CrowdSec

Crowdsec

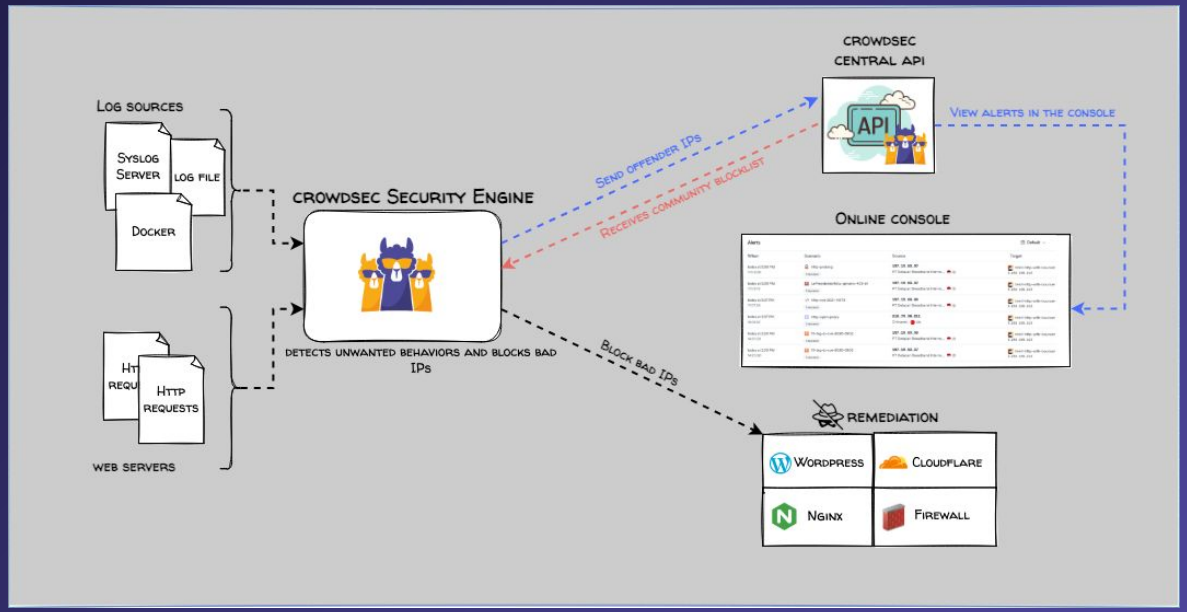
Application Security

Open WAF Day Barcelona

Actionable Collective Threat Intelligence

What is Crowdsec

- Community-oriented project
- Analyzes your logs (or HTTP request) in real time
- Automatically shares information about attacks you detect
- You receive a blocklist of the most aggressive IPs in our network



Why did we implement a WAF

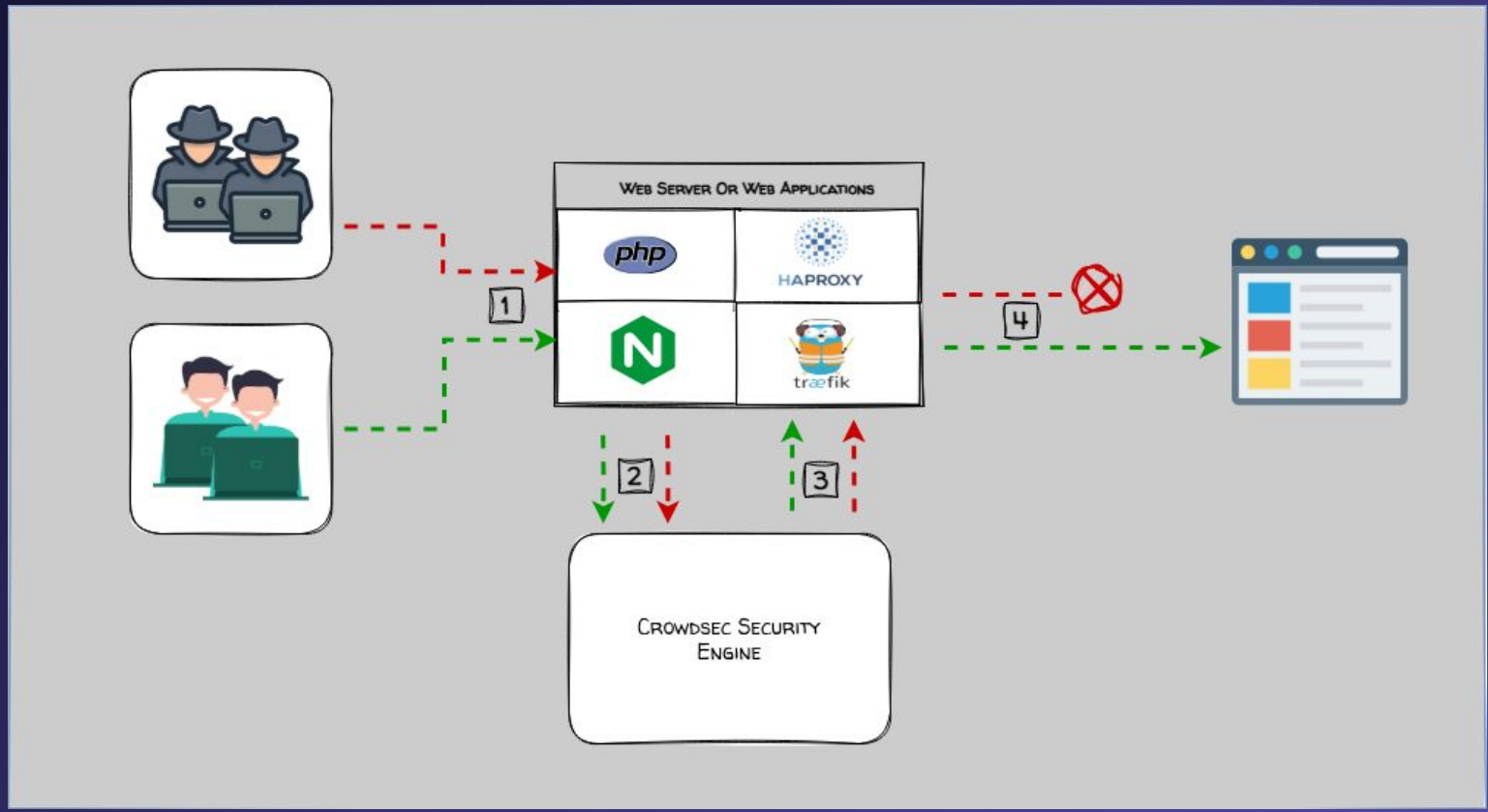
> “Any security software that can do pattern matching will be used as a poor man’s WAF at some point.”

- Majority of our users are only exposing/protecting HTTP servers
- At the time, modsecurity was EOL with no clear path for the future
- Some of them were using crowdsec as a WAF
  - But because we read logs, the block only happens after the request
  - Logs only give a very incomplete picture of the request

Why Coraza

- We did not want to roll our own WAF engine
  - Already did that in the past (naxsi)
- Established project
- Implemented in Go
- modsecurity rules compatibility (although not 100%)
  - CRS compatibility

Architecture



Our Approach

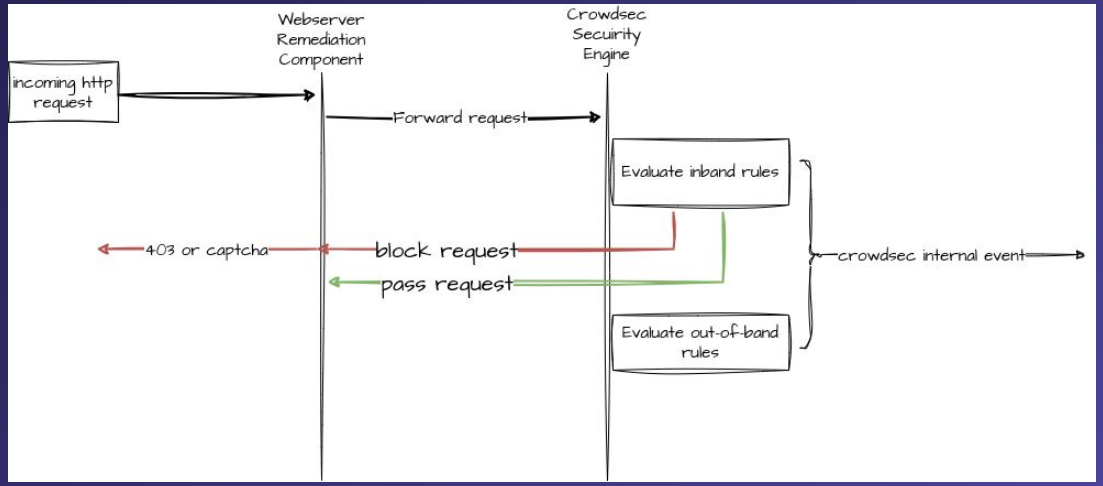
- CRS handles the generic detection
- We are focusing on virtual patching rules
  - We aim for the setup to be as simple as possible
- Seclang is hard, hide it as much as possible from the user

Extending Coraza capabilities

- Out-Of-Band evaluation
- Behavioral detection
- Easier runtime configuration with hooks

Extending Coraza capabilities : Out of band evaluation

- Testing a new rule directly in production
- Expensive rules
- Detect repetitive actions (eg, scrapping)
- We provide CRS as out-of-band by default:
  - Allows to use them on any website without configuration
  - But will not block any requests on its own



Extending Coraza capabilities : Integration with crowdsec

- When a rule is matched:
  - A crowdsec event is generated
  - The event goes through the parsers/scenarios pipeline
  - Allow to take decisions upon out-of-band matches
  - Allow to take long term decisions against repeating offenders

```
type: leaky
format: 3.0
name: crowdsecurity/appsec-vpatch
description: "Identify attacks flagged by CrowdSec AppSec"
## See appsec-native.yaml for reasons why we created a negative startsWith here, we want to ignore
filter: "evt.Meta.log_type == 'appsec-block' && evt.Meta.rule_name not startsWith 'native_rule'"
distinct: evt.Meta.rule_name
leakspeed: "60s"
capacity: 1
groupby: evt.Meta.source_ip
blackhole: 1m
```

Extending Coraza capabilities: hooks

- Allow for runtime configuration
- Think `SecRuleUpdateByXXX`, `SecRuleRemoveByXXX` but more flexible

Extending Coraza capabilities: on_load

- Executed during coraza initialization
- Can be used to globally disable rules (`SecRuleRemoveByXXX` alternative)
- Set a specific remediation (ban or captcha) globally

```
name: crowdsecurity/my-appsec-config
default_remediation: ban
inband_rules:
 - crowdsecurity/base-config
 - crowdsecurity/vpatch-*
on_load:
 - apply:
    - RemoveInBandRuleByName("my_rule")
    - SetRemediationByTag("my_tag", "captcha")
```

Extending Coraza capabilities: pre_eval

- Executed before the request is passed to coraza
- Can be used to disable rules based on the client request
- Set a specific remediation (ban or captcha) based on the request
- Full HTTP request is available

```
name: crowdsecurity/my-appsec-config
default_remediation: ban
inband_rules:
 - crowdsecurity/base-config
 - crowdsecurity/vpatch-*
pre_eval:
 - filter: IsInBand == true && req.RemoteAddr == "192.168.1.1"
   apply:
    - RemoveInBandRuleByName("my_rule")
```

Extending Coraza capabilities: post_eval

- Executed after coraza returns
- Intended for debugging or threat hunting
- Can dump the full request to disk
- Has access to the request for filtering

```
name: crowdsecurity/my-appsec-config
default_remediation: ban
inband_rules:
 - crowdsecurity/base-config
 - crowdsecurity/vpatch-*
post_eval:
 - filter: IsInBand == true
   apply:
    - DumpRequest().NoFilters().WithBody().ToJSON()
```

Extending Coraza capabilities: on_match

- Only called if a request has matched
- Has access to the full request
- Last chance to change the remediation
- Can prevent an event (or alert) from being created

```
name: crowdsecurity/my-appsec-config
default_remediation: ban
inband_rules:
 - crowdsecurity/base-config
 - crowdsecurity/vpatch-*
on_match:
 - filter: IsInBand == true && req.RemoteAddr == "192.168.1.1"
   apply:
    - CancelAlert()
    - CancelEvent()
  - filter: |
      any( evt.Appsec.MatchedRules, #.name == "crowdsecurity/vpatch-env-access") and
      req.RemoteAddr = "192.168.1.1"
    apply:
    - SetRemediation("allow")
  - filter: evt.Appsec.MatchedRules.GetURI() contains "/foobar/"
    apply:
     - SetRemediation("allow")
```

Custom rules format

```
rules:
  - and:
    - zones:
      - URI
      transform:
      - lowercase
      match:
        type: endsWith
        value: /wp-admin/admin-ajax.php
    - zones:
      - ARGS
      variables:
        - action
      match:
        type: equals
        value: duplicator_download
    - zones:
      - ARGS
      variables:
        - file
      match:
        type: contains
        value: ".."
```

- Seclang can be very terse
- Lots of gotchas if you are not familiar with it
- Temporary solution
- End goal is `req.URI endsWith "/wp-admin/admin-ajax.php" && req.args.action == "duplicator_download" && ".." in req.args.file`

Custom rules format

```
SecRule REQUEST_FILENAME "@endsWith /wp-admin/admin-ajax.php"
"id:100238081,phase:2,deny,log,msg:'crowdsecurity/vpatch-CVE-2020-11738',tag:'crowdsec-crowdsecurity/vpatch-CVE-2020-11738',t:lowercase,chain"
SecRule ARGS_GET:action "@streq duplicator_download"
"id:90315028,phase:2,deny,log,msg:'crowdsecurity/vpatch-CVE-2020-11738',tag:'crowdsec-crowdsecurity/vpatch-CVE-2020-11738',chain"
SecRule ARGS_GET:file "@contains .."
"id:956980145,phase:2,deny,log,msg:'crowdsecurity/vpatch-CVE-2020-11738',tag:'crowdsec-crowdsecurity/vpatch-CVE-2020-11738'"
```

Testing

- Two main goals:
  - Make contributing new rules easier:
    - Anybody can submit new rules for integration, so we need to have an easy way to test them (for correctness and false positives)
  - Make reviewing (and understanding them) rules easier

Testing for true positives

- Make sure a new rule blocks exploitation attempts
  - Use existing exploits
  - Reverse the patch to write it
- Based on crowdsec existing testing framework
- Use nuclei templates to describe HTTP requests
- Test for behaviour, not internal state
- End-to-end test:
  - Client -> web server -> crowdsec

Testing for true positives

```
appsec-rules:
- ./appsec-rules/crowdsecurity/base-config.yaml
- ./appsec-rules/crowdsecurity/vpatch-CVE-2024-8963.yaml
nuclei_template: test-CVE-2024-8963.yaml
```

Testing for true positives

```
id: test-CVE-2024-8963
info:
  name: test-CVE-2024-8963
  author: crowdsec
  severity: info
  description: test-CVE-2024-8963 testing
  tags: appsec-testing
http:
  - raw:
      - |
        GET /client/index.php%3F.php/gsb/users.php HTTP/1.1
        Host: {{Hostname}}
        Content-Type: application/x-www-form-urlencoded

    cookie-reuse: true
    matchers:
      - type: dsl
        condition: and
        dsl:
          - "status_code_1 == 403"
```

Testing for false positives

- Make sure a new rule will not trigger false positives
- Finding a good dataset of legitimate queries is hard !
  - You can either build your own (very time consuming)
  - Or try to find an existing one
- Settled on open-appsec legitimate queries dataset
  - Contains about 1 millions HTTP request made on 185 websites
- CI job spawns nginx + crowdsec and replay all the requests

Community

- ~100k active crowdsec installations
  - ~3.5k WAF users
  - One installation can cover a lot of websites

Most reported virtual patching rules

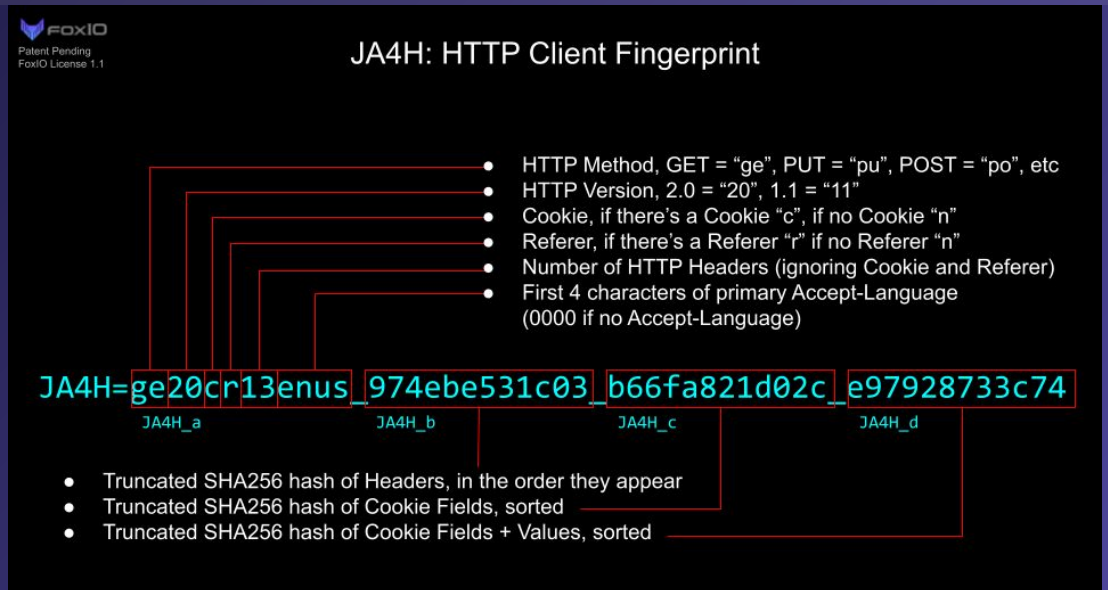
total_signals	scenario	total_ips
976,196	crowdsecurity/vpatch-env-access	16,950
473,228	crowdsecurity/vpatch-git-config	13,024
56,598	crowdsecurity/vpatch-symfony-profiler	3,729
36,853	crowdsecurity/vpatch-laravel-debug-mode	2,796
63,294	crowdsecurity/vpatch-CVE-2017-9841	1,464
3,277	crowdsecurity/vpatch-CVE-2022-41082	1,121
10,148	crowdsecurity/vpatch-CVE-2024-4577	864
1,846	crowdsecurity/vpatch-CVE-2018-20062	287
1,133	crowdsecurity/vpatch-CVE-2023-23752	275
811	crowdsecurity/vpatch-CVE-2022-35914	187

Most reported CRS rules

total_signals	crs_id	total_ips
606,619	930130	7,124
56,374	920350	1,898
89,669	942100	1,875
69,266	920440	1,556
60,786	953100	1,420
62,714	920420	1,392
68,061	932235	1,337
72,842	933150	1,193
27,672	941100	1,096
43,789	941160	969

Bot Classification

- When a request is blocked, JA4H hash is computed
- Most bots don't even bother to properly impersonate a web browser
  - They might have the proper UA
  - But no accept-language, send very little headers, …



What's next

- Automate rule creation
  - PoC for now, we had great success using an LLM to automatically generate a virtual-patching rule from an exploit
- OpenAPI schema validation
  - Goes further than just JSON schema validation

CrowdSec

Crowdsec Documentation: https://docs.crowdsec.net/

WAF Documentation: https://docs.crowdsec.net/docs/next/appsec/intro

Discord: https://discord.com/invite/crowdsec

Rules:

https://github.com/crowdsecurity/hub/tree/master/appsec-rules/crowdsecurity