Authors

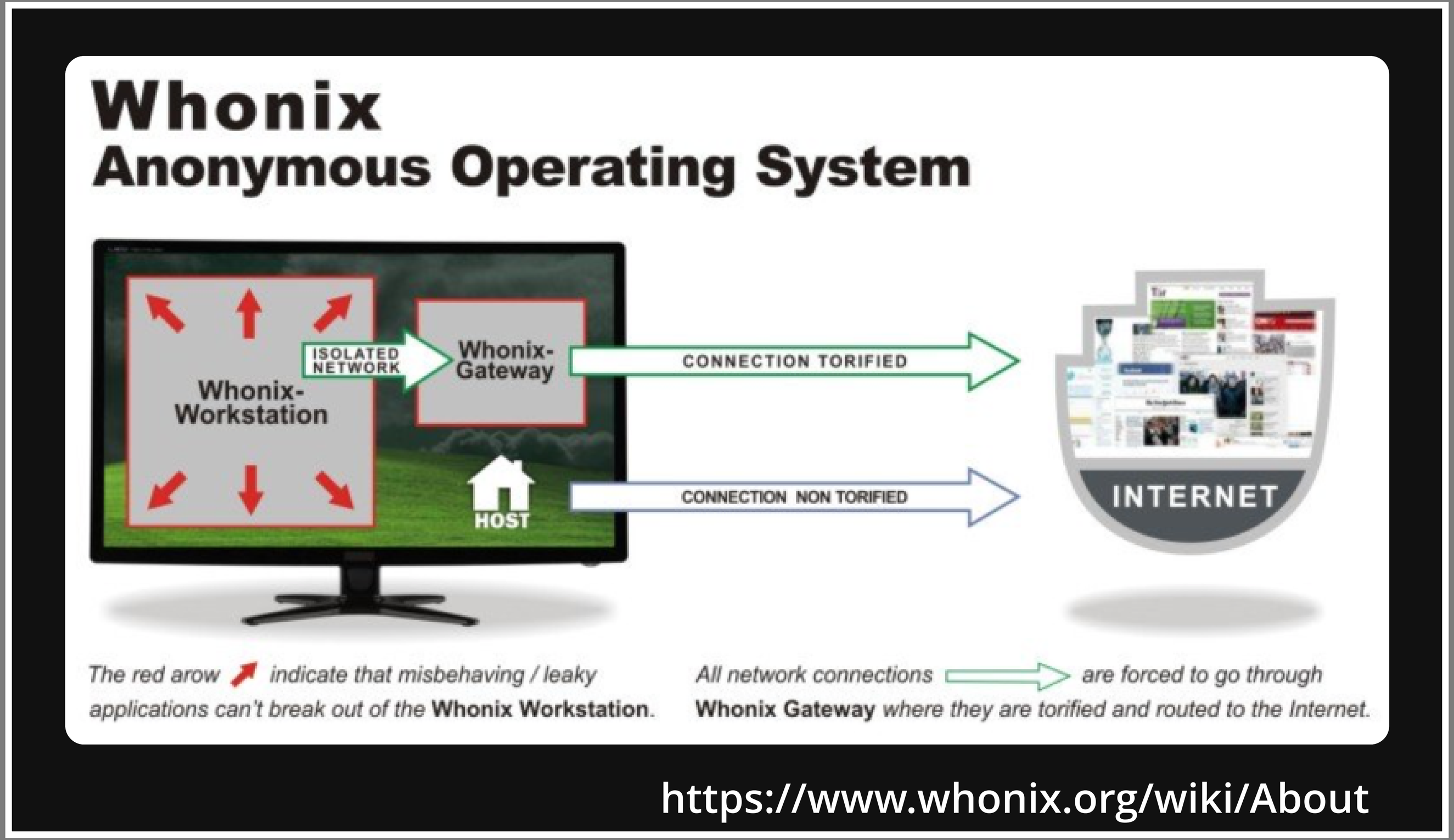
By John Quinn, Cameron Dey, and Evan Tanner
CSCI 462 Software Engineering Practicum
Spring 2020
Faculty Mentor: Jim Bowring

whonix Automated Testing Suite

https://github.com/johncameronquinn/wats-senior-capstone

COLLEGE of CHARLESTON
COMPUTER SCIENCE

Problem Description:. Currently, testing for Whonix is all performed manually, wasting time and introducing errors where volunteer developers do not test as stringently [1]. The developers need a testing suite for features such as internet browsing, chat clients, encrypted usb disks, and others [1].



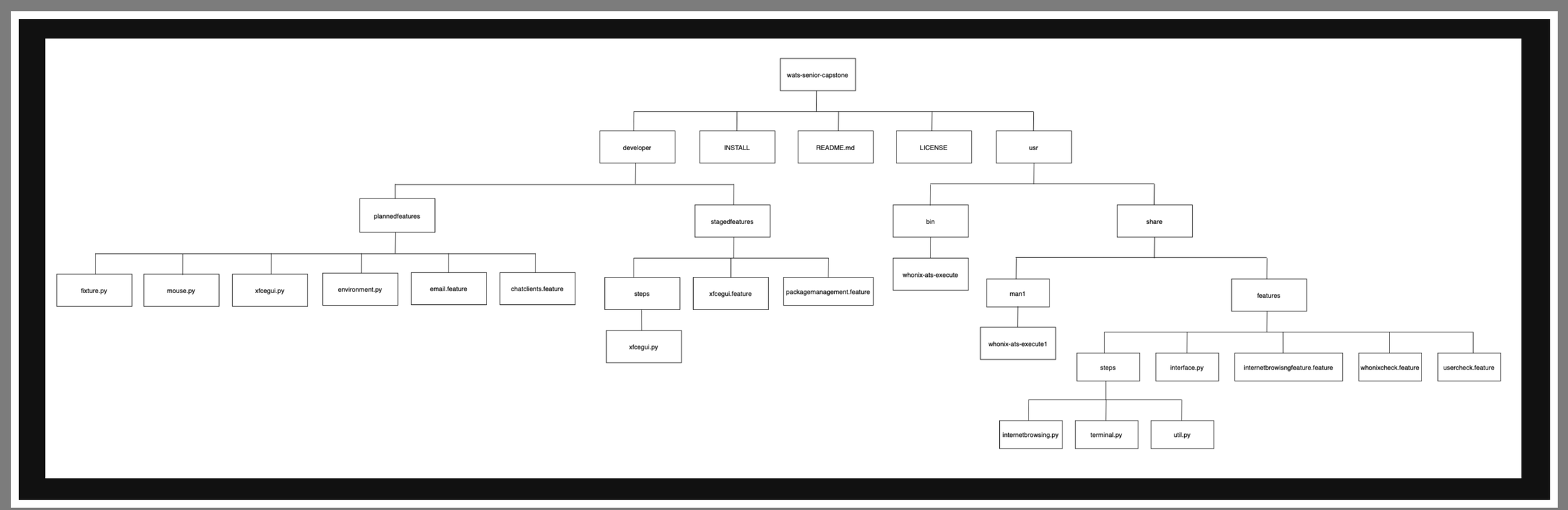
https://www.whonix.org/wiki/About

What is Whonix Automated Test System?

An automated testing framework for Whonix built on Python utilizing behavior driven development for testing the operating system's features and graphical user interfaces.

Directory Structure

Features- Functional features developed by our team.
Staged Features- Features ready for with lack step implementation.
Planned Features- Features to be implemented in the future.



Abstract

Abstract: *Whonix Automated Testing Suite* is a behavior-driven testing framework developed to access the functionality of Whonix features, behaviors, and graphical interfaces. As stated by the founder, "current Whonix development is slowed down by the time spent on manual testing as well as finding the causes for bugs" [1]. The contribution of an automated test suite allows for developers to run tests without manually executing them therefore increasing the efficiency, uniformity, and accuracy of development cycles. With the testing suite utilizing Python 3, Dogtails, and Behave, the software suite alleviates the tedious task of running individual tests and lets the software undertake the role of testing execution. Consisting of feature files and step implementations, a major benefit of the software is the flexibility in allowing for future and customizable feature tests. In conclusion, our contribution is intended to assist Whonix developers by contributing a testing suite that automates feature testing and the discovery of bugs

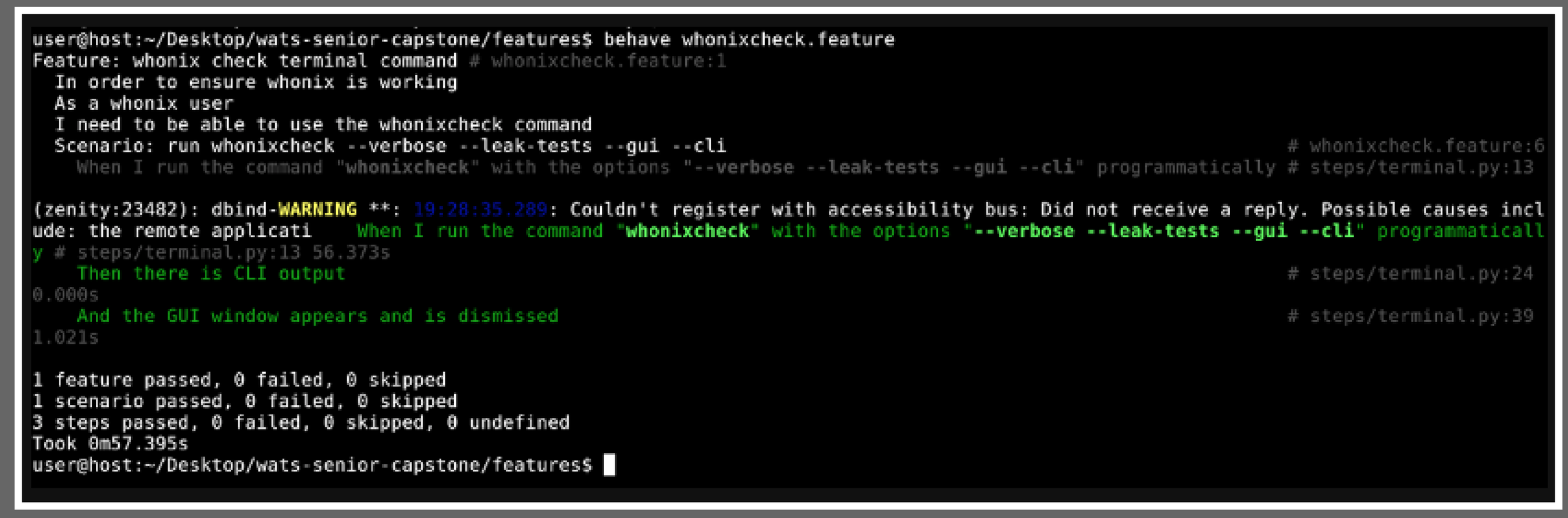
Collaboration with Whonix Community

https://forums.whonix.org/t/cs-student-capstone-bachelors-senior-project-for-whonix

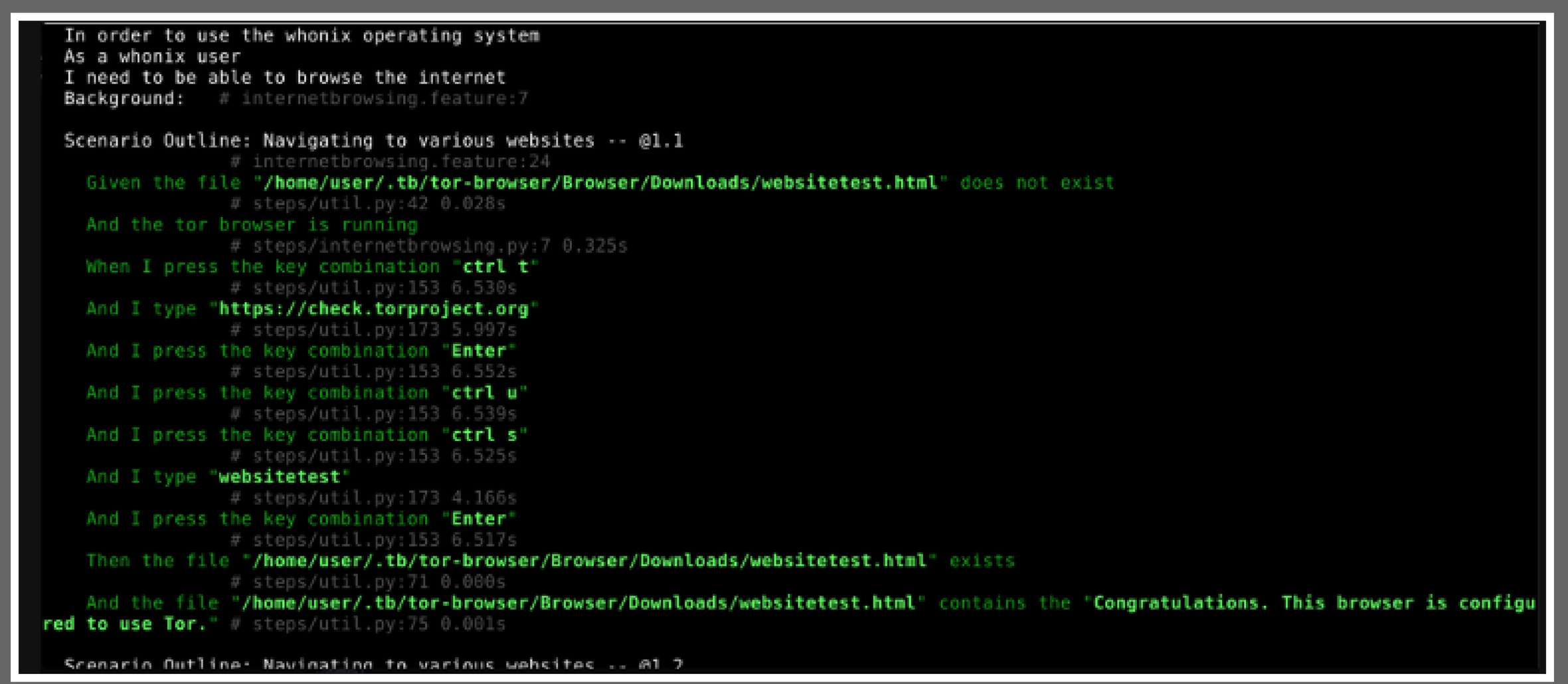
Our initial contact with the Whonix community began by posting on their online forum about our interest in contributing to the operating system. On the forum post, we asked for recommendations of what problems needed to be solved including any help with bug fixing, testing, or feature additions. One of the founders responded to inform us that an automated testing suite was their highest priority in needed contributions. After deciding to take on the contribution, we were notified that another community member was working on their own test suite as well, and that we could develop our system to eventually overlap with each other. With this member focusing mostly on the integration of kernel tests, our testing suite would supplement with the role of testing Whonix's GUI features, behaviors, and their retrospective outcomes. In our early conversations with the Whonix community, brainstorming was a key aspect when defining how exactly we should implement our test suite. The founder suggested that we could use Tail's testing suite as a model for our own implementation of a Whonix test suite. With Tails using Cucumber, a behavior driven development software that uses Ruby, we decided to use a similar software, Behave. With our team having experience in Python, we decided that using Behave along with Dogtails would benefit our approach to the suite's development. While the founder's requirements were not overly strict, he stated the testing suite should access the functionality of new Whonix builds, notify the user to feature failures or successes, have few performance constraints, be packaged as a .deb, along with a few others. Throughout the rest of the semester, our team interacted with the community by giving updates and receiving feedback of our progress through either the forum or Telegram where we had a private conversation between the founder and the member working on the kernel test suite.

Overview of The Whonix Automated Testing System

Whonix Check



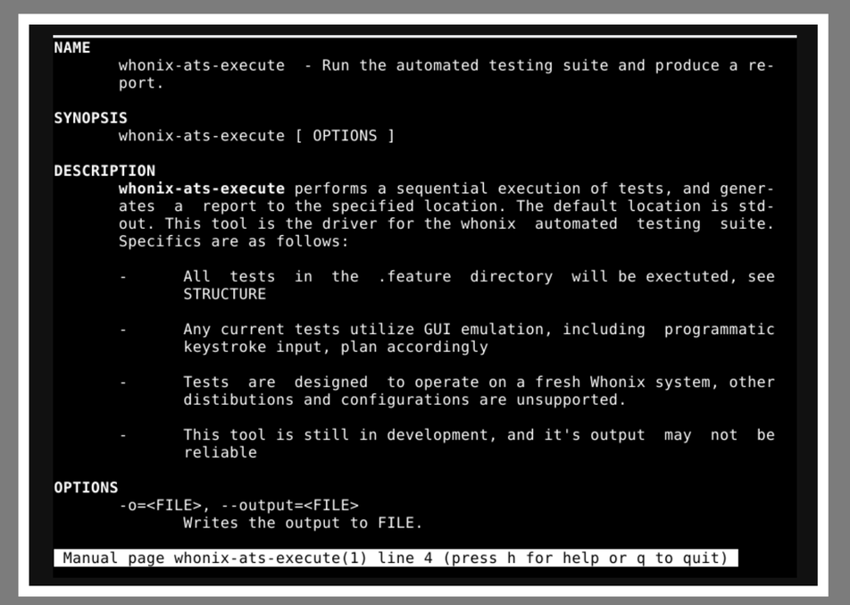
Internet Browsing



Requirements

-Contain in a public repository
-Executed Locally
-Few Perfomance Constraints
-Open Source License
-Packaged as Debian
-Adequate documentation including a readme, wiki, credits, and authors
-Shows output of the successes and failures of features
- Capability for allowing future additions of feature tests

Manual File



Why an Automated Testing System?

An automated testing suite would make Whonix testing more efficient, uniform, and systematic. In addition to these benefits, the testing suite would allow developers to create their own branches for customizable tests and future contributions . Through the integration of a testing suite, Whonix developers and the community will be able to use a singular system for a generalized criteria of what substantiates a successful build or feature task.

How WATS Works: Internet Browsing

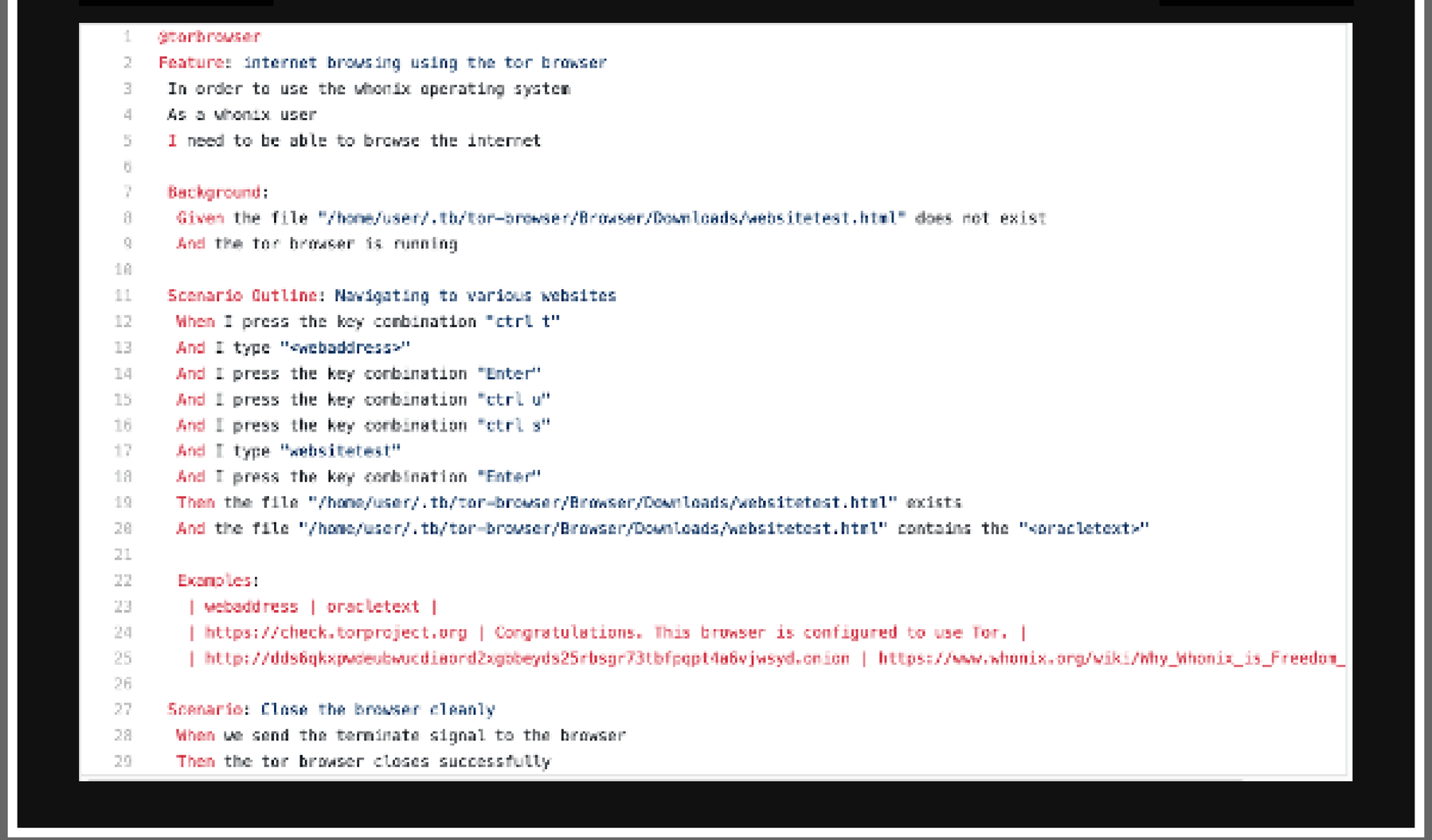
Features

Feature Files: Can be written by a business analyst, sponsor, or any anybody [2]. Written in a natural language (Gherkin), features are composed of scenarios that are implemented in the step files [2].

Components of a Feature File

Feature- Feature name and description [2].
Background- Executed before and adds context to scenarios before their execution [2].
Given-"Puts the system in a known state before user (or external system) interacts with system" [2].
When- "Describes the key action that the user (or external system) performs" [2].
Then- "Observation of the outcomes" [2].
And, But- "Mimics the steps that precedes them" [2].
Tags (@)- "Selectively check parts of your feature set" [2].

internetbrowsing.feature



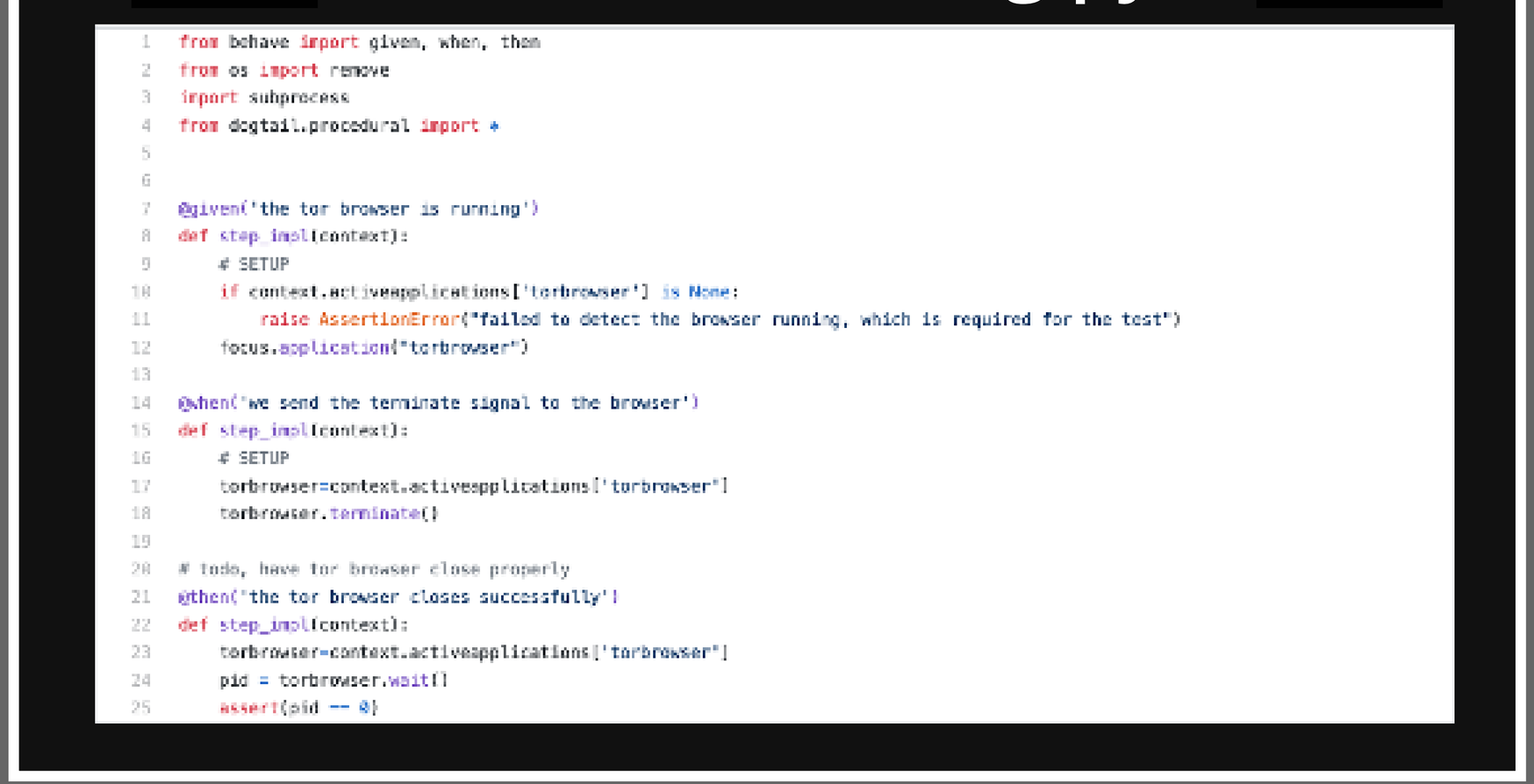
Steps

Step Files: Written in Python, step files are .py implementations of the feature files [2]. Step files are identified using decorators which match the predicate from the feature file [2].

Components of a Step File

-Context- Variable Behave uses where information is stored and shared [2].
-Step- "Decorator matches the step to any step type, "given", "when", or "then" [2].
-beforeall(context, step), afterall(context,step)- "These run before and after the whole shooting match" [2].
-before_feature(context, step), after_feature()- "These run before and after each feature is exercised"[2].
-before_scenario(context,step), after_scenario(context,step) -"These run before and after every scenario" [2].
-before_step(context,step), after_step(context,step)- "These run before and after every step" [2].
-before_tag(context,step), after_tag(context,step)- "These run before and after every tag" [2].

internetbrowsing.py



Libraries Utilized In Development:

-**Python 3**
-**Behave:** Behavior Driven Development Python Library
-**Dogtails:** Library For GUI Access and Manipulation

Features:

-**Internet Browsing Test Feature:** Navigates to various sites (both on clearnet and tor) using key combinations then checks for a successful exit.
-**Wonix Check Test Feature:** Runs Whonix Check and validates CLI output on terminal

Sources:

[1] Shleizer, P., 2020. *CS Student Capstone (Bachelor's Senior Project) For Whonix.* [online] Whonix Forum. Available at: <https://forums.whonix.org/t/cs-student-capstone-bachelors-senior-project-for-whonix/> [Accessed 17 April 2020].

[2] Jones, R., Rice,B. and Engel, J., 2020. *Tutorial - Behave 1.2.7.Dev1 Documentation* [online] Behave.readthedocs.io. Available at: <https://behave.readthedocs.io/en/latest/tutorial.html> [Acccessed 16 April 2020].