money-laundering detection system at a financial institution. This

program automatically looks through millions of financial transactions using sophisticated machine learning technologies and strategies to alert bank employees of suspicious activity. While detecting money-laundering helps combat current criminal activity, it is also essential to design future online financial platforms to be more secure and reliable in order to prevent other possible criminal activity. Increasing the number of competent software engineers and programmers working in cybersecurity is an essential step in preventing cyber crimes; current hiring techniques throughout the tech industry are antiquated and place unnecessary emphasis on speed and efficiency with little regard to the security implications. In order to address the lack of emphasis on cybersecurity throughout the tech sector, I propose an STS project in which I will explore and analyze the failure of industry-wide conventions for hiring applicants using a technological momentum framework. I will explore how the origins of today's software engineering hiring process are from a time in which speed and efficiency were of greater importance than security and analyze how said hiring process gained momentum over time and has ultimately become detrimental to the software engineering industry today.

Technical Problem

Last summer I participated in a software engineering internship at a financial institution in which I worked on an anti-money laundering team to develop an application that helps detect and protect against financial crimes. The application was already under development when I arrived, but my team was tasked with implementing additional features before the application was to be deployed. The existing application harnessed AI to detect suspicious activity and flag suspected fraudulent transitions to open an investigation. For my internship, I worked to create and test an admin module in which administrative users could view and manage investigations.

In order to manage a complex platform that relies both on algorithms and investigators, I created an administrator dashboard that shows analytics and relevant information about investigators and investigations such as how many investigations have been completed in a certain time frame, how many investigations an investigator has completed in a certain time frame, how many transactions have been flagged, and other data that helps in managing the platform. Previously, analytics for the platform needed to be generated manually and were only done so once a year because of the extensive work required. The admin dashboard I built alleviates the issue of manually generating analytics and reports regarding investigations by continuously and automatically generating live analytics on a dashboard that platform administrators can view and access at any time.

Throughout my internship, I worked with many tools including the Go programming language, Git for collaboration and code management, AWS serverless functions, Agile development methodology, and many DevOps tools. While all of these tools are fairly new, I was surprised to learn that many of these tools are already considered an industry "norm" and that they are prevalent in almost any software engineering job since I had not encountered them as part of my computer science (CS) education (U.C. San Francisco, 2021). With many of these tools being used at almost every tech company, it would make sense to learn these technologies at school. I was also surprised by the strong emphasis on writing secure code and thoroughly testing all components of a project; for my CS assignments at UVA I had probably spend 95% of my time developing code and 5% of my time testing whereas at my internship I had spent around 50% of my time developing code and 50% of my time testing it. While studying CS at UVA had certainly given me a strong background and helped build my intuition in problem-solving, it was

clear to me that there was a strong mismatch between the skills and tools that were taught in class and those used in industry.

Having taken part in the pilot program while at UVA, I found Software Development Essentials (SDE) to be the most useful course in preparing me for my internship experience. Of all CS classes offered at UVA, SDE was by far the most applicable to my internship as it taught many essential skills that are commonplace throughout the tech industry such as Git, database management, and Agile development (University of Virginia, 2018). While SDE was certainly useful, the issue arises from the fact that it is the only CS class I have taken at UVA that teaches material that is essential and currently being used throughout the software engineering industry. Additionally, the UVA CS curriculum does not place enough emphasis on security or testing. Students can opt to take a cybersecurity class on its own, such as in Intro To Cybersecurity, but those classes are not graduation requirements and many students never take them (University of Virginia, n.d.). Additionally, the existing cybersecurity classes at UVA tend to place more emphasis on offensive cybersecurity techniques and exploiting vulnerabilities whereas the industry is reliant on software engineers having a strong grip on writing secure code and mitigating risks.

In my opinion, the UVA CS curriculum should be improved by teaching newer industry-applicable skills such as Git, DevOps, and AWS early on in a student's course of study so that students have a strong foundational knowledge of industry-prevalent tools and practices and do not need to play catch-up, potential creating security vulnerabilities, when heading into internships or jobs. Additionally, the UVA CS curriculum should be improved by integrating cybersecurity as a theme throughout all CS classes so that students are cognizant of how their choices in designing software affect the security and safety of their projects.

STS Problem

The computer science and software engineering hiring processes remain one of the most stagnant and detrimental components of the industry as a whole. In the early days of computer science, career options were far more limited than they are today. Before the emergence of the internet, computer scientists would usually choose one of three career paths: developing specialized code and algorithms for a three-letter agency or the military, researching advanced computer science in academia, or working for a specialized tech company like IBM. The average company did not need computer scientists and the idea of self-employment would be a fantasy. On a separate note, the physical technology available at the time was far more limited than it is today. The amount of storage and processing power we have on our phones today would have been unthinkable just a few decades ago. While we can easily buy a 10 gigabyte flash drive today for around $10, the same amount of storage in 1980 would cost upwards of $400,000 (University of Missouri–St. Louis, 2012). Since storage was so expensive, computer scientists at the time needed to cut down on costs and placed a strong emphasis on making code as efficient as possible, which means minimizing the memory and storage needed to perform a task. Since all of the career options available at the time relied on a strong knowledge of algorithms and the efficiency of an algorithm was paramount in keeping costs low, the hiring process for new computer scientists was designed to test an applicant's ability to develop efficient algorithms.

As the tech industry exploded in the early 2000s and every company wanted an online presence, the demand for computer scientists skyrocketed. As new companies entered the tech space, they continued using the existing hiring process that tests an applicant's ability to develop efficient algorithms to solve a problem. While a strong understanding of algorithms and

efficiency are still essential throughout many parts of the tech industry, there are many new components like graphics, design, networks, and cybersecurity that are also important in today's world. However, the antiquated hiring process means that nearly all candidates in all areas of computer science are selected based on their ability to develop efficient algorithms (North Carolina State University, 2020). This mismatch leads to nonsensical situations such as a graphic designer for a website being chosen based on their ability to invert a binary tree, or in non-CS terms, hiring an artist based on their SAT math score.

To analyze this phenomenon, I will utilize the framework of Technological Momentum. This framework states that as a technological system gains momentum and popularity over time, it transitions from being shaped by society to becoming the shaper of society (Hughes, 2009). Technological Momentum is essentially a combination of both Technological Determinism, which states technology influences society, and the Social Construction of Technology (SCOT) , which states society influences technology. Technological Momentum incorporates both of these frameworks by observing the transition from SCOT to Technological Determinism over a period of time instead of offering a static analysis. I will analyze how the CS hiring process was initially shaped by the needs of society, but gained momentum over time and eventually came to influence the curriculum of universities and shape the landscape of the tech sector. In doing so, I seek to better understand the flaws in the current system and how they can be mitigated or changed altogether.

Conclusion

To better prepare computer science students in building a better and more secure future, I propose a two-part solution consisting of 1) suggestions for changes to the CS curriculum to

better reflect skills and tools currently being used in industry 2) an analysis of the momentum of the CS hiring process over time. Understanding the mismatch between what is taught in universities and what is commonplace in industry as well as the history that has led up to this mismatch will provide a strong background for fixing the problem and stopping the increasing gap between what a computer scientist needs to know during their hiring process and their job.

References

Barlow, R. (2017, January 13). *Why Julian Assange and WikiLeaks Aren't Heroes*. Boston University. Retrieved October 31, 2021, from https://www.bu.edu/articles/2017/julian-assange-wikileaks-not-heroes/

Finklea, K. (2017, March). *Dark Web*. Congressional Research Service. https://biotech.law.lsu.edu/blog/R44101.pdf

Industries Lagging in Cyber Security. (2017, December 21). Maryville Online. https://online.maryville.edu/blog/industries-still-lagging-behind-in-cyber-security/

Hughes, T. P. (2009). Technological momentum. In Johnson, D. G. & Wetmore, J. M. (Eds.), Technology and Society: Building our Sociotechnical Future, 141-149.

North Carolina State University. (2020, November 11). *Tech sector job interviews assess anxiety, not software skills*. North Carolina State University College of Engineering. Retrieved October 31, 2021, from https://www.engr.ncsu.edu/news/2020/11/11/tech-sector-job-interviews-assess-anxiety-not-software-skills-2/

UC San Francisco. (2021). *Cultivating a DevOps Practice to Support Innovation*. UCSF Technology. Retrieved October 31, 2021, from https://tech.ucsf.edu/devops

University of Missouri–St. Louis. (2012). *Moore's Law*. University of Missouri–St. Louis Information Theory. Retrieved October 31, 2021, from https://www.umsl.edu/~siegelj/information_theory/projects/Bajramovic/www.umsl.edu/_abdcf/Cs4890/link1.html

University of Virginia. (2018). *UVA CS 2020 Curriculum*. University of Virginia Computer Science. Retrieved October 31, 2021, from http://pilot.cs.virginia.edu/category/courses

University of Virginia. (n.d.). *Bachelor of Computer Science (BSCS) Requirements*. University of Virginia Computer Science. https://engineering.virginia.edu/sites/default/files/common/departments/computer-science/files/2-%20BSCS%20Info%20and%20Curriculum%202017%20ws.pdf

Wolff, J. (2020, June 8). *How to improve cybersecurity for artificial intelligence*. Brookings. Retrieved October 31, 2021, from https://www.brookings.edu/research/how-to-improve-cybersecurity-for-artificial-intelligence/