WEB APPLICATION FIREWALL

By
Team ENIGMA
Amogh Avadhani - 110945879
Ankit Agrahari - 110946823
Prachi Poddar - 110815897
Sudeshna Pal - 110938222

Prepared in the partial fulfillment of the course
CSE 509 System Security

At
State University of New York – Stony Brook
Department of Computer Science

Table of Contents

1. Introduction

A **Web Application Firewall (WAF)** is an HTTP application firewall, which sits in front of the application server and monitors all the incoming traffic. The Web Application Firewall provides a layer of cover to the web server against various attacks. The WAF monitors and filters out malicious traffic, and only allows legitimate traffic to go through. The WAF can guard the server form known XSS attacks, SQL injection attacks and can also filter out requests that vastly deviate from previously seen requests. The WAF is independent of the server it is protecting and can be

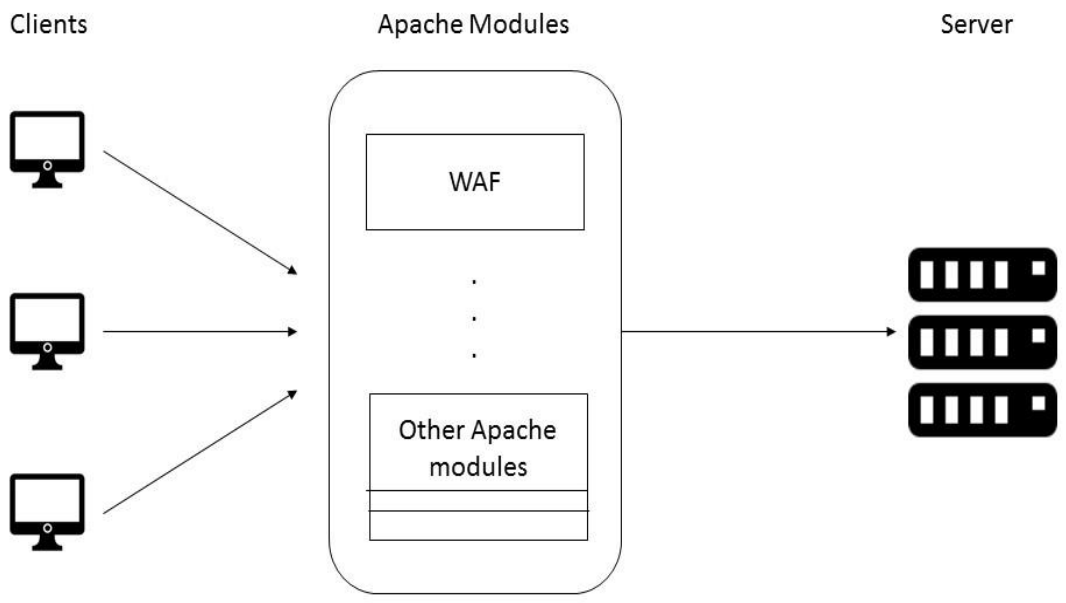


Figure 1.1

implemented as a Reverse Proxy for a server, or as an Apache module. The WAF in this project has been implemented as an Apache module for the Apache HTTP server.

2. High Level Design

Requests sent towards the server are first made to pass through the Web Application Firewall. The WAF checks the requests for malicious content and drops the request if found. The WAF allows only legitimate traffic to hit the server. The detection of malicious content in the request is done in two parts:

- **Signatures**

The WAF scans the request and checks for the presence of attacks based on the rules defined in the signatures. The signatures store contains rules for the WAF to check for, in an encoded format, against known attacks like XSS, SQL injection etc. If any new attacks become known, we can add the signature of that attack to the signature store based on the proper encoded format, which will then be picked up by the WAF for subsequent requests.
Examples for signatures:

```
HEADER:User-Agent,CONTAINS:"bot"
REQUEST_METHOD:GET,PARAMETER:*,CONTAINS:"union all select"
REQUEST_METHOD:POST,PARAMETER:foo,CONTAINS:"../../../../"
```

- **Anomaly Detection**

In anomaly detection, the WAF first learns how the legitimate traffic looks like. Based on this collected information, the WAF makes calculations about legitimate requests and proceeds to discard all the requests which far deviates from the norm.

Anomaly detection can be split into two phases, training phase and detection phase. In training phase, the WAF does not stop any requests allowing all the requests to hit the server. The purpose of this phase is only to collect information regarding the requests. The collected data is used to calculate the following parameters,

  - Maximum number of parameters for all requests across all pages.
  - Maximum number of parameters for each request.
  - Average length of every specific parameter for each request.
  - Character sets (allowed characters) of every specific parameter for each request.

In the detection phase, the WAF monitors the incoming requests based on the calculations. Requests which exceed the maximum number of parameters across all pages or for a page, are not allowed to go through. Requests that have parameter lengths which fall outside the range of *average +- 3 * standard deviation* are also dropped by the WAF. Requests with parameters having characters not seen in character set of the request parameter are ignored too.

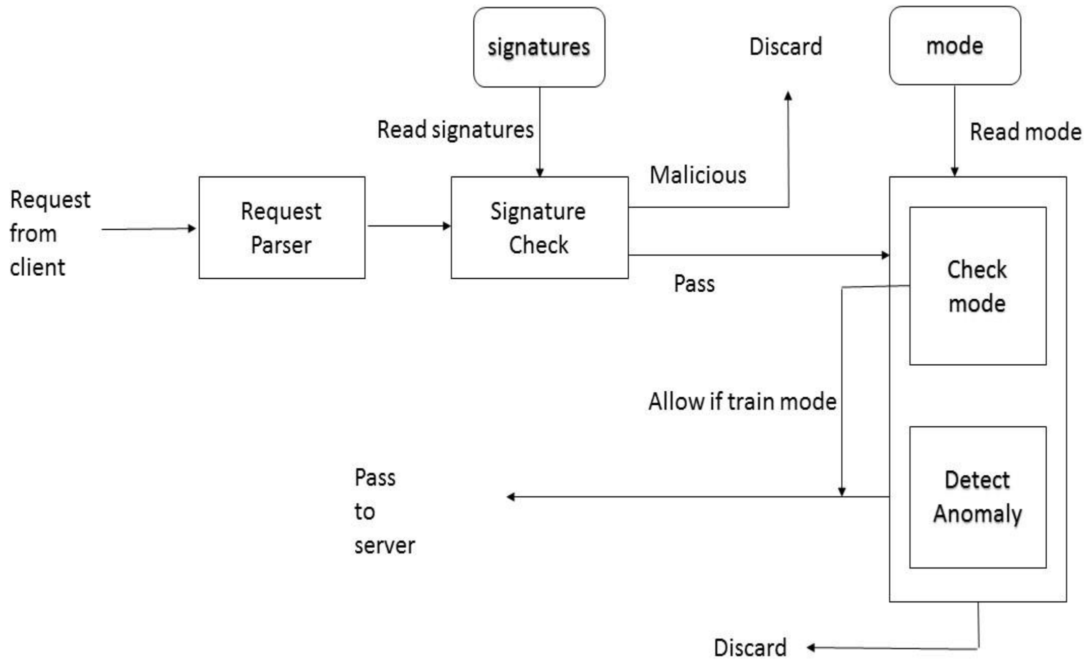


Figure 2.1

3. Low Level Design

The Web Application Firewall has been implemented as an Apache module. Apache has a modular design which enables us to write explicit functionality blocks for the server. The new module must be hooked on to the server at the appropriate location in the process flow. A hook function provided by Apache is used to implement this. Since we want our WAF module to be called at the beginning of every process request, we declare the hook using APR_HOOK_FIRST.

The two parts of filtering in the WAF are checking signatures and anomaly detection.

3.1. Signatures (mod_waf.c, waf_parse_signature.c)
By Amogh Avadhani, Ankit Agrahari, Prachi Poddar, Sudeshna Pal

Cross Site Scripting (XSS) are a type of injection attacks where malicious scripts are injected into benign websites. Similarly, **SQL injection** is a type of attack where SQL statements are entered into an entry field for execution. These attacks are well known attacks and signatures are a good way of guarding against such well known attacks. A signature record is stored in the WAF, which contains the rules for handling well known attacks in an encoded format. The WAF checks for the presence of these defined formats in the uri passed and filter accordingly.

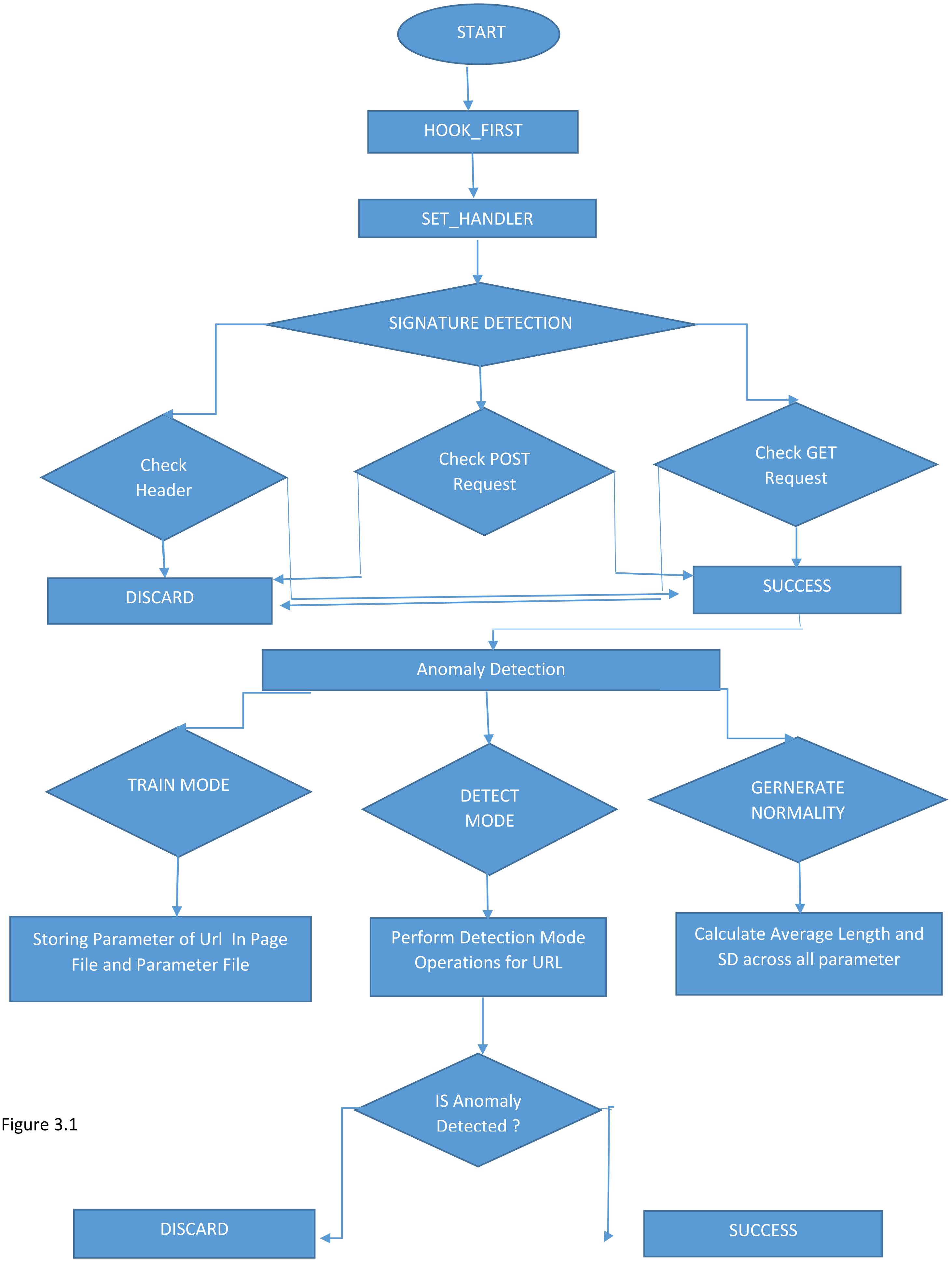

START
HOOK_FIRST
SET_HANDLER
SIGNATURE DETECTION
Check Header
Check POST Request
Check GET Request
DISCARD
SUCCESS
Anomaly Detection
TRAIN MODE
DETECT MODE
GERNERATE NORMALITY
Storing Parameter of Url In Page File and Parameter File
Perform Detection Mode Operations for URL
Calculate Average Length and SD across all parameter
IS Anomaly Detected ?
DISCARD
SUCCESS

Figure 3.1

In order to check for the right signatures, the module first reads the encoded rules from the signature file. The WAF then proceeds to check the header parameters and the method parameters (GET or POST), that is passed. If any parameters are found to contain malicious signatures, the request is dropped immediately without proceeding further. Any new signatures can be added to the signature file in the proper encoded format, which will be taken up by the WAF for subsequent requests.

3.2. Anomaly Detection (anomaly_detection_phase.c, waf_train_mode.c, waf_generate_profile.c, waf_detection_mode.c)

Checking for signatures can only protect against known attacks present in the signature records. For attacks with unknown signatures, we analyze legitimate traffic and then filter out those requests which deviate to a great extent from the legitimate ones. Anomaly detection is divided into three modes, namely, training mode, generate normality mode, detection mode.

3.2.1 Training mode (anomaly_detection_phase.c, waf_train_mode.c)

By Amogh Avadhani, Ankit Agrahari

In training mode, the WAF does not filter out anything. All the requests are allowed in this mode. Assumption is made that the requests in this mode are all legitimate ones. The goal here is to collect information about the requests. All the information for a particular page, i.e. url and maximum number of parameters, and also all the information for a particular parameter, i.e. url and parameter length are collected and stored in file storage. The data written to the file is in an encrypted format to avoid misuse of the data if it falls in the hands of adversaries.

3.2.2 Generate normality profile mode (anomaly_detection_phase.c, waf_generate_profile.c)

By Ankit Agrahari, Sudeshna Pal

The collected information is fetched from file storage to calculate the following parameters:

For pages

- Maximum number of parameters across all pages
- Maximum number of parameters for each page

For parameters

- Average length of parameter values
- Standard deviation of parameter values
- Allowed character set for the parameter

The parameters calculated are again put back into the file storage.

3.2.3 Detection mode (anomaly_detection_phase.c, waf_detection_mode.c)

By Amogh Avadhani, Ankit Agrahari

During detection mode, every request is checked thoroughly using the parameters generated in the previous mode. If for any request, the number of parameter exceeds the overall maximum parameter, the request is dropped. If for any request, the number of parameters are within the overall limit, but exceed the specific number of maximum parameters for that particular url, the request is dropped. For every parameter of every request, only the parameters having a value length of *average +- 3 * standard deviation,* are allowed to go through. The character sets of every parameter for every request are also checked, and the requests which have characters not present in its character sets are dropped. The following are the various flags returned based on anomaly detection filtering.

`EXCEED_ALL_MAX_PARAM:` Returned when number of parameters exceed the total number of parameters across all pages. The request is dropped.

`EXCEED_MAX_PARAM:` Returned when number of parameters exceed the total number of parameters for the particular page. The request is dropped.

`MAX_PARAM_PASS:` Returned when number of parameters is within the allowed number. The request is allowed to go through.

`PARAM_LEN_ILLEGAL:` Returned when the length of parameter value lies outside the bounds taken for the normal distribution. The request is dropped.

`UNKNOWN_PARAM:` Returned when the request contains a parameter not seen during training mode. The request is dropped.

`CONTAINS_NO_SEEN_CHAR:` Returned when any parameter value contains a character outside the scope of its character set. The request is dropped.

`PASS_DETECTION:` Returned when all the criteria is met. The request is allowed through.

Based on whether all the defined criteria are met or not, the request is either discarded or allowed to pass through and hit the web server.

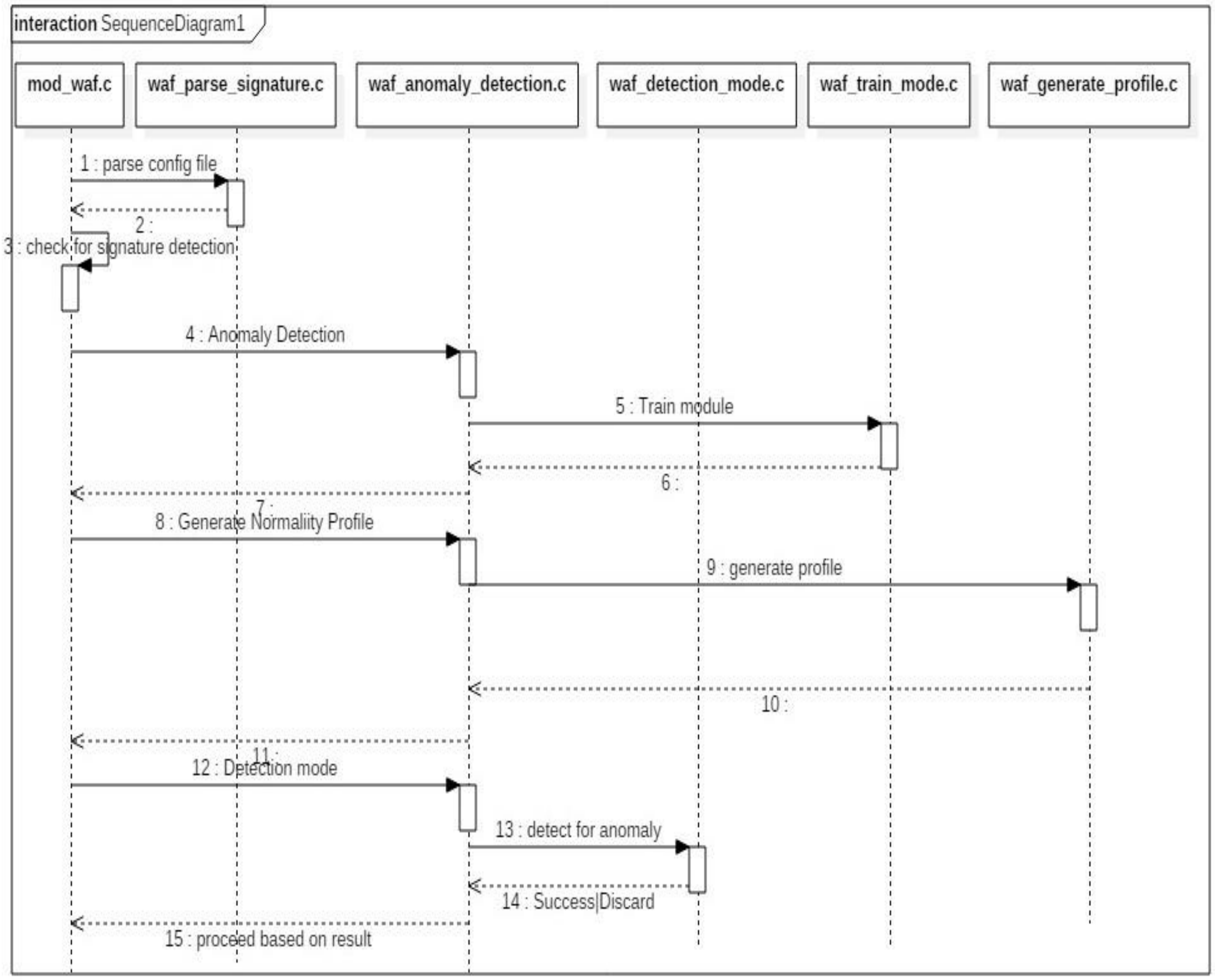


Figure 3.2

3.3 Testing Environment (Joomla)

By Amogh Avadhani, Prachi Poddar, Sudeshna Pal

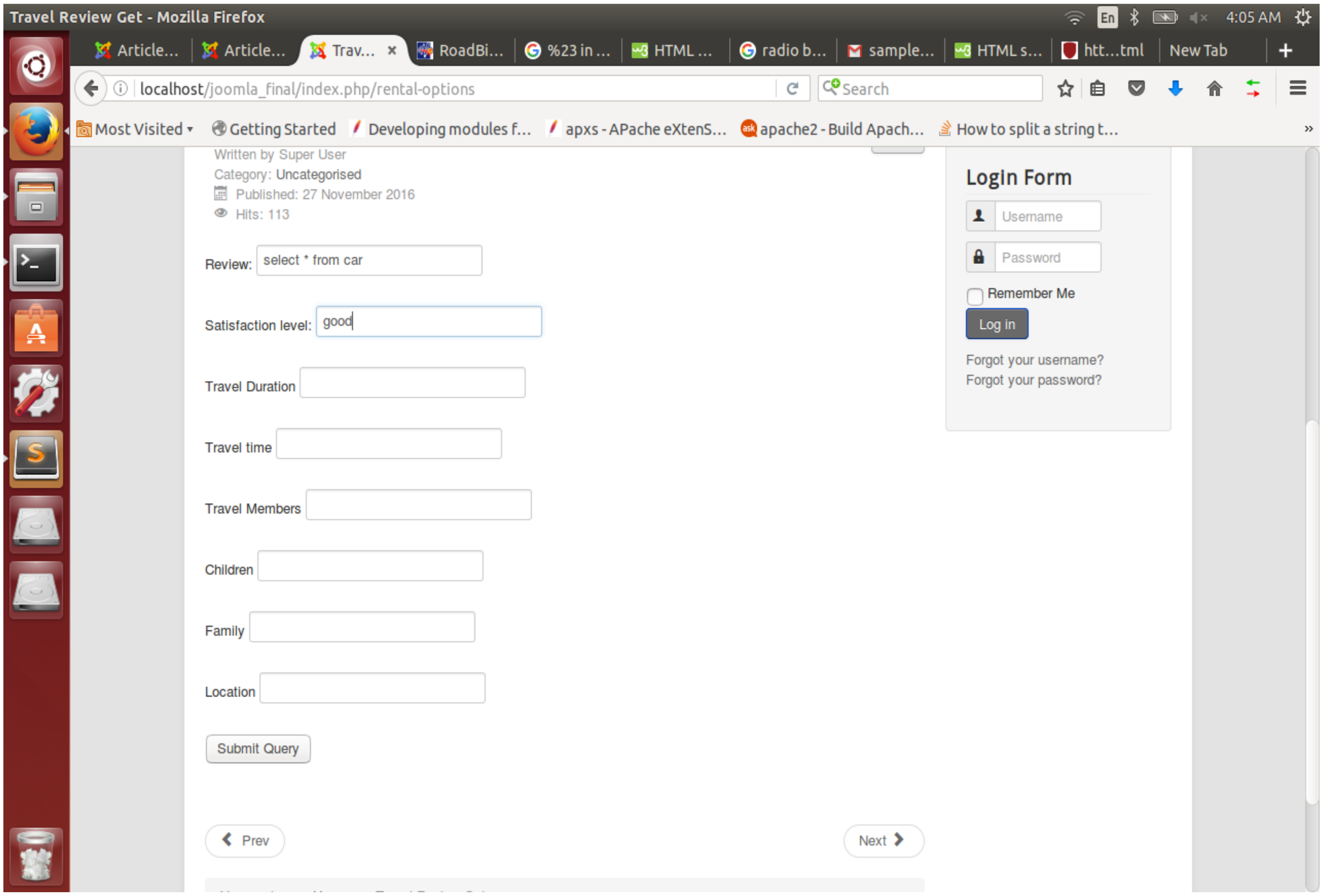
Joomla is an open source Content Management System (CMS) in which websites and web applications can be created. We have created a sample e-commerce website using Joomla CMS, which is hosted on the Apache server sitting on the local machine. The Joomla websites contain different pages which can be navigated back and forth. Provision is made in the website to send

GET and POST requests to other pages. All the requests are first forwarded to the WAF before hitting the server to retrieve the actual content present in those pages.

4. Use Cases

4.1. Signature verification

Detecting SQL injection signature from the request. “Select” query is tried to run from the text field.



Written by Super User
Category: Uncategorised
Published: 27 November 2016
Hits: 113

Review: select * from car

Satisfaction level: good

Travel Duration

Travel time

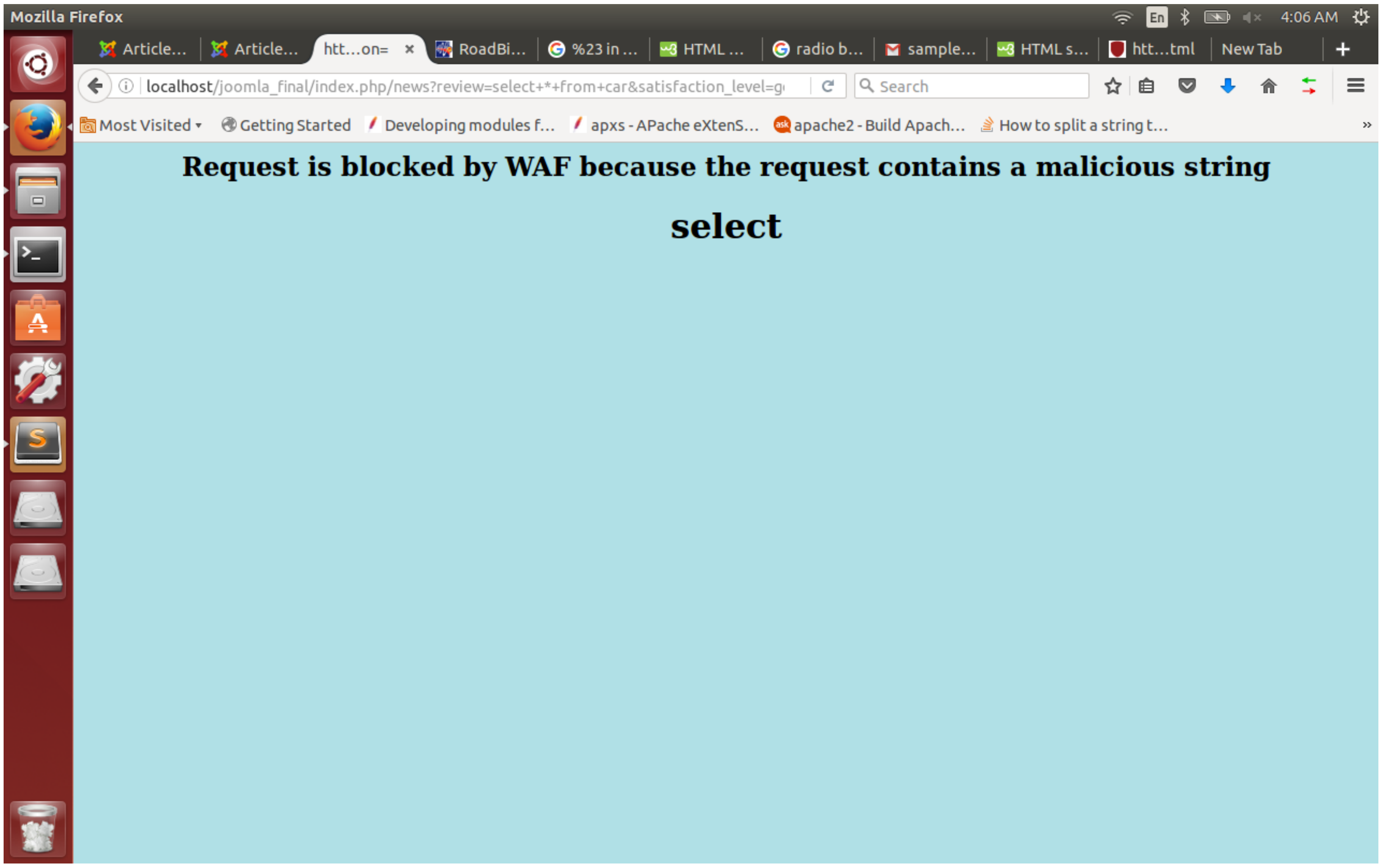
Travel Members

Children

Family

Location

The SQL select query injection attack is dropped.



localhost/joomla_final/index.php/news?review=select+*+from+car&satisfaction_level=g

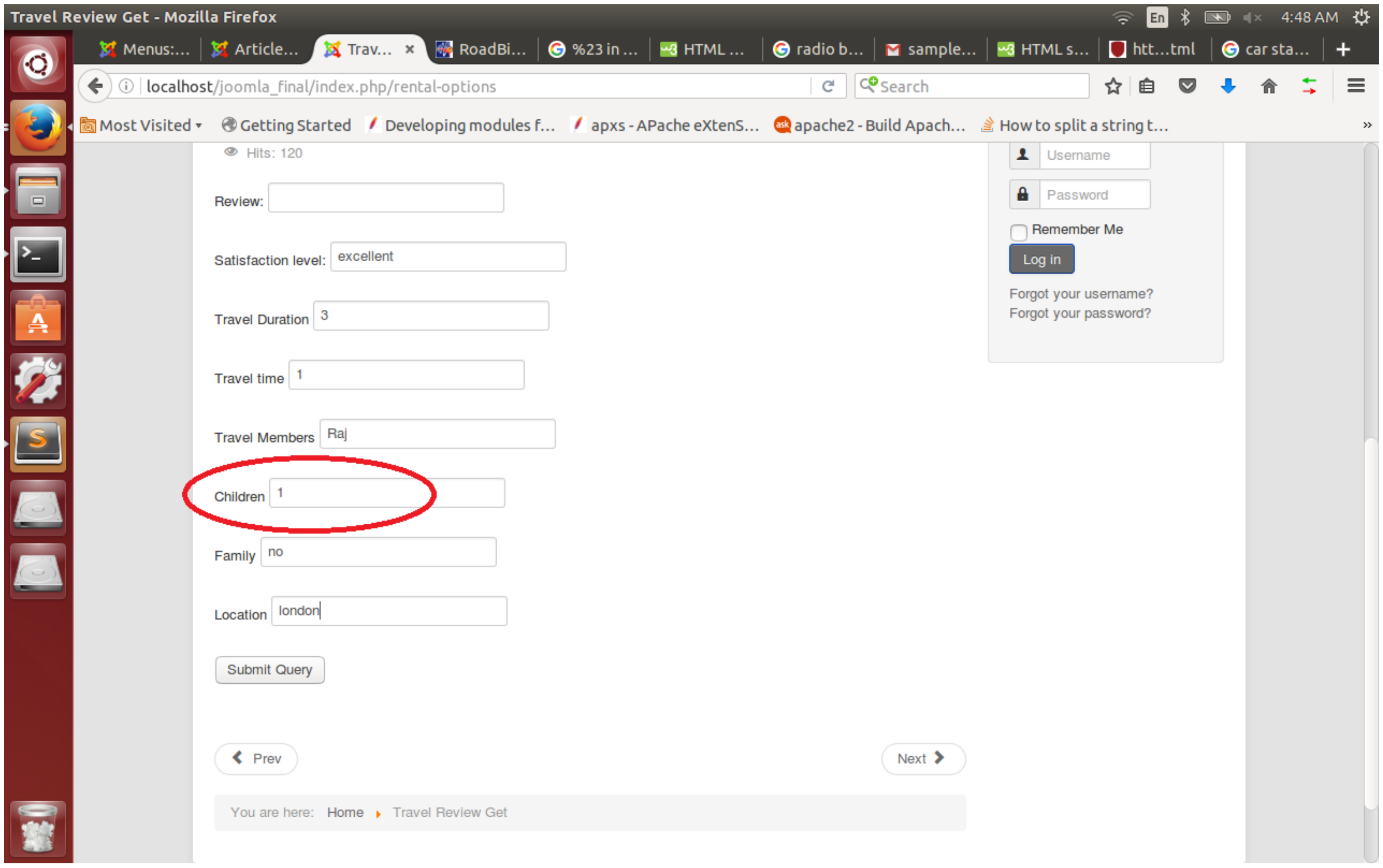
Request is blocked by WAF because the request contains a malicious string

select

4.2. Anomaly Detection

Training:

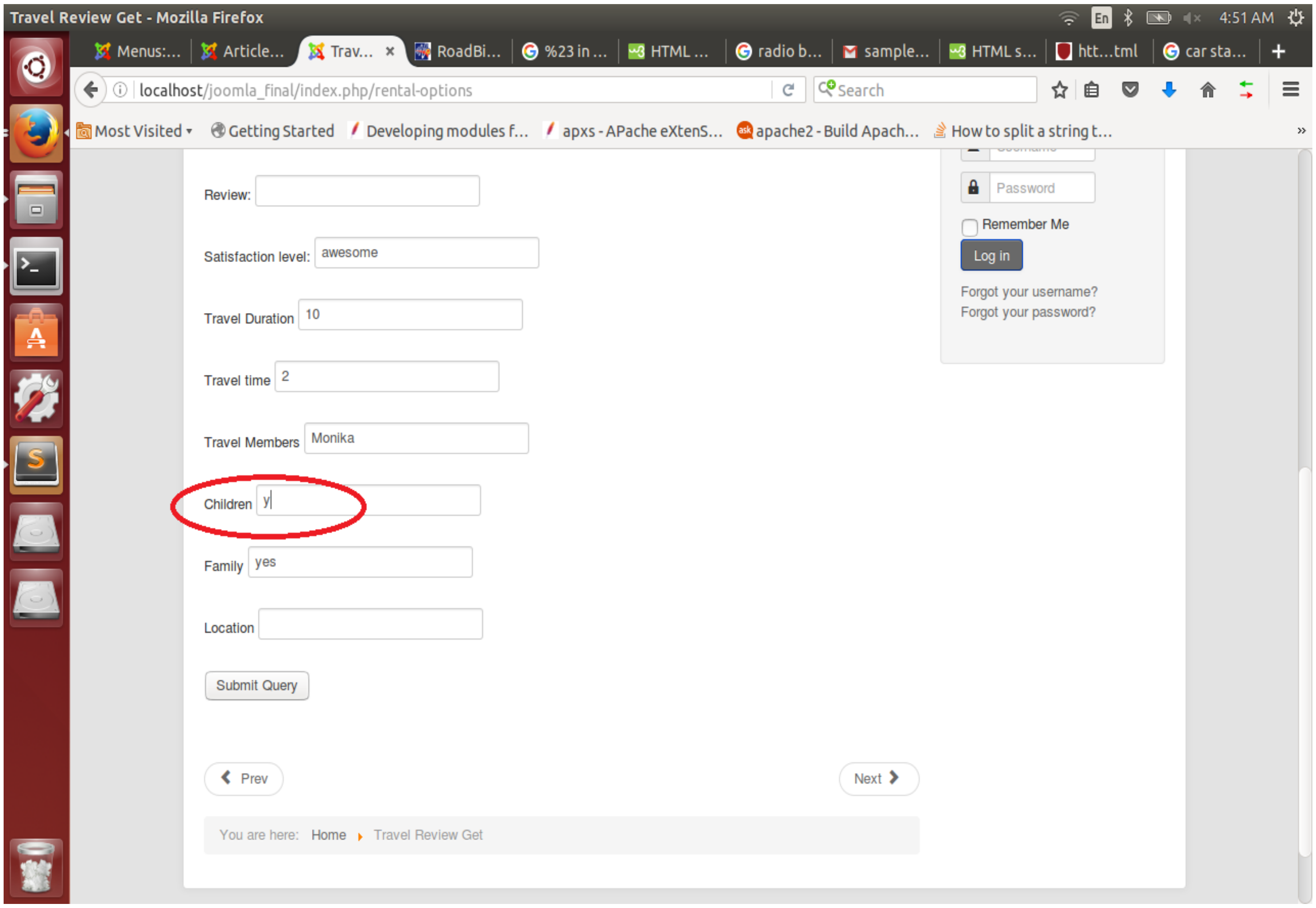
Character set of the parameter “Children” is numeric during training mode.



Review:	
Satisfaction level:	excellent
Travel Duration	3
Travel time	1
Travel Members	Raj
Children	1
Family	no
Location	london

Detection:

Letters are given in the "Children" field during testing.


Travel Review Get - Mozilla Firefox
localhost/joomla_final/index.php/rental-options
Password
Remember Me
Log in
Forgot your username?
Forgot your password?
Submit Query
Prev
Next
You are here: Home Travel Review Get

Review:

Satisfaction level: awesome

Travel Duration 10

Travel time 2

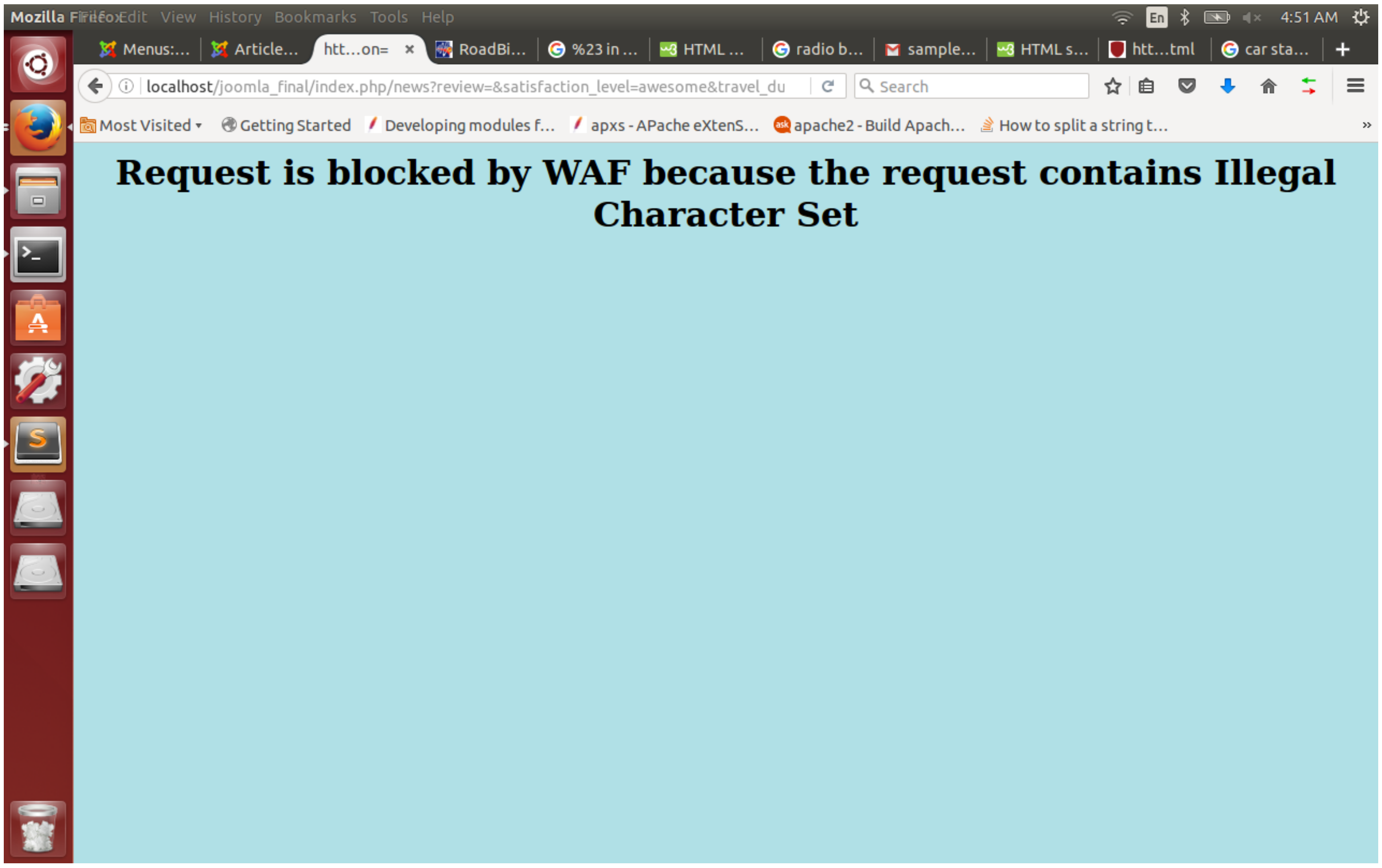
Travel Members Monika

Children y

Family yes

Location

The request is dropped by the WAF because of illegal character set entry.



Request is blocked by WAF because the request contains Illegal Character Set

5. Conclusion

A Web Application Firewall was implemented as an Apache module. The module sits in front of the web server filtering all the requests, stopping the malicious ones and allows only the non-malicious ones to go through to the server.

6. References

https://httpd.apache.org/docs/2.4/developer/modguide.html
https://httpd.apache.org/docs/current/programs/apxs.html
https://en.wikipedia.org/wiki/Web_application_firewall
https://github.com/SpiderLabs/ModSecurity
https://www.joomla.org/
http://www.w3schools.com/