THE UNIVERSITY OF SYDNEY

INFO2222 Assignment 2

Due: Friday of Week 10, 11:59pm

This assignment is worth 5% of your final assessment

It's an often repeated maxim that systems should be designed with security as the focus, not something to be slapped on after the fact. Unfortunately, as with most other good advice, the demands of getting a system working often outweigh the demands on ensuring that it is secure.

In this assignment you will be building on your website from assignment 1. If your website has not been completed, you will be required to finish it.

Task description

There are two components to this assignment, a core section and a list of options. You will be required to implement all the components from the core section, along with **two** of the optional components.

As a team you should pick and implement two of the options listed below, and you should provide complete and thorough documentation of your site and what security features you have implemented. Your options are:

- Confidentiality - HTTPS support
- Integrity - Web Application Firewall
- Availability - WSGI + Web Server deployment
- Auditability - API + Database

Core Components

You are required to complete both of these core components.

Deploying the Site [2 Marks]

Real websites are deployed on dedicated servers.

To complete this component, you will be provided with a server and ssh access and will need to deploy your site to the server.

You will be marked both for getting your site functioning on this server, and how you have configured the server. Consider how git keys and ssh keys should be stored, and how groups and users should be set up.

It's worth noting that the servers may be rebooted periodically, you will want some method of ensuring that your web application restarts when the server restarts. You may find systemctl, or init scripts useful for this.

- 1 mark for a successful deployment
- 0.5 marks for an appropriate use of users and groups
- 0.5 marks for a successful startup script

In assignment three the availability of your site will be assessed. Bottle by itself is not particularly efficient, and you may find the cherrypy python package useful in quickly improving the stability of your site.

Virtual Users [2 Marks]

Bots are prevalent on the Internet, from crawling and data-mining to automated up-voting and sharing, the actions of an anonymous virtual user can be nearly indistinguishable from that of a real person. These virtual users are also useful debugging tools, and when run can swiftly ferret out any broken links or functionality.

You will be required to write a number of virtual actors for your site. They should span all regular user, and administrator roles and should perform a number of actions that you would consider standard for the site such as logging in, trawling the site sending messages, and banning and unbanning each other.

You will need to write a separate Python class (in its own file) that when run, selects a virtual user at random and navigates them through the site, performing tasks that a regular user would (filling in forms, viewing pages etc). The actions of your virtual users should span all possible routes for your site. You may find it useful to re-purpose your think alouds as the paths that your virtual users take through your site.

Your virtual users should make use of a headless browser to ensure that Javascript and AJAX can be correctly loaded.

Template code will be provided. You may find that the `requests` and `selenium` packages are helpful in completing this section.

- 1 mark for successfully setting up your bots and demonstrating at least one think aloud
- 1 mark for completing all other tasks on your site

Options

Each option is worth 2 marks. You are required to complete two options but may complete as many as you wish. Completing further options will not award additional marks.

Confidentiality - HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication and encrypts data between the

If you select this option, you will be required to obtain a and configure your site to use a SSL certificate obtained from some certificate authority (Let's Encrypt provides these for free). You should attempt to explain your choice of SSL settings in your report.

- 1 mark for ensuring HTTPS for all routes
- 1 mark for explaining your choice of configuration options

Integrity - Web Application Firewall A (WAF)

Many modern websites have a parser (such as Naxsi for Nginx) that checks incoming requests for malicious strings and sanitises them. This attempts to clear strings of potential SQL injections, XSS attacks and other nefarious schemes.

If you pick this option you will be required to build your own parser that checks incoming requests to your site and cleans them of any nefarious contents. The parser should operate as a separate Python class in its own file, and should detect and report malicious strings to the site's Administrators. The WAF should handle attacks gracefully; failing a check might sanitise the user's input, redirect to an error page or otherwise inform the user of the error without impacting on the availability of the site.

This web application firewall should be hosted on a separate port to your web application (so that multiple web applications could make calls to the same WAF). The application should be written such that the target port and address are a not hard coded, and are preferably loaded from a configuration file when the server is started.

Template code will be provided.

- 0.5 marks for separation of the WAF from the application
- 0.5 marks for explaining in the report for explaining your WAF rules
- 1 mark for the implementation and demonstration of the WAF

Integrity - Web Application Firewall B (WAF)

Often attackers may begin by scanning and scraping a target website. With this option you are to develop a WAF that detects and rejects scrapers and automated scripts. You should consider `curl`, `wget` `python requests`, `selenium` and other common approaches to automating the scanning and scraping of a website.

This web application firewall should be hosted on a separate port to your web application (so that multiple web applications could make calls to the same WAF). The application should be written such that the target port and address are a not hard coded, and are preferably loaded from a configuration file when the server is started.

You will need to find a method of authenticating your own virtual users if you select this option; without proper authentication your virtual users should be blocked by this WAF.

Template code will not be provided.

- 0.5 marks for separation of the WAF from the application
- 0.5 marks for explaining in the report for explaining your WAF rules
- 1 mark for the implementation and demonstration of the WAF

Availability - WSGI + Web Server Deployment

You may have noticed that the bottle framework is not particularly fast, and that under a reasonably small load availability may start to suffer. This option involves hosting your application using a more powerful web server to ensure faster response times and better availability.

uWSGI and nginx are likely to be useful tools for completing this task, completing this option may also change how your web application is launched at startup. These are incredibly useful tools with thousands of options, you are not required to have a perfect setup, merely a working one.

This option may be difficult if you are uncomfortable with the Linux command line.

- 1 mark for a working nginx setup
- 1 mark for a working uWSGI setup

API + Database

It is common practice to divide a website into a series of different processes. One of the most coarse grained approaches to this is to have a separate API that provides the appropriate data while the front end serves page requests. If you pick this option you will be required to write a separate Python server that hosts the required API endpoints.

As a secondary part of this option, you will need to further segregate your database from your main application. Your database should exist as a third, separate application on its own port. Your web application itself should only ever get data by requesting it from the API, it should not communicate with the database itself.

Your API and database should be communicated with using an appropriate Python networking module. Your database does not require SQL (but you may use it), it can simply contain all the Python objects that contain the data you require for this site and can be backed up to a text document. You may chose to use a pre-existing SQL or NoSQL database.

Lastly, **all API calls should be logged** somewhere on your server for the purposes of auditing.

Your web server and API applications should be written such that the target port and address are a single variable that can be easily modified. please do not hard code them into every request. You may find that your MVC structure naturally lends itself to this sort of division, as a result, no template code will be provided for this option. Instead consider splitting your model and database components across multiple bottle servers

- 1 mark for separation of the API
- 0.5 marks for separation of the database
- 0.5 marks for the API log

Documentation

Your application should come with 1000-2500 words of documentation dictating how the site works and what security decisions have been made. In particular it should discuss what data is collected and stored, how it can be accessed and viewed. It is suggested that you include relevant sections of code, and diagrams where appropriate. For example, if you store data in cookies, you should state what data is stored and how it is stored, or if you communicate with another python server as part of an option for assignment 2, you should state how this communication is handled.

The documentation **should not disclose any administrative or other passwords or keys that are used in your site**, but should otherwise describe in detail what the security goals of the site are, and how they have been achieved.

Additional Criteria

The following are some additional criteria that will be required of your web application.

- Your site should not make external calls to on-line APIs, CDNs or other repositories as it needs to be able to run in an isolated environment. External requests will result in the loss of marks.
- **DO NOT TAMPER WITH THE STAFF ACCOUNTS ON YOUR SERVERS**, these are the ‘ec2-user’ account and the ‘marker’ account. These accounts will be used for marking, if staff cannot access them, they cannot mark your website and you will receive a 0 for this assessment.
- Broken functionality may result in a marks deduction; if your usability project submission was not feature complete then you should finish your implementation before the end of this project. Marks will be deduced based on the severity of the unfinished features.
- If your site is not in working order and hosted on your virtual machine by the due date then you will be awarded a mark of zero for this assessment.

Marking

You will receive a mark out of 10 for this assignment. The security of your implementation of your web application will not be assessed in this assignment, but will be assessed in Assignment 3.

- A mark out of 4 for the core functionality. Each component will be marked out of 2. A mark of 0-1 indicates that the work does not meet the requirements outlined above, it is missing key features or requirements and would be difficult to deploy. A mark of 1-1.5 meets most the requirements but may not be complete in every case. A mark of 1.5-2 indicates that the work meets all of the requirements.
- A mark out of 4 for your options, each option will be marked out of 2. A mark of 0-1 indicates that the work does not meet the requirements outlined above, it is missing key features or requirements and would be difficult to deploy. A mark of 1-1.5 meets most the requirements but may not be complete in every case. A mark of 1.5-2 indicates that the work meets all of the requirements.
- A mark out of 2 for the quality of the documentation. A mark of 0-0.5 indicates that the document is sparse and incomplete. A mark of 0.5-1 indicates that the documentation is missing large section of required details and a reader would not be able to understand the architecture of the site from this document. A mark of 1-1.5 indicates that the document contains all of the key details required and is presented in a readable fashion. A mark of 1.5-2 indicates that the contents of the document are clear, and well explained and include appropriate snippets of python code and diagrams.
- Up to five marks may be penalised for non-completion of the functionality of the website from assignment one. Wire frames are not sufficient. Your site should support registration, logging in, messaging, posting and account administration, user administration should allow the changing of account details, while administrators should be able to restrict, promote or delete other users.

Submission Details

The following details should be read carefully regarding the submission of this assignment

Group Assessment

As with your usability project you will be required to complete a group assessment form using spark-plus. The marks awarded for this assessment will be proportional to the contributions of each group member.

Code Submission

Your report should include a link to your last git commit before the submission deadline.

The code that you have on your site will be assessed. READMEs, code commenting and other forms of documentation will be looked upon favourably. Spaghetti code, missing documentation, tabs rather than spaces, and a lack of compliance with PEP8 standards will be looked upon unfavourably. Code is more often read than written; code that cannot be understood will be assumed to be non-functional.

Report Submission

You will be required to submit your report on Canvas in a pdf format. Latex is encouraged (and is a useful skill to learn).

If you have any questions, or are unsure about the submission process or any aspects of this assignment, please ask sooner rather than later!

Academic declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgment from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Computer Science, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of Computer Science for the purpose of future plagiarism checking.