

**WAF Module Implementation Overview**

**Language & Framework:**

* The WAF is implemented in Python using the built-in http.server and socketserver modules for HTTP handling, and the requests library for forwarding sanitized requests.

**Key Implementation Steps**

**1. HTTP Server Setup**

* The WAF listens on port 7070, acting as a reverse proxy between the client (e.g., BurpSuite) and your Go frontend server.
* It uses a threaded server (ThreadingTCPServer) to handle multiple concurrent requests efficiently.

**2. Request Parsing**

* For every incoming HTTP request, the WAF reads and parses:
  + All HTTP headers (case-insensitive)
  + The full request body (if present)

**3. Detection Logic**

* The core detection is implemented in the detect\_smuggling(headers, body) function.
* This function checks for all major HTTP Request Smuggling vectors:
  + Presence of both Content-Length and Transfer-Encoding headers (CL.TE/TE.CL)
  + Multiple or duplicate CL or TE headers (TE.TE/CL.CL)
  + Obfuscated header names (e.g., Transfer\_Encoding, Content-Length)
  + Malformed or conflicting header values (e.g., Transfer-Encoding: chunked, identity)
  + Chunked body not properly terminated
  + Mismatched Content-Length and actual body length

**4. Logging & Blocking**

* If any detection rule is triggered, the WAF:
  + Logs the suspicious request and the reason for detection to a log file (waf\_smuggling.log)
  + Responds to the client with a 400 Bad Request and a message indicating smuggling was detected
  + Does **not** forward the request to the backend

**5. Forwarding Clean Requests**

* If the request passes all detection checks:
  + The WAF forwards the request (with headers and body) to the Go frontend server using the requests library
  + The response from the Go frontend is relayed back to the original client

**6. Error Handling**

* If forwarding fails (e.g., backend unavailable), the WAF logs the error and returns a 502 Bad Gateway to the client

**Integration Points**

* **Upstream:**
  + The WAF receives all HTTP traffic from clients or security tools (e.g., BurpSuite)
* **Downstream:**
  + The WAF forwards only sanitized, validated requests to your Go frontend (localhost:7070)
* **Logging:**
  + All detections and errors are logged for auditing and debugging

**Extensibility**

* The detection logic is modular and can be extended to handle new smuggling techniques or additional HTTP protocol anomalies.
* Mitigation logic (e.g., header normalization, request rewriting) can be added in the same handler function after detection.

**Summary Table**

|  |  |
| --- | --- |
| Component | Functionality |
| HTTP Server | Listens for incoming requests on port 7070 |
| Detection Function | Inspects headers and body for smuggling signs |
| Logging | Records all detections and errors |
| Forwarding | Sends clean requests to Go frontend |
| Error Handling | Responds with errors if detection or forwarding fails |

This implementation ensures that all incoming HTTP requests are rigorously inspected for smuggling attacks before reaching your application infrastructure, providing a robust security layer.