Cloudflare WAF AI

DESIGN DOCUMENT

SDMAY21-16
Client: Cylosoft
Advisors: Dr. Yong Guan, Dr. Berk Gulmezoglu

Ben Feddersen: Stakeholder Correspondent
Eric Reuss: Documentation Manager
Giovanni Mejia: Report Manager
Jordan Heim: Documentation Manager
Presiian Iskrenov: Meeting Scribe
Ryan Burgett: Meeting Leader

sdmay21-16@iastate.edu
http://sdmay21-16.sd.ece.iastate.edu

Revised: October 4, 2020//V1

Executive Summary

Development Standards & Practices Used

- Agile
- REST
- TCP/IP
- HTTP

Summary of Requirements

Microsoft ISS will generate text-based log files of each web request. A console-based application will be created to find the current log file for the site that is being monitored. The application will then send the log file to the web service to be stored and processed. An AI application will be created to monitor the log files and identify suspicious web activity using the Cloudflare API. When suspicious activity is detected on the server, the potentially dangerous IP will be blocked for twenty-four hours.

Applicable Courses from Iowa State University Curriculum

- COM S 227
- COM S 228
- COM S 252
- COM S 309
- COM S 311
- COM S 321
- COM S 329
- COM S 339
- COM S 352
- COM S 363
- CPR E 430
- CPR E 431
- COM S 474

New Skills/Knowledge acquired that was not taught in courses

- .NET servers
- Implementing code involving artificial intelligence
- ISS log files and general web traffic moderation

Table of Contents

List of figures/tables/symbols/definitions

- Tables:
  - 2.2: Main Tasks (Page 8)
  - 2.6: Personnel Effort Requirements (Page 12)
- Figures:
  - 2.4: Project Schedule (Page 10)
  - 3.7: Design Plan (Page 17)
- Definitions:
  - AI: Artificial Intelligence, a computer application that can learn, adapt to changing environments based on previous experiences.
  - API: Application Programming Interface, a software intermediary that allows two applications to talk to each other.
  - WAF: Web Application Firewall, an application firewall that applies a set of rules to an HTTP conversation.
  - Agile: A software development strategy based around developing in iterations or sprints.

1 Introduction

1.1 ACKNOWLEDGMENT

Technical advice and guidance were given by Dr. Yong Guan, Dr. Berk Glumezoglu, and Dr. Akhilesh Tyagi. Andrew Dakin provided technical advice and product specifications and a test domain from Cylosoft.

1.2 PROBLEM AND PROJECT STATEMENT

Problem Statement: Cylosoft is a company that designs, codes, and hosts websites. These websites are often probed and tested by bots, hackers, and spammers. The web servers at Cylosoft generated text log files as URLs are hit, and Cloudflare acts as a web application firewall (WAF). Cloudflare has both Cloudflare developed rules and customer-generated rules, and there are gaps in the rules that can be improved.

Solution Approach: This problem can be solved by creating a web server-side service that gathers logs and sends them in real-time to a cloud database. The database would then have code run against it, looking for potential bots, hackers, and spammers and generate firewall rules accordingly. The project's output will be real-time WAF rules generated by artificial intelligence based on web traffic patterns.

1.3 OPERATIONAL ENVIRONMENT

Our project's component is software that will run on hardware (computers, servers.) located at the Cylosoft office. While we do not need to consider conditions such as heat, weather, and moisture for our product, we must ensure that every aspect of our code can run smoothly on Cylosoft's system.

1.4 REQUIREMENTS

Functional:

- Microsoft ISS will generate text-based log files of each web request.
- A console-based application will be created, which will find the current log file for the site that is being monitored. The application will then send the log file to the web service to be stored and processed.
- The Azure web service will process and store relevant information from IIS log files.
- The AI should have access to the processed IIS log data.
- An AI application will be created to monitor the log files and identify suspicious web activity using the Cloudflare API.
- When suspicious activity is detected on the server, the potentially dangerous IP will be blocked for twenty-four hours.

Non-Functional:

- The system should be able to perform network inspection for as much uptime as possible.
- Building and scaling the system should be easy to manage.
- The system should be able to accommodate large volumes of traffic.

- Access to the system should be restricted to only approved users.
- The system should block suspicious network traffic upon identifying it.

Environmental:

- Servers and databases should be able to function indoors.
- Rules and IP blocks of network firewalls should be updated per Cylosoft standards.
- Servers should function with web requests from any distance.

Economic:

- All testing should be completed using one domain/server, which will be provided by the client.
- All other design and implementation should be completed using development platforms, IDEs at no extra cost.
- The project should be completed with six team members, with each team member working three hours per week.
- The project should be designed and implemented by May of 2021.

1.5 Intended Users and Uses

Our product's intended end-user is Cylosoft, with the product operating on Cylosoft's existing software system. Our work is intended to be scalable to meet the needs of Cylosoft as they expand their software in the future.

1.6 Assumptions and Limitations

Assumptions:

- Supplies to accomplish Cylosoft will provide this system.
- Team members working on the project will be able to access the resources needed to accomplish the task.
- All equipment utilized will be in working condition.
- The scope of the project will not change for the duration of its life cycle.
- The project source code will be utilized after the project is completed.

Limitations:

- Existing network bandwidth can't be tied up by traffic generated from our project.
- Project planning shall be done by the end of the fall semester.
- The design and implementation of the project shall be done by the end of the spring semester.

1.7 Expected End Product and Deliverables

Expected End Product: ***Security Application with learning capabilities***

We will have a complete design document at the end of this project to help stakeholders understand the design process we used. A console-based application will read log files from Microsoft IIS. These log files will be parsed and fit the Azure web service database template. Each log file will be assigned to a corresponding website to ensure ease of accessibility of necessary information. The AI will be developed to have access to the processed IIS log information and help monitor them to identify suspicious activity. When the AI detects suspicious activity, it will remove the IP and block them from accessing the site for twenty-four hours.

Deliverables:

Status Reports-

- We will be providing bi-weekly reports on our progress and report any shifts in our timeline. The team will also give the date of all meetings held and the discussions had.

User Manual-

- We will provide a README file as well as a LICENSE file contained within the GITHUB of the project. The README file will have detailed instructions on how to use the console application and make modifications to it. The LICENSE file will contain any legal documentation associated with Frameworks/IDE's/API's and other relevant material used in developing this project.

Project Design Document-

- We will provide a detailed plan and description of the process we used during this application's development. This document will outline many things, including the Gantt chart, price cost, and technologies used and relevant to the project.

Azure Web Service-

- We will create an Azure web service that receives Microsoft IIS logs and stores the relevant data from them in a database.

Console Application-

- We will develop a console application to parse through Microsoft IIS log files and return them appropriately to fit Azure database fields. Log files will also be assigned their corresponding website so that they can be easily tracked

AI-

- Develop an AI that will find "obvious" threats that Cloudflare would typically not catch. These "obvious" threats will be recorded as rules to improve future responses to that type of attack. AI will then remove the threat.

Test-Case-

- We will provide an example of the log files association to a website and the proper formatting of Microsoft IIS log files inside the Azure database. Also, a simulated attack to show the performance of the AI in detecting and removing a threat.

2. Project Plan

2.1 Task Decomposition

Our initial approach to creating the solution was to break apart the problem into three smaller problems: a console application, a cloud web service, and an artificial intelligence algorithm to make Cloudflare rules. The console application will pull Microsoft IIS logs for a site being monitored and then send them to the web service. The web service will transform the records and fit them into our data structure. Our AI algorithm will then use the web service logs to generate Cloudflare WAF rules to be applied to the monitored site.

2.2 Risks And Risk Management/Mitigation

Main Tasks:

Building console application - Risk Score: 0.6

- This task was given a risk score of 0.6 because there will be many different things to come together for this task to work correctly. This application will have to make connections between the Microsoft IIS log files and our web service. It also has to look at the IIS log files and extract meaningful information and translate that to a condensed log file to send over.

Building web service - Risk Score: 0.5

Building database service - Risk Score: 0.2

Building AI application - Risk Score: 0.9

- If the AI cannot identify malicious traffic to block correctly, we will need to find a different AI approach. This will likely involve figuring out another machine learning/AI learning algorithm that will identify it instead. This task has a high-risk score because our team has little experience with machine learning/AI and will likely have to change models.

Testing/Quality Assurance - Risk Score: 0.8

- Testing and making sure our systems work together correctly without failure might prove to be complicated. As testing

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Proposed Milestones:

Research-

Main Task:

- Research all things related to the project (Programming language, Frameworks, API)
    - Subtask:
        - Share information with the group and discuss the practicality

Environment Setup-

Main Task:

- Install IDE to be used for console application and AI and corresponding frameworks

Console Application -

Main Task:

- The console-based application will find the current log file for the site being monitored and send the log file to the web service
    - Subtasks:
        - Establish a connection between code and IIS log files
        - Establish a relationship between code and web service
        - Find log file specific to the website
        - Parse log file for necessary information
        - Send parsed log file to web service

Web Service

Main Task:

- Azure web service will store relevant information from logs into the database
    - Subtask:
        - Establish a connection between code and web service
        - A parsed log file will store incorrect data fields in the database

AI-

Main Task:

- AI will detect attacks and write new rules that do not exist on the Cloudflare API to improve detection accuracy.

- Subtasks:
  - AI would look through Cloudflare rules and remove the user if conditions met
  - Machine learning algorithm will add new regulations not existing within Cloudflare

2.4 Project Timeline/Schedule

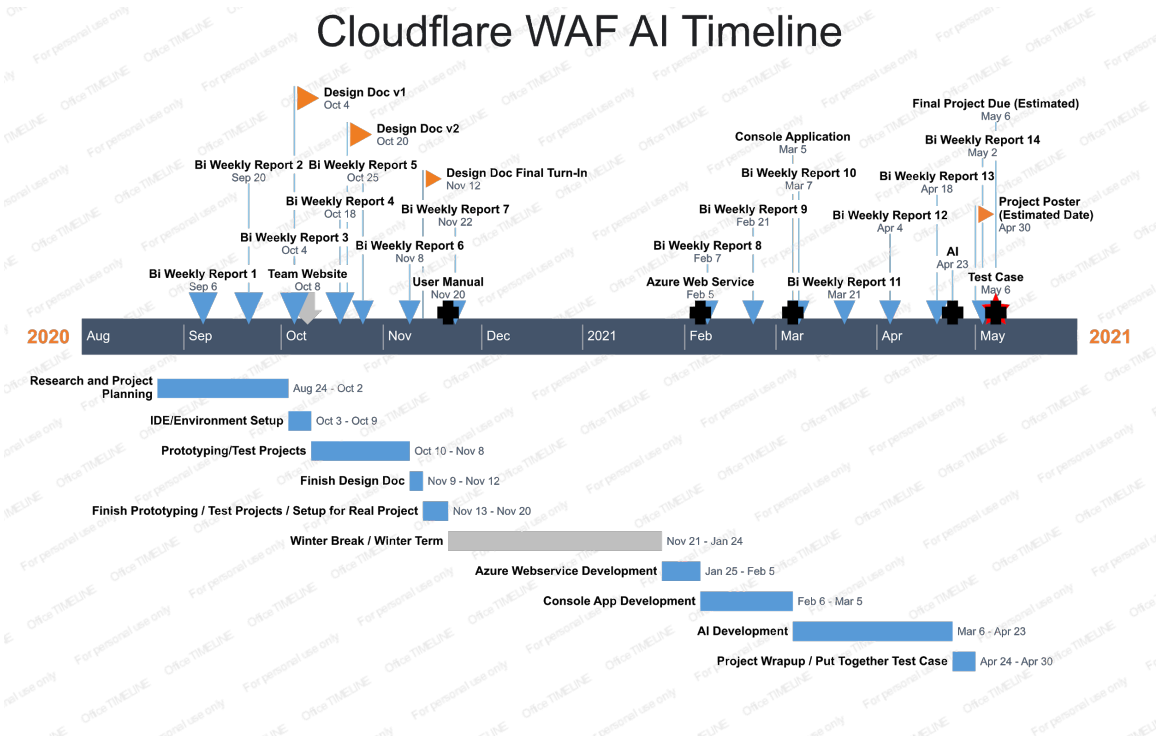

Figure 2.4 Project Schedule

Note on the Gantt chart: Listing all subtasks made the above chart unreadable, so each subtask's timelines are spelled out below.

Note on our development strategy: We plan on using an Agile development strategy with one-week sprints instead of the traditional two-week sprints. Our project has many small tasks that would be difficult to lump together in two-week sprints, and with this semester being remote, we were planning on meeting weekly from the beginning.

Breakdown Into Subtasks

- Research and Planning
    - Information gathered through research was shared with the group each week.
- IDE/Environment Setup
    - The IDE and necessary libraries are to be installed and set up for each developer by October 9th
- Prototyping / Test Projects
    - There should be a test project for the web service after the second week, a test console app after the third, and a test AI project by the final week of this task.
- Finish Design Doc
    - The design document should be completed and turned in by November 12th
- User Manual
    - There should be a rough user manual (subject to change throughout the project) by November 20th
- Azure web service
    - The first week will be dedicated to setting up a database
    - The second week will set up the database's automatic filling with test data sent to the web services.
- Console Application
    - The console application should be able to find log files specified by the end of the first week.
    - It should be able to parse the log files by week 3.
    - By the end of the fourth week, the console app should send the relevant data to the webservice.
- AI
    - By the end of the second week, the AI application should set Cloudflare rules by manual input.
    - The rest of the time for AI will be used to develop the machine learning algorithm
- Project Wrap Up/ Test Case
    - By the project deadline, there should be a presentable test case scenario for the program, and our project sponsors should fully understand how to duplicate our setup and take over the project from us.

2.5 Project Tracking Procedures

With the semester progressing, we have utilized different tools to help manage as our project grows. We are using slack to communicate with our clients and address any upcoming issues with our project. For project discussion among our group members, we heavily use discord to manage meetings and share resources that pertain to our project. It allows us to brainstorm and finalize ideas for our project. We have also set up a Github repository and GitLab manager, ready once we start development in our project. These repositories will help manage changes in our project among all of our members. We will all be able to make changes once our project starts without interfering with anyone's work. Our team also discussed the possibility of using Trello to help keep track of our project's achievements and significant milestones. From experience, we can all say that Trello is an excellent tool for development management. Aside from these useful management

tools, we can all tell our team has utilized these to the fullest. We hope to come across any more valuable tools as we start development with our project!

2.6 Personnel Effort Requirements

TASK	ORDER	PERSON-HOURS
Research	1st	108
Environment Setup	2nd	18
Console Application	3rd	72
Web Service	4th	36
AI Application	5th	126

Table 2.1 Personnel Effort Requirements

All team members will initially take on each of the tasks. As the tasks get more in-depth, team members will choose a specific job to focus on. The tasks are ordered because some tasks depend on the previous implementation. For example, how the database is formatted depends on how the console application formats the IIS logs. The person-hours are estimates and will depend on how many obstacles our team runs into as we learn how to implement the project's tasks.

2.7 Other Resource Requirements

- Virtual environment for testing, provided by Cylosoft
- Microsoft Visual Studio, provided by Iowa State University

3 Design

3.1 Previous Work And Literature

There is a three-part process for our project. The first part involves grabbing Microsoft IIS files and parsing through them to get the necessary information; this is done using the console application. The second part consists of sending data to the web service Azure and ensuring that the data is processed and stored correctly. Lastly, the third and final part involves creating an AI with machine learning capabilities to detect attacks based on the Cloudflare API and add its own rules as it detects new types of attacks.

To grab log files from IIS programmatically, we'd have to use the ILogScripting COM Interface (Archiveddocs). As mentioned on the website by Microsoft

"If you want to create a custom component that implements the ILogScripting interface, you need to make automation (IDispatch) object implementing the methods of the ILogScripting interface.

Your interface should be designed and implemented to handle only one log file query session per module instance. Therefore, each call to ILogScripting::SetInputLogFile and ILogScripting::SetInputServerInstance should reset the record read state."(Archiveddocs)

The second part is relatively straightforward; we'd just check the database to ensure that the IIS log files were correctly parsed and formatted to fit; we could query this using SQL.

Lastly, the AI we create will be the most challenging part of the project and will take most of our time. We will be using machine learning and the Tensorflow library to assist us. We could have used many different libraries like Keras Python, Theano Python, Scikit-learn python, and many more ("Top 8 Pytho..."). However, we decided to go with Tensorflow because it seemed to have the most resources available. Also, TensorFlow syntax is considered more straightforward than the others. When combining those two factors and the power of TensorFlow, the decision was easy. The best resource we've found has been the TensorFlow website; they have tutorials and guides for anyone ranging from the beginner level to advanced ("Machine Learning...").

There are many distinguished web security companies out there that are advanced. These companies include Crowdstrike, Gigamon, Fortinet, and many more (Admin). Crowdstrike has been used for many high profile cases, including the controversy surrounding the 2016 Russian investigation (Admin). Crowdstrike uses AI to detect threats similarly to our project ("About..."). Although we may not be able to compete with as much funding and resources as Crowdstrike, we can do our best to create an efficient and self-sufficient AI through machine learning.

3.2 Design Thinking

We, as a group, based our design thinking on the perspective of the client. The project was described into three major parts that build on top of each other. The group researched to find the best resources and programming languages to ensure a high-quality product. We have been deciding based on what is most compatible and efficient to help in the progression of the project and the client's perspective. Our console app takes in IIS files and will require it to be written in a specific programming language. Our AI aspect of the project will need to be programmed with machine learning techniques to ensure proper functionality. Each part of the project design is primarily associated with specific resource and language requirements, so group design choices were focused on compatibility and efficiency as a whole.

3.3 Proposed Design

As mentioned in previous sections, the work we need to complete can be broken into a couple of components.

- Web Server
    - This is the webserver that contains the website our project is in charge of protecting. Our task should be able to work for various web servers, so I will not spell out the details here.

- Console Application
    - This application will be responsible for finding, parsing, and sending relevant information from IIS log files to the Azure web service. We have a prototype version, written in Python, that can parse IIS logs and extract the relevant data. It does not yet find log files or send relevant information. This application will make use of REST protocols for transmitting data.
- Azure web service
    - This web service will consist of a database for storing IIS log information relevant to the AI. This will use REST protocols for transferring data between the database, the console application, and the AI. Since we are using Azure, the database will be scalable should the number of web servers under our protection increase.
- AI
    - The AI will be responsible for analyzing the relevant data from the database and creating new Cloudflare rules based on that data. It will also make use of REST protocols in receiving data and updating the Cloudflare rules.
- Cloudflare WAF
    - This is the firewall that will block unwanted traffic from reaching the webserver. Its rules will be updated in real-time by our AI through its API. Administrators will also set rules manually in case of an incorrect assumption by the AI or an update to Cylosoft standards.

3.4 Technology Considerations

-Visual Studio IDE

- Strengths-
    - Created by Microsoft and is compatible with Azure, SQL, Python
    - Easily manage frameworks
    - Most members of our team are familiar with the UI of Visual Studio.
- Weaknesses-
    - Can be slow at times
    - Uses a large amount of memory that results in slowing down your computer.

Alternatives to Visual Studio include IntelliJ, Notepad++, Eclipse, and many more. We decided to go with Visual Studio because it came from Microsoft and compatible with Microsoft services such as IIS.

-Python Language

- Strengths-
    - Python is capable of interacting with most other languages, thanks to the Python Package Index.
    - Has an extensive standard library, which results in a reduction of code length
    - Python is open source
    - Python is Object-Oriented and has its unit testing framework.

- Weaknesses-
    - Python executes with help from an interpreter, which causes it to slow down.
    - Has design restrictions that are often reported by other Python developers.

Alternatives to this language are C, Java, C++, and many other languages. However, we chose to work with Python because it is a growing language and has much documentation and framework support. After researching AI, we realized that most of the material was done in Python.

-TensorFlow (AI Library)

- Strengths-
    - High Performance
    - Large Community Support
    - Highly Parallel can pipeline easily
    - Great Graph visualizations
    - Frequent updates, backed by Google
- Weaknesses-
    - Benchmark tests are low
    - Very low level with a high learning curve
    - The unique structure resulting in potential difficulty while debugging
    - No windows support
        - Can be installed on windows through the use of Python package library

Alternatives to TensorFlow include Keras Python, Theano Python, Scikit-learn python, and many more. We decided to go with TensorFlow because of its high performance and great visual tools. There is also a lot more information online for us to reference and learn.

3.5 Design Analysis

So far, our design has worked. Our console application can read IIS logs and output relevant information promptly. However, as we continue the project by completing our weekly sprints, we may need to modify our design. Each week, team members will go over their progress from the week before. If any aspect of the system needs to be changed, all members will discuss and choose a new plan. We have not yet had to change the design, but based on experience and the Agile set of best practices, I believe this process will be sufficient to update the system as needed.

3.6 Development Process

We, as a team, are following the *agile* development process for this project. With this project primarily being software-based, we unanimously agreed this was the appropriate choice. We had already implemented continuous planning and collaboration with our project in the beginning stages of our development. We have produced realistic timelines and team expectations to produce a high quality finished product. With each of us bringing different perspectives to this end goal, we

are continuously learning and working together to make sure everything is in order. Once we hit the prototype stage, our group will gather feedback and adapt to the necessary changes that may occur in this development process.

3.7 Design Plan

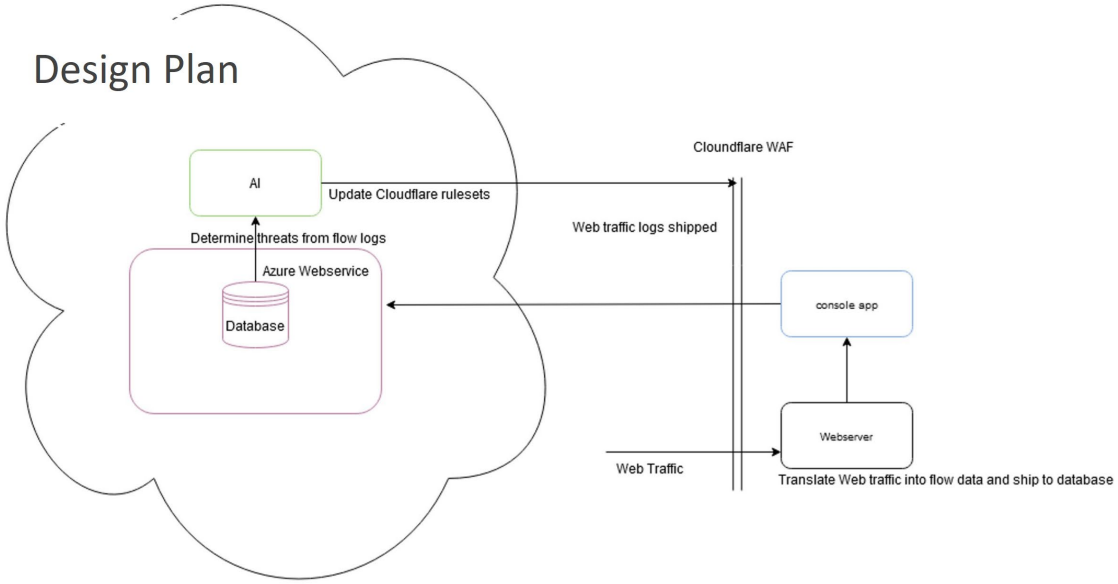

Figure 3.1 Design Plan

(For a full list of requirements, see section 1.4)

- Meeting Functional Requirements:
  - The web server and console application modules will meet the functional needs of generating log files for web requests and collecting log files.
  - The Azure web service database module will meet the functional requirement of storing and processing log files.
  - The AI Cloudflare WAF modules will meet the functional requirements of monitoring and updating usage rules and denying access to suspicious users.
- Meeting Non-Functional Requirements:
  - The Azure web service module will be active at all times to keep the system running for as much uptime as possible.
  - The database and console application modules will not have limits on the number of logs that they can process to ensure that the system will handle large volumes of traffic and be easy to scale.
  - Security checkpoints will be implemented on the database and console application modules to ensure that it can only be accessed by approved personnel.

- The Cloudflare module will fulfill the non-functional requirement of blocking suspicious network traffic.

4 Testing

Testing for our project will be critical. To accomplish this, testing will be broken down into the following types of tests: Unit Testing, Interface Testing, and Acceptance Testing.

Each type of testing will follow roughly the same procedure. Our team will come up with what needs to be tested. Upon agreeing on what requires testing, tests should be written to help determine if what is tested performs as intended. When the team is done writing tests, the next step is to carry out the tests on the intended units, interfaces, and systems involved. After the tests come back, the team will meet to discuss what went as expected and what didn't. From this meeting, we can decide on the next steps to perform to get everything working correctly. Keeping accurate documentation on this whole process will also be vital in ensuring everyone is on the same page about testing that has happened and directions to take proceeding the meeting.

4.1 Unit Testing

Our project consists of three main components or units that need to be tested in isolation. The first is the console application; it needs to be tried separately from the rest since it will need to be fully functional on its own before the rest of the application can be tested.

The second unit that will need to be tested is the Azure web service that handles our application's data flow. The web service will need to be tested in isolation to ensure the data is getting transformed correctly. This will make sure that the information being fed into our web service is error-free.

The final isolated test will be on the AI we are creating to build Cloudflare rules. Our AI will need to be tested throughout the development and training process for the algorithm to develop. The AI model will also need to be tested after the model is done to ensure the model is correct.

4.2 Interface Testing

Each of our three components will be connected to at least one of the other parts, which means the interfaces between them will need to be tested. One crucial interface is the web API that sits between the console application and the web service. The interface must be tested not only for correctness but also for efficiency. We will test the two components by giving the console application a known test file and then checking the web service's database to make sure the file was interpreted and stored correctly.

There will be four interfaces in our project; The one discussed in the previous paragraph, an interface between the web service and the database, one between the web service and the AI algorithm, and one between the AI and Cloudflare.

4.3 Acceptance Testing

Our application's end goal is to block irregular and potentially malicious traffic on the servers of our client. After testing that our code compiles correctly and runs without errors, we will try our application functions properly by simulating suspicious requests to the server and observing how the application responds.

We will also ask our client to contribute some suspicious connections to test our application. This will ensure that our application will satisfy our client's expectations and provide a more diverse set of tests that will more effectively verify our project's requirements.

4.4 Results

Testing for our team has been strictly for our research and understanding of the project components. We have begun individually testing out tools and methodologies that will make our lives easier when it comes time to get hands-on with the systems we will be standing up. Major research topics we have been focusing on have been revolving around IIS logs generated from the web server and different approaches to the AI element of our project. As we meet each week, our discussions regarding these topics have got increasingly technical, indicating that our progress on getting a handle on these topics has increased.

Citations

About CrowdStrike: Cloud-Native Endpoint Protection. August 5th 2020, www.crowdstrike.com/about-crowdstrike/.

Admin. Admin. July 22nd 2020, www.thesoftwarereport.com/the-top-25-cybersecurity-companies-of-2019/.

Archiveddocs. "IIS Logging Overview." Microsoft Docs, docs.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms525410(v=vs.90).

"Machine Learning Education : TensorFlow." TensorFlow, www.tensorflow.org/resources/learn-ml.

"Top 8 Python Libraries for Machine Learning & Artificial Intelligence." Hacker Noon, October 20th. 2020, hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-y08id3031.