Magdeburger Journal zur Sicherheitsforschung

Gegründet 2011 | ISSN: 2192-4260
Herausgegeben von Stefan Schumacher
Erschienen im Magdeburger Institut für Sicherheitsforschung
http://www.sicherheitsforschung-magdeburg.de/publikationen/journal.html

This article appears in the special edition »In Depth Security – Proceedings of the DeepSec Conferences«.
Edited by Stefan Schumacher and René Pfeiffer

# Building Your Own Web Application Firewall as a Service

## And Forgetting about False Positives

**Juan Berner**

---

When a Web Application Firewall (WAF) is presented as a defensive solution to web application attacks, there is usually a decision to be made: Will the solution be placed inline (and risk affecting users due to outages or latency) or will it be placed out of band (not affecting users but not protecting them either). This paper will cover a different approach you can take when deciding how to use any WAF at your disposal, which is to try and get the best of both worlds, making the WAF work in passive mode out of band detecting attacks and in active mode by selectively routing traffic through your WAF to decide if it should block the request or allow it.

To achieve this the paper will show how to abstract the WAF around a web service, something that developers are commonly used to working with, which can result in delivering security in a targeted and scalable manner. In this network agnostic setup, a WAF web service functionality can grow horizontally, allowing you to enhance the WAF decisions with your own business knowledge. This will mean that the decision to block or to route traffic through the WAF will not only depend on the WAF's decision but also on data about your application and its context, which can significantly reduce the false positive rate up to the point of practically not existing.

This paper will go through how such a service can be built with open source examples, what alternatives are there, depending on the flexibility of the WAF used, and how this approach can be used to manually decide on the false positive rate wanted and the desired business risk depending on the attack type and it's possible impact.

**Keywords:** WAF, Web Application Firewall, Security Architecture, Web Application

---

Citation: Berner, J. (2020). Building your own web application firewall as a service: And forgetting about false positives. *Magdeburger Journal zur Sicherheitsforschung*, *19*, 987–994. Retrieved June 25, 2020, from http://www.sicherheitsforschung-magdeburg.de/uploads/journal/MJS_069_Berner_WAF.pdf

# 1 Introduction

Current modern websites allow the capture, processing, storage and transmission of sensitive customer data (e.g., personal details, credit card numbers, social security information, etc.) for immediate and recurrent use. To ensure that this can happen safely, organizations need to employ different types of controls and tools that allow them to increase their capacity to detect and respond to threats to their network.

One of the main challenges in implementing these tools surfaces due to the requirement that for them to analyse and decide whether the traffic must be blocked they need to be placed in the middle of the traffic, adding latency to each request which can become prohibitive for applications that depend on a low latency response. Another challenging obstacle when deploying them is caused due to the false positive rate, or how common these tools decide a normal user is malicious and might block their activity, which can make the adoption of these tools much harder than they would expect.

## 1.1 Web Application Firewall

One of the tools which are used to protect websites from application attacks is called a Web Application Firewall (WAF). This is an application firewall for HTTP applications which applies a set of rules to an HTTP conversation. Generally, these rules cover common attacks such as Cross-site Scripting (XSS) and SQL Injection.

These are usually deployed in one of the following architectures:

### 1.1.1 Inline

When on inline mode, a WAF appliance is placed in the middle of the traffic between a user and a web application, allowing it to inspect and block attacks in a transparent manner to web servers, as shown in Fig. 5.

### 1.1.2 Out of band

In this mode the WAF would have the ability to inspect the traffic sent to the web server but unable to react to it since it would only see a copy of the traffic, as shown in Fig. 2.

### 1.1.3 Agent

When using an Agent mode for a WAF, software is placed in the web server imitating an inline mode with a hardware setup. While this allows an easier network placement it can become more invasive on the deployment environment and lead to less efficient resource allocation for web servers, as shown in Fig. 3.

### 1.1.4 Cloud

When using a cloud provider as a WAF solution, web servers can benefit from a simple setup and what would seem like unlimited scaling capacity. This can be achieved by allowing a third party to be placed in the middle of all traffic between web servers and their customers. The drawback of such a setup would be an increased latency incurred due to the traffic going through the cloud provider (which can be reduced if it's the same provider used for the web application) and the risk of having the data go through a third party, as shown in Fig. 4.

## 1.2 Typical problems with WAF setups

### 1.2.1 Network placement

As mentioned with the out of band or inline architectures, depending on the scale of the organization what network placement strategy to use can become a challenging part of placing a installing a WAF solution. In environments where there might exist multiple datacenters hosting the applications needed to protect, ensuring that there is a complete coverage of a WAF (either placed in-line or always capturing copies of the traffic) can become a daunting task. This can lead to inefficient use of resources or a heavy investment on network changes to accommodate to the WAF solution.

### 1.2.2 False positive rate

One of the biggest problems with WAF solutions acting in blocking mode is the false positive rate they can generate. The false positive rate can indicate how many customers are being blocked while not actually performing an attack, which can translate into financial loss and reputational damage. One of the major reasons why WAF solutions are not placed in blocking mode is due to the false positive rate affecting enough customers that security teams are forced to stop blocking attacks, turning the solution into simply a visibility tool without the ability to stop attacks.

### 1.2.3 Latency added

Given that the WAF needs to inspect the traffic and decide if it should be blocked or not, latency is added by WAF applications. Depending on the network placement, and how costly the analysis operations they perform might be, this can become prohibitive to applications that depend on low latency responses to function or engage their users.

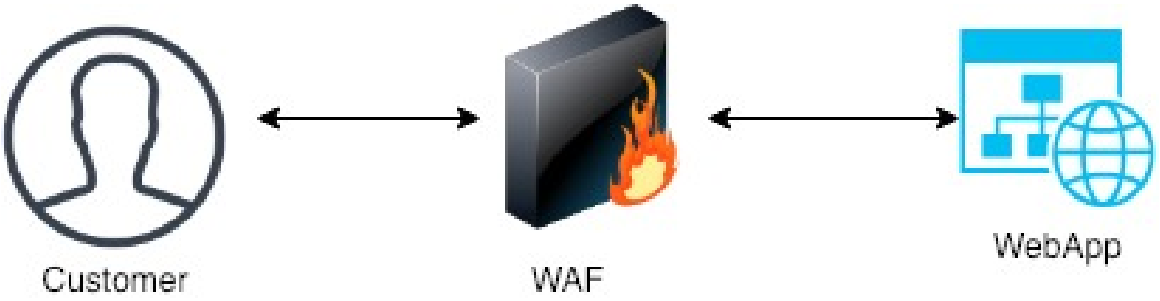



Figure 1: Inline Mode

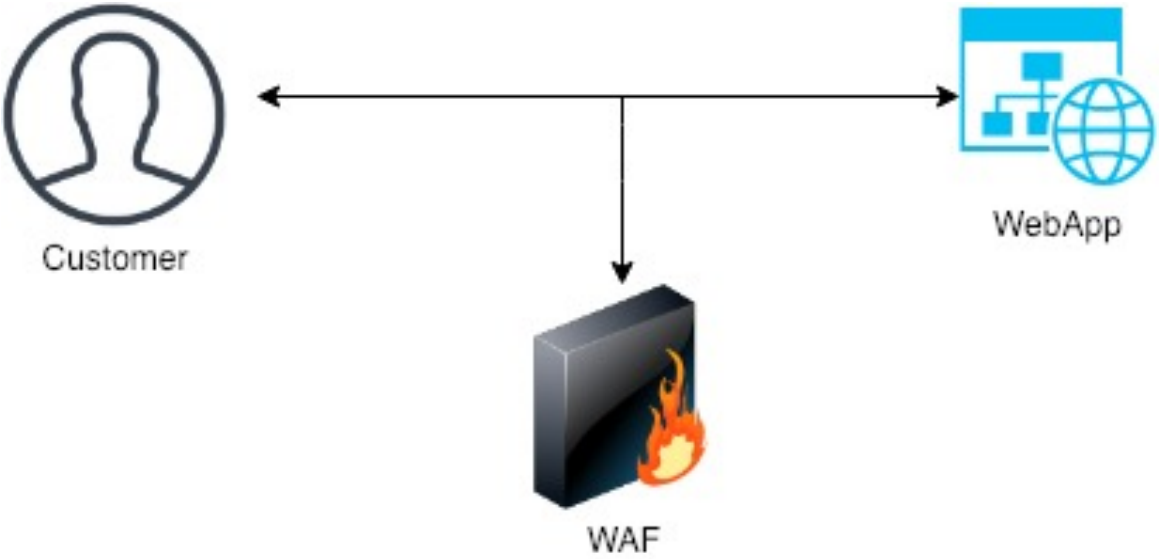



Figure 2: Out of Band Mode

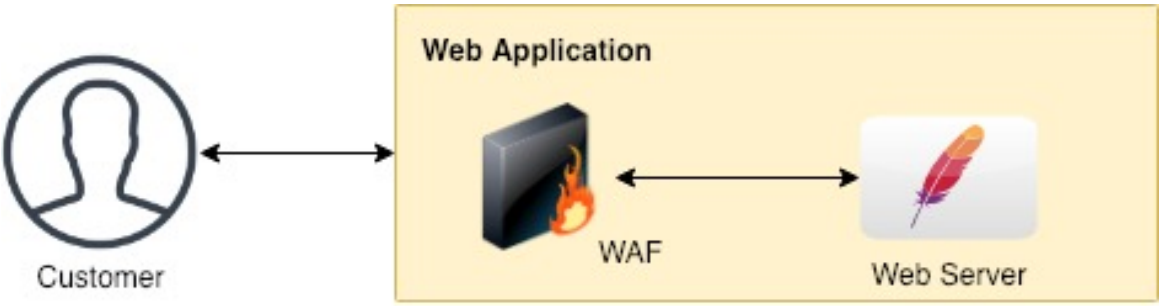



Figure 3: Agent Mode

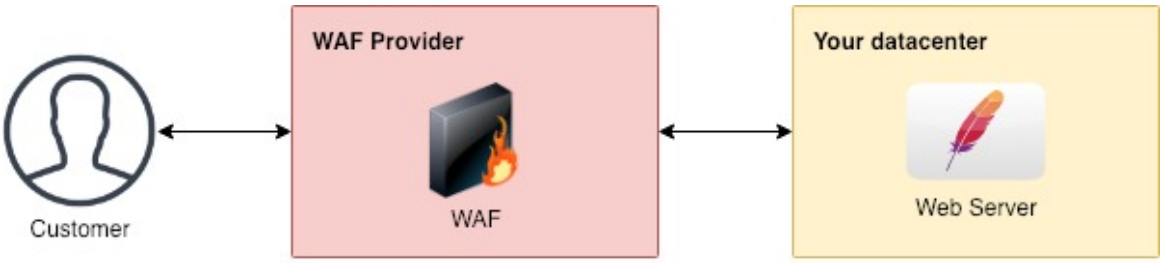



Figure 4: Cloud Mode

# 2 WAF as a Service

## 2.1 Problems it should solve

### 2.1.1 Simple network placement

By using an Agent, the network placement is simple to implement (since the Agent would live in the same environment as the web server) and by only using it as a relay to the WAF service it's resource requirements are small enough to avoid affecting the web server or it's resource allocation.

### 2.1.2 No false positives should be generated

Using context, historical information and business logic of the application, the service would allow the teams to manage the false positive rate making it virtually non existent if necessary.

### 2.1.3 Avoid adding latency for regular users

A focus on this solution would be to remove the latency impact on regular users by performing out of band analysis on the traffic. This is achieved by only turning it into an inline WAF for threats performing web application attacks against the website.

## 2.2 The solution

To solve these problems I developed a system that was able to work in both inline and out of band mode. By taking advantage of a small agent that would only interact with a WAF service, it's possible to take advantage of the Agent's architecture benefits without its drawbacks, and allow the application developers to interact with it as with any other service. This agent would not only generate logs of requests -that the web servers produced- but also decide in what situations further analysis by the WAF service was required (which would impact the latency of the request).

The decision on whether or not the WAF service should be involved in a particular request would not be decided by the agent itself, but by an out of band process. This process would be:

1. Inspecting all the requests
2. Replaying them against the WAF service (acting in an out of band mode)
3. Updating a state store so only certain segments of the traffic which were seen as riskier (due to being involved in web application attacks or other suspicious activity) to be routed through the WAF service (acting in an in-line mode).

This architecture is able to provide the best of the out of band mode (which is a lack of latency added for regular users) and of an in-line mode (the ability to stop web application attacks before they execute on a web server). During the next section I will cover how such a system can be built and what components are needed.

# 3 Components

## 3.1 Web Application

The web application will be the one to decide how they want to interact with the WAF service. The developers can choose to be fully in-line mode (which means it would be always adding latency to its users), exist in an out of band mode (without having the ability to block) or work in a hybrid mode. In this mode they can also decide the false positive rate they would want to accept to block attacks (if any).

The web application will also be responsible for sending logs of each request -with business information about it- to allow the out of band evaluation to detect possible attacks.

## 3.2 Agent

The agent will act as a proxy for the web application with minimal footprint. It will behave differently depending on what application is using it, allowing the application to decide how to respond to attacks. The Agent will be the component to communicate in behalf of the web application to the WAF service.

An alternative to an agent is using a library that can be embedded in the web application itself and performing checks against the WAF service before and after a request is processed. This can be simpler to implement, although there is a drawback with this approach: if an attack is able to compromise the web application (for example its request handling code), all visibility and defenses against web application attacks on the web server would also be compromised.

## 3.3 Historical database

A historical database is needed to provide the ability to understand, not just the current context, but the historical context of a request too. This can involve activity on an endpoint through time, business value of particular users or patterns of behaviour that can allow to avoid false positives due to changes on the application or the user base. An example of this would be Google's BigQuery, which allows to perform interactive analysis of massively large datasets.

## 3.4 State store

The state store will allow us to store configuration that the Agent can consume, which should be fast enough to avoid latency impacting users. An example of this would be the redis in-memory key-value database.

### 3.5 Real time messaging service

To allow web applications to send encapsulated web request in the form of logs and have those processed, we will be using a real-time messaging service. An example of this would be Google's PubSub.

### 3.6 Log processing

The log processing component will consume from the real time messaging service and from the encapsulated web requests in the form of logs recreate them and replay them to the WAF service. This will also be able to calculate risk scores through windows of time to benefit of real time context, helping deciding what traffic needs to be routed through the WAF service in an inline mode. An example of this would be Google's Dataflow.

### 3.7 WAF service

The WAF service is the core component of this architecture. It receives web requests and is able to respond with risk scores based on its plugins. Due to its pluggable architecture, different types of checks can happen in parallel to evaluate a request. Based on its output, an application can decide how it wants to react to the request.

The WAF service can perform several checks in parallel with:

- Open source components: Examples of them could be Modsecurity or Naxsi which are open source WAF solutions.
- Custom modules: They could be built to apply business logic checks or evaluate machine learning models against the requests.
- Proprietary software or appliances: Adding them would allows to reduce the complexity of their installation and also add a path for a simple evaluation procedure.

### 3.8 Detection component

The detection component will be the one to perform analysis on the result of the log processing component, replaying the requests against the WAF service out of band. It will use the information from the historical database to update the state store, which the Agent relies upon to decide which traffic would be in-line or out of band.

### 3.9 Visualization

To be able to understand the activity currently impact the web servers, and have visibility on attacks as well as performance metrics, a visualization solution should be placed. An example of this would be using an ELK stack (Elasticsearch, Logstash and Kibana) to store all processed requests for visualization of the historical and real time activity of the WAF service.

## 4 Architectural diagram

## 5 Blocking web application attacks

To take advantage of this model, we need to be able to efficiently decide what traffic to block. This will minimize the latency incurred on regular users and remove the possibility of false positives affecting them. To achieve that we will need to decide what traffic should be placed in an 'inline' mode in the following ways:

### 5.1 Traffic routing

Traffic routing is the way by which we can decide that portions of the traffic should work in an in-line mode, having every single request analysed by the WAF and blocked if it is considered a web application attack. This allows applications that have a low tolerance for latency to be able to have their traffic inspected and only add latency when a threat is detected and malicious requests need to be blocked. This can be achieved in the following ways, either by human or an automatic decision making process:

#### 5.1.1 Fingerprint based routing

By analysing the traffic automatically in the log processing component, we can extract fingerprints that are performing web application attacks and only have those go through the WAF service (adding latency to them). These fingerprints would be extracted by combination of parts of the request (IP address, client ID, User Agent or combinations) or by particular fingerprints that might be automatically or manually added (as for 0day fingerprints or known attack patterns).

While the Log processing component would be automatically creating these fingerprints and adding them to the State store component, so that the Agent is aware that the traffic must be router to the WAF service, this can also be triggered by an analyst which decides that a particular fingerprint needs to be routed through the WAF.

#### 5.1.2 Net block based routing

Another option for routing traffic is based on a network block. This means that particular ISPs, hosting providers or other known services that have a higher risk of attacks coming from them (such as open proxies or anonymity networks like TOR) can be routed by default to the WAF service. This would happen by updating the state store with the IP address net blocks for these providers or members of the networks so that the Agent is aware that it must route such traffic to the WAF.

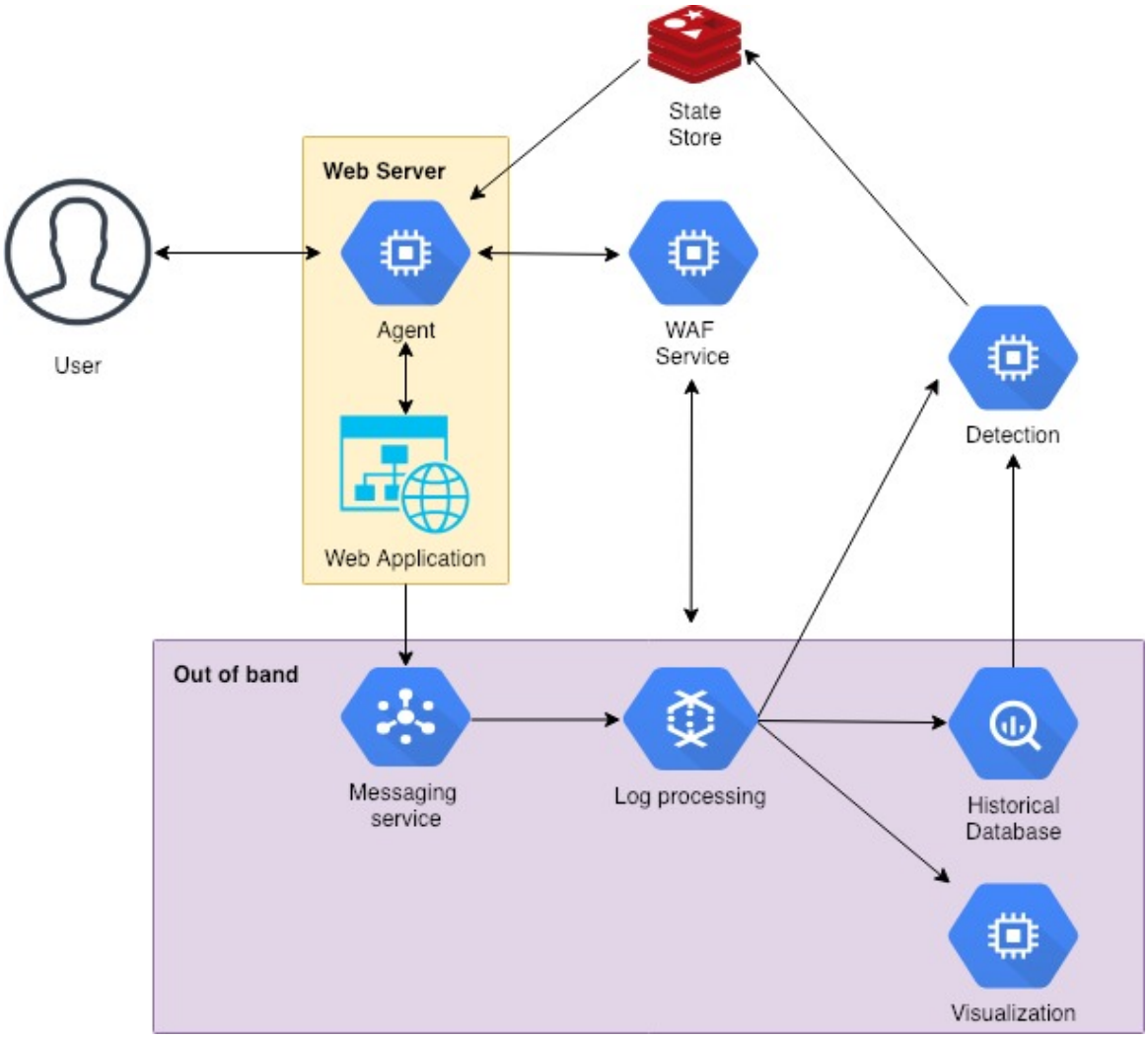



Figure 5: Architectural diagram

#### 5.1.3 Virtual patching

For situations where a vulnerability is known to exist on particular endpoints, or where these endpoints have a higher degree of risk and need to have the WAF service inspecting every single call (not only when a threat is detected), we can enable virtual patching. This means that particular endpoints are always routed to the WAF service for analysis of requests coming to them, either for the full endpoint or a combination of parameters that might be vulnerable to attacks.

### 5.2 False positive rate management

To find a way to avoid blocking users which might not be performing web application attacks, we need to first differentiate between a false positive on the detection process (DFP) and a false positive on the blocking process (BFP). While we will accept having false positives on the detection process (which means thinking a particular request might be a web application attack when it's not) we will focus on not having false positives in the blocking process (affecting those requests and blocking a normal user from accessing resources on the web server).

An important point to take into account is that while it's possible to remove the probability of false positives at all, different applications might want to balance what false positive rate they are willing to accept against the risks of increasing the likelihood of attacks from succeeding. Given the fact that the false positive rate can be decreased by getting more context around a request, the lower false positive rate we want the more context we might require, which also means more time is given to an attack to attempt to exploit the web application. Any application will need to balance its threat model against the impact of affecting users to find the right false positive rate to accept.

This section will focus on how to avoid blocking user's requests taking advantage of the information we have available, before deciding which traffic needs to be routed to the web server.

#### 5.2.1 Business Logic Analysis

By looking at the business logic of the application we can get information relevant to change the probability of the request being part of an attack or not. This can involve the trust we can give to some request attributes, such as the IP address or the client, what business activity has happened for the user in a period of time and what impact we would have by blocking them. Leveraging these data points can allow us to reduce the impact of incorrectly blocking accounts and routing traffic through the WAF service.

#### 5.2.2 Historical Analysis

Looking at the history of the requests and its fingerprints can help understand the risk of the request and its possible impact. This allows to compare the request against other requests which were similar or what is the rate of DFP's (detection false positive) for that particular endpoint or type of request.

#### 5.2.3 Context Analysis

The context analysis of a request will be a key part of avoiding BFP (blocking false positive) even if we have a high amount of DFP. This would happen given the fact that most web application attacks require many requests to be able to find an actual vulnerability on the web application. By looking at the context of a request in terms of how many requests have been marked as a possible attack, we can specify a particular false positive rate we are willing to accept and only place the suspicious traffic on the in-line mode for the WAF service if the context matches our desired BFP.

#### 5.2.4 An example of setting the desired BFP

In the situation where we have an application that wants a BFP of 0,00001% (one request would be incorrectly blocked for every 10 million) but we have a DFP of 0.1% (one request is incorrectly considered an attack every 1 thousand). Given that the probability of a DFP is independent of other requests, by only placing traffic in an in-line mode once a score of 5 requests marked as DFP is reached we can guarantee the BFP of 0.00001%. Depending on the type of attack and threat model of the application, the BFP and its related score can be modified to get the best possible balance in terms of security and impact to users.

## 6 Conclusion

To find a way to solve the problem of latency and BFP that normal WAF setups introduce, this paper describes a strategy that mixes both the in-line and out of band modes, in a hybrid architecture, that can dynamically choose what parts of the traffic should be placed in in-line mode and which ones should continue in an out of band analysis mode. By leveraging this architecture, this does not only avoid affecting users with latency and incorrect blocking, but also improve the response capabilities by allowing multiple components to be placed as plugins of the WAF service working in parallel to perform decisions and analyse the traffic. Leveraging open source components, any organization can implement such an architecture to improve their ability to protect its users and improve their experience at their platform.

## 7 About the author

Juan Berner is a security researcher with over 9 years of experience in the field, currently working as Security Lead Developer, SME for Application Security and Architect for security solutions at Booking.com. He has given talks in the past on how to build an open source SIEM (https://www.ekoparty.org/security-monitoring-like-the-nsa.php) and on exploiting A/B Testing frameworks (Exploiting A/B Testing for Fun and Profit).

## 8 References


- https://www.acunetix.com/websitesecurity/web-application-attack/
- https://www.owasp.org/index.php/Web_Application_Firewall
- https://www.elastic.co/elk-stack
- https://redis.io/
- https://modsecurity.org/
- https://github.com/nbs-system/naxsi
- https://cloud.google.com/bigquery/
- https://cloud.google.com/pubsub/
- https://cloud.google.com/dataflow/