Domain Monitor

Programming Fundamentals Assignment

Introduction

Cyber attacks, specifically in the form of, Phishing and Pharming, Social Engineering type attacks, are becoming increasingly prolific within the UK Legal industry.

For example, scammers will register a domain similar to that of a genuine law firm domain and put up a fake website or a clone of the actual genuine firm website. They will then attempt to redirect traffic to the fake website either via exploitation of vulnerabilities in DNS server software, or through Phishing emails to potential targets.

We decided that we would write a command line application in Python, using the Flask framework, in order to try and identify possible fraudulent domain names and websites.

In short, Domain Monitor works by taking a genuine law firm name and producing multiple possible domain name variations of that name. We then attempt to resolve each variation in the hope that we can identify a potential scam as early as possible and report to the relevant governing body should it be necessary.

Because neither of us had any previous experience with Python, as well as having to learn the Flask web development microframework (http://flask.pocoo.org/), we also had to research and understand several other Python standard library modules and third party packages including, but not limited to:

- **Click** - For creating command line interfaces (http://click.pocoo.org/5/).
- **Peewee** - A small, expressive Object Relational Mapping library for managing our database and our interaction with it (http://docs.peewee-orm.com/en/latest/).
- **Requests** - A simple HTTP library (http://docs.python-requests.org/en/master/).
- **CSV, DateTime, Typing, Regex and UnitTest** - from the Python standard library.

As a byproduct of using some of these libraries and frameworks, we also had to extend our understanding of programming by researching and understanding classes, generators and Python decorators.

Project Details

As we had varying levels of previous exposure to programming, we decided that the best way to approach this project would be to do the majority of the work in person together. This enabled us to leverage each other's strengths, brainstorm ideas, share knowledge and problem solve as a team. We were also able to pair program, which as well as increasing productivity, also increased our understanding by, both, teaching and listening.

Wanting to replicate a professional development environment as closely as possible, we decided to take a test driven approach to development. As the core functionality of our program relied on generating specific variations, it made sense for use to write a test first and then develop our algorithms to meet the test criteria. Taking a TDD approach also made collaboration easier by defining a set criteria that we could both work towards, limiting the risk of potential scope creep.

Although completing the majority of the work in person, we were also able to collaborate remotely using Git version control. We created a shared repository for the project on Bitbucket (https://bitbucket.org), which enabled us to make further amends and work as individuals, whilst always staying up to date with what the other was working on.

Because of the scope of our project idea, one of the hardest parts of the project was deciding on a set of features that we could realistically produce within the timeframe, that would utilise our combined skill set, rather than just focusing on the skill set of a particular individual.

Overall, we thoroughly enjoyed working together and feel that our differing skill sets complimented each other well ensured we formed a very strong team. We would have no hesitation working together in the future.

Evaluation

As we had a chosen such an ambitious project, we quickly realised that we would have to concentrate on developing only a small subset of the desired features whilst still producing a working product.

This project is very much a first iteration and has several weaknesses. There are many more domain variations that we can produce and although we check whether a domain resolves, this doesn't necessarily mean it is malicious or provide us with any useful information. Future plans would include scraping the content of the page to identify it's content which would also enable us to gauge the severity of threat.

During the project, and because of our initial idea, we quickly got pigeon-holed, by tying our program too closely with our business domain, in this case the legal sector. It became apparent that actually we could totally de-couple the logic of our project from the 'subject'. This would allow the program to be used to to identify fraudulent domains for any industry, based on a certain config. If we were to start the project from scratch, we would definitely add this further layer of abstraction and increase the usability of our software.