Ubuntu 18.04 ?? ModSecurity?? Nginx?????

下面给你一套 **Ubuntu 18.04** 部署 **ModSecurity**（含 **Nginx**）最完整、最稳定、一次成功的安装方式，适合生产环境。

? ????????

Ubuntu 18.04 上 ModSecurity 有两种方式：

?? A?Nginx + ModSecurity v3??????????

- 需要编译 ModSecurity v3 库
- 再重新编译 Nginx + ModSecurity Connector
  适合需要稳定 + 高性能。

?? B???? libapache2-mod-security2?? Apache ???

不适用 Nginx，不推荐。

? ??????? Nginx + ModSecurity v3 ????????

---

1. ????

```
sudo apt update
sudo apt install -y git build-essential autoconf automake libtool \
```

```
libpcre3 libpcre3-dev libssl-dev libxml2 libxml2-dev \
libyajl-dev libgeoip-dev pkg-config doxygen \
libcurl4-openssl-dev liblua5.3-dev
```

---

2. ????? ModSecurity v3???

```
cd /usr/local/src
sudo git clone --depth 1 https://github.com/SpiderLabs/ModSecurity
cd ModSecurity
sudo git submodule init
sudo git submodule update

sudo ./build.sh
sudo ./configure
sudo make
sudo make install
```

生成库文件:

```
/usr/local/modsecurity/lib/libmodsecurity.so
```

---

3. ????? Nginx + ModSecurity v3 Connector

3.1 ?? Connector

```
cd /usr/local/src
sudo git clone --depth 1 https://github.com/SpiderLabs/ModSecurity-nginx
```

3.2 ??????? Nginx????

（示例使用 nginx 1.18, 换成你自己版本号即可）

```
NGINX_VERSION=1.18.0

cd /usr/local/src
sudo wget http://nginx.org/download/nginx-$NGINX_VERSION.tar.gz
sudo tar zxvf nginx-$NGINX_VERSION.tar.gz
cd nginx-$NGINX_VERSION
```

3.3 ???? Nginx ??????????

```
nginx -V
```

你会看到一大串 `--with-xxx` 参数
需要包含全部参数，并额外加：

```
--add-module=/usr/local/src/ModSecurity-nginx
```

例如：

```
sudo ./configure \
 --with-http_ssl_module \
 --with-http_v2_module \
 --add-module=/usr/local/src/ModSecurity-nginx
```

然后编译并替换 nginx：

```
sudo make
sudo cp objs/nginx /usr/sbin/nginx
```

---

4. ?? ModSecurity ??

4.1 ????

```
sudo mkdir /etc/nginx/modsec
cd /etc/nginx/modsec
```

4.2 ??????

```
sudo wget https://raw.githubusercontent.com/SpiderLabs/ModSecurity/v3/master/modsecurity.conf-recommended
sudo mv modsecurity.conf-recommended modsecurity.conf
sudo wget https://raw.githubusercontent.com/SpiderLabs/ModSecurity/v3/master/unicode.mapping
```

4.3 ????

```
sudo nano /etc/nginx/modsec/modsecurity.conf
```

□□□

```
SecRuleEngine On     # □□□□□□
```

---

5. ?? OWASP CRS ?????????

```
cd /etc/nginx/modsec
sudo git clone https://github.com/coreruleset/coreruleset.git
sudo mv coreruleset rules
sudo cp rules/crs-setup.conf.example rules/crs-setup.conf
```

---

6. Nginx ??? ModSecurity

□□ nginx.conf □ vhost□

```
modsecurity on;
modsecurity_rules_file /etc/nginx/modsec/modsecurity.conf;
```

□□□

```
server {
    listen 443 ssl;

    modsecurity on;
    modsecurity_rules_file /etc/nginx/modsec/modsecurity.conf;
```

```
    root /var/www/html;
}
```

---

7. ??????? Nginx

```
sudo nginx -t
sudo systemctl restart nginx
```

? 8. ??????

访问：

```
http://your_server/?id=' OR '1'='1
```

如返回 **403 Forbidden**，说明成功。

? ?????? nginx -V ??

我可以为你生成一条你直接复制粘贴即可编译成功的 **configure** 命令，避免你手动拼接失败。

也可以根据你当前的生产环境，为你做：

□ □□□□□□
□ □□□□□□
□ □□□□□□□□□ fail2ban + CRS）
□ □□ HTTPS + WAF 一次成功的完整工程方案

你需要我为你的环境直接生成一键安装脚本吗？

下面我根据你当前环境 **nginx:1.20.2** 官方 **Docker** 镜像的编译参数，给你一套完全适配 **Docker** 环境的一键部署 **ModSecurity v3 + CRS** 规则的方案。

你现在用的是：

- **nginx:1.20.2** 官方镜像（**Debian**）

- 需要重新编译 Nginx + ModSecurity v3 Connector（因为官方镜像不带 WAF）
- 又是 Docker 环境 → 最佳方式是构建一个自定义镜像

? ??????????? “?? ModSecurity ? Nginx 1.20.2” ??

下面我已经根据你的 **nginx -V** 的 **configure** 参数自动生成了完整可直接使用的 **Dockerfile**。

你只需要复制即可使用。

? ??????

```
mkdir -p /home/shuncom/build/nginx-modsec
cd /home/shuncom/build/nginx-modsec
```

? ???? Dockerfile????????

> 这个 Dockerfile 能构建出:
> ✔ Nginx 1.20.2
> ✔ ModSecurity v3
> ✔ ModSecurity-nginx Connector
> ✔ 完整支持 CRS 规则
> ✔ 与你当前 nginx 配置完全兼容

Dockerfile?

```
FROM debian:bullseye

# □□□□□
RUN apt update && apt install -y \
    git build-essential autoconf automake libtool \
    libpcre3 libpcre3-dev libssl-dev libxml2 libxml2-dev \
    libyajl-dev pkg-config doxygen curl wget \
    libcurl4-openssl-dev liblua5.3-dev ca-certificates && \
    apt clean

# □□□□□□ ModSecurity v3
RUN cd /usr/local/src && \
    git clone --depth 1 https://github.com/SpiderLabs/ModSecurity && \
    cd ModSecurity && git submodule update --init && \
    ./build.sh && ./configure && make && make install

# □□ ModSecurity-nginx connector
RUN cd /usr/local/src && \
    git clone --depth 1 https://github.com/SpiderLabs/ModSecurity-nginx

# □□ Nginx 1.20.2 □□
RUN cd /usr/local/src && \
    wget http://nginx.org/download/nginx-1.20.2.tar.gz && \
    tar zxvf nginx-1.20.2.tar.gz

# □□□□□□ nginx -V □□□□□ Nginx
RUN cd /usr/local/src/nginx-1.20.2 && \
    ./configure \
      --prefix=/etc/nginx \
      --sbin-path=/usr/sbin/nginx \
      --modules-path=/usr/lib/nginx/modules \
      --conf-path=/etc/nginx/nginx.conf \
      --error-log-path=/var/log/nginx/error.log \
      --http-log-path=/var/log/nginx/access.log \
      --pid-path=/var/run/nginx.pid \
      --lock-path=/var/run/nginx.lock \
      --http-client-body-temp-path=/var/cache/nginx/client_temp \
      --http-proxy-temp-path=/var/cache/nginx/proxy_temp \
      --http-fastcgi-temp-path=/var/cache/nginx/fastcgi_temp \
      --http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp \
```

```
      --http-scgi-temp-path=/var/cache/nginx/scgi_temp \
      --user=nginx \
      --group=nginx \
      --with-compat \
      --with-file-aio \
      --with-threads \
      --with-http_addition_module \
      --with-http_auth_request_module \
      --with-http_dav_module \
      --with-http_flv_module \
      --with-http_gunzip_module \
      --with-http_gzip_static_module \
      --with-http_mp4_module \
      --with-http_random_index_module \
      --with-http_realip_module \
      --with-http_secure_link_module \
      --with-http_slice_module \
      --with-http_ssl_module \
      --with-http_stub_status_module \
      --with-http_sub_module \
      --with-http_v2_module \
      --with-mail \
      --with-mail_ssl_module \
      --with-stream \
      --with-stream_realip_module \
      --with-stream_ssl_module \
      --with-stream_ssl_preread_module \
      --with-cc-opt='-g -O2 -ffile-prefix-map=/data/builder/debuild/nginx-1.20.2=. -fstack-
protector-strong -Wformat -Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -fPIC' \
      --with-ld-opt='-Wl,-z,relro -Wl,-z,now -Wl,--as-needed -pie' \
      --add-module=/usr/local/src/ModSecurity-nginx && \
    make && make install

# □□□□
RUN mkdir -p /etc/nginx/modsec /var/cache/nginx/

# □□ ModSecurity □□□□
RUN cd /etc/nginx/modsec && \
    wget https://raw.githubusercontent.com/SpiderLabs/ModSecurity/v3/master/modsecurity.conf-
recommended && \
```

```
    mv modsecurity.conf-recommended modsecurity.conf && \
    wget https://raw.githubusercontent.com/SpiderLabs/ModSecurity/v3/master/unicode.mapping

# 下载 CRS 规则
RUN cd /etc/nginx/modsec && \
    git clone https://github.com/coreruleset/coreruleset.git && \
    mv coreruleset rules && \
    cp rules/crs-setup.conf.example rules/crs-setup.conf

CMD ["nginx", "-g", "daemon off;"]
```

? ??????

```
cd /home/shuncom/build/nginx-modsec
docker build -t nginx:1.20.2-modsec .
```

? ???? docker-compose.yml ?????

把你的:

```
image: nginx:1.20.2
```

改成:

```
image: nginx:1.20.2-modsec
```

? ??? nginx.conf ?? ModSecurity

在你的 server 或 http 区块里加入:

```
modsecurity on;
modsecurity_rules_file /etc/nginx/modsec/modsecurity.conf;
```

? ??????

```
docker-compose down
docker-compose up -d
```

? ????? Nginx ????? ModSecurity v3 + CRS WAF?

? ??????????????

如果你需要，我可以继续为你：

? ???????????????

? ?????????? CRS ???? POST JSON?

? ???????????

? fail2ban ?? WAF ?????????

? ????? `/etc/nginx/modsec/modsecurity.conf` ????

如果你愿意，我还可以：

◫ 直接给你一个适配你环境的“完整可运行” **compose.yml + Dockerfile** 打包工程目录。

要继续吗?

好 ——我已经为你把自动放行健康检查 **/**静态文件、**CRS** 对 **POST JSON** 的误报缓解、业务豁免规则、**fail2ban** 联动 **ModSecurity** 阻断日志自动封禁，以及一份经常用到的 **`/etc/nginx/modsec/modsecurity.conf`**（最佳实践版）都整理好了。你能直接把下面的片段复制到你当前容器 / 镜像里的对应位置（我按路径写好了）并逐条应用、测试。先把 **ModSecurity** 设为 **DetectionOnly**（检测而不阻断）进行 **24–72** 小时观察，再切换到 **On** 阻断模式。

?????

1. 先备份 `/etc/nginx/modsec/modsecurity.conf` 与现有规则目录。
2. 在生产环境启用“阻断”模式前，先用 `SecRuleEngine DetectionOnly` 观察误报。
3. 我用的 rule id 范围（1000000~ 1000100）为自定义规则，和 CRS 原有 id 不冲突。
4. 下面给出的 nginx 示例为你 docker-compose 环境可用的配置片段。
5. 若你希望我把这些直接写入你生成的镜像 /compose，我可以生成一个补丁包（不需要你再确认，直接应用也安全）——但我先把规则和说明放在这里，你可以分步执行。

---

1) ??????????????ModSecurity ???

把下面内容放到 `/etc/nginx/modsec/requests_exclusions.conf`，并在 `modsecurity.conf` 里 `Include` 这个文件（后面示例会显示如何 Include）。

```
# /etc/nginx/modsec/requests_exclusions.conf
#
# 放行健康检查（URI 前缀、User-Agent 或来自负载均衡的特定 IP）
# 注意：用 phase:1 在最早阶段控制事务，id 必须唯一且 > 1000000 避免冲突

# 1) 放行 /health 和 /healthz 等健康检查路径（全部放行）
SecRule REQUEST_URI "@beginsWith /health" \
  "id:1000001,phase:1,pass,nolog,ctl:ruleEngine=Off"

SecRule REQUEST_URI "@beginsWith /healthz" \
```

```
  "id:1000002,phase:1,pass,nolog,ctl:ruleEngine=Off"

# 2) 放行特定探测 User-Agent（如 k8s kube-probe、ELB health）
SecRule REQUEST_HEADERS:User-Agent "kube-probe|ELB-HealthChecker" \
  "id:1000003,phase:1,pass,nolog,ctl:ruleEngine=Off"

# 3) 放行来自内网 LB 的 IP 列表（用空格分隔多个 IP）
# 例: 10.0.0.5 为负载均衡器
SecRule REMOTE_ADDR "@ipMatch 10.0.0.5/32 10.0.1.0/24" \
  "id:1000004,phase:1,pass,nolog,ctl:ruleEngine=Off"

# 4) 静态文件（扩展名）直接放行（避免大文件/静态被误判）
SecRule REQUEST_URI "\.(?:css|js|png|jpg|jpeg|gif|webp|svg|ico|woff2?|ttf|map)(?:$|\?)" \
  "id:1000005,phase:1,pass,nolog,ctl:ruleEngine=Off"
```

解释：`ctl:ruleEngine=Off` 会关闭该次请求的规则检查（适用于确实只读、无风险的健康 / 静态请求）。如果你更保守，也可以把 `ctl:ruleEngine=DetectionOnly`。

---

2) ???? CRS ? POST JSON ????????????? Method A?

目的：避免 CRS 将正常的 `application/json` POST 内容识别为 SQLi/XSS 等（常见误报场景）。

A??? A????—— ? JSON ???? JSON ?????? Body ??

把以下放进 `/etc/nginx/modsec/json_tuning.conf`：

```
# /etc/nginx/modsec/json_tuning.conf
# 在 phase:1 将 requestBodyProcessor 设为 JSON（更稳定），并对常见会误报的规则作“局部关闭/降低”或开启检测模式

# 1) 对所有 Content-Type: application/json 的请求:
SecRule REQUEST_HEADERS:Content-Type "application/json" \
  "id:1000010,phase:1,pass,nolog,ctl:requestBodyProcessor=JSON,ctl:ruleEngine=DetectionOnly"
```

```
# 2) 对特定 URI 前缀（例如 /api/）优先把评分阈值提升（如果你使用 Anomaly Scoring）
# 这里将 transaction 的评分阈值在后续阶段提高（示例: +10）
SecRule REQUEST_URI "@beginsWith /api/" \

"id:1000011,phase:1,pass,nolog,ctl:requestBodyProcessor=JSON,ctl:ruleRemoveByTag=OWASP_CRS/WEB_ATTACK/PROTOCOL_VIOLATION"

# 说明:
# - 第一条把 JSON 请求的 body 解析器设为 JSON（比默认的 urlencoded/text 解析更稳）
# - 同时把 ruleEngine 设为 DetectionOnly（检测模式），以便观察具体哪个规则误报
# - 第二条示例展示了如何按 tag 移除一类 CRS 规则（替换成你实际要移除的 tag）
```

> 备注: 如果你不想把整条 JSON 请求都设为 DetectionOnly, 可把 `ctl:ruleEngine=DetectionOnly` 去掉, 改为 `ctl:ruleRemoveById=xxxx,ctl:ruleRemoveByTag=tagname` 精确移除若干误报规则（但需要先通过检测模式查出误报的 rule id 或 tag）。

B??? B????—— ??????? request body ????????

例如 `/api/webhook/`, 如果这类接口经常被误报, 可以完全禁用 Body 检查:

```
# /etc/nginx/modsec/json_tuning.conf (补充)
SecRule REQUEST_URI "@beginsWith /api/webhook" \
  "id:1000015,phase:1,pass,nolog,ctl:requestBodyAccess=Off,ctl:ruleEngine=DetectionOnly"
```

> 注意: `requestBodyAccess=Off` 会完全跳过对 body 的检查 —— 适合那些你非常信任且只需要 header 校验的 webhook。

---

3) ???????????? URI?? header ?? IP?? cookie?

将下面文件保存为 `/etc/nginx/modsec/business_exceptions.conf`：

```
# /etc/nginx/modsec/business_exceptions.conf

# 1) 特定管理后台路径放宽（示例）
SecRule REQUEST_URI "@beginsWith /admin/" \
  "id:1000020,phase:1,pass,nolog,ctl:ruleEngine=DetectionOnly"

# 2) API Key header 的服务间调用放行（例 header: X-Internal-API-Key）
SecRule REQUEST_HEADERS:X-Internal-API-Key "@rx ^[A-Za-z0-9\-_]{20,}$" \
  "id:1000021,phase:1,pass,nolog,ctl:ruleEngine=Off"

# 3) 对某些 Cookie（内网 SSO）放宽
SecRule REQUEST_COOKIES:SSO_SESSION "@rx ^[A-Za-z0-9\-_=]{10,}$" \
  "id:1000022,phase:1,pass,nolog,ctl:ruleEngine=DetectionOnly"
```

原则：越明确的匹配越安全（例如固定 URL 前缀、固定 header 名称或仅在内网 IP 生效）。

---

4) fail2ban ?? WAF??? Nginx error.log ? ModSecurity ?????????

4.1 ? modsecurity.conf ??????? nginx error.log ???????????

在 `modsecurity.conf` 中确保 `SecAuditLog` 与 nginx error_log 配置合理；同时 nginx/ErrorLog 通常会打印 ModSecurity 拒绝语句，例如：

```
[error] 12345#0: *12345 ModSecurity: Access denied with code 403 (phase 2). Pattern match
"..." at ARGS:username. [...]
```

（你的 nginx 已经以 `--error-log-path=/var/log/nginx/error.log` 编译）

4.2 fail2ban filter????

`/etc/fail2ban/filter.d/modsecurity.conf`?

```
# /etc/fail2ban/filter.d/modsecurity.conf
[Definition]
failregex = \[error\].*ModSecurity: Access denied with code 403.*client: <HOST>
            \[error\].*ModSecurity: Access denied with code 403.*client: <HOST>\]
ignoreregex =
```

> 说明：某些系统的 error.log 里格式不同，上面是常见匹配。你启动 DetectionOnly 后，可触发一次人工封堵，观察 nginx error.log 中实际行格式，再调整 `failregex`（我可以帮你精确化）。

4.3 fail2ban jail????

`/etc/fail2ban/jail.d/modsecurity.local`?

```
# /etc/fail2ban/jail.d/modsecurity.local
[modsecurity]
enabled  = true
filter   = modsecurity
action   = iptables-allports[name=ModSecurity, port="http,https"]
logpath  = /var/log/nginx/error.log
maxretry = 3
findtime = 600
bantime  = 3600
```

4.4 ?? / ?? fail2ban

```
sudo systemctl restart fail2ban
# 检查状态
sudo fail2ban-client status modsecurity
sudo fail2ban-client status
```

测试方法：临时在 nginx 上触发一个可被 ModSecurity 拦截的请求（在检测模式下生成 error.log 行），观察 fail2ban 是否识别并封禁 IP。生产中建议把 `bantime` 和 `maxretry` 调到合适值。

---

5) ????????? `modsecurity.conf`????

在 `/etc/nginx/modsec/modsecurity.conf` 末尾添加：

```
# include your tuning files (order matters: exclusions should be early)
Include /etc/nginx/modsec/requests_exclusions.conf
Include /etc/nginx/modsec/json_tuning.conf
Include /etc/nginx/modsec/business_exceptions.conf
```

（或在 `nginx.conf` 指定 `modsecurity_rules_file` 指向一个总入口规则文件，这个入口再 Include 上述子文件。）

---

6) ??? `modsecurity.conf` ?? / ??????????

下面是一个精简但实用的 `/etc/nginx/modsec/modsecurity.conf`（把常用且重要的选项都列上），你可以直接替换或合并到现有配置中。注意注释——按需修改。

> 将下面内容保存为 `/etc/nginx/modsec/modsecurity.conf`（替换原文件前请备份）。

```
# /etc/nginx/modsec/modsecurity.conf (best-practice compact)

# 基本运行模式：先用 DetectionOnly 监测 24-72 小时，确认没大量误报再改为 On
```

```
SecRuleEngine DetectionOnly

# 日志与审计
SecAuditEngine RelevantOnly
SecAuditLog /var/log/modsec_audit.log
# Parts: A (request headers), B (request body), F (response headers), H (audit log trailer)...
SecAuditLogParts ABIJDEFHZ
SecAuditLogType Serial
SecAuditLogStorageDir /var/log/modsec_audit/

# 全局默认动作（Anomaly scoring 推荐）
SecDefaultAction "phase:1,log,pass"
SecDefaultAction "phase:2,log,deny,status:403"

# 请求体/响应体访问控制（默认开启请求体检查）
SecRequestBodyAccess On
SecRequestBodyLimit 13107200       # 12.5 MB （根据业务调整）
SecRequestBodyNoFilesLimit 131072  # 非文件 body 限制（128KB）
SecRequestBodyInMemoryLimit 131072 # 128KB

# 最大文件上传限制（CRS 有时也需要）
SecUploadDir /var/cache/modsecurity/uploads
SecRequestBodyLimitAction Reject

# 响应体分析（默认关闭，除非特定需求）
SecResponseBodyAccess Off
# 若开启，限制大小以免 OOM
SecResponseBodyLimit 5242880       # 5 MB

# utf-8 / unicode 映射
SecUnicodeMapFile /etc/nginx/modsec/unicode.mapping

# 安全策略：允许临时禁用某些规则
# （自定义 rules 文件在后面 Include）

# 兼容 CRS：指定 CRS path（CRS 安装后）
Include /etc/nginx/modsec/rules/*.conf

# 方便调试（生产建议关闭调试）
#SecDebugLog /var/log/modsec_debug.log
```

```
#SecDebugLogLevel 3

# 速率限制集成点（如与你的 nginx limit_req 联动）
# 若使用 Anomaly Scoring，设定阈值（示例）
#SecAction "id:900000,phase:1,pass,initcol:ip=%{REMOTE_ADDR},setvar:ip.score=0"

# 自定义 include（放在最后，保证你自定义的排除/豁免被加载）
Include /etc/nginx/modsec/requests_exclusions.conf
Include /etc/nginx/modsec/json_tuning.conf
Include /etc/nginx/modsec/business_exceptions.conf
```

> 说明：
>
> - `SecAuditEngine RelevantOnly`：只记录含有触发或异常的事务，避免太大日志。也可改为 `On` 全记录（日志量大）。
> - `SecRequestBodyLimit` 必须高于你 APP 最大 POST 大小（例如文件上传）。
> - `SecResponseBodyAccess Off` 推荐默认关闭（响应体检查 CPU/内存消耗较高）。

---

7) nginx ?????? modsecurity ??????????

在你的 `nginx.conf` `http {}` 或具体 `server{}` 中加入：

```
# 在 http {} 顶部（或 server/ location）
load_module modules/ngx_http_modsecurity_module.so;

modsecurity on;
modsecurity_rules_file /etc/nginx/modsec/modsecurity.conf;
```

（你在镜像里编译时，已经用 modsecurity-nginx connector 集成，确保 load_module 路径与你镜像实际模块路径一致。官方编译为动态模块时需要 load_module；静态编译可直接用 `modsecurity on;`）

---

8) ? DetectionOnly ? On ???????????

1. 在 `modsecurity.conf` 保持 `SecRuleEngine DetectionOnly`，启动并观察 48–72 小时。
2. 收集误报：`/var/log/modsec_audit.log` 与 nginx error.log。找出误报的 rule id（记录会标出匹配 id）。
3. 对高误报的 rule 使用 `ctl:ruleRemoveById=ID` 或写更精确的 `SecRule` 以排除。也可以把特定 URI 设为 `ruleEngine=Off`。
4. 调整 Anomaly Scoring 阈值（CRS 有 `threshold`）或把某些 router 设置为 DetectionOnly。
5. 确认误报控制在可接受范围后，将 `SecRuleEngine On` 切换为阻断模式。
6. 继续定期检查 audit logs，更新豁免规则。

---

9) ?????????????

- 把 `health` 与 `static` 放在最前面（phase:1），减轻 WAF 负担。
- 对 JSON heavy API：优先使用 JSON 解析器（`ctl:requestBodyProcessor=JSON`）。
- 大流量场景开启 `SecAuditEngine RelevantOnly`，并把 `SecAuditLog` 存到独立磁盘或集中收集（ELK）。
- 配合 `limit_req` / `limit_conn` 做 L7 防刷，并用 fail2ban 及时封禁扫描源。
- CRS 默认规则集经常更新，务必定期 `git pull` 并在预演环境运行测试。
- 保持规则 id 表的文档（哪些是你添加的、哪些被移除），便于审计。

---

10) ??????????

- 健康检查：`curl -I http://yourhost/health` → 应该返回 200，不被 WAF 阻断
- 静态文件：`curl -I http://yourhost/static/image.jpg` → 200
- JSON POST：`curl -X POST -H 'Content-Type: application/json' -d '{"name":"a"}' http://yourhost/api/test` → 应不被误报（先确认 DetectionOnly 模式里没有规则触发）
- 模拟注入（仅检测模式）：`curl "http://yourhost/?id=' or 1=1 --"` → 应触发日志（如果 DetectionOnly，会记录但不阻断）

如果你要，我可以立刻：

- 基于你当前 docker-compose 路径，生成一份补丁包，包括 Dockerfile 的小改动（把文件写入镜像）和 `docker-compose.override.yml` 的示例，或

- 根据你48小时检测日志（把一段`modsec_audit.log`的sample粘过来），我来直接帮你写精确的`ctl:ruleRemoveById`列表和fail2ban正则优化。

你想直接把这些规则放进容器（我给出准确命令）还是先试运行并把检测期的audit日志贴过来让我分析误报?

下面我给你总结一套**Nginx + ModSecurity v3 + CRS WAF**在实际生产环境中能实现的完整安全能力（涵盖OWASP Top 10、常见Web攻击、Bot防护、协议校验、日志与响应体系），便于你评估是否满足你的公司平台安全需求。

? ?????ModSecurity v3 + CRS = ??? Web ????? WAF

具备完整应用层防护（L7 Security），覆盖:

- OWASP Top 10（SQL、XSS、文件包含、权限绕过…）
- 业务层安全保护（参数校验、攻击语义分析）
- 协议校验（HTTP RFC）
- 智能启发式攻击识别
- 日志、审计、行为分析
- 与Nginx完美集成，性能远优于ModSec v2

? ???? & ?????

ModSecurity v3 + CRS对HTTP协议提供严格校验:

? HTTP RFC ????

- 检查HTTP头是否合法
- 检查非法方法（如`TRACE`、`CONNECT`）
- 检查过长URI、Header、Body
- 防止Request Smuggling（请求走私）
- 防止Response Splitting（响应拆分）

? ????????

- FD（文件描述符）耗尽
- 无效 Content-Length
- Chunked 伪造攻击
- 不完整的 multipart 请求

? ??????? OWASP Top 10 ????

1. SQL ???SQLi?

CRS 能检测：

- 单引号、注释符尝试
- `UNION SELECT`
- `OR 1=1`
- `sleep(5)` / 盲注
- MySQL / PostgreSQL / MSSQL 语义特征

支持高级语义识别：绕过编码、空格替换、**%00**、**HEX** 编码、**ASCII** 编码等。

---

2. XSS ??????

包括：

- `<script>` 注入
- onload、onclick、javascript:、data:
- DOM XSS
- 编码绕过（HTML/URL/Base64）

---

3. ???? / ???????LFI?

拦截：

- `/../../etc/passwd`
- `php://filter/resource=`
- `file:///`

---

4. ???????RFI?

检测:

- `http://evil.com/shell.jpg`
- 远程　PHP 包含
- URL 伪装　/重定向攻击

---

5. ?????CMD Injection?

识别:

- `; rm -rf /`
- `| cat /etc/passwd`
- "`$()`"、"`"反引号
- PowerShell / Bash 命令

---

6. ????

- PHP、JSP、Lua、Python 注入
- eval(), system(), exec()

---

7. Java ???????

- CommonsCollections payload
- JRMP
- ysoserial 攻击流量特征

---

8. ???? / JSON ??

- JSON 结构污染
- 注入脚本
- 流量中含反序列化特征序列

---

9. ?? UA/Referer ?????

- BurpSuite、sqlmap
- Nmap、masscan
- Python requests, curl
- 自写攻击脚本流量模式

? ??Bot ?????CRS + ?????

CRS 自带 Bot 检测：

- 自动识别扫描器（sqlmap、nmap、Nikto）
- 自动识别爬虫 UA
- 识别异常访问模式
- 可选：灰名单 + 速率限制
- 与 Nginx limit_req 配合可实现 Anti-DDoS L7 防护

? ????????

拦截：

- 上传包含恶意 Payload 的文件
- PHP 藏木马
- 伪造图片 “JPG+PHP”
- multipart 边界异常
- Content-Type 伪造

可限制：

- 文件类型
- MIME / 后缀匹配
- 文件大小
- 文件内容特征匹配

? ???????? / ????? (DLP)

CRS 能发现敏感信息:

- 身份证号
- 手机号码
- 银行卡号
- 邮箱地址泄露
- Base64 隐写数据
- 内部系统路径泄露

? ??????????Heuristic Detection?

CRS 具有“语义检测”:

- 判断攻击意图，而不仅是关键字
- 多规则分段评分 → 很难绕过
- 自动组合行为判断
- 流量模式检测（多次试探攻击）

这类智能特征识别能力比传统 Web 防火墙更强。

? ??????????????????

ModSecurity v3 支持:

? ???????????

包括:

- 请求头
- 请求 Body
- 响应头、Body
- 匹配的 WAF 规则
- 攻击来源 IP & 时间

可以接入：

- ELK（ Kibana）
- Loki / Grafana
- SIEM（如阿里云、 Wazuh 等）

? ??????????????

- CRS 支持 “Anomaly Scoring 模式 ”（推荐，不会随便 403）
- 低误报、多级评分
- 业务参数豁免（白名单）
- 自定义规则（放行或增强）

ModSecurity v3 + CRS 是市场上误报处理体验最好的开源 **WAF**。

? ???????????

你可以编写自己的规则：

- 阻止某类接口访问
- 参数只允许特定格式
- 某些 IP 禁止访问
- 某些路径强制加密
- 防止越权访问（基于 Referer / Cookie）

语法非常灵活，能满足绝大多数企业业务安全场景。

? ???ModSecurity v3 + CRS ??????

? ?? OWASP Top 10 ??

? SQL/XSS/Upload/LFI/RFI/CMD ?????

? Anti-Bot / Anti-Scanner

? ??????

? HTTP ??????

? ???? / ????

? ??????????

? ??? Web ??????

适合:

- 企业内网平台
- B/S 架构项目
- Java/PHP/Python/Go 任意 Web 系统
- API 网关安全支持

? ?????

我可以继续提供:

? 1????? **?? ModSecurity ?????**??????

? 2??????? CRS ??

? 3???????? WAF ????

? 4????? + ????

? 5?? Nginx ???“???? WAF/CRS”??

需要的话告诉我你的业务接口类型（ REST、表单、上传等），我可以为你生成 定制化规则 。

Revision #3
Created 25 November 2025 03:30:39 by Admin
Updated 25 November 2025 03:45:38 by Admin