

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: November 15, 2020//V3

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
- IIS log files and general web traffic moderation

Table of Contents

1	Introduction	
1.1	Acknowledgement	5
1.2	Problem and Project Statement	5
1.3	Operational Environment	5
1.4	Requirements	5
1.5	Intended Users and Uses	6
1.6	Assumptions and Limitations	6
1.7	Expected End Product and Deliverables	7
2	Project Plan	8
2.1	Task Decomposition	8
2.2	Risks And Risk Management/Mitigation	8
2.3	Project Proposed Milestones, Metrics, and Evaluation Criteria	9
2.4	Project Timeline/Schedule	10
2.5	Project Tracking Procedures	11
2.6	Personnel Effort Requirements	12
2.7	Other Resource Requirements	12
3	Design	12
3.1	Previous Work And Literature	12
3.2	Design Thinking	13
3.3	Proposed Design	13
3.4	Technology Considerations	14
3.5	Design Analysis	15

3.6	Development Process	15
3.7	Design Plan	16
4	Testing	17
4.1	Unit Testing	17
4.2	Interface Testing	17
4.3	Acceptance Testing	18
4.4	Results	18
5	Citations	19
6	Closing Material	19
6.1	Conclusion	19
6.2	References	19

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). Cylosoft uses Cloudflare as its firewall, which has both Cloudflare generated 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 console application that gathers logs and sends them in real-time to a cloud database. The database will then be queried by machine learning code, which will look for potential bots, hackers, and spammers and generate firewall rules accordingly. The project's output will be real-time WAF rules for the Cylosoft web servers, 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, etc.) 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.
- The system should be able to run in a "log-only" mode for testing on a live site.

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, etc. 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 this system will be provided by Cylosoft.
- Team members working on the project will be able to access the resources needed to accomplish the project.
- 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 in further understanding of the design process we used. A console based application that 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 that will have access to the processed IIS log information and will 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 provide the date of all meetings held and the discussions held.

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 contain detailed instructions on how to use the console application as well as make modifications to it. The LICENSE file will contain any and all legal documentation associated with Frameworks/IDE's/API's and other relevant material used in the development of this project.

Project Design Document-

- We will provide a detailed plan and description of the process we used along the development of this application. Many things will be outlined in this document, some of which includes: 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 in a proper format 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 be able to find “obvious” threats that Cloudflare would normally not catch. These “obvious” threats will be recorded as rules to improve future responses to that type of attack. AI will then remove threat.

Test-Case-

- We will provide an example of the log files association to a website as well as 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 a lot of different things that will have to come together for this task to work properly. This application will have to make connections between the Microsoft IIS log files as well as our web service. To add to that, it also has to look at the IIS log files and extract important 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 is unable to correctly identify malicious traffic to block, we will need to find a different approach to the AI. This will likely involve figuring out a different machine learning/AI learning algorithm that will be able to 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 properly without failure might prove to be difficult. As testing

2.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

Proposed Milestones:

Research-

Main Task:

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

Environment Setup-

Main Task:

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

Console Application -

Main Task:

- Console-based application will find current log file for site being monitored and send the log file to the web service
 - Subtasks:
 - Establish connection between code and IIS log files
 - Establish connection between code and web service
 - Find log file specific to 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 database
 - Subtask:
 - Establish connection between code and web service
 - Parsed log file will store data in correct fields in 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 will look through cloudflare rules and remove user if conditions met
 - Machine learning algorithm will add new rules not existing within Cloudflare

2.4 PROJECT TIMELINE/SCHEDULE

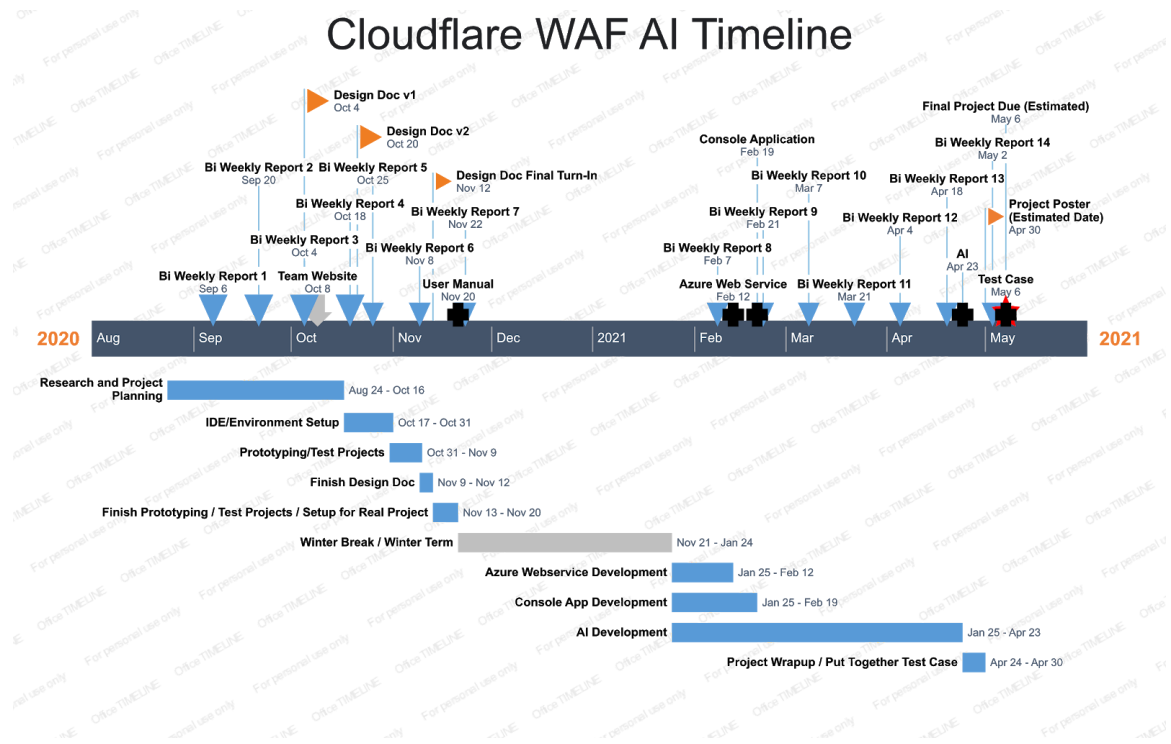


Figure 2.4 Project Schedule

Note on the gantt chart: Listing all subtasks made the above chart unreadable, so the timelines for each subtask 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 31
- 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 12
- 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 and third weeks will be used to set up automatic filling of the database with test data sent to the webservice.
- Console Application
 - The console application should be able to find log files specific by the end of the first week.
 - By the end of the third week, the console app should be able to send the relevant data to the webservice.
- AI
 - By the end of the second week the AI application should be able to 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 client and manage 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 gives us the opportunity to brainstorm and finalize ideas for our project. We have also set up a Github repository and GitLab manager so it's 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 achievements and major milestones of our project. From experience, we can all say trello is a great tool for development management. Aside from these useful management tools, we can all say our team has utilized these to the fullest. We hope to come across any more useful 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

Each of the tasks will initially be taken on by all team members. As the tasks get more in-depth, team members will choose a specific task to focus on. The tasks are ordered because some tasks depend on the previous implementation. For example, the way that 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 tasks for the project.

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 are many distinguished web security companies out there that are really 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 makes use of AI to detect threats in a similar fashion to our project (“About...”). Although we may not be able to compete with as much funding and resources as CrowdStrike we can definitely do our best to create an AI that is efficient and self-sufficient through the use of machine learning.

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

In order 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 create an 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 properly parsed and formatted to fit, we could query this by using SQL.

Lastly, the AI we create will be the most difficult part of the project and will take the majority of our time. We will be using machine learning and Tensorflow library to assist us. There are many different libraries we could have used 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 simpler 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...”).

3.2 DESIGN THINKING

We as a group, based our design thinking in the perspective of the client. The project was described into three major parts that build on top of each other. The group altogether conducted research 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 perspective of the client. Our console app takes in IIS files and will require it to be written in a certain programming language. Our AI aspect of the project will need to be programmed with machine learning techniques to ensure a proper functionality. Primarily each part of the project design is 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 web server that contains the website our project is in charge of protecting. Our project should be able to work for a variety of different 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 sending data.

- Azure web service
 - This web service will consist of a database for storing IIS log information relevant to the AI. This will make use of 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 which will block unwanted traffic from reaching the web server. It's rules will be updated in real time by our AI through its API. Administrators will also be able to 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 comes from Microsoft and is made to be 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 a large standard library which results in reduction of code length
 - Python is open source
 - Python is Object Oriented and has its own 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 a lot of documentation as well as framework support. Also, after researching AI we came to the realization 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 high learning curve
 - Unique structure resulting in potential difficulty while debugging
 - No windows support
 - Can be installed on windows through 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 from.

3.5 DESIGN ANALYSIS

So far our design has worked. Our console application is able to read IIS logs and output relevant information in a timely manner. 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 design 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 past experience and the Agile set of best practices, I believe this process will be sufficient to update the design 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. In the beginning stages of our development, we had already implemented continuous planning and collaboration with our project. 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 be able to gather feedback and adapt to the necessary changes that may occur in this development process.

3.7 DESIGN PLAN

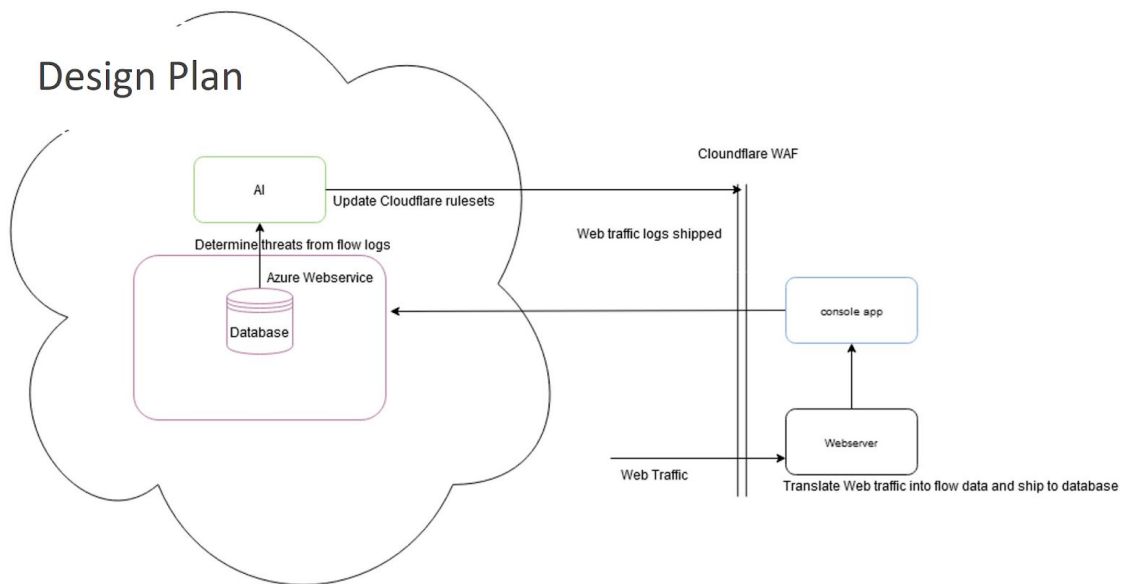


Figure 3.1 Design Plan

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

- Meeting Functional Requirements:
 - The web server and console application modules will meet the functional requirements 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 be capable of handling 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 as well 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 come to a decision on next steps to perform to get everything working properly. Keeping accurate documentation on this whole process will also be vital in making sure 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 tested 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 training 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

The end goal of our application 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 test 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 the expectations of our client, and will also provide a more diverse set of tests, which will more effectively verify our project's requirements.

4.4 RESULTS

Testing for our team has been strictly for our own 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, the discussions we have regarding these topics have got increasingly more technical indicating that our progress on getting a handle on these topics has increased.

5 Implementation

- Web Server
 - Setup access (user/password).
 - Setup firewall rules for tunnel to Azure web service.
 - Make sure log files are being written to where we expect them to be.
- Console Application
 - IDE setup in Visual Studio with Python
 - Created a parser for the IIS log file example provided
 - Setup a github to store our console application code
- Azure web service
 - Found resources online that can be used to help us when using Azure
- AI
 - IDE setup in Visual Studio with Python
 - Configure the AI to monitor and scan the patterns of attacks.
- Cloudflare WAF
 - Setup API authorization
 - Make sure we can create a test rule using the API
 - Review existing rules set up by Cylosoft

6 Closing Material

6.1 Conclusion

At this point in the project, we have completed research that will be critical to our implementation phase of our project next semester. We have begun writing code and testing for our console application and database, which we will be able to build on during the implementation phase. While we are slightly behind where we wanted to be at this point, we have a solid groundwork that can be extended to fulfill all our requirements. With an extended winter break in the coming months due to Covid-19, we plan on staying in touch as a team and completing initial versions of the console application and database. This will give us a very strong start to our implementation, as we work toward fulfilling all functional and nonfunctional requirements next semester.

6.2 References

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

Admin. Admin. 22 July 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\(v=vs.90\)](https://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, 20 Oct. 2020,
hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-yo8id3o31.