Final Paper:

My Experience as a Cloud Developer Intern at ODU

David Allen

School of Cybersecurity, Old Dominion University

CYSE 368: Cybersecurity Internship

Professor Teresa Duvall

December 1, 2025

Fall 2025

Employer: James Tate

Company: Old Dominion University (ODU)

Table Of Contents

Introduction

When graduation was getting closer, I wanted to be able to get a little work experience in my field. One of the best ways to do this was to get an internship. I want to be able to pad as much experience as possible to increase my chances of getting hired. I already have an Associate's degree in Information Systems Technology and two CompTIA certifications, Network+ and Security+. A problem I encountered when searching for my internship is that I wasn't hearing back from anyone I applied to. I tried to contact them, but I never got any responses. I would just wait for a rejection email eventually, which is all I ever got back. Before the Fall semester started, I reached out to the Cybersecurity Internship Director, Teresa Duvall. I was told about an internship available for a cloud developer at Old Dominion University. Although I do have an interest in programming, it wasn't my greatest skill. I thought that this could be a great opportunity to improve my programming skills and gain work experience, so I decided to become a cloud developer at ODU. While working at ODU, learning objectives I hoped to achieve are code security fundamentals working with real-world projects, collaboration with other IT professionals, and web development and operations practices.

When starting my internship, I basically had two onboarding sessions, one with my supervisor about my position, and another with staff from my division, Digital Transformation and Technology (DTT). In my onboarding session with my supervisor, James, we went over information related to our team in web development. All information is stored in the ITS Application Development Notebook on OneNote. It covers documentation about our code repositories and other helpful links for developers. Our team uses GitLab to develop and maintain our code for each repository, with documentation available in both GitLab and OneNote, but OneNote is where documentation is primarily stored. The main application that the

cloud team is responsible for is the MyODU portal. It's used for students and staff to be able to access their school information, like course information, financial aid, email, and other school activities or personal information. Before I was able to get started, I had to have a staff account with an email and access to the school VPN. Without VPN access, I wouldn't be able to access the code repositories on GitLab to perform my tasks. Hours are highly flexible in the internship and for staff. We are able to make-up time throughout the week by working extra hours a day if we need a day off. The internship is 10 hours on average a week, but we can work under or over these hours accordingly. I work 15 hours typically to make sure I'm getting enough time towards my total internship hours. Most of the code used in the cloud development team is a form of JavaScript, like React or Node. The front end contains HTML, CSS, and other forms of JavaScript as well. NodeJS and PHP are on the back end of the servers. Other DevOps technologies mentioned include containers, such as Docker and Kubernetes.

In the second onboarding meeting, I got to be able to learn more about ODU as a whole and each department in it, along with a little history on the school. ODU was originally a division of William & Mary, eventually breaking off into its own university. It has many ties to different military divisions over the years, such as establishment of the Aircraft Instruments Institute. ODU has had a history of enhancing distance learning, such as founding TELETECHNET to provide satellite learning for those overseas in the Navy. ODUOnline was eventually rolled out to replace the former satellite system and provide an online experience. ODUOnline was rebranded as ODUGlobal to promote their learning experience to a variety of markets worldwide. Recently, the Division of Digital Learning and Office of Information Technology Services merged to create the new Division of Digital Transformation and Technology. General policies were covered, such as maintaining professional and academic

integrity, being respectful and responsible for our actions, and following service standards to assist customers, including other students and staff. Strategic planning, discrimination policies, and the student code of conduct were also discussed to promote a safe and secure environment for ODU and the ability to achieve goals at a local and global scale. Office supplies can be requested as needed from our supervisors, as needed.

The beginning of my internship and onboarding started off good. There was a little time when I had to wait for my laptop and staff account to be created, which slowed me down a little in the beginning. In the meantime, I was given some self-learning opportunities to get started. James told me to look over the Git documentation to become familiar with how Git works. I was also told to review JavaScript syntax, since I was dealing with a lot of JavaScript when performing tasks. I already enjoyed being a student at ODU, as they have an excellent cybersecurity program with a wide variety of classes to take. I feel like it was only natural that I would enjoy becoming an intern here at ODU.

Management Environment

For my internship, supervision was lenient, although progress of my work was given daily as I had questions about a task. When given a task, I would post screenshots or explain any issues I had with my supervisor in Microsoft Teams. While most communication was performed in Teams, I also had meetings with him over Zoom when it was more appropriate. For example, when there was something visually that I needed to look at, we would conduct a meeting through Zoom to demonstrate what the next task would be. He’s always busy with meetings throughout the day, which is understandable as a cloud architect that is the leader of a team.

There's a lot of discussions with members from other teams and vendors on how certain technology should be incorporated. During bi-weekly sprint reviews, there's typically a lot of discussion about how other team members are implementing their own parts to a certain project, sometimes with a demonstration. It's a good way to collaborate and give feedback on how platforms or code can be implemented with a project.

I believe supervision overall is well done. There are plenty of meetings, although they are not always required as an intern, where progress is monitored, along with text communication through Teams. James was always there to help whenever I had a question, or I ran into issues. He gave good recommendations whenever I had a problem with something. For example, I was having trouble installing a program called NPM, which came as part of NodeJS. I told him about the issues I was having, but while I waited for responses, I also looked things up because other people might have the same issue as mine. When he responds back, he always gives me great answers, depending on the information I've provided. I try to send as much information as possible to resolve issues. We eventually resolved the issue once he explained how to change environment variable on my computer correctly after I had them misconfigured. James was very informative, and I would recommend him to anyone as a great supervisor for my internship.

Major Work Duties, Assignments, and Projects

Most of my major work duties involved running security scans. I was tasked in primarily performing SAST scans, or Static Analysis Security Testing scans. I would take a copy of the code repository into my own personal workspace or make a branch within GitLab. Branching in GitLab makes it easier to avoid any compatibility issues that could arise working in a forked repository, which acts as a completely separate project that isn't entirely connected to the

original. I would have to edit a configuration file in the root of the repository. This would then scan the code within the repository and results were output into a text file. This allowed me to look at potential vulnerabilities involving static code in the repository. Once I had done that, I was tasked with finding out the vulnerabilities that were considered false positives. This allowed me to determine what vulnerabilities were real and which ones were fake. If I saw something interesting or had a question about a listed vulnerability, I would message James about it and go from there. The main takeaway when searching for false positives is that I had to check and see what's hardcoded, or if a variable had a set value that doesn't have to be entered by the user. It all depended on what it was exactly, like a manipulative URL for example. Remediation involved removing variables within the code, like in an XSS vulnerability I discovered, that also outputs an unnecessarily redundant response for the user, more about that later. But, if there weren't any ways to fix the code, the scanner could've been confused by something in the code by mistake. An easy fix for this was to exclude the line from scans by adding a comment to that line. There were a lot of lines I had to go through in multiple repositories, so it took me about 2 weeks to go through each repository. One interesting find was an outdated MD5 hash that had to be updated, so I let James know about that and they would resolve that once compatibilities were fixed with the hash algorithm.

I also got to learn how development is coordinated within the team on GitLab. Earlier I mentioned an XSS vulnerability that had to be fixed. I was told to create a merge request to resolve this issue. This involved creating a branch within our develop branch on the cloud portal repository, where all changes are made to the code. When I create a separate branch for this issue, it allows me to make changes separate from the original code. Changes can only be made within a branch to prevent a random person from deleting the actual code in the repository or

making unauthorized changes. Branches make it easier for others to review code before it's implemented into the repository. I had to go into my branch and locate the line where the vulnerability was located. At first, I didn't know how to solve this issue and deleted the block associated with that line, but it would actually break the code as certain other functions required that block. There were only two things that needed to be changed: removing a variable and changing the output for the user. I deleted the variable associated with user input, to avoid a user from manipulating a URL that might lead to other nefarious actions in the cloud portal. After that, I changed the output to not include the fact a user made a request, as this is already known by the user. The output now states the URL is invalid. Once changes were made, the merge request in my own branch was pushed back into the develop branch for approval. This was actually brought up during the sprint review as a demonstration to make others aware and give me more experience with sprint reviews. I had to explain to other team members about the issue, alongside James, and we made others aware of this potential vulnerability, along with the necessary fixes, for approval into the main code repository.

Another major assignment I performed also involved scanning, but this time I used a software package called NPM. NPM is used to scan for vulnerabilities based on current software dependencies for the code, such as React, which a lot of the code is based on so that the web applications can behave properly. While the previous SAST scanning involved a lot of manual intervention to fix errors, NPM involves automatic fixes. Sometimes manual intervention is required, but vulnerabilities can be patched by using an NPM audit fix or NPM update command to update outdated dependencies. When manual intervention is required, it's because dependencies are conflicting with each other. Information about dependencies are usually found

in a package file, where each version of a dependency is listed. The package file can be edited to update each version to resolve conflicts.

The last of my major projects was learning about Amazon Web Services (AWS). My supervisor was able to help us out in learning more about services used on AWS. AWS hosts a variety of services for web applications running on the cloud. The whole idea behind it is serverless computing and Infrastructure as Code (IaC). It enables organizations to utilize applications without requiring functioning servers, since everything is being handled by Amazon to develop and secure code. First, I had to understand the difference between accounts and users in the context of AWS. Accounts are the actual projects being worked on within AWS, which are the cloud portal and a non-production version of the cloud portal. Users are the actual users on the account or working within the projects.

Some services provided on AWS include EC2, IAM, and CloudWatch. EC2 is a service that allows servers to be created to host web applications, which brings us back to serverless computing. EC2 launches virtual servers in instances. Like many services on AWS, they can cost money. When running services, we were told to close them if they cost any money, although some are free. For EC2, servers are ran using Amazon Machine Images (AMIs). This includes operating systems, an application server, and the applications that need to be ran in the instance. Some available operating systems include an Amazon Linux distribution, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. Network settings can be configured, such as web traffic allowed on the server instance. This allows developed applications to be tested in a separate and secure environment utilizing a virtual server.

Going over the other services briefly, IAM stands for Identity and Access Management. This allows management over resources provided on AWS. Here users and groups can be

managed with various policies. AWS provides its own policies, but policies can also be created by users if given permission. For example, a resource might only be allowed for users in certain regions. If a user is in another region, then they will not be able to have access to certain. This ensures users have access to the required resources and makes sure data integrity is maintained by only allowing authorized users. CloudWatch is used for monitoring and alerts. Alarms can be set up to monitor resources and send out alerts, such as billing alerts when resources reach a certain dollar threshold. Logs can be found to troubleshoot issues, such as logging in to specific resources. Log anomaly detectors can be created, which is something I actually did, because I didn't have a lot of access within AWS. It took about 24 hours to create the log anomaly detector, although nothing was being scanned, probably because nothing was being used for that specific log group.

Use of Cybersecurity Skills and Knowledge

Before I was an intern, I thought I already knew a lot about cybersecurity. I already learned a little bit about web application attacks and tools. For example, I was already familiar with what an XSS attack was and how to mitigate it. XSS stands for cross-site scripting. It's when an attacker tries to manipulate input to retrieve information through a URL form. Scripts can be executed in the URL to steal data from the user, like session cookies, or impersonate a trusted user. XSS attacks can be considered persistent or non-persistent attacks. Non-persistent attacks are attacks that are not part of the webpage, usually sent in the form of a link by the attacker and running a script on their website that can harvest information about a victim. Persistent attacks are stored on the actual web server, where users might click on a link on an actual website and a malicious script can run and attack the user, in some cases multiple users.

This can be mitigated by validating variables in the code and making sure the code is sanitized. A user should not be able to inject malicious scripts if untrusted user input is cleaned.

Working as an intern allowed me to expand my previous knowledge as cybersecurity student into a role as a cloud developer. It allowed me to take my knowledge, apply it, and learn more than I was taught in a classroom setting. For example, I mentioned that I was familiar with attacks. Another type of attack that I was familiar with was a Denial of Service (DoS) attack. During my internship, I learned about different types of DoS attacks that were never mentioned to me before. For example, I learned about Regular Expression DoS (REDoS) attacks. These attacks are related to the size of an input value, usually repeated values, such as a user typing in the same letter multiple times, followed by a random different letter. In algorithms, multiple paths are taken when input is provided, but in the event of REDoS, there can be a loop based on the repeated values given. When being processed, if an input value fails to find a path, backtracking occurs to the previous value where the loop occurs. This can cause a regular expression to work very slowly for a web application and give the attacker the opportunity to perform other types of attacks.

Curriculum Preparation

I believe the curriculum here at ODU has provided me with a great introduction to what I may encounter in the real world. When performing tasks in my internship, I'm able to take my knowledge from my courses and apply it to new tasks that I was not previously familiar with. For example, this semester I am taking Ethical Hacking and Penetration Testing. One topic that was discussed in this course was malware analysis. While discussing malware analysis, multiple types were mentioned, such as static analysis, dynamic analysis, memory analysis, and hybrid

analysis. Static analysis analyzes the code without executing it, dynamic analysis analyzes the malware in a controlled environment to observe its behavior, memory analysis analyzes system memory during execution, and hybrid analysis is a combination of both static and dynamic analysis to gain a better understanding of how malware behaves.

One lab assignment from this course required us to download malware in a compressed folder and upload it to a sandbox platform for testing on a system. We downloaded two different types of malwares, one related to botnets and another related to keylogging. In this sandbox, both are executed to analyze their behavior. When executing the behavior of the botnet, I noticed that multiple different Ips were being requested and that they all lead back to one central IP address. This type of malware takes control of the victim's computer and makes it part of the botnet, with the central IP address being a command-and-control server that runs the botnet. For the second type of malware, links pointed to a single IP and multiple files were executed. Based on the behavior of this malware, files were created in the startup directory and personal data was being stolen, along with credentials found in a web browser, although I wasn't actually typing in anything personal or private. It was detected after I randomly started typing on my keyboard, which is how the sandbox detected the keylogger software.

Although I did not specifically perform malware analysis in my internship, I did have to analyze a lot of code. Instead of analyzing malware, I analyzed web applications utilizing static analysis. Another student in the internship was responsible for dynamic analysis and integrated OWASP ZAP into GitLab and generated a report of their own. I was primarily focused on my own analysis of the web applications tested, but both of us made our supervisor aware of changes we made and any signs of new vulnerabilities when found. When combining both static and

dynamic analysis together, we can provide our supervisor and other team members with the most information possible to gain a better understanding of the development of our web applications.

Another course that I'd like to mention was beneficial, although I already knew a lot of the content before I took this course was Cybersecurity Fundamentals. This course is meant as an introductory course for computer science students in the field of cybersecurity. Cybersecurity Fundamentals is a senior level course that discusses networking and security protocols. The course then starts discussing attacks and various mitigation methods for each attack. Attacks discussed include Distributed Denial of Service (DDoS), XSS, SQL injection, and many other attacks. This course is very descriptive with its content and goes in-depth about protocols and cyber-attacks. Many examples of attacks are given. For example, since this course has a technical focus for computer science students, many examples of attacks using different types of code are given, such as XSS using JavaScript. While this course felt like a lot of review, it helped reinforce what I already knew to prepare me for my internship. This review allowed me to dive deeper, such as learning about other types of DDoS attacks, like REDoS attacks.

Internship Fulfillment

Throughout my internship I was able to accomplish many learning objectives. The first learning objective I accomplished was the ability to collaborate with other IT professionals. I enjoyed having discussions with other team members during my time as cloud developer intern. I was able to participate in daily stand-up meetings with other professionals that support ODU's web applications. In our daily stand-up meetings, we discussed what we accomplished the day before, what we plan on doing today, and is there anything blocking us from accomplishing our plans. The meetings are very short. They only take about five minutes. Sometimes other team

members might be busy and can't always make it. There's a lot of meetings with other vendors, such as Google, that require collaboration based on the project that's being worked on. Some team members work internationally and are busy doing other things based on their time zone, so they join whenever they can. When it comes to developing web applications, I've noticed some team members work on certain aspects based on the programming language they are familiar with. One team member might have the task of connecting databases with the web application, so they work with SQL. Another team member might handle authentication or hosting the web application on a server, so they handle PHP related tasks. There's a lot of coding involved, and since there are multiple team members, it can be beneficial to split up tasks based on the code being used, just to avoid any confusions with syntax. It's easier to just focus on one type of code rather than multiple types at once when given a task.

I also participated in sprint reviews. In sprint reviews, we discuss what we liked, what we lacked, and what we learned based on the topic of discussion. They are great for everyone to come together and collaborate on the current project, providing different perspectives and insights into the tasks being completed. Team members might come up with demonstrations to show off what they've completed. This allows the team to evaluate how a web application will behave when performing actions within the web application. For example, we had one demonstration where a team member was working on a database. They showed off how information is pulled from the database and certain results were displayed, basically student records. Some team members provided input about this function, asking if certain inputs are allowed based on student ID numbers. They recommended allowing student ID numbers into a form without the zeros in the front to make it easier for advisors when searching for students in the database.

I was also able to have my own demonstration during one of the sprint sessions. While performing a SAST scan, I came across a XSS vulnerability in one of the repositories. I brought it up to my supervisor. When it came time to the next sprint meeting, he said it would be alright to demonstrate the vulnerability I found in the repository. I explained the issue I found was a XSS vulnerability and that I provided remediation that needed approval. Other team members were offered questions, in which there were no questions. Once I made this aware to the team, it was then approved in GitLab as a merge request and pushed into the develop branch before final changes are made to the rest of the code.

Motivating Aspects

When I started my internship, I was tasked with setting up SAST scans. There were motivating, discouraging, and challenging aspects to this, but for this section, I'll discuss the motivating aspects of this. After I set up the scanner, I had to analyze the scans for false positives. Many false positives related to these scans included validated input. Variables were hardcoded into the web application and didn't pose a threat, as a user wouldn't be able to steal sensitive information via SQL injection or XSS. I felt like as I was analyzing these scans, it did seem monotonous, but made more sense as I was doing it in the process. I felt more inclined to do it as I kept going through each scan, making it enjoyable.

Another part of this internship that motivated me was being able to participate in a team. This one aligns with my goal of working with other IT professionals, but being in meetings makes me feel welcome as part of the team. I was able to take accountability in my work like the rest of the team. When I take accountability in my work, I have a great sense of pride that makes me want to continue. It makes me believe that I can potentially work in the field of cybersecurity

as a cloud developer. Other staff members were also helpful during my internship. I was having trouble with installing some software on my computer. I contacted ITS about the issue and they were very helpful in resolving the situation. They were able to set up an appointment and connect to my computer locally. I needed administrative approval to download the software that was required to do my job.

Discouraging Aspects

While this job was highly motivating for me, there were also some discouraging aspects of it as well. One of the first discouraging aspects of the job was trying to resolve software dependency vulnerabilities using NPM. NPM is a software package that is included in Node.JS that allows users to scan their own code for vulnerabilities based on the installed dependencies. I was having a lot of problems when I was trying to run NPM commands on the code repositories I was allowed to use. First, I use NPM audit to check how many vulnerabilities there are for the repository. Next, I use NPM update to update vulnerable and outdated software dependencies. I ran into a lot of errors when trying to update the outdated errors. One thing I noticed when updating errors was that NPM would update some dependencies to a deprecated dependency with a warning. I tried NPM audit fix, but there were always conflicting dependency errors. Software dependencies are compatible with certain dependencies, otherwise they will not work properly. This was definitely the hardest thing I tried doing. While it was great that some of these dependencies were able to be updated to reduce the amount of vulnerabilities in the repository, some fixes can introduce breaking changes or there wasn't a fix available at the time. The repositories I used were old, so it's probably why NPM wouldn't work. For another repository, I ran into errors that Python needed to be installed on my computer and NPM had to have it configured. I tried NPM update again. After that, I received errors that Microsoft Visual Studio

(VS) needed to be installed. After I was able to install VS on my computer, I received a lot of compiling errors that I had no idea how to solve. I knew that I wasn't going to be able to solve every issue right away, so I resolved what I could and moved on.

Another problem I encountered was power outages. I was trying to do research on how to resolve the NPM errors I was receiving. This was the morning of the Amazon outage, although it did not impact me for my internship except for accessing Canvas the entire day. The bigger problem was that I had two separate power outages where I lived. My internship is usually from 9 a.m. to 12 p.m. The first power outage occurred around 10 a.m. I was looking at articles, and I thought I had found a solution for one of the NPM errors I was having, and the next thing I know, I'm sitting in the dark. I was hopeful, because it came back on after a few minutes. I continued to work and moved on to utilize the solution I had found. I began typing a command in NPM. Now, around 10:45 a.m., I was sitting in the dark again. At this point, I was a little annoyed. Whenever I tried to get something done, I physically couldn't do my job like I wanted. I thought maybe it wasn't meant to be today, so I decided to take a break for the rest of the day. I would've done schoolwork like I wanted to, but Canvas was down. The day after was a much better day as I continued to work on resolving my NPM issues without any interruptions.

I also had issues trying to run a copy of my Student360, a web app, repositories locally. One repository was the frontend, the other was the backend. My plan was to run a server for it locally off my computer and to test it behaviorally after the NPM changes I made. I did use an NPM command that resolved some vulnerabilities and introduced breaking changes, so it was only right to test it and see what these changes were. Turns out it was more complicated than I originally thought. When I hosted the repository off my computer, it only took me to the actual files of the repository from the Internet. It didn't go according to how I planned it. I tried some

other solutions, but I did not receive the result I was looking for. I was suggested by my supervisor to try to run it using Docker, which is what I'm currently trying as I write this. So far, I'm not having any luck because NPM is still an underlying issue. I've been considering using an alternative to NPM to deal with these dependency issues, such as Yarn, but I'm unsure if it would work with the repositories. I would need to install it and learn how it works compared to NPM.

Challenging Aspects

My internship has brought me some challenges. These challenges have been great experience for me to enhance my experience working in cybersecurity. They were not the easiest tasks I have done or understood, but they were not the hardest either. One task that I thought was challenging was setting up my SAST scans in GitLab. When I first started my internship, I had to get familiar with Git and GitLab before I could even implement vulnerability scans into GitLab. I was given the Git documentation and read the first few chapters to get myself with how Git works. Once I accomplished that, I was given the task of setting up SAST scans in GitLab. To me, this task was a challenge because I had never used any version of Git. I've heard of GitHub and how people use it to store their own code and share it with others publicly. Our school uses GitLab to store their own code. We recently migrated from the Community Edition to the Enterprise Edition to support our authentication needs. As for the authentication scanner, I had to be familiar with JavaScript and GitLab, a form of bash scripting, syntax. I have to configure a file that is able to run this scanner in the root directory. The file was in a YAML format, known as .gitlab-ci.yml. After I configured this file, I created a pipeline to run the job. The job was in four stages. I added an extra security module that would perform the scan using Semgrep, a free open-source scanner built-in with GitLab. I found a video that helped me create this, since I didn't know how to set this up. I used my own forked copy of the directory to perform SAST

without interfering with the main code repository. I also had the option to create a branch which works within the code repository but in my own personal workspace where I can merge changes into the personal branch, once changes are approved by an administrator. There are four stages that are ran within the pipeline: build, security, test, deploy. The build stage builds the pipeline. The security stage performs security testing of the repository. The test phase tests the pipeline. The deploy stage deploys the pipeline. Once the pipeline runs and jobs are finished, the pipeline is displayed as successful. If a stage fails, which it has, the pipeline will show up as failed and not run. One situation I had to actually create a branch due to tagging issues with a forked repository. My pipeline would fail in a forked repository, so I had created a separate branch and performed my changes. For the security stage, all potential vulnerabilities were displayed in a separate JSON file. After that, I was able to analyze the vulnerabilities and provide changes as needed.

Another challenging task I had to accomplish was learning how to use Docker. Docker is a way to turn applications into containers and run the application. Docker is a separate environment. Containers are a way to isolate applications from the rest of the network and develop code securely. Once your application is developed, you can perform testing using the isolated container running off an open port. To start, I did a tutorial on how to set up a Docker container. I used a Docker command to build a container using a local code repository. I hosted the container on port 8080. To access the front end of the server, I would click on the local host port link to access the application. That part was fairly easy with the example application that helped me understand how Docker works. What I was tasked with was to run a front end and back end using Docker. I ran into a lot of problems when trying to build a Docker container for both the Student360 front end and back end. First, I was still having issues with NPM as the

Docker configuration file required using NPM. I had to install dependencies for the container. At first, I didn't realize that local code repositories and Docker containers were separate. I tried running NPM commands locally and while building the Docker configuration file. It resulted in different results. When I ran an NPM install command locally, I did get a compiler error, but not an error that NPM required Python. It was the other way around when I tried to build a Docker configuration file, where I didn't receive any compiling errors, just that Python needed to be installed. When I received this error, I went and searched for a solution on the Internet. Once I found my solution, I needed to add a line of code to the Docker configuration file that installs Python to the Docker container. Another issue I had was using an NPM build command. What I found out about this NPM command is that I had to remove it from the Docker configuration file as it was not needed. When I first ran the back end, I had an error where a module wasn't found. All I had to do to fix this was to add the 'app/serve' directory into the repository, since it was required to serve the application. Once I resolved these issues, I was able to successfully build a container for both Student360's front end and back end. Currently, when I try to access the front end of the application, I get a 404 error. I am actively trying to resolve this issue. I am assuming it has something to do with the way my Docker configuration file is set up. I will continue to look more into that in the coming weeks as I get closer to the end of my internship. As an update from this, I am currently trying out a different repository since the one I was previously trying to run is too old with many conflicting dependencies. I am working on running the ODU cloud portal project using Docker. I am running into some of the same errors, such as directories being displayed instead of the application running within Docker. I changed the working directory within the configuration file, but that just displays the directory in Docker when connecting to the port. I changed the serve command to serve an output of the necessary files, but that just

gives me a blank page. I am actively changing things in the configuration file to be able to display the application correctly.

Recommendations for Future Interns

There are many things I will recommend for future interns if they decide to become a cloud developer intern here at ODU. The number one thing I would recommend is knowing about programming. You have to be able to read JavaScript and some other forms of code, like PHP, which isn't that much if I'm honest. It's mostly JavaScript. You have to be able to know how it works within an application and what behaviors are being performed by the code. JavaScript is responsible for things like clicking on buttons, so every developer should know how to read and write in the given language. Knowing how to debug code helps with knowing program knowledge. If you are familiar with reading and writing in code, then you should be able to fix the code, although it's not always as easy as it sounds Another recommendation is to learn how to use Git. Git and its many forms are used to store code. We used GitLab and a little bit of GitHub for outside projects. When working with a team, there are many functions within GitLab that I had to know. I had to know how to properly perform a merge request and provide a certain level of detail based on the changes I made.

Docker was also useful for the tasks I performed. Docker allows developers to deploy their applications to see how they perform. Once deployed, developers can interact with the page from a port on the container connecting to the local host port on their computer. The most challenging aspect when using Docker was setting up a configuration file. It's easy with the example project, but complex with large-scale projects, like the ones in this internship. Many of these projects utilized different types of code that interacted with each other in different ways.

With this in mind, there's a lot that goes together to make a web application function. Getting an application to work in one environment might be entirely different from getting it to work in another. It's a trial-and-error process that requires a lot of patience. If you are not patient, you might struggle, so learning how to be patient helps. Don't expect instant results. Technology isn't always perfect. This includes the designers of the application, as bugs are bound to happen. It's always good to learn the best coding practices and study up on programming or any new technology for that matter, especially in cybersecurity. Things become outdated all the time. There might be a new software dependency available, so it would be good to update outdated dependencies, if compatible. An easier tool to compile code might be available, so there's always something new to learn and expand your skill set. Be motivated to learn. If you don't, you will get left behind as new technology is being created and implemented at work.

Conclusion

Overall, I enjoyed my time during my internship at ODU. I learned a lot about what it takes to become a cloud developer. It's definitely not an easy job, or my first choice, but I took a chance on myself. I thought maybe I could be good at this even though programming was one of my weaker skills. I found it an opportunity for me. I can get my hands on actual projects someone would work on as a cloud developer. I would be able to interact with Git and learn about code repositories. I was able to update outdated dependencies for the repositories I worked on. I learned how to deploy an application using Docker. I got to interact with other IT professionals and learn about what they do on-the-job, which was honestly a lot of meetings, but many other things too. Everyone was helpful to me when I asked for assistance. For example, if there was an issue with a program, I contacted ITS. It was a great opportunity to learn, and I wouldn't have it any other way.

As for the remainder of my time at ODU, this is my last semester. Anything I learn in the internship, I can relate it back to other courses I take. If I have anything to showcase that I learned from the internship, it can be highly beneficial for other classmates to understand the topics. I could learn something in my class and then apply it to my internship. A great example of these experiences was learning about code analysis in both my internship and in Ethical Hacking and Penetration Testing. Although malware analysis is what was covered in the course, I learned the theoretical aspects of analyzing code and was able to apply it to my internship by practicing static analysis. This helped me further understand the importance of performing analysis as a cloud developer.

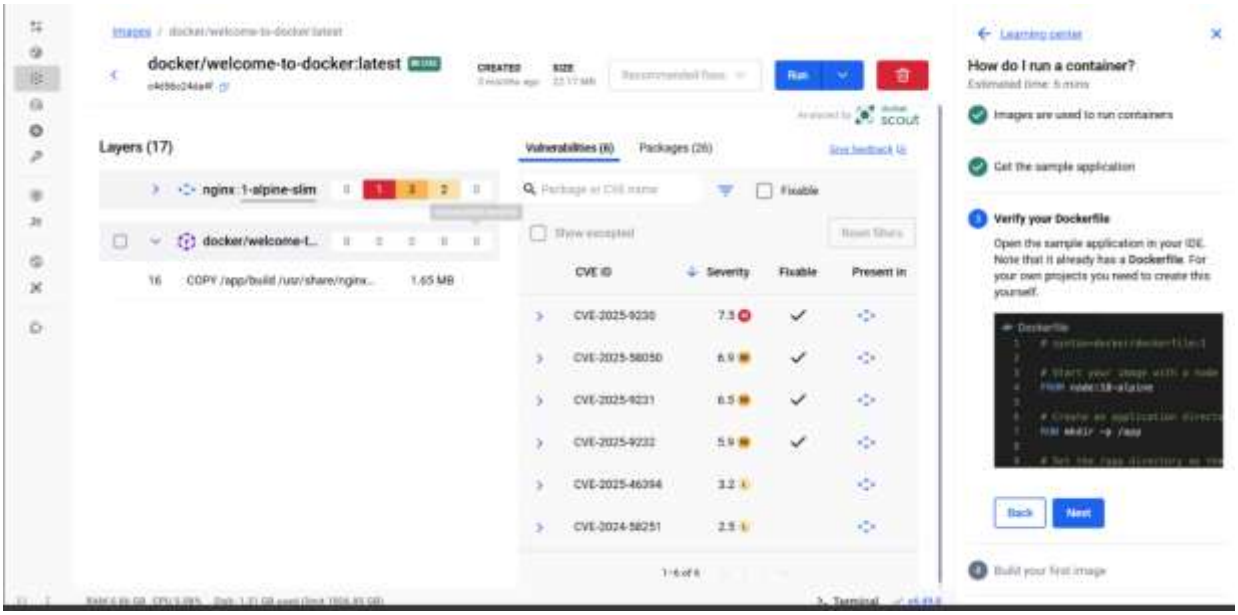
As for my future, after I graduate, I will consider applying to developer positions now that I have experience as a cloud developer. This internship has had the chance to open the door for more opportunities. It doesn't even have to be a programming role. An employer might just find it interesting, and it gets me picked for an interview. This has also encouraged me to learn more code in the future. If I want to learn more about programming, I will need to learn as much about coding that I can. I might actually get back into coding in my free time when I get a chance. I used to do coding exercises on my phone using an app. I found it very entertaining to learn code. The application was called SoloLearn. It's great for learning the basics, and even more advanced topics. It offers many languages, like Python, PHP, and JavaScript. If I had a choice in favorite programming languages, it would be Python, although this internship has had me more interested in JavaScript, so I would learn more about that as well. This internship has made me more interested in programming and enthusiastic about my potential future in the field as a cloud developer. Hopefully, I get a chance to actually become a cloud developer at some point in my career.

Appendix

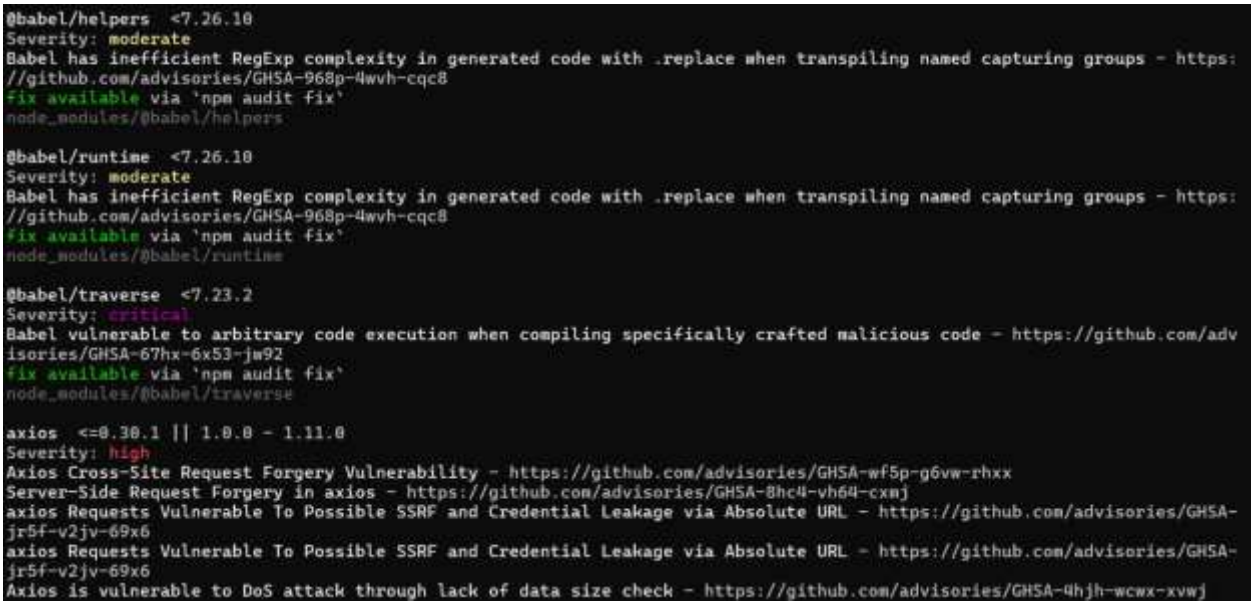
```
47
48  sast:
49      stage: security
50      artifacts:
51        paths:
52          - gl-sast-report.json
53        untracked: false
54        when: on_success
55        access: all
56        expire_in: 30 days
57
```

Setting up SAST in GitLab with a stage

Results of some of the scans in GitLab



Docker container security scan



Running NPM scans to check for vulnerabilities

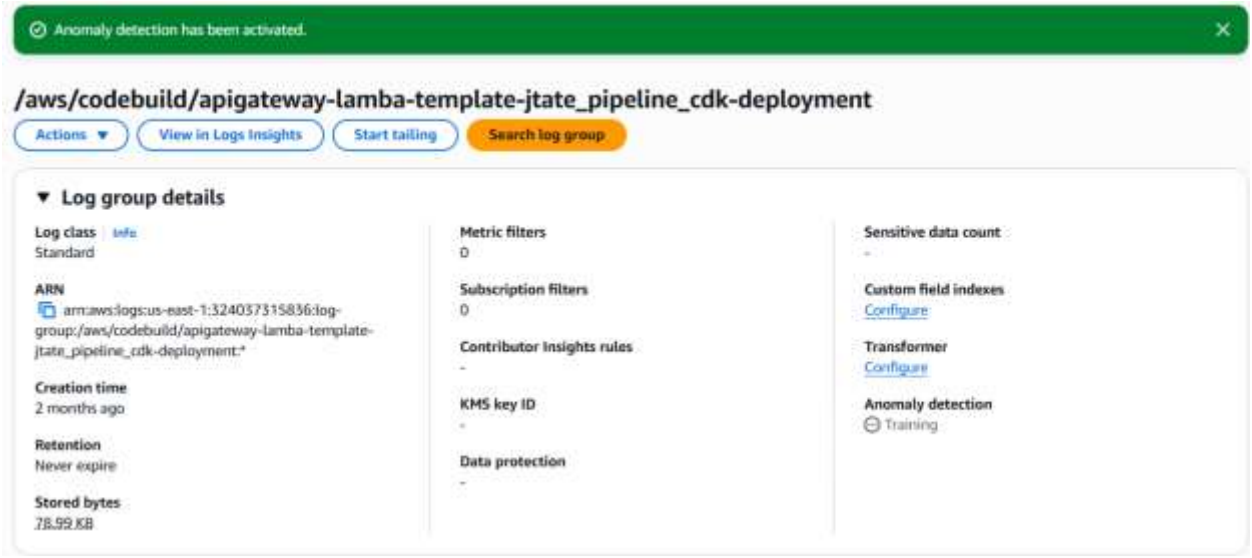
Dockerfile.txt

```
# Start your image with a node base image
FROM node:22-alpine

# The /app directory should act as the main application directory
WORKDIR /app

# Copy the app package and package-lock.json file
COPY package*.json ./

# Copy local directories to the current local directory of our docker image (/app)
COPY . .
COPY ./public ./public

EXPOSE 3000

# Start the app using serve command
CMD [ "serve", "-s", "build" ]
```

Docker configuration file template

```
View build details: docker-desktop:///dashboard/build/desktop-linux/desktop-linux/bnqmp4ov4rmnfexmfzq9ta2c8
PS C:\Users\dalle042\Repository\student-success-web-service-personal-test> docker build -f Dockerfile.txt -t student-success-test .
[+] Building 24.3s (12/12) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile.txt                                                          0.0s
 => => transferring dockerfile: 538B                                                                              0.0s
 => [internal] load metadata for docker.io/library/node:22-alpine                                                 0.6s
 => [internal] load .dockerignore                                                                                 0.0s
 => => transferring context: 2B                                                                                   0.0s
 => [1/7] FROM docker.io/library/node:22-alpine@sha256:b2358485e3e33bc3a33114d2b1bdb18cdbe4df01bd2b257198eb51beb  0.0s
 => => resolve docker.io/library/node:22-alpine@sha256:b2358485e3e33bc3a33114d2b1bdb18cdbe4df01bd2b257198eb51beb  0.0s
 => [internal] load build context                                                                                 0.0s
 => => transferring context: 22.04kB                                                                              0.0s
 => CACHED [2/7] WORKDIR /app                                                                                     0.0s
 => CACHED [3/7] COPY package*.json ./                                                                            0.0s
 => CACHED [4/7] RUN apk add --update --no-cache python3 build-base gcc && ln -sf /usr/bin/python3 /usr/bin/pyth  0.0s
 => CACHED [5/7] RUN npm install --legacy-peer-deps                                                               0.0s
 => [6/7] COPY . .                                                                                                0.2s
 => [7/7] RUN npm install -g serve                                                                                4.6s
 => exporting to image                                                                                           18.6s
 => => exporting layers                                                                                          12.6s
 => => exporting manifest sha256:f50b585b2877bbe0fc6f329f2b6692681154624dd224d3d02c58b4b1458dfcce                  0.0s
 => => exporting config sha256:5742f575d7307f17122bcca9305feddfb61f2ce3b336e0ce2528a41398e39f61                   0.0s
 => => exporting attestation manifest sha256:5d67107fb8214ad48e7fa041b3b779896e7c29ba97ffa76a77434238c444aaf1     0.0s
 => => exporting manifest list sha256:22d35508e2d2db43758171b721265421fad1237a9fd42160f2e692880c3eac22            0.0s
 => => naming to docker.io/library/student-success-test:latest                                                    0.0s
 => => unpacking to docker.io/library/student-success-test:latest                                                 5.9s
```

Creating Docker builds using the Docker configuration file



Setting up log anomaly detection in AWS