

# Apache Tomcat and Beyond



Team: Open Source Web Services


Members: Anthony Bustamante, Nur Dincer, Yang Yu, and Sunjil Gahatraj



# Web Application Firewall- Requirements

---

Functional	Non- Functional
<ul style="list-style-type: none"><li>• Determining whether a web application request is benign or malicious is a crucial aspect of web application security. This is where a web application firewall (WAF) comes into play, as it helps analyze and filter incoming requests to identify and block potentially malicious ones while allowing legitimate traffic to pass through</li><li>• WAF can define rules based on specific conditions, such as URL patterns, HTTP headers, and request addresses.</li><li>• The Content Security Policy is a response header that informs the browser about the allowed sources of content on the page</li></ul>	<ul style="list-style-type: none"><li>• The WAF can scale horizontally to accommodate increasing traffic and growing application needs. This is accomplished by efficient scaling option provided by Google Cloud as an offering where both WAF application and trained model are hosted.</li><li>• The WAF is highly manageable. It can adopt different pattern and additional checks as well as use different machine learning model to validate request.</li></ul>

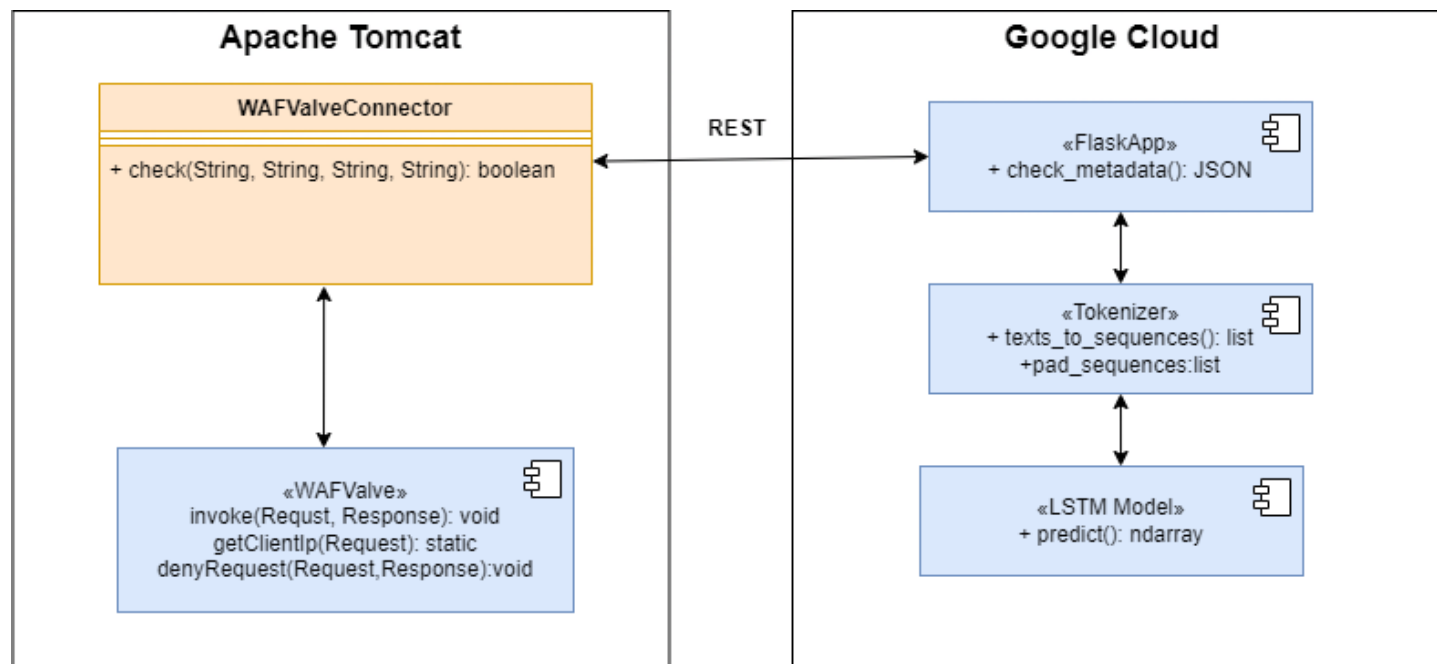


# Content Security Policy- Requirements

---

Functional	Non- Functional
<ul style="list-style-type: none"><li>• The Content Security Policy is a response header that informs the browser about the allowed sources of content on the page</li><li>• The SCPValve component should allow configuring various aspects of the Content Security Policy (CSP) by providing setter methods for properties such as defaultSrc, styleSrc, scriptSrc, mediaSrc, fontSrc, connectSrc, and imgSrc. These properties define the allowed sources for different types of content.</li></ul>	<ul style="list-style-type: none"><li>• Performance: The component should be efficient and have low overhead since it is invoked for each request. It should not introduce significant latency or resource usage.</li><li>• Security: The component should adhere to best practices for security. It should validate and sanitize user input to prevent any potential security vulnerabilities, such as injection attacks or unauthorized access to files.</li></ul>

# Web Application Firewall- Architecture

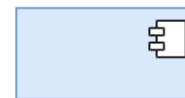


## Legends

Connector



Component





# Web Application Firewall- Architecture

---

## Request and Response Flow

- WAFValve.java (Component) intercept incoming web request
- WAFValve.java extract request details i.e protocol, domain, ip, and path
- WAFValveConnector.java (Connector) encapsulate these details into JSON data
- WAFValveConnector.java calls Flask Web Application Firewall Application (WAF) component hosted on Google Cloud using REST service
- Flask Application calls Tokenizer class responsible for tokenizing the text data
- Flask Application internally calls Long Short-Term Memory (LSTM) Model with tokenized data which represents the loaded LSTM model used for prediction. Response traverse same request path

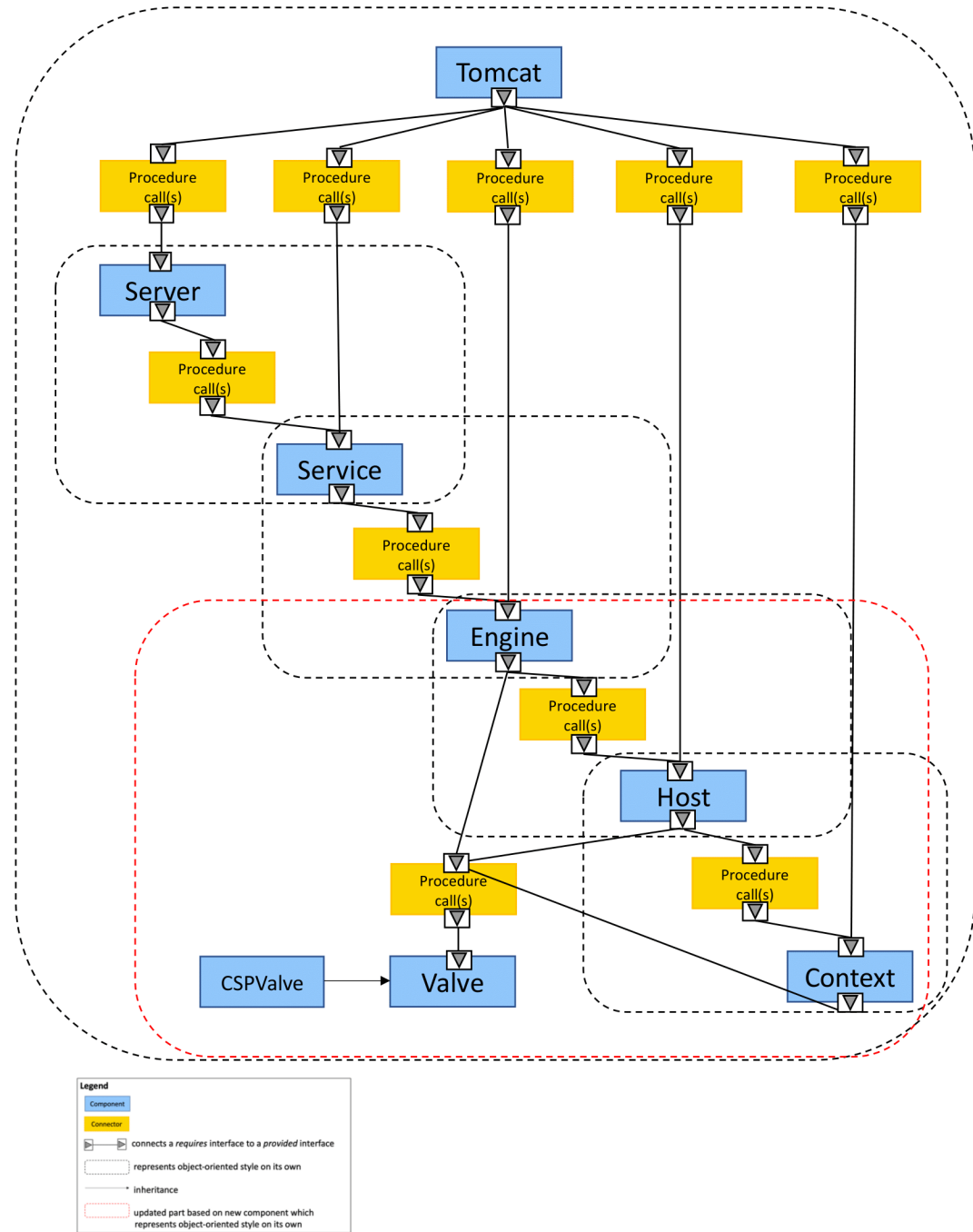
## Non-functional Requirement

**Flexibility:** LSTM is trained model and can be swapped with another compatible model

**Scalability:** WAF component is stateless and can be scaled horizontally as per need using Google Cloud scaling feature

**Security:** Mitigating potential security risks such as cross-site scripting (XSS) attacks.

# Content Security Policy- Architecture





# Content Security Policy- Architecture

---

## **Request/Response Processing**

- The CSPValve component is invoked during the request/response processing pipeline of the Tomcat server.
- When a request reaches the CSPValve, its invoke method is called, passing the current Request and Response objects.

## **Customizable Properties:**

- The CSPValve component provides setter methods for various properties that define the Content Security Policy (CSP) configuration.
- These properties include defaultSrc, styleSrc, scriptSrc, mediaSrc, fontSrc, connectSrc, and imgSrc.
- Developers can set these properties to specify the allowed sources for different types of content.

## **Security Value Validation**

- The CSPValve component includes a helper method called checkValues to validate the configured security values.
- It splits the values by spaces and checks if each value is present in the allowed\_values list, which contains the predefined set of allowed values.
- The method allows wildcard domains (starting with ".\*") for flexible domain matching.



# Source code

---

## Language

- WAFValve.java: Java
- WAFValveConnector.java: Java
- CSPValve.java: Java
- WAF Flask application: Python
- LSTM Model: Python

## Design Pattern

- Representational State Transfer (REST):  
Communication
- Client- Server: Apache Tomcat and WAF Flask  
Application
- Web Application Firewall(WAF): Security
- Object-Oriented: Inheritance (Valve interface)

Actual implementation of the WAF and CSP features align with as intended architecture proposed on G3.