CGI

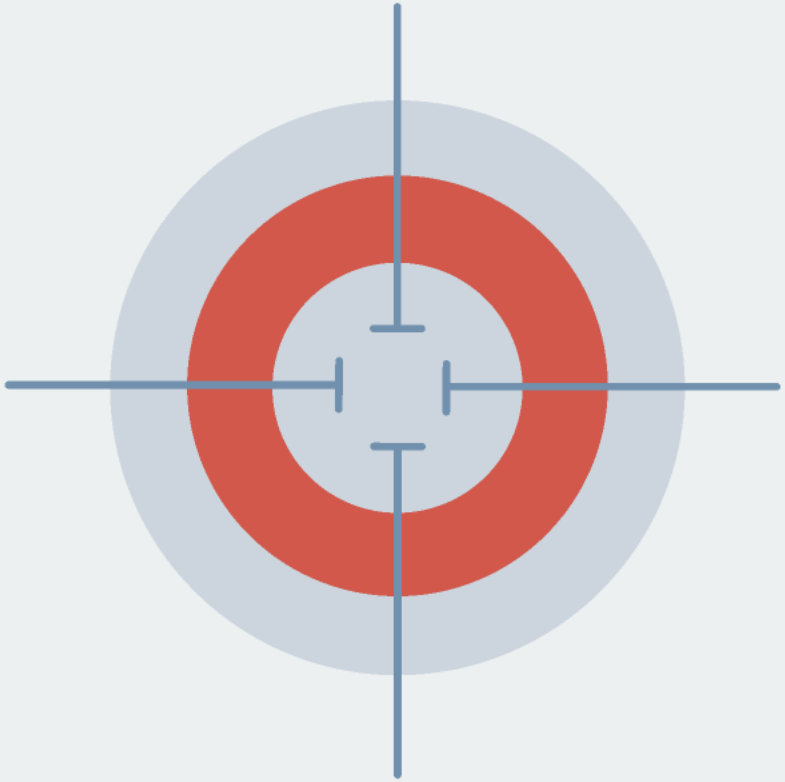
DNS Url Filtering System

Enhancing Web Security Through Advanced Filtering Technology

Ghazal Ghafari

July 3, 2024

Business Question



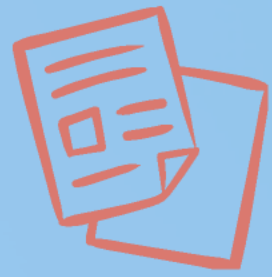
- The client is interested in enhancing their information security.
- They prefer using AI over traditional blocklists to identify malicious URLs.
- The goal is to implement a system that can accurately classify and block malicious URLs while minimizing disruptions to legitimate web usage by employees.

The primary business question

"How can we improve our information security system to effectively identify and block malicious URLs using artificial intelligence, without imposing significant restrictions on employees?"

Agenda

1. Project Overview
2. Implementation Plan
3. Cloud Infrastructure
4. Roles and team members
5. System Setup and Configuration
6. Initial Data Analysis
7. Machine Learning Integration
8. Decision Explanation and Monitoring
9. Product Features & Product Comparison
10. Roadmap Timeline
11. Q&A Session



Project overview

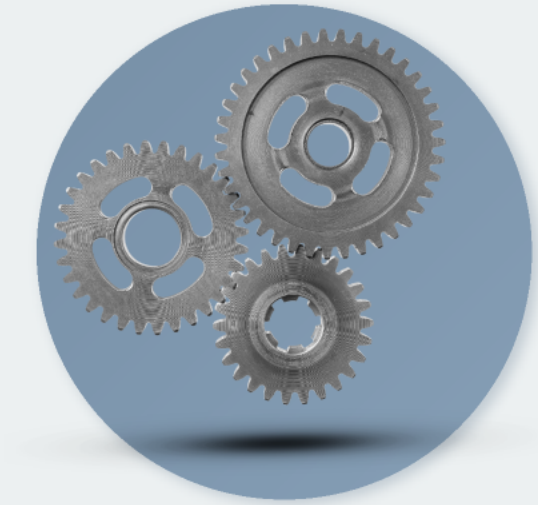
What is DNS Filtering?

A method of blocking access to specific websites or IP addresses based on DNS queries.
Enhances network security by preventing access to malicious or inappropriate content.

GOALS

1. Enhance Information Security
2. Leverage AI for URL Classification
3. Proof of Concept and Implementation Plan
4. Real-time Monitoring and Scalability

Implementation Plan



Technical requirements

Chosen Technologies:

- **Flask**
- **dnsmasq**
- **Prometheus**
- **Grafana**
- **Google Cloud**

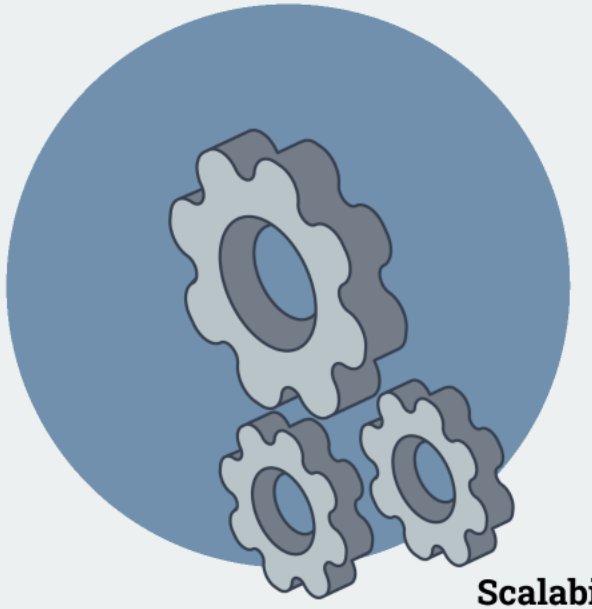
Hardware Components:

- **Servers:** High-performance servers to host the DNS filtering application and machine learning models.
- **Networking Equipment:** Routers and switches to manage network traffic effectively.

Software Components:

- **DNS Filtering Application:** Built with Flask to classify URLs in real-time.
- **Machine Learning Models:** For adaptive URL classification and automated threat detection.
- **Monitoring and Alerting:** Using Prometheus and Grafana for real-time system health monitoring.

Cloud Infrastructure



Google Cloud Platform (GCP):

- **Compute Engine**: Virtual machines for running the DNS filtering application and machine learning models.
- **Cloud Storage:** For storing datasets, logs, and model checkpoints.
- **Cloud Pub/Sub:** Messaging service for real-time event handling and communication.
- **Cloud Monitoring:** Integrated with Prometheus and Grafana for comprehensive monitoring and alerting.
- **VPC (Virtual Private Cloud):** For secure and scalable network management.

Scalability and Reliability:

- **Auto-Scaling:** Automatically adjusting resources based on demand.
- **Load Balancing:** Distributing traffic evenly across multiple instances to ensure reliability.
- **High Availability:** Redundant systems to minimize downtime and ensure continuous operation.

NotebNote: While this project plan utilizes Google Cloud Platform (GCP), it is equally feasible to implement the solution using other cloud platforms such as Amazon Web Services (AWS) or Microsoft Azure, depending on specific requirements and preferences.

Introduce the team members and their roles

- **Project Manager :** oversees project and ensures milestones are met.
- **DevOps Engineer:** manages deployment and cloud infrastructure on Google Cloud.
- **Data Engineer:** handles data preprocessing and integration.
- **Machine Learning Engineer:** develops and fine-tunes the machine learning model.
- **Security Analyst:** ensures system security and compliance.

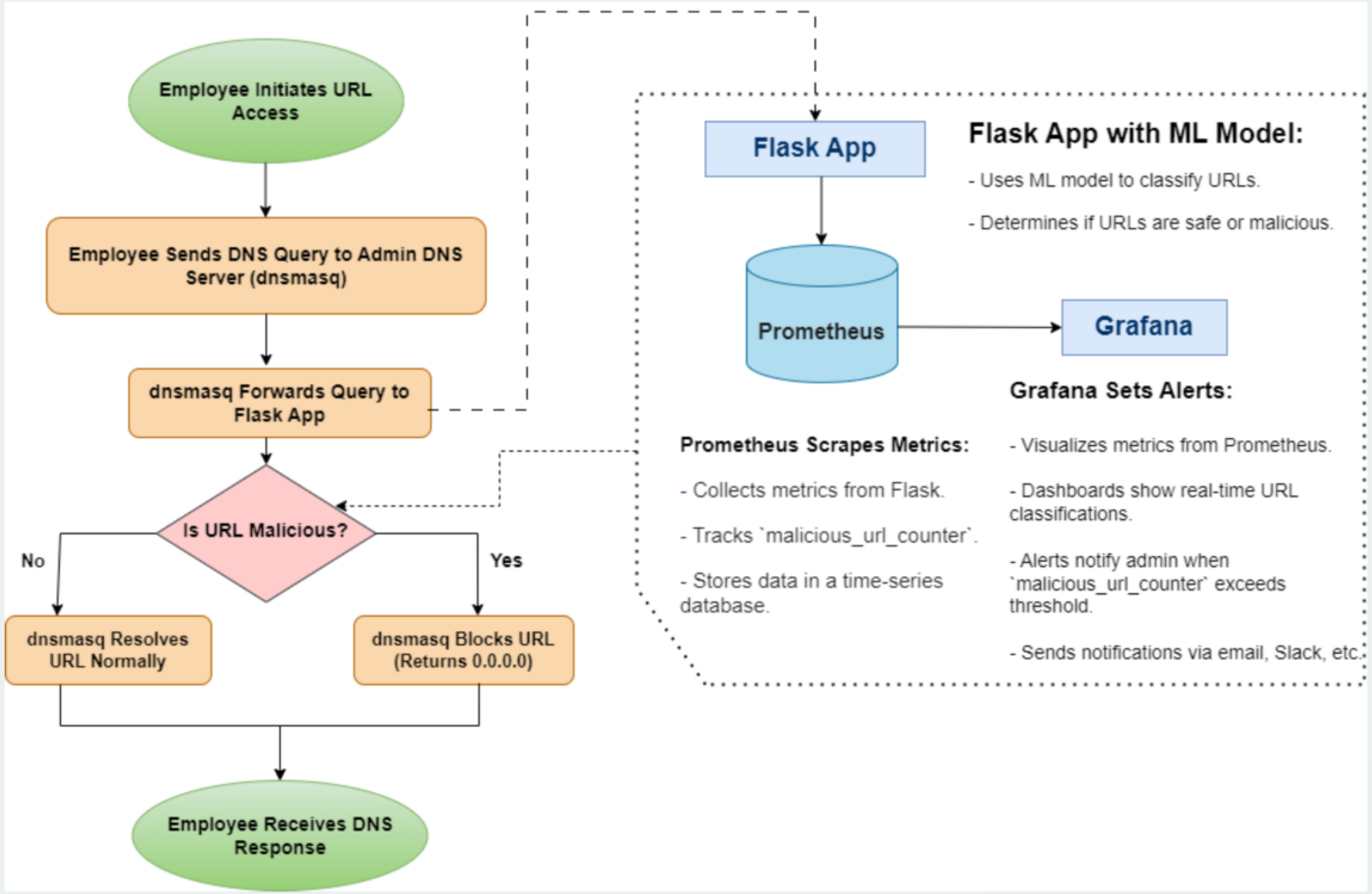
Note:

Each role does not have to be fulfilled by a different person. A person can also have two roles depending on their skills and the overall project length. For a simple AI filtering solution, a 5-person team setup might be excessive and expensive for the client.

System Setup and Configuration

- **Install and configure Flask application and dnsmasq on the Google Cloud VM instance.**
- **Initial configuration steps:**
  - **Flask:** A micro web framework used for developing the URL classification application.
  - **dnsmasq:** A lightweight DNS forwarder used for DNS query redirection to Flask.
  - **Deployment:** Steps to deploy Flask and dnsmasq on Google Cloud.

How the system works

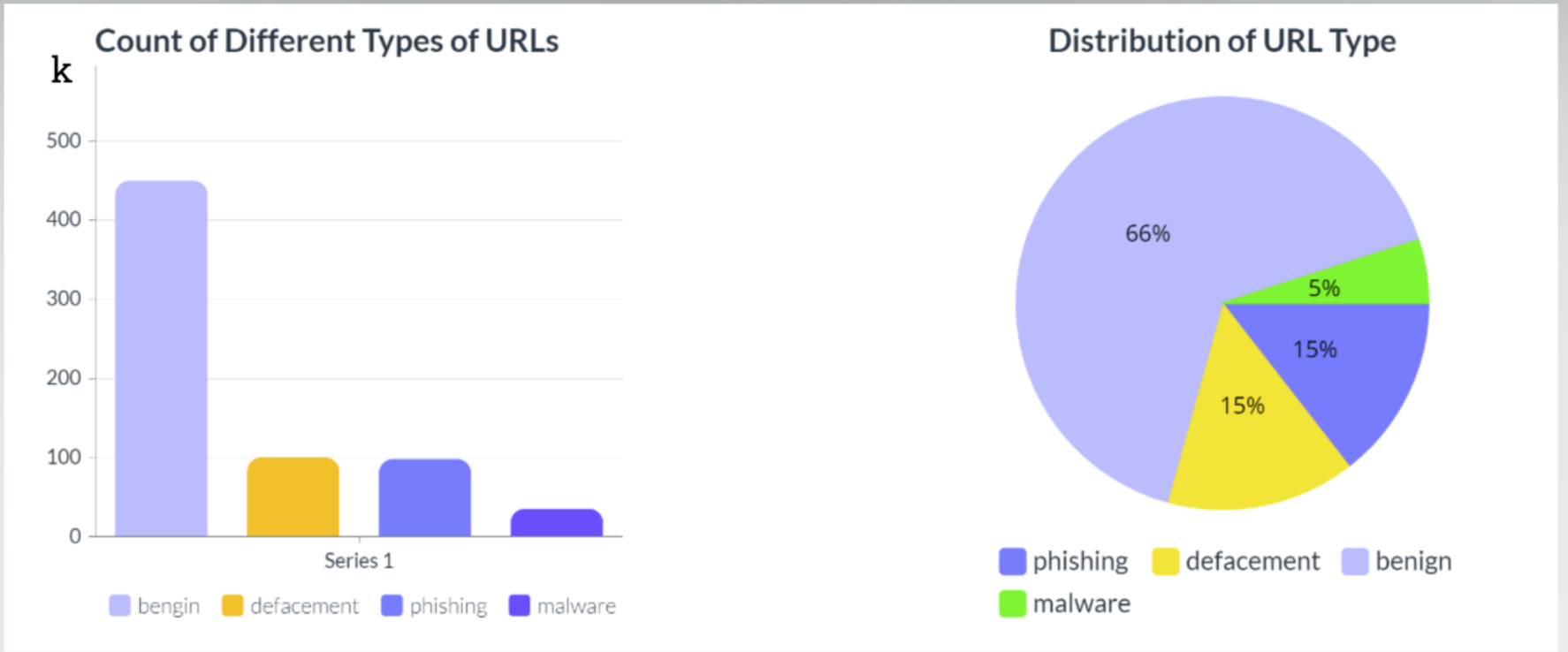


Initial Data Analysis

The dataset contains URLs and their corresponding types:

URL: The URL to be classified.

Type: The type of the URL (benign, defacement, phishing, malware).



Count of Different Types of URLs:

- ❑ The bar plot shows the count of URLs for each type.
- ❑ The dataset is highly imbalanced, with a majority of the URLs being benign.

Distribution of URL Types:

- ❑ The pie chart illustrates the proportion of each URL type.
- ❑ Benign URLs constitute 65.7% of the dataset, followed by defacement, phishing, and malware.

Initial Data Analysis

Dataset Inspection:

Source: The dataset was sourced from Kaggle, containing labeled examples of malicious and benign URLs.

Content: The dataset includes features such as URL length, presence of special characters, and domain information.

Data Quality: Initial inspection shows a diverse range of URLs with varied characteristics, suitable for training a machine learning model.

Feasibility & Risk:

Feasibility: The dataset appears comprehensive and well-suited for developing a machine learning model to classify URLs. The features provided are relevant for identifying malicious patterns.

Potential Risks:

- **Data Imbalance:** There might be an imbalance between malicious and benign URLs, which could affect model performance. Techniques like oversampling or undersampling may be needed.
- **Feature Relevance**: Some features may not be as predictive of malicious activity and could introduce noise. Feature selection and engineering will be crucial.
- **Evolving Threats**: Malicious tactics evolve over time. The model will need regular updates with new data to maintain its effectiveness.
- **False Positives/Negatives**: Incorrect classifications could lead to blocking legitimate sites or allowing malicious ones. Continuous monitoring and model retraining will be necessary to minimize these risks.

Machine Learning Integration

Process 1: Initial Data Analysis

- preprocess the dataset of URLs.
- Visualize data
- Use bar plots and pie charts for insights.

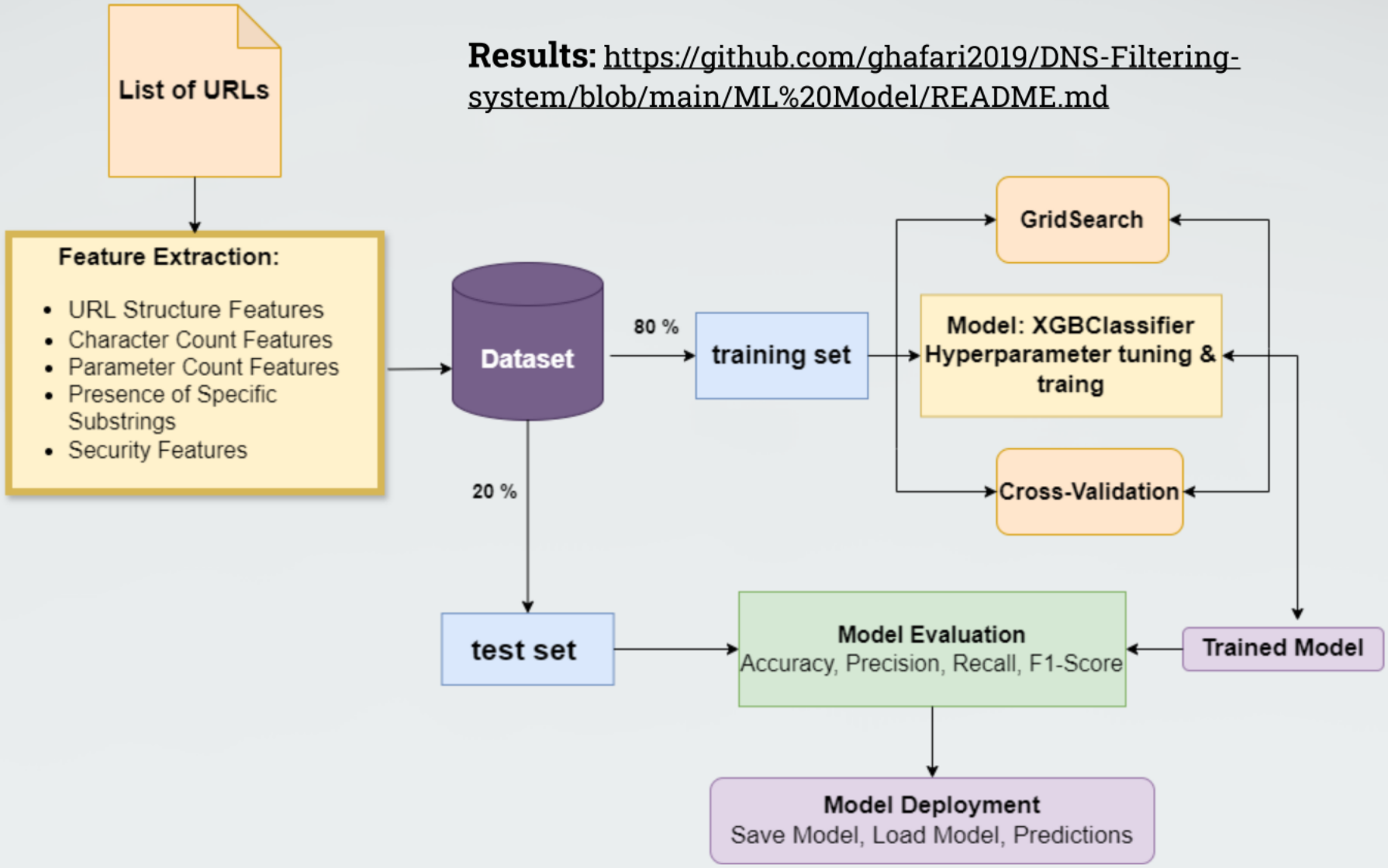
Process 2: Feature Extraction and Model Training

- Extract features from URLs
- Train the XGBoost classifier with hyperparameter tuning and cross-validation.
- Evaluate using accuracy, precision, recall, and F1-score.

Process 3: Model Evaluation and Deployment

- Assess model performance on the test set.
- Save and prepare the trained model for deployment.
- Integrate into the DNS filtering system for real-time classification

Results: https://github.com/ghafari2019/DNS-Filtering-system/blob/main/ML%20Model/README.md



Decision Explanation and Monitoring

Decision Explanation:

- **Transparent Logging:** Detailed logs of URL classification decisions.
- **Justification Reports:** Automated reports explaining why a URL was classified as malicious.
- **Incident Analysis:** Tools to analyze and understand false positives and negatives.

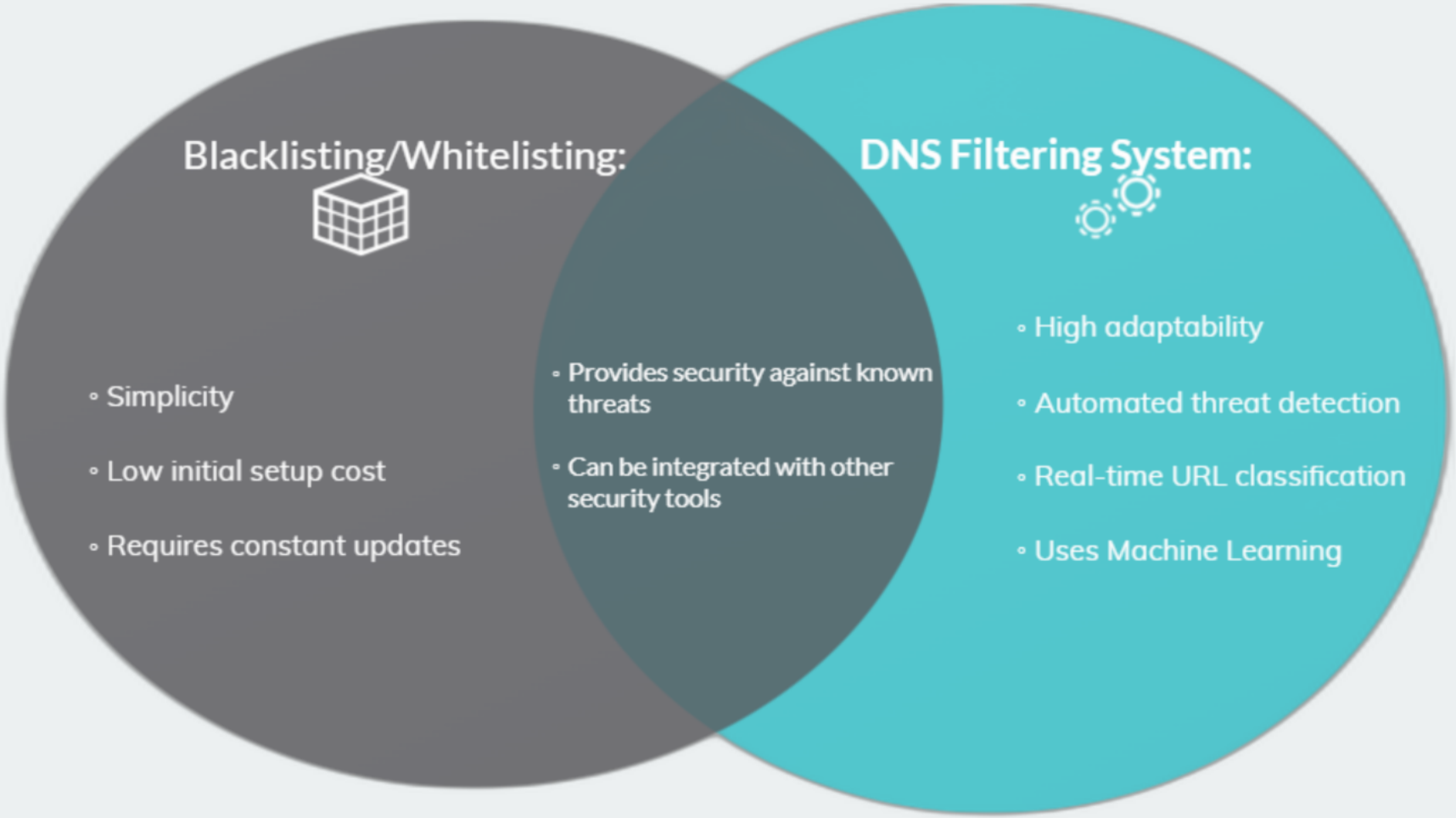
Product Comparison

Comparison of Filtering Methods: This slide provides a comparison of different DNS filtering methods, evaluating them based on key criteria such as simplicity of implementation, effectiveness against new threats, centralized control, comprehensive filtering, and adaptability over time.

Filtering Method	Simple Implementation	Effective Against New Threats	Centralized Control	Comprehensive Filtering	Adaptive & Improves Over Time
Blacklisting/Whitelisting	√	×	√	×	×
Signature-Based Filtering	√	×	×	√	×
Proxy-Based Filtering	√	×	√	√	×
Content Filtering	×	√	×	√	√
Machine Learning-Based Filtering	×	√	×	√	√
Our DNS Filtering System	√	√	√	√	√

VENN DIAGRAM

Comparison of Approaches: The Venn diagram contrasts the traditional blacklisting/whitelisting method with the DNS filtering system, highlighting their unique advantages and shared benefits.

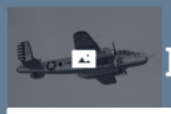


Timeline

2024-08-14: milestone (deliverable): documented and signed-off (by client) requirement list

Initial Setup

2024-08-15
Setup Flask, dnsmasq, initial ML model, and Prometheus



Deployment

2024-10-10
Deploy to production, continuous monitoring, audits

Hypercare Phase

2024-10-24
Bugfixing, stabilization, maintenance

2024-08-01
Requirement analysis with CGI experts and client

Business Analysis

2024-09-19
integrate Grafana, enhance scalability

Optimization

2020-10-09: milestone : code-freeze (software development end)

2024-11-20
End of Project

github link of Project:

https://github.com/ghafari2019/DNS-Filtering-system

Ghazal Ghafari