DateEscape: The Innovative New Way to Escape a Commitment

Jeffrey M. Dinackus

Northern Virginia Community College

DateEscape is the automated way to have a friend give you a phone call during a date so you can duck out with an excuse. It uses Python to take information and make Twilio API requests on behalf of the user to send text messages and calls directly to the user's phone. The call will play a recorded message from the user and the texts can be anything that is text. The result is a much more convenient way to automate the receiving of calls and texts. It is also highly customizable; API requests can be sent seconds or minutes apart to simulate a real interaction with some modification to the script. DateEscape can be used for any interaction if you know you may want to escape in advance or if can use your phone during the interaction. It relies on AWS EC2 for hosting the server that runs the Python script, after it is cloned from GitHub. This can be obtained for free by using an AWS free tier eligible t.2 micro instance and the script instructions are configured for Ubuntu. The user will also need an API key from Twilio and a Twilio phone number which can be obtained for free. The user is also reccomended to save the Twilio phone number as a contact in their phone before the running of the script adding realism as the calls and texts will say the name of anything they enter! DateEscape can be found for free on my GitHub (Jeffrey Dinackus, 2023).

There are 3 main scripts used by DateEscape, initial-setup.py, test-connection.py, and make-a-call.py. Initial setup takes in the variables needed from the user, such as his or her phone number, his or her Twilio phone number, and API keys then it saves them to individual files. Then, it prints them so the user can check his or her inputs. Test-connection.py sends a text and a call to the user using his or her input. The user is free to edit his or her information anytime by changing the values in his or her files. If the user receives them, he or she is good to go. Make-a-call.py reads the inputs and a creates a timer which counts down to when it is supposed to call based on user input. This is where the API requests are made. I did a lot of refactoring

throughout the project to make the project cleaner and more readable. For example, instead of having two API call sections, one for if the user enters “now” to be called now, and one for timed calls, combined into one with a function. This will make modifications to the script (the order or number of calls and texts) much less confusing for the user.

During development, the services I used were AWS EC2 (also required for running the program) and AWS CodeGuru Security (“Find Expensive Code – Amazon CodeGuru Security – AWS,” n.d.). The total time developing was at least 40 hours, as this project was originally a hackathon project and I worked almost nonstop on it for 2 days (Jeffrey Dinackus & Trombo, 2023). Post hackathon, the development was probably 10 more hours due to design, refactoring, documentation, and testing. Some major challenges include the referencing of the user information and API keys, which were originally desired to be environmental variables per the hackathon rules. Python had a hard time reading the environmental variables and later I decided that it was a better user experience for the user to be able to open a file and simply edit the information. The timer was a major hinderance in the original DateEscape, called QuickEscape. Due to the short time period of the hackathon, I was not able to complete it in time, it is fully working for the new project. The Python library time operates via seconds, but the input is minutes and hours, meaning a conversion was necessary. The original project also had to poll a MongoDB Atlas DB to receive the information needed to make the calls. The original script was continually polling the Atlas DB for new events that would take place within 10 seconds then activate. This was far from ideal. The reason for this was that there was a React front end which the user entered their information into. This front end has been completely scrapped in the current version for simplicity. The current setup for this project also isn’t ideal. It requires the creation of a Twilio account, verifying a phone number with Twilio, receiving a Twilio Phone

number, and probably some troubleshooting to get it all to work. I have done the heavy lifting of writing the API calls and structure. In the original deployment we would have had to charge the user for the API requests they make or require them to enter their API keys every time. In the current deployment, the user can run it for free or very cheap by getting his or her own API keys. I have written documentation to ease this. It is a one-time setup, after you get it working you can run it many times as long as you keep the instance running or save the script between uses. The user can create his or her desired script by copying and pasting the API requests he wishes to make. In the future I may want to create a front end and manage API keys for the user, like the original deployment.

AWS EC2 is required to make this project work and is much simpler to use than a traditional server. I also used AWS CodeGuru Security to do a security check of my code by uploading a zip of my project file a few times. I resolved one security flaw by adding some type checking of one of the variables. The other nine vulnerabilities only apply to Python 2, and as my project uses Python 3, are false positives. Screenshots of CodeGuru Security have been added to the Appendix of this paper.

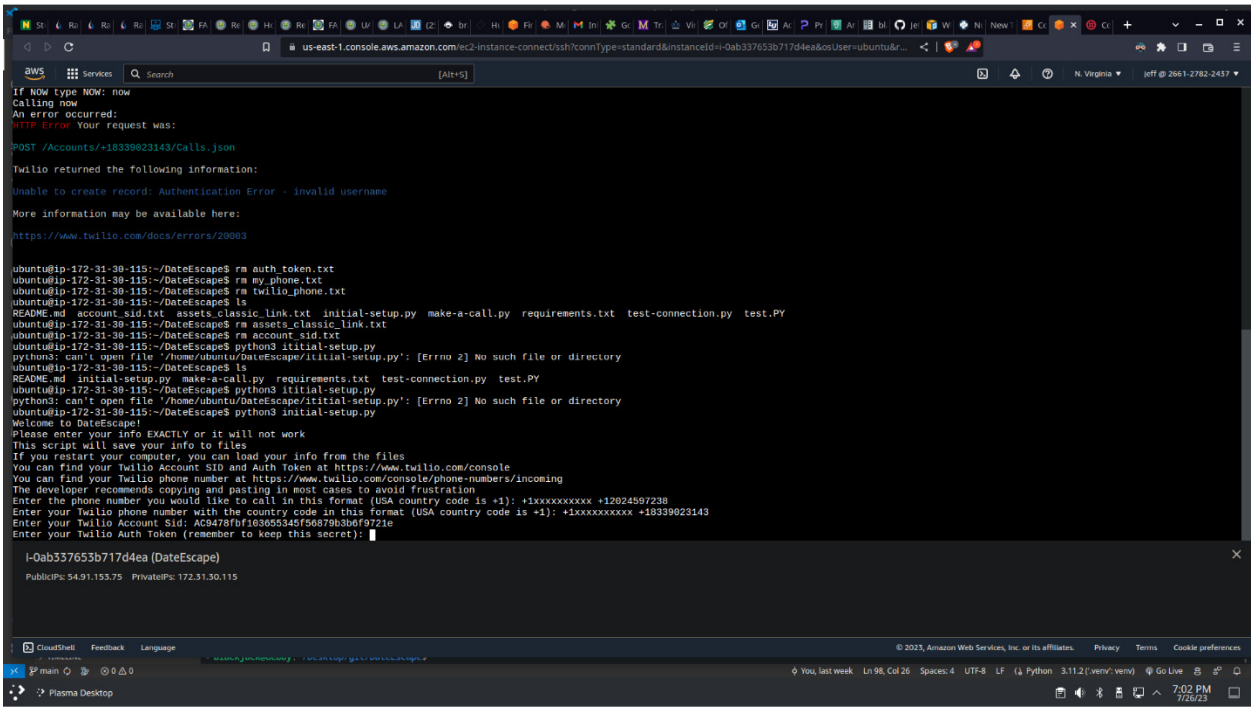
References

Jeffrey Dinackus. (2023, July 28). GitHub - JeffreyDinackus/DateEscape: This app allows you to make prerecorded calls and texts to yourself so you can get out of a date. Retrieved July 28, 2023, from https://www.github.com/jeffreydinackus/dateescape

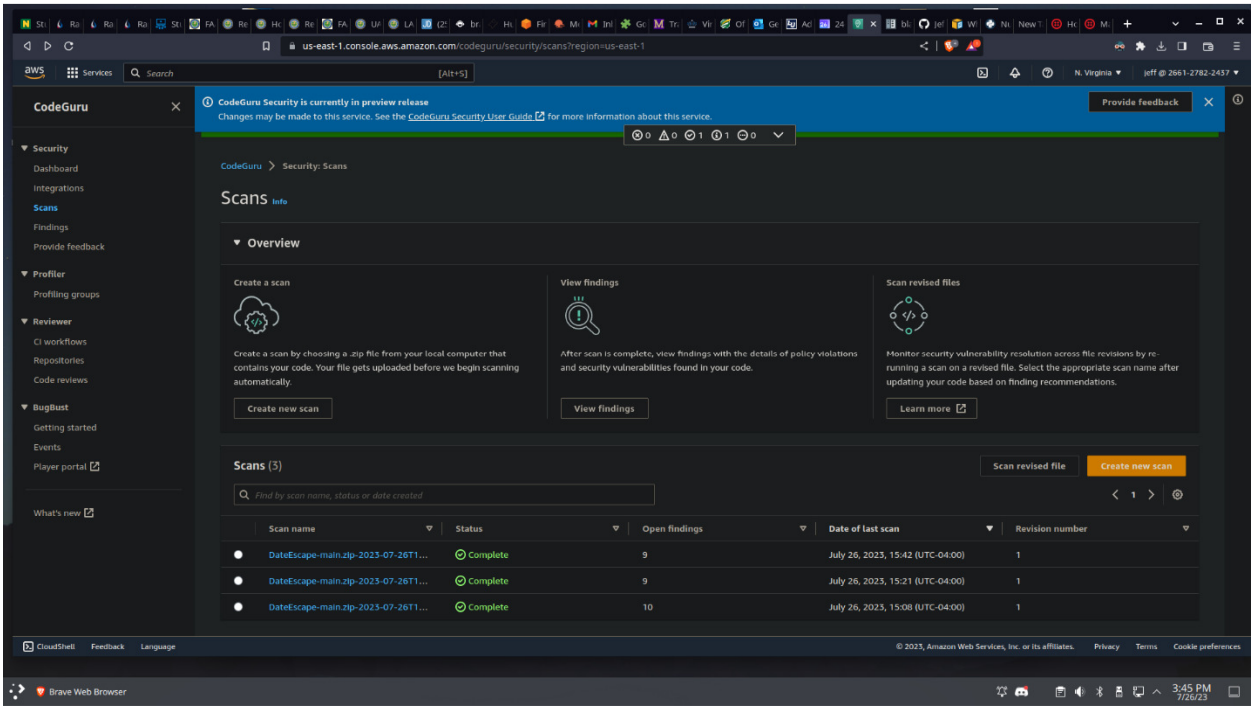
Jeffrey Dinackus, & Trombo, J. (2023, March 25). GitHub - JeffreyDinackus/QuickEscape.tech. Retrieved July 28, 2023, from https://github.com/JeffreyDinackus/QuickEscape.tech

Find expensive code – Amazon CodeGuru Security – AWS. (n.d.). Retrieved from https://aws.amazon.com/codeguru/

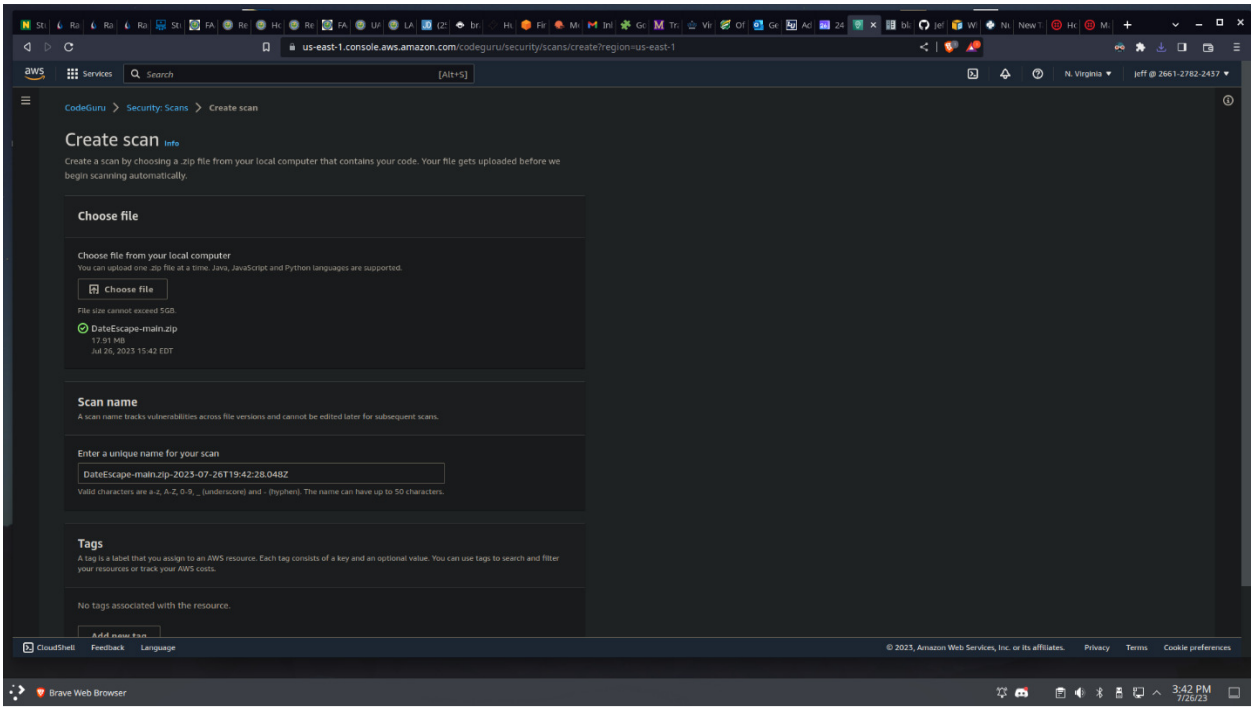
Appendix



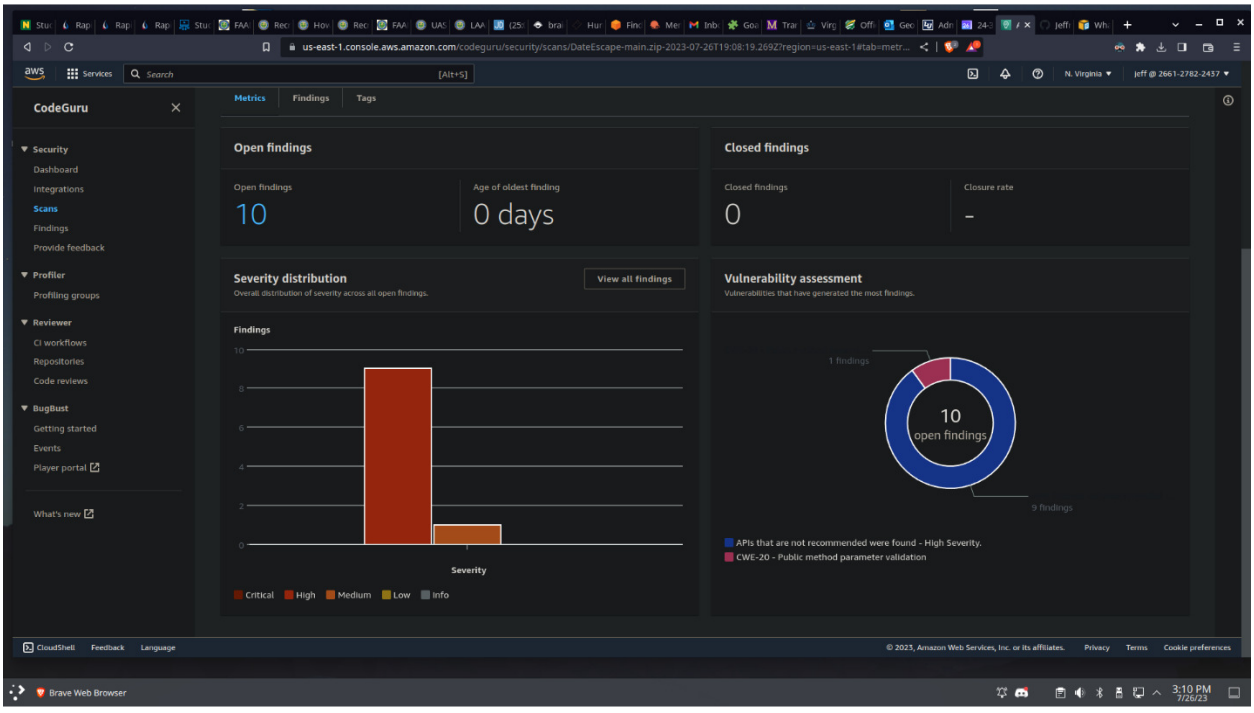
Running the initial-steup.py Script on my Ubuntu EC2 Server. Via EC2 Connect



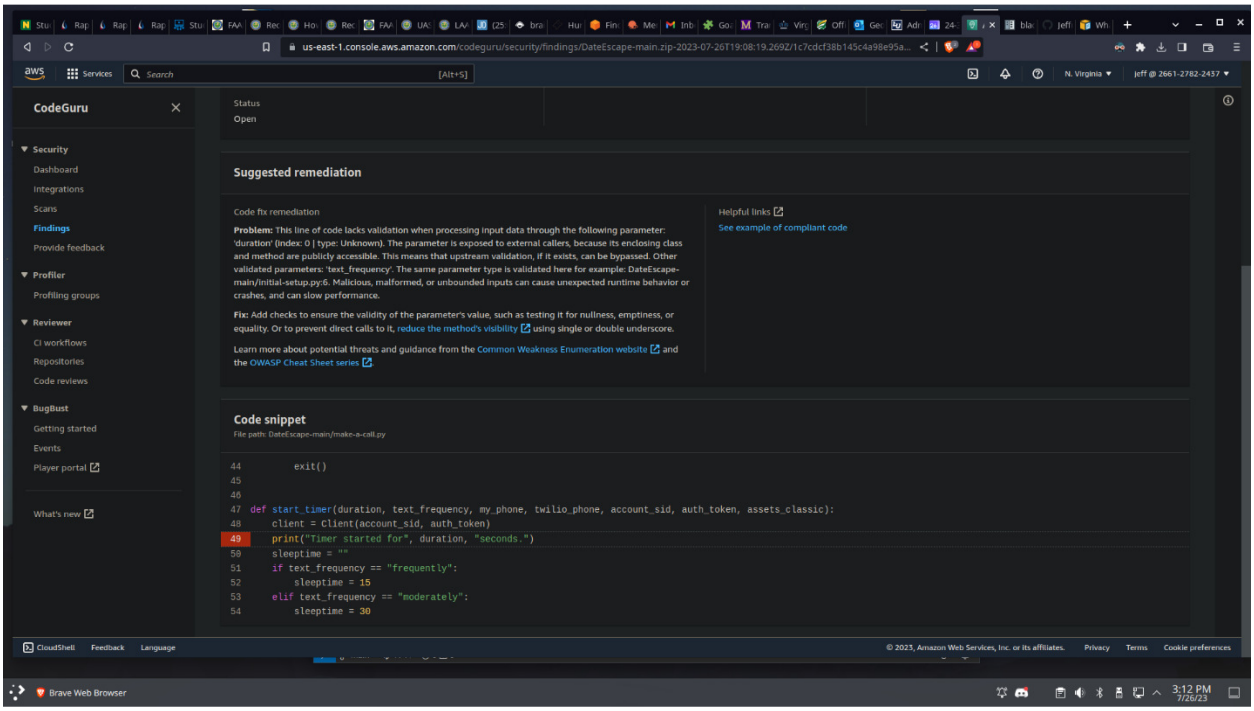
All 3 scans in CodeGuru Security.



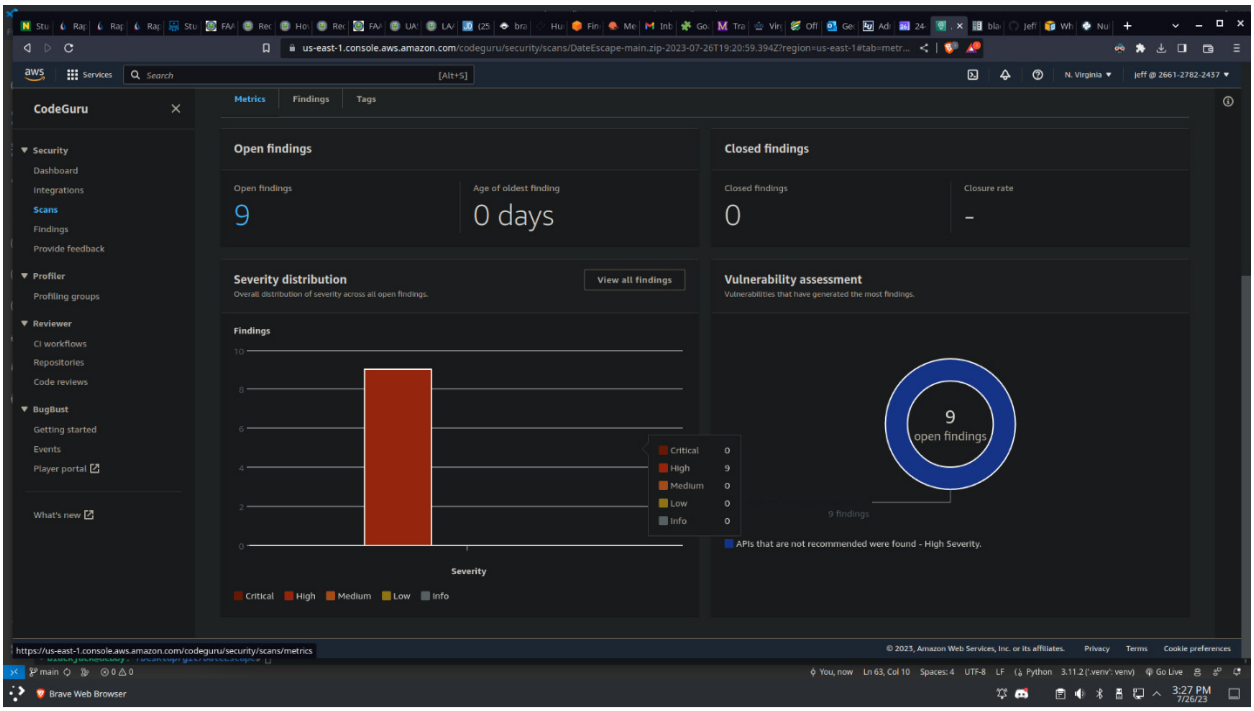
Creating a scan



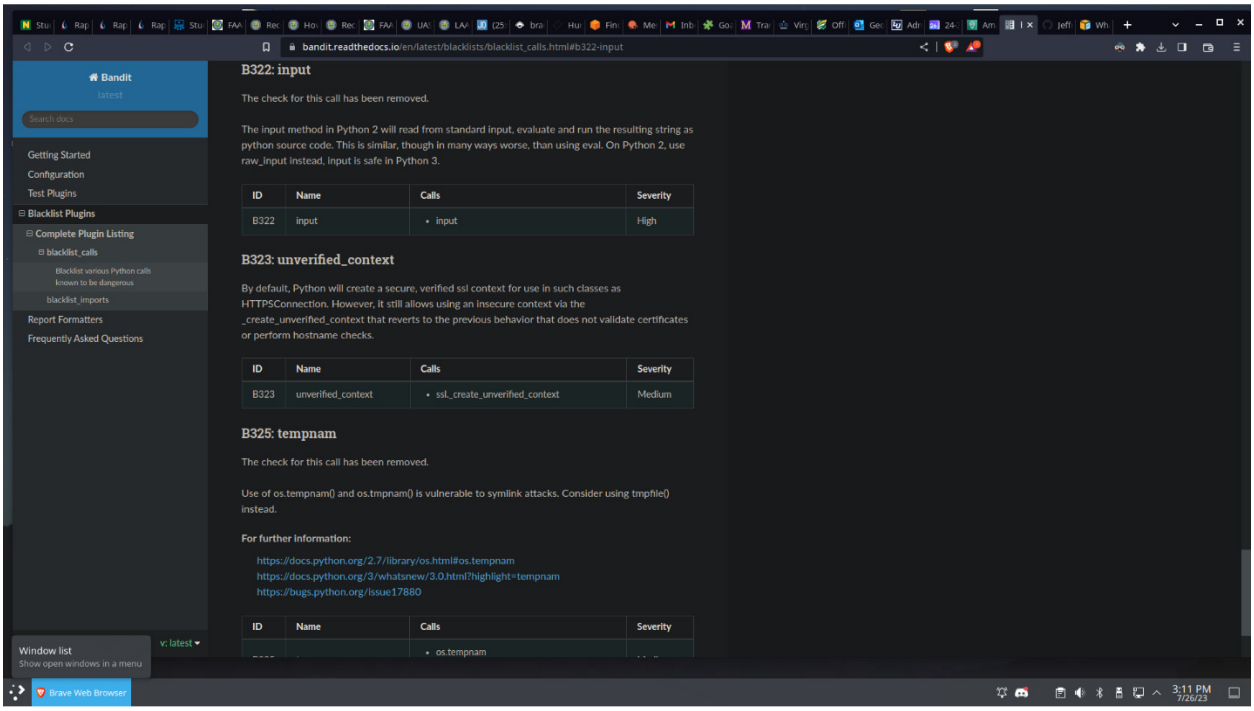
First scan results, nine red/orange are false positive, orange one is variable type checking as seen in next picture.



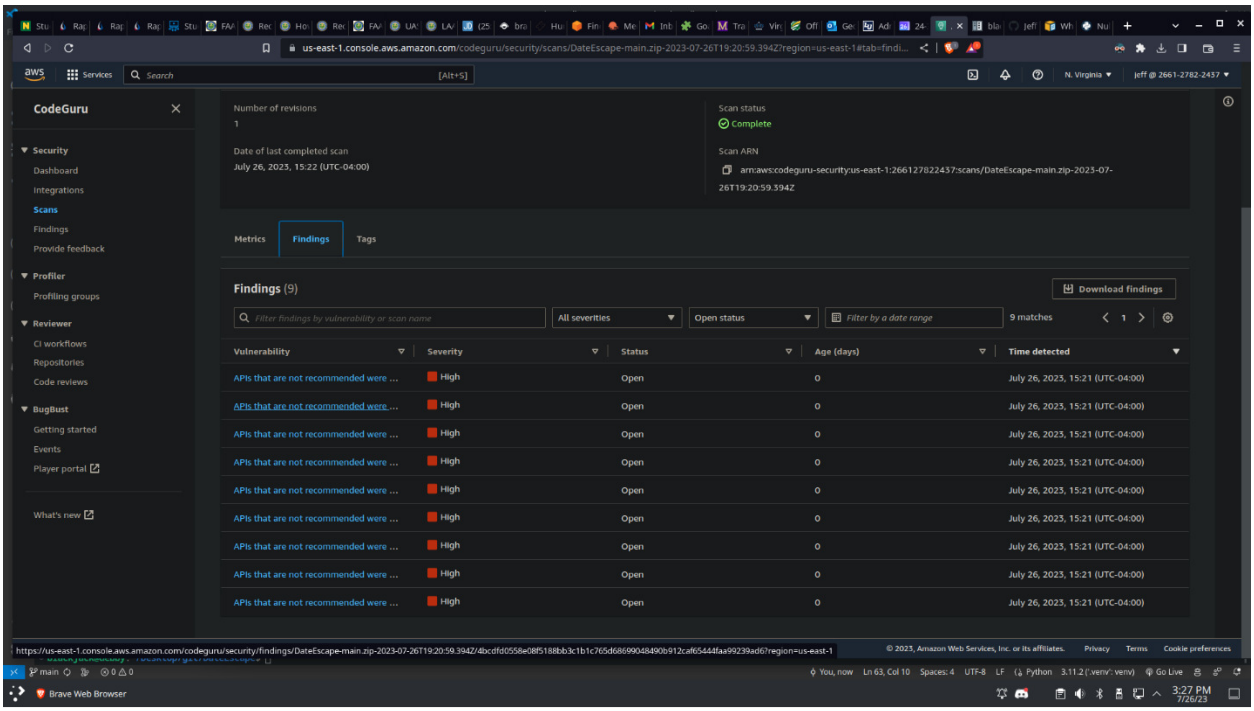
The type check error



After I fixed the true vulnerability, these are false positive that are only a issue with Python 2



Information about the false positive, it applies to Python 2



After resolving error on third scan.