coraza

WEB APPLICATION FIREWALL

The way to WAF in 2023

About the speakers

Juan Pablo Tosso

- Author and co-leader of OWASP Coraza Web Application Firewall
- 10 years industry experience
- Father of two
- API security nerd
- Research Engineer at Traceable AI
- Golang developer

About the speakers

Felipe Zipitría

- CRS co-leader
- Coraza co-leader
- Long time supporter of OWASP projects

1990s World Wide Web

- We used marquees, background music, and fancy pointers
- XSS was crazy dangerous
- It was all server rendering or static HTML
- We moved from a one-line telnet-friendly protocol (HTTP/0.9) to complex headers and body
- Netscape was a thing

WAF in the early 2000s

AppShield

AppShield was the first WAF, it was capable of applying policies to HTTP requests and HTML content.

OWASP TOP 10

First draft of the document that would shape the structure of web vulnerabilities until today.

Public CRS

CRS was released on Github for public consumption. It used to be a commercial ruleset.

ModSecurity Apache Module

ModSecurity provided WAF capabilities for Apache by creating SecRules.

WAFEC

Standard created to define what is a WAF and how it should behave.

ModSecurity

- First Open-Source WAF, mod_security
- Designed as an Apache module
  - Heavily relied on Apache httpd capabilities
- A language for rules definition and engine configuration was created, know as *SecLang*
- Reaching EOL in 2024

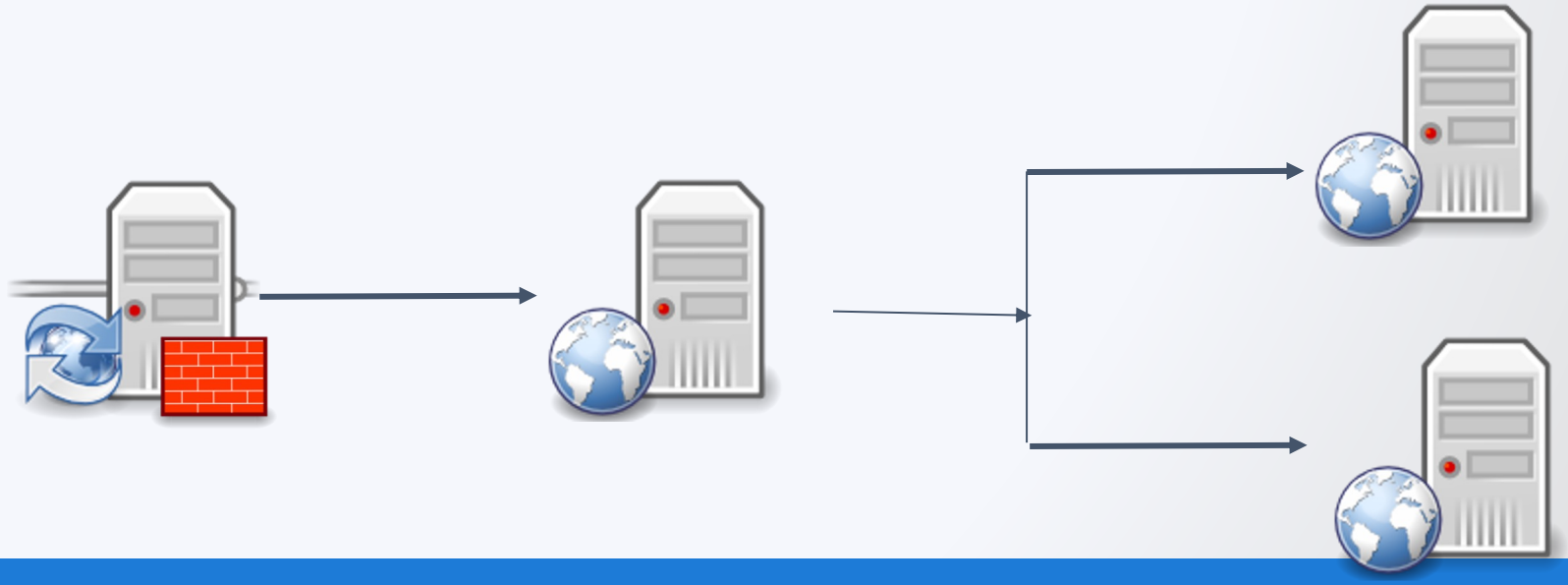
Implementation techniques

Inline:

Reverse/Transparent Proxy
Part of the webserver

Out of band:

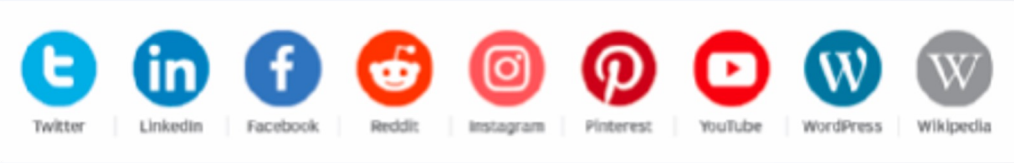
Traffic mirroring



Core Rule Set

- With the creation of the WAF engine, there was the idea of having a set of rules for everyone
- The *Core Rule Set* was born!
- Initially, inside Trustwave
- Wisely, handed over to the community and now we have the OWASP ModSecurity Core Rule Set

Web 2.0


Twitter
LinkedIn
Facebook
Reddit
Instagram
Pinterest
YouTube
WordPress
Wikipedia

Rich Content Applications (Ajax)

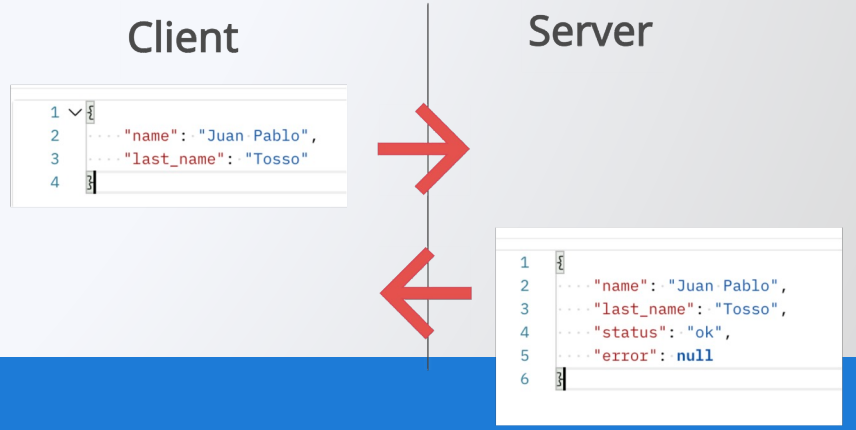
- Game changer in web development, it brought rich content applications to life
- JSON, RPC, among others
- More client side logic validation

```
function createXHR() {
    if (typeof XMLHttpRequest !== "undefined") {
        return new XMLHttpRequest();
    } else {
        var versions = ["MSXML2.XmlHttp.6.0",
            "MSXML2.XmlHttp.3.0"];

        for (var i = 0, length = versions.length; i < length; i++) {
            try {
                var xhr = new ActiveXObject(versions[i]);
                return xhr;
            } catch (error) {}
        }
    }

    alert("Your Browser Doesn't Support XmlHttp");

    return null;
}
```

APIs & JS frameworks

Rich content applications forced the developers to create APIs and share structured data between client and server, like JSON.

- New frontend frameworks like Angular, Vue and React
- Business logic became the fun attack vector
- Every company was speaking API



🆕 ways of integrating

eBPF

- eBPF attaches a code into the kernel to access information from processes. It can be used with httpd and openssl.
- We can read encrypted traffic
- We cannot terminate a session

Tracing

A distributed trace is a set of events, triggered as a result of a single logical operation, consolidated across various components of an application.

- Traces can be used to asynchronously analyze requests using a WAF.
- Doesn’t support blocking.

GRPC

GRPC provides fast and lightweight communication for protobuf structures.

HTTP requests and responses can be serialized into protobufs

The result can be exchanged with a grpc server for WAF

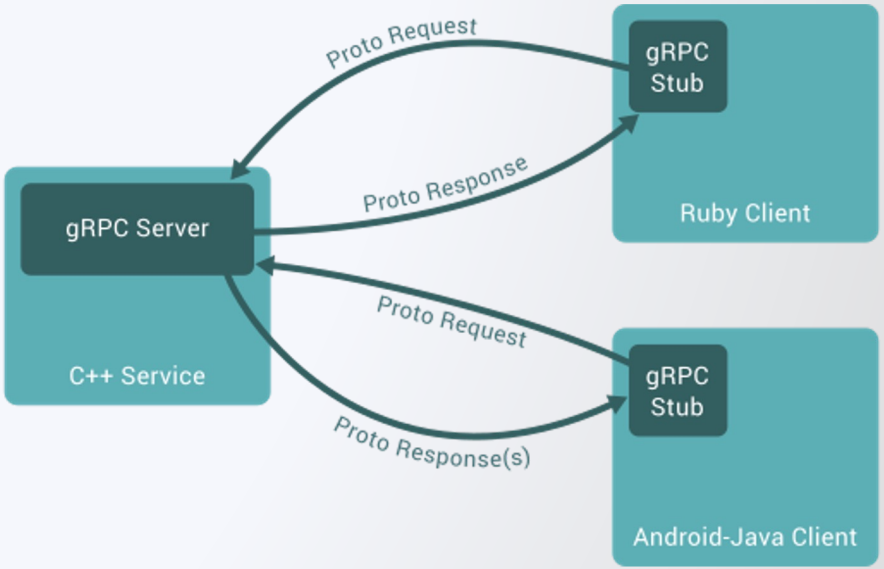
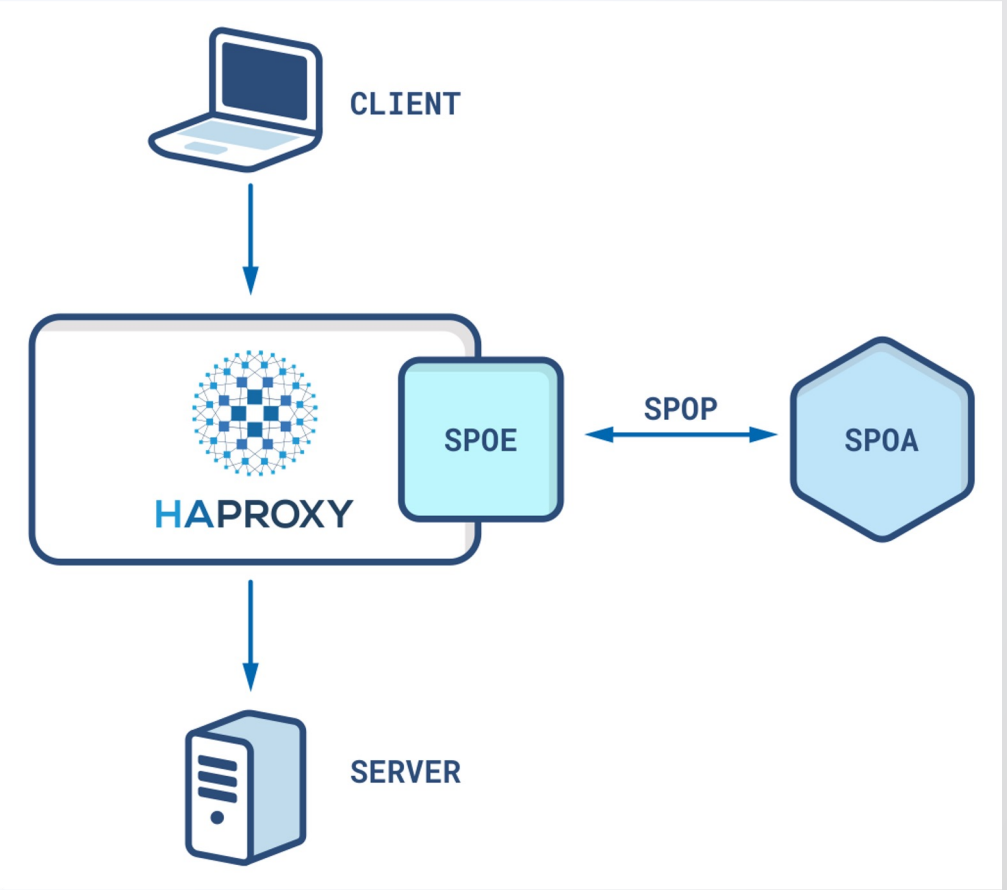


Image: GRPC.io website. CC 4.0

SPOA

- HAProxy specific
- Stream Processing Offload Engine
- Provides blocking and mutating capabilities



Web Assembly (WASM)

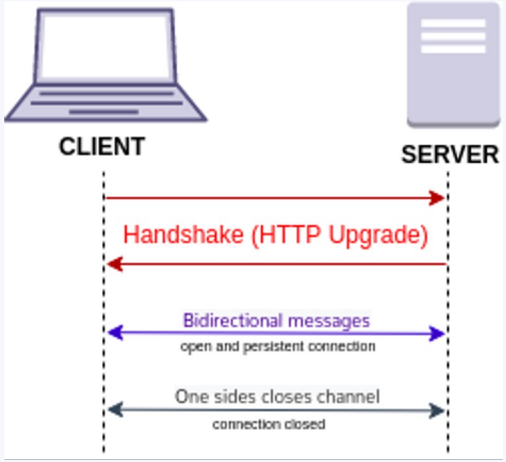
WASM provides a universal binary that can be consumed from multiple languages and frameworks.

It’s currently used by Envoy Proxy for plugins

2023 Challenges

Websockets

Designed to work over HTTP ports 443 and 80 as well as to support HTTP proxies and intermediaries



- Transmitted data is not standard HTTP
- How can we protect it?

GraphQL

GraphQL is an API standard that provides dynamic access to data using queries instead of URL based calls.

- There are multiple specific vulnerabilities related to GraphQL implementations
- The GraphQL query and result are treated as a string by WAFs

Edge Termination

- What happens if the WAF is outline?
- How can we stop the attack?
- How can we stop users?

	Application	Web Server	Load Balancer	Outline
Generates lag	Yes	Yes	Yes	No
Can stop real time	Yes	Yes	Yes	No
Can delete session	Yes	Yes	No*	No*
Can block IP Addr	Yes	Yes	Yes	No

XSS is not fun anymore

- XSS was part of the OWASP TOP 10 for more than ten years
- In some scenarios, it can still lead to dangerous attacks
- Web browsers are capable of stopping most of the attacks
- Frontend frameworks might also stop XSS

Horizontal Scaling

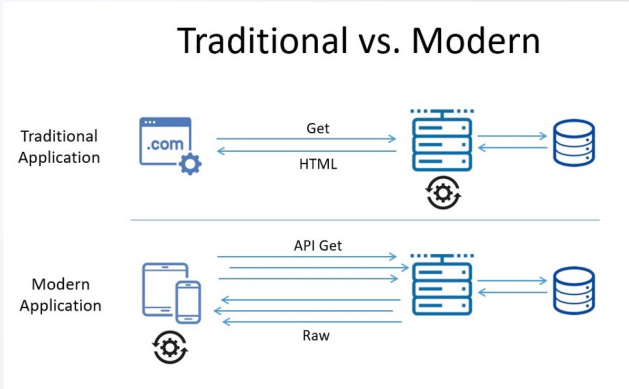
Web Application Firewalls should be able to scale horizontally, and not only vertically, so we can meet expectations for microservices and complex auto-scaling infrastructures.

0 False Positives

- If rules are too generic we get false positives, if rules are too specific we get false negatives
- We rely on signature based rules

API Security

- API security includes more body types and parameters in the path
- It's still vulnerable to OWASP top 10, but with a bigger focus on business logic



Compete against CDN WAF

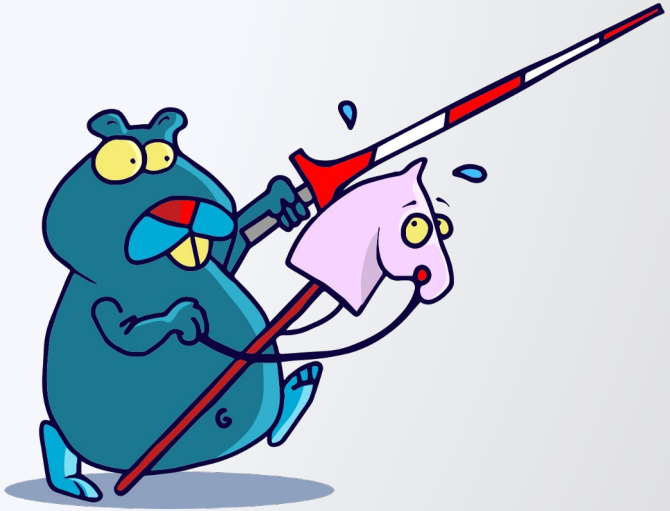
CDN vendors like Cloudflare and Akamai has great WAF features, but companies are realizing they are not enterprise-grade WAFs and they need more detailed control of the protection.

Block the user, not the IP

- A single session could share multiple IP addresses
- Multiple users could share the same IP address
- “IP + USER_AGENT” is not a good user identifier

OWASP Coraza

- Modern focus on rich applications
- High Performance
- Multiple connectors
- Tested in rough environments
- 100% OWASP Core Ruleset compatible
- Our mascot’s name is Sancho

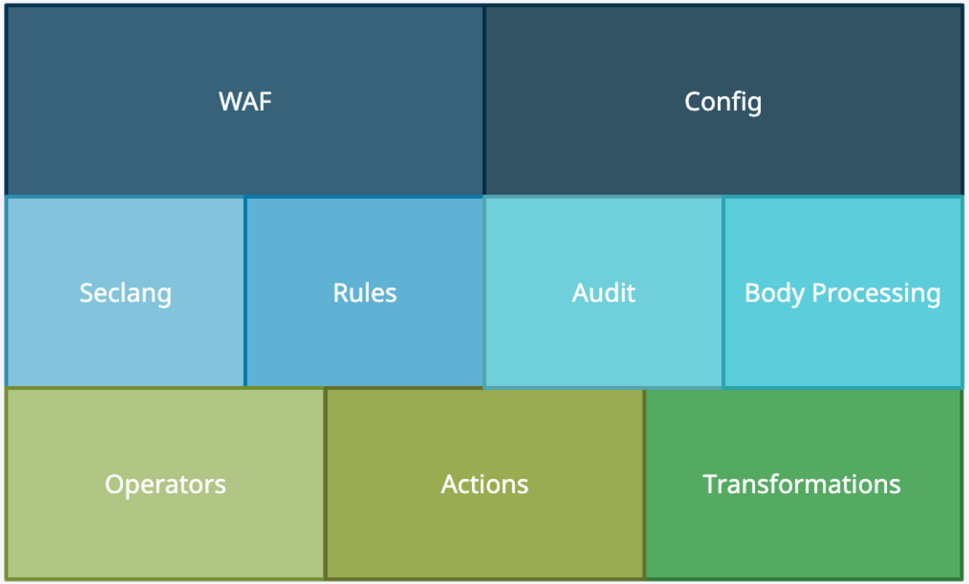


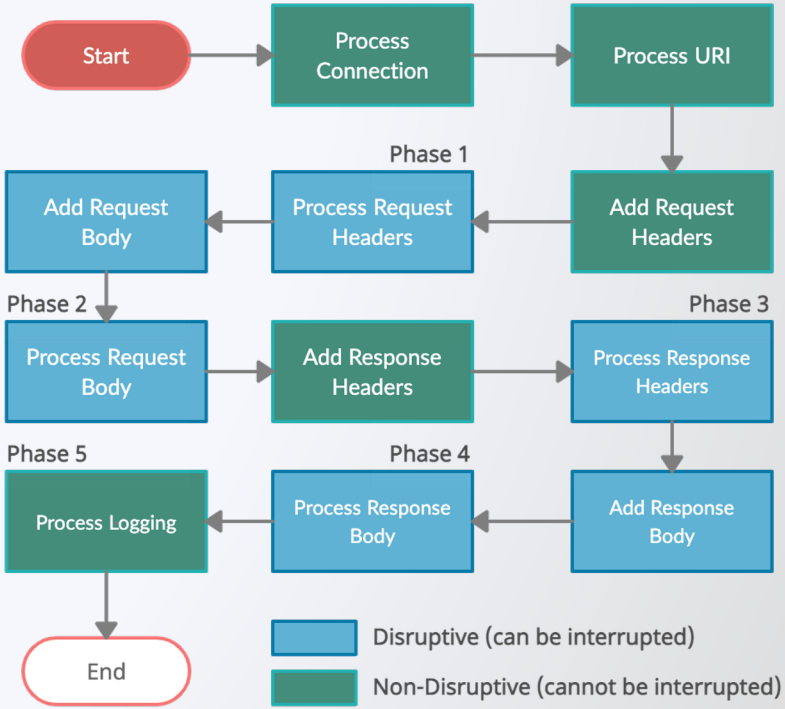
Community. Period.

Github stats

- **6** active developers
- **950+** stars
- **753** commits
- **119** forks
- **209** issues solved
- **600+** daily clones
- **~60** new visitors per day

High Level Design





Sample Usage

Coraza uses an immutable pattern to initialize WAF transactions.

CRS can be integrated with only a few lines

```go
package main

import (
	"fmt"
	"github.com/corazawaf/coraza/v3"
)

func main() {
	// First we initialize our waf and our seclang parser
	waf, err := coraza.NewWAF(coraza.NewWAFConfig().
		WithDirectives(`SecRule REMOTE_ADDR "@rx .*" "id:1,phase:1,deny,status:403"`))
	// Now we parse our rules
	if err != nil {
		fmt.Println(err)
	}

	// Then we create a transaction and assign some variables
    tx := waf.NewTransaction()
	defer func() {
		tx.ProcessLogging()
		tx.Close()
	}()
	tx.ProcessConnection("127.0.0.1", 8080, "127.0.0.1", 12345)

	// Finally we process the request headers phase, which may return an interruption
	if it := tx.ProcessRequestHeaders(); it != nil {
		fmt.Printf("Transaction was interrupted with status %d\n", it.Status)
	}
}
```

Benchmarks

	Coraza v2	Coraza v3	Libmodsecurity
Simple Post JSON	792	69458	1035
Giant JSON POST	786	70482	1009
Giant post multipart	763	69589	1026
Simple post multipart	753	70036	1056
Simple get	835	72157	1018
Simple post urlencoded	759	70650	1039
Giant post urlencoded	776	69444	1008

- 16GB memory
- Apple M1 processor
- Tested against libmodsecurity **using CGO**

https://github.com/jptosso/coraza-benchmark-2

Deployment Options

- **Web Assembly**
  - Envoy
  - Nginx
  - Kong
  - HTML5 applications

- **Caddy**: Caddy Module
- **HAProxy:** Through SPOA
- **In-App:** Standard HTTP middleware

Extensibility

- **Operators:** New content validators
- **Actions:** New disruptive actions
- **Transformations:** New encodings/decodings
- **Body Processors:** GraphQL, custom formats
- **Logging:** Log to Elastic
- **Audit Logging:** New formats and log to server
- **Directives:** New configuration options

Roadmap

...

Conclusions

- WAFs can live along “next generation WAFs”
- Existing Open-Source WAFs must evolve to embrace API security and new data structures
- Detaching detection from blocking in an async way allows for Edge termination
- Enterprise-Community communication is essential

Find Us!

- Docs
  - https://www.coraza.io/
- Github
  - https://github.com/corazawaf/coraza
- OWASP
  - https://owasp.org/www-project-coraza-web-application-firewall/
- Slack
  - https://owasp.slack.com/archives/C02BXH135AT