WWW.HAPROXY.COM

Sigurnost Web aplikacija sa HAProxy LB-om

ACLs, maps, stick tables, enterprise modules, DoS filtering, WAF, etc.

Dinko Korunić

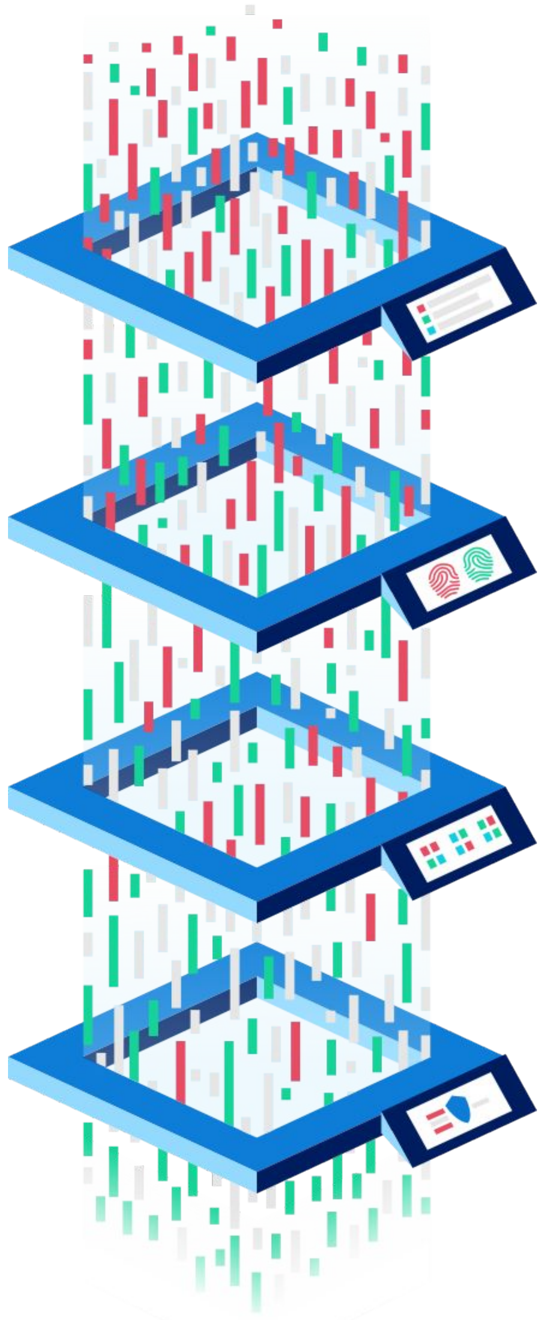
Senior Systems Engineer @ HAProxy Technologies

Sadržaj

Sadržaj

- Uvod u HAProxy
- Osnovne građevne jedinice: ACLs, stick tables, maps
- Obrana od zlonamjernog prometa botova, filtriranje aplikacijskih napada i DoS prometa itd.
- Sinkronizacija praćenja između HAProxy LB-ova
- Enterprise moduli: fingerprinting, WAF, Stream Processing Offload Engine i ostala proširenja

Napomena: Puno je previše sadržaja i slajdova, mnoge teme ćemo samo površno proći… Bit će refresher za postojeće korisnike i uvod u koncepte za zainteresirane.



Multilayered Security

Identifying and stopping threats in today's ever-changing security landscape requires a multilayered approach. HAProxy delivers peace of mind by immobilizing threats at the edge without sacrificing the best-in-class performance that it's known for.

LAYER 1 – Access Control Lists (ACLs)

The first layer is our flexible Access Control Lists (ACLs). They match on custom-defined criteria, allowing you to make routing decisions and implement protection mechanisms based on anything found within the request/response headers or metadata. You can easily create policies that match clients and requests by IP range, SSL data, headers or paths, geolocation, and device type.

ACL, Map files, and TLS ticket keys can be updated from a central location at a defined interval using the dynamic update module included in HAProxy Enterprise.

LAYER 2 – Client Fingerprinting

A second layer unmasks clients that try to sneak past with forged request data. HAProxy attaches fingerprints to clients and is able to triangulate on the data to form an accurate ID.

Bots and scanners are identified immediately, before they have a chance to do harm.

LAYER 3 – Realtime Cluster-wide Tracking

The third layer of defense deploys behavior-analysis across your entire cluster of proxies. HAProxy performs real-time tracking of client requests and stores that data to form big-picture insights about what a client may be trying to do.

Track behavior based on IP address, User-Agent string, session ID, and request path, and much more. Generated metrics include requests/sec, total number of requests made, errors/sec, total number of errors, byte rates, and more.

LAYER 4 – Web Application Firewall (WAF)

HAProxy provides a fourth layer of defense: an integrated Web Application Firewall (WAF). The WAF detects and stops Layer 7 attacks including SQL injection and cross-site scripting.

The HAProxy WAF comes with support for ModSecurity rulesets, whitelist-only mode, and an optional, simplified, set-and-forget SQLi / XSS WAF mode.

Razvoj i CE/EE razlike

Korisnici

Booking.com
STACKPATH
GitHub
tumblr.
Instagram
twilio
vimeo
stackoverflow
stripe
reddit
Red Hat OpenShift
airbnb
Alibaba.com
DISQUS
THREDUP Secondhand Clothes. Firsthand Fun.
EGNYTE
fedora
Virgin america
W3C®
TubeMogul
imvu
imgur
GloboTech COMMUNICATIONS
BIBLIO .com
tuenti
bitpusher
a0labs

HAProxy CE

- Novi major release svakih 6 mjeseci, novi LTS jednom godišnje (parni branch)
- Community Edition: HAProxy je OpenSource, trenutni LTS je 2.4 branch (supportan do 2026-Q2) a development je 2.5-dev: https://www.haproxy.org/
- Niz povezanih OpenSource projekata:
  - Kubernetes Ingress Controller: https://github.com/haproxytech/kubernetes-ingress
  - Data Plane API: https://github.com/haproxytech/dataplaneapi
  - Docker images (Alpine, Debian, Ubuntu + multiarch manifests): https://hub.docker.com/u/haproxytech
  - Helm Charts (HAProxy, IC): https://github.com/haproxytech/helm-charts
  - Connector for Consul Connect: https://github.com/haproxytech/haproxy-consul-connect
  - Lua libraries (Oauth/JWT, helpers, filters, ACME)
  - SPOA libraries and implementations
  - benchmarks, cloud (AWS, Azure etc.) integration blueprints

HAProxy EE

- Enterprise Edition: ultra-stable codebase + feature backports
  - LTS: 2.2.r1 (EOL Feb 2025)
  - non-LTS: 2.3r1 (EOL Nov 2022)
- Razlike: support + dodatni moduli!
  - Fingerprint module
  - Realtime Cluster-wide Tracking (Stktagg)
  - WAF: Simple (XSS/SQLi), Advanced (black+white), ModSecurity (+CRS)
  - Dynamic ACL updates (lb_update)
  - Antibot (JS challenge) module
  - reCAPTCHA (v2/v3) Lua library/module
  - Search Engine Verification (SPOA verify crawler)
  - Real-Time Dashboard
  - DataPlane EE

HAProxy EE (2)

- Specijalno:
  - PacketShield: high-perf (TCP: ACK, SYN, RST flood) firewall (NDIV - XDP)
  - custom Keepalived (VRRP) branch
  - razni dodatni moduli tipa Verify Crawler, SSO, itd.
  - ALOHA HAProxy HW
  - WIP: Aloha MT, Fusion Control Plane, ...
- Distribucije i OS:
  - RHEL 7, RHEL 8, Ubuntu 20.04 Focal Fossa, Ubuntu 18.04 Bionic Beaver, SUSE Linux Enterprise Server, VMware Photon OS, FreeBSD, itd.
  - Azure Marketplace, AWS Marketplace (EC2 AMI)
  - Docker images (samo Ubuntu)
  - Kubernetes Ingress Controller EE
- Fusion Control Plane: deployment, kontrola (Data Plane API), agregacija (stick tables i logovi), cloud integracija (AWS)

Performanse

EBtree

- http://git.1wt.eu/web/ebtree.git/

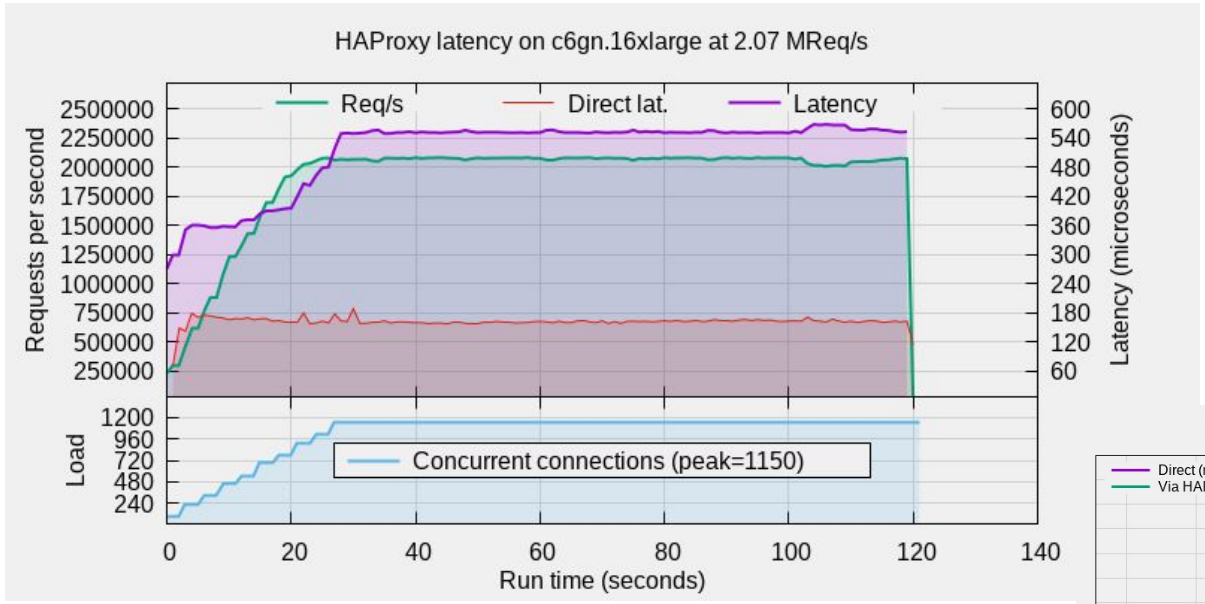
- Elastic Balanced Tree
- optimized for frequent inserts and deletes without additional memory management
  - 100ns inserts
  - >200k TCP conn/s
  - >350k HTTP req/s
  - resulting in 3-5% CPU usage!

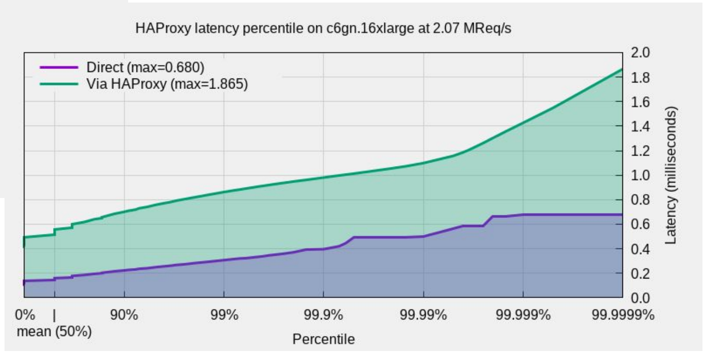
- Halog: 4M log line/s
- 450000 BGP routes table: >2M lookup/s

- HAProxy use: timers, schedulers, ACL, stick-tables (stats, counters), LRU cache...

Performanse

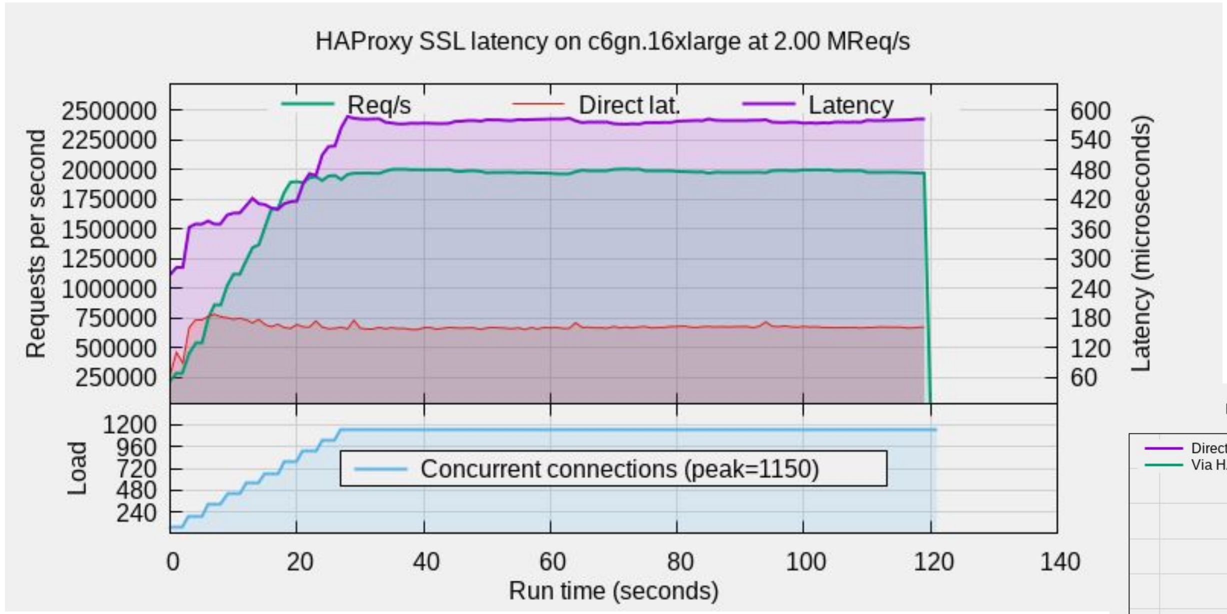
- AWS c6gn.16xlarge Graviton2 instanca (64-core): **100Gbps** wire-speed i **2M HTTPS req/s** forwarding
- HTTP - oko 1.87ms dodatna latencija per req

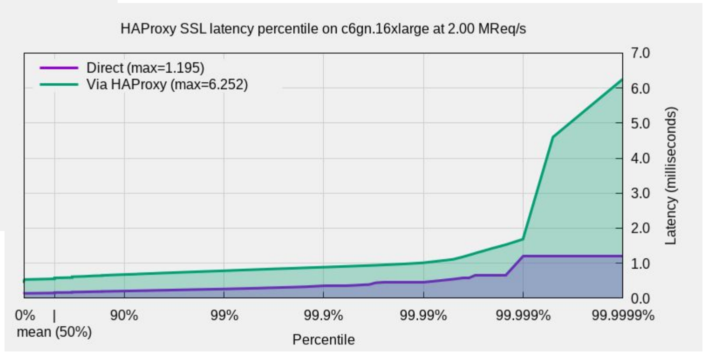




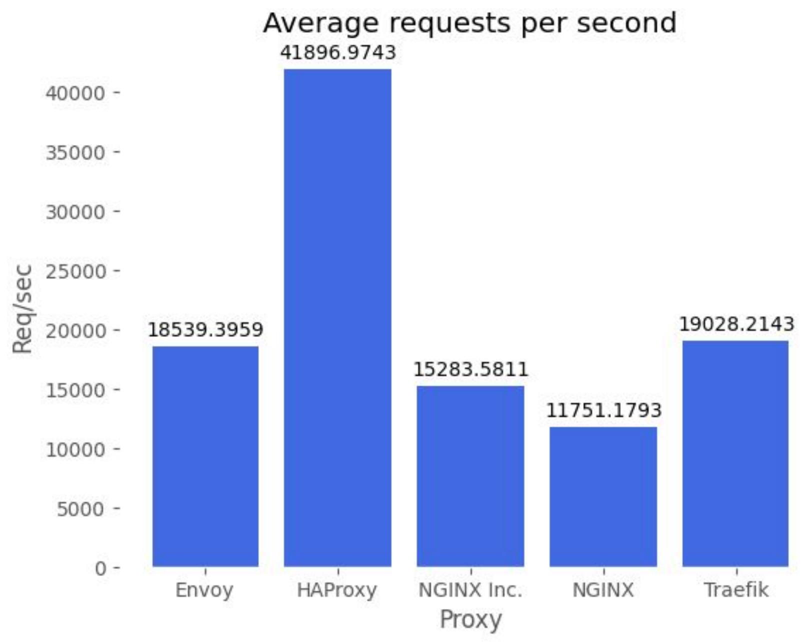
Performanse (2)

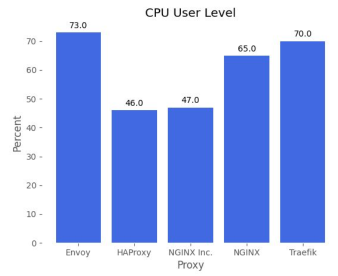
- TLS - do 5ms dodatne latencije per req
- TLS for free!

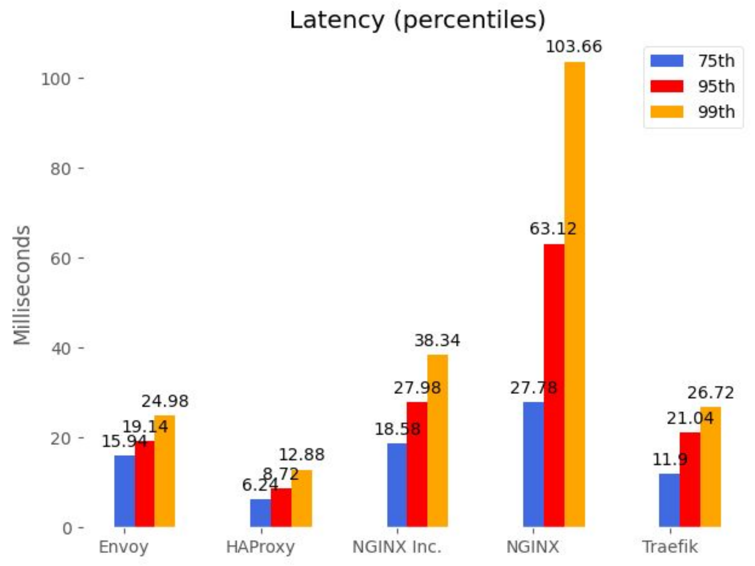




Benchmarking ICs







ACLs - Access Control Lists

ACL - Uvod

- testiranje uvjeta i izvršavanje **akcija** nad HTTP upitom ili odgovorom
- traženje **nizova** znakova, uzoraka, IP adresa, provjera request ratea, TLS statusa, itd.
- **akcije** - odluka o usmjerivanju, redirekciji upita, vraćanje odgovora, itd.
- **imenovani** (named) i **anonimni** (anonymous, in-line) ACL

```
acl is_static path -i -m beg /static  # named ACL
use_backend be_static if is_static  # uses named ACL
use_backend be_static if { path -i -m beg /static }  # in-line ACL

http-request deny if { path_beg /api } !{ src 10.0.0.0/16 }  # NOT
http-request tarpit if { path_beg /rest } { src -f
/etc/hapee-1.9/blacklist.acl }  # reads from file into EBtree
```

ACLs - Uvod (2)

```
acl starts_evil path -i -m beg /evil
acl ends_evil path -i -m end /evil
http-request deny if starts_evil || ends_evil  # OR

acl evil path_beg /evil
acl evil path_end /evil
http-request deny if evil  # AND
```

- ACL se sastoji od **izvora** informacija (fetch) i **niza znakova** s kojim se radi usporedba
- moguće dodavati različite **zastavice** (npr. -i) i **metodu** (npr. beg)

ACL fetches

- `src` - izvorišna **IP adresa** klijenta
- `path` - request **path**
- `url_param(foo)` - vrijednost URL **parametra** foo
- `req.hdr(foo)` - vrijednost HTTP request **zaglavlja** (npr. Host)
- `ssl_fc` - da li je upit dan preko SSL konekcije
- ...

ACL converters

```
path,regsub(^/static,/)
```

- `lower` - lowercase uzorka
- `upper` - uppercase uzorka
- `base64` - base64 encode uzorka ili niza znakova
- `field` - selekcija polja kao što radi awk: field(|,3)
- `bytes` - dohvat bajtova iz binarnog uzorka
- `map` - dohvat uzorka iz mape

ACL flags

- moguće višestruke zastavice u svakom ACL-u
  `path -i -m beg -f /etc/hapee/paths_secret.acl`

- `-i` - case insensitive
- `-f` - uzorak je datoteka na disku (učitava se u EBtree)
- `-m` - odabire se tip usporedbe (u idućem poglavlju)

ACL matching methods

- `str` - uzorci moraju biti identični niz znakova
- `beg` - traži se uzorak od početka niza
- `end` - traži se uzorak od kraja niza
- `sub` - traži se uzorak unutar niza
- `reg` - uzorak se koristi kao regularni izraz (obično se izbjegava zbog performansi)
- `found` - istinit ako je uzorak pronađen (npr. željeni tip zaglavlja u upitu ili uzorak u mapi)
- `len` - vraća dužinu uzorka

Redirekcije

```
http-request redirect location
http://www.%[hdr(host)]%[capture.req.uri] unless { hdr_beg(host) -i
www }

http-request redirect prefix /foo if  !{ path_beg /foo }

http-request redirect scheme https if !{ ssl_fc } # 302
http-request redirect scheme code 301 https if !{ ssl_fc } # 301
```

Odabir backenda

- HTTP način rada (`mode http`):

```
use_backend be_stats if { path_beg /stats }
use_backend be_%[path,map_beg(/etc/hapee-1.9/paths.map,mydefault)]
default_backend be_foobaz
```

- TCP način rada (`mode tcp`):

```
tcp-request inspect-delay 10s
use_backend be_ssl if { req.ssl_hello_type gt 0 }
```

Izmjene HTTP zaglavlja

- moguće i u upitu (`http-request`) i u odgovoru (`http-response`)

- `add-header` - dodaje zaglavlje, dozvoljeni duplikati
- `set-header` - dodaje zaglavlje ili mijenja postojeće
- `replace-header` - regex zamjena postojećeg zaglavlja (npr. dodavanje cookieja)
- `del-header` - brisanje zaglavlja (npr. X-Forwarded-For)

Izmjene URL patha

- transparentno prepisivanje patha, primjerice po backendu

```
http-request set-path /foo%[path] if !{ path_beg /foo }
http-request set-path %[path,regsub(^/netdata/,/)]
```

Blokiranje upita

- `http-request deny`- vraća **403** i prestaje sa procesiranjem (ne dolazi do backenda)
- `http-request tarpit`- drži **otvorenu konekciju** do timeout tarpit i vraća 500 (dodatna obrana…)
- `http-request silent-drop`- **ne vraća RST** klijentu (dodatna obrana, nije praktično ako je ispred FW zbog conntrackinga)

```
http-request deny if HTTP_1.0
http-request deny if { req.hdr(user-agent) -m sub evil }
http-request deny if { req.hdr(user-agent) -m len le 32 }
http-request tarpit if { path -m sub /. }
```

```
dynamic update  # EE feature
  update id /etc/hapee-1.9/whitelist.acl url
http://192.168.122.1/whitelist.acl delay 60s
```

Stick Tables

Uvod

- pratiti ponašanja korisnika tijekom obrade dobivenih upita
- **skladištenje događaja** (upiti, odgovori) i **kategorizacija** po ključu (IP adresa, URL, itd.)
- StackExchange - prvi korisnik, kontrola “loših” klijenata, zaštita od botova, tracking
- najvažnija upotreba:
    - **server persistence** / sticky sessions: cookie-bazirano ili consistent hashing nije uvijek dobro, moguće je koristiti IP adresu, više cookieja ili byteove u bodyju (username ili kakav ID iz ne-HTTP protokola) i povezati sa pojedinim backend serverom
    - **detekcija botova i napada**: request floods, brute-force attacks, vulnerability scanners, Web scrapers, slowloris attacks
    - **metrike**
- u praksi je to K/V store sa automatskim ažuriranjem vrijednosti
- sadržaj svih stick tablesa je moguće ispisati iz CLI-ja

```
echo "show table per_ip_and_url_rates" | socat stdio
/var/run/hapee-1.9/hapee-lb.sock
```

Definiranje i korištenje

- odrediti kapacitet (`size`), trajanje podataka (`expire`), tip podataka (`type`) i deklarirati vrijednosti (`store`)

```
backend st_src_global
    stick-table type ip size 1m expire 10s store http_req_rate(10s)
frontend fe_main
    bind *:80
    http-request track-sc0 src table st_src_global
```

- svaki unos koristi 50 bajtova interno + dužina svakog ključa (npr. 50 bajtova za IP) + brojači koji se koriste
- napomena: reload servisa stvara novi proces sa novom mapom, istovremeno će biti aktivno više procesa
- brojač sc 0 - 12

Tipovi ključa

- `ip` - 50 bajtova i sprema IPv4 adresu (npr. fetch src ili req.hdr(x-forwarded-for))
- `ipv6` - 60 bajtova, sprema IPv6 ili IPv6 mapiranu IPv4 adresu
- `integer` - 32 bajta, npr. Client ID iz cookieja, zaglavlja, frontend ID itd.
- `string` - zauzima len bajta, primjerice session ID-jevi, API keys itd.
- `binary` - zauzima len bajtova, za binarne uzorke iz TCP streamova, ili npr. base32 (IP+URL fetch)

Tipovi vrijednosti

- `http_req_rate` - broj HTTP **upita** koji je ključ napravio kroz definirano vrijeme

```
stick-table type ip size 1m expire 10s store http_req_rate(10s)
tcp-request inspect-delay 10s # TCP phase, hold until HTTP info
tcp-request content track-sc0 src
#http-request content track-sc0 src
http-request deny if { sc_http_req_rate(0) gt 100 }
```

- `conn_cur` i `conn_rate` - koliko ključ ima aktivnih **konekcija** ili koliko ih brzo stvara (prosjek kroz vrijeme)

```
stick-table type ip size 1m expire 10s store conn_cur
tcp-request content track-sc0 src
tcp-request content reject if { sc_conn_cur(0) gt 10}
```

Tipovi vrijednosti (2)

- `http_err_rate` - HTTP upiti koji završavaju sa **4xx** odgovorima, npr. vuln. scanneri

```
stick-table type string len 128 size 2k expire 1d store
http_err_rate(1d)
tcp-request content track-sc0 path
```

- `bytes_out_rate` - egress **promet** po ključu, npr. po pathu (capacity planning, anomaly/exfiltration detection...)

- `bytes_in_rate`…

IP - TLS version tracking

```
backend st_ssl_stats
    stick-table type ip size 200 expire 1h store http_req_rate(1d)

frontend fe_main
    tcp-request inspect-delay 10s
    tcp-request content track-sc0 src table st_ssl_stats if {
ssl_fc_protocol TLSv1.1 }
```

Sticky sessions

- `stick on` - podatak se koristi kao ključ za određivanje **backend servera**

```
backend mysql
    mode tcp
    stick-table type integer size 1 expire 1d
    stick on int(1)
    on-marked-down shutdown-sessions
    server primary 192.168.122.60:3306 check
    server backup 192.168.122.61:3306 check backup
```

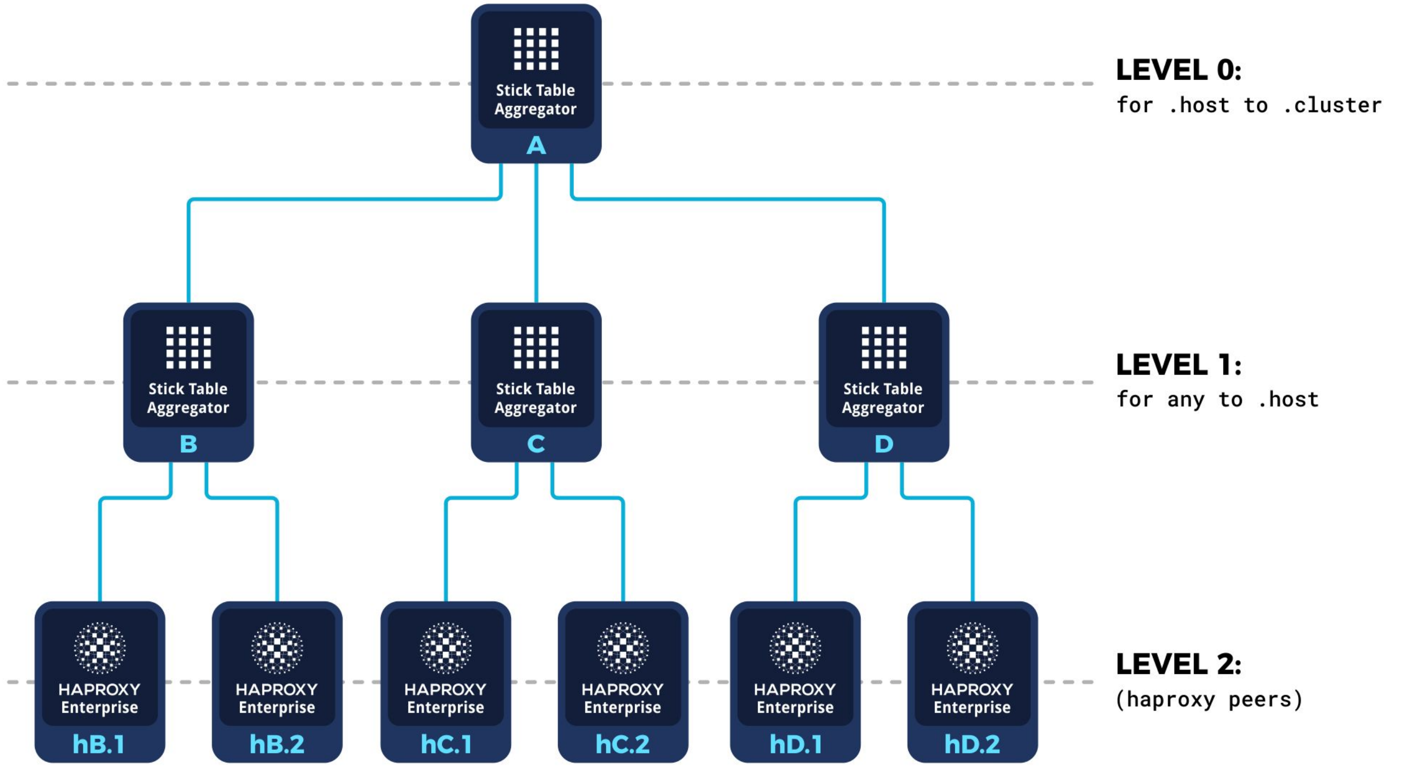
Peers - sinkronizacija

- više poslužitelja razmjenjuje stick tables vrijednosti
- nema agregiranja, samo se **prepisuju** vrijednosti!

```
peers mypeers
    peer centos7vert 192.168.122.64:10000
    peer shorepoint 192.168.122.1:10000
stick-table type string len 32 size 100k expire 30m peers mypeers
```

- EE: Stick Table Aggregator (Stktagg)
  - zbraja vrijednosti i vraća rezultate u **odvojene** stick tablese
  - podržava složenije arhitekture i HA (clusteri, grupe clustera) u Active-Active
  - cluster-wide tracking, različite topologije

Stick Table Aggregator



Maps

Upotreba

- izvor za mapu je tekstualna datoteka koja se učita u EBtree
  ```
  static.example.com be_static
  www.example.com    be_static
  ```

- `map` converter - staza do datoteke i opcionalna default vrijednost
- hitless-reload: novu instanca sa **novom mapom** bez gubitaka konekcija
- EE: lb-update može dinamički osvježavati mape

```
frontend fe_main
    bind :80
use_backend
    %[req.hdr(host),lower,map(/etc/hapee-1.9/maps/hosts.map,
be_static)]
```

Konverteri

- `map` - niz znakova mora odgovarati cijelom ključu
- `map_beg` - vrijednosti koje odgovaraju unosu tražeći od početka
- `map_end` - traži se od kraja vrijednosti (iterativno pretraživanje)
- `map_sub` - traži se podniz (iterativno)
- `map_ip` - IP ili CIDR pretraživanje
- `map_reg` - pretražuje regularni izraz
- `map_str` - alias za map

Blue-green deployment

- bluegreen.map:
  ```
  active be_blue
  ```

```
frontend fe_main
    bind :80
    use_backend %[str(active),map(/etc/hapee-1.9/maps/bluegreen.map)]

backend be_blue
    server server1 10.0.0.3:80 check
    server server2 10.0.0.4:80 check

backend be_green
    server server1 10.0.0.5:80 check
    server server2 10.0.0.6:80 check
```

Rate limiting by URL path

- limiti u datoteci:
  ```
  /api/routeA 40
  /api/routeB 20
  ```

```
frontend api_gateway
    bind :80
    default_backend api_servers

stick-table type binary len 8 size 1m expire 10s store
http_req_rate(10s)

# track client by base32+src
# = 32bit hash(Host hdr + URL path) + 8-20 bytes srcIP
http-request track-sc0 base32+src
```

Rate limiting by URL path (2)

```
# get rate limit for path from map file
http-request set-var(req.rate_limit)
path,map_beg(/etc/hapee-1.9/maps/rates.map)

# get client request rate
http-request set-var(req.request_rate)
base32+src,table_http_req_rate(api_gateway)

# is request_rate > rate_limit true?
acl rate_abuse var(req.rate_limit),sub(req.request_rate) lt 0

# deny if true
http-request deny deny_status 429 if rate_abuse
```

DDoS zaštita

Uvod

- napade je moguće uspješno presresti na LB-u bez opterećivanja backenda
  - vrlo razumni HW zahtjevi
  - podrška za moderne Linux kernele (kernel TCP splice, TFO, …)
  - moguće raditi sa vrlo velikim brojem konekcija istovremeno bez potrebe da se propuštaju na backend
- building blocks:
  - high-perf ACL-ovi (non-regex, EBtree)
  - high-perf maps (EBtree)
  - high-perf SSL/TLS stack + threading
  - cluster-wide real-time tracking
  - detailed logging
  - EE: PacketShield
  - EE: fingerprinting, JS challenge, reCaptcha, WAF itd.

HTTP flood

- upiti na jedan ili više URL-ova (npr. search engine) sa što većom frekvencijom
- LOIC, HOIC attacks, itd.
- ograničavamo broj upita u sekundi po IP-u

```
backend per_ip_rates
    stick-table type ip size 1m expire 10m store http_req_rate(10s)

frontend fe_mywebsite
    bind *:80
    http-request track-sc0 src table per_ip_rates
    http-request deny deny_status 429 if { sc_http_req_rate(0) gt 100
}  # 429
    #timeout tarpit 5s
    #http-request tarpit if { sc_http_req_rate(0) gt 100 }  # 500
    ##http-request silent-drop if ...
```

Slowloris Attacks

- potrebno postaviti dovoljno visok globalni `maxconn`
- pojedini frontend treba imati manji maxconn nego globalni, tako da jedan frontend ne dominira
- bufferira se HTTP body do http-request timeouta

```
timeout http-request 5s
option http-buffer-request
```

Blacklisting, greylisting

- npr. po IP rasponima ili GeoIP bazi (MaxMind ili DigitalElement su EE)

```
http-request deny if { src -f /etc/hapee-1.9/blacklist.acl }
http-request deny if { src -f /etc/hapee-1.9/greylist.acl } {
sc_http_req_rate(0) gt 5 }
```

Lower priority backend

- moguće imati više backenda za isti server sa različitim postavkama za maxconn
- ovisno o brzini upita moguće je odbijati ili usmjeravati na backend sa restriktivnijim postavkama

```
backend be_normal
  server server1:80 maxconn 3000

backend be_website_bots
  Server server1:80 maxconn 100

frontend fe_mysite
  stick-table ...
  bind *:80
  http-request track-sc0 src
  http-request deny if { sc_http_req_rate(0) gt 200 }
  use_backend be_website_bots if { sc_http_req_rate(0) gt 100 }
  default_backend be_normal
```

Antibot

- EE feature
- detekcija da li je klijent podržava Javascript
- **JS challenge**: klijent rješava jedinstveni i dinamički generiran (ovisno o detektiranom pregledniku i konfiguraciji) matematički problem
- može nastaviti tek kad vrati ispravan rezultat kroz cookie
- opcionalni pop-up (Yes/No)

reCAPTCHA

- EE feature
- podržana v2 i v3 reCAPTCHA
- Lua biblioteka
- za Web scrapere koji izvršavaju JS
- dobra kombinacija za smanjenje false positiva u slučaju detekcije moguće zlonamjerne aktivnosti (Web scraping itd.)

```
http-request use-service lua.request_recaptcha unless {
lua.verify_solved_captcha "ok" } { sc_get_gpt0(0) eq 1 }
```

Zaštita od botova

Botovi

- korisni: crawling/indexing, agregiranje informacija, alerting na promjene...
- zlonamjerni: Web scraping, spamming, request flooding, brute-forcing, vulnerability scanning…
- potrebno je prepoznati zlonamjerno ponašanje i neutralizirati bez utjecaja na korisne botove
- nerijetko su i korisni botovi neispravno konfigurirani i/ili utiču na normalni rad Web aplikacija

- EE servis Verify Crawler (koristi SPOE) moguće je u pozadini i bez blokiranja HAProxy servisa raditi višestruke provjere i upravljati valid_crawler i invalid_crawler stick tablesima:
  - UA check
  - source IP forward and reverse DNS check

Web scraping

```
backend per_ip_and_url_rates
    stick-table type binary len 8 size 1m expire 24h store
http_req_rate(24h)
backend per_ip_rates
    stick-table type ip size 1m expire 24h store gpc0,gpc0_rate(30s)

frontend fe_main
    bind :80
    http-request track-sc0 src table per_ip_rates  # IP track
    http-request track-sc1 url32+src table per_ip_and_url_rates unless
{ path_end .css .js .png .jpeg .gif }  # IP+URL track
    acl exceeds_limit sc_gpc0_rate(0) gt 15  # thresh: 15 uniq / 30s
    # increase GPC in per_ip_rates if it is first visit for IP+URL and
below thresh
    http-request sc-inc-gpc0(0) if { sc_http_req_rate(1) eq 1 }
!exceeds_limit
    http-request deny if exceeds_limit  # 403, other option is 429
    default_backend web_servers
```

Web scraping (2)

- kad se detektira aktivnost koja identificira bota:
  - usmjeravanje na backend sa manjim prioritetom i max brojem konekcija
  - usmjeravanje na backend sa HAProxy **cacheom**
  - **tagiranje** (General Purpose Tag gpt0) u per_ip_rates sa expireom od 24h tako da smanji CPU overhead i odmah odbacuje upite (early jail)

```
backend per_ip_rates
    stick-table type ip size 1m expire 24h store
gpc0,gpc0_rate(30s),gpt0

http-request sc-set-gpt0(0) 1 if exceeds_limit
http-request deny if { sc_get_gpt0(0) eq 1 }
```

- moguće je smanjiti false-positivese koristeći reCAPTCHA modul umjesto deny

```
http-request use-service lua.request_recaptcha unless {
lua.verify_solved_captcha "ok" } { sc_get_gpt0(0) eq 1 }
```

Brute-force

- npr. repetitivni POST na isti URL (login)

```
backend per_ip_and_url_bruteforce
    stick-table type binary len 8 size 1m expire 10m store
http_req_rate(3m)

http-request track-sc2 base32+src table per_ip_and_url_bruteforce if
METH_POST { path /login }  # track POST to /login per IP
http-request deny if { sc_http_req_rate(2) gt 10 }
```

Vulnerability scanners

- WAF je nužan, ali možemo reagirati i ranije na jednostavne indikatore
- visok 404 rate najčešće nije uobičajen

```
backend per_ip_rates
    stick-table type ip size 1m expire 24h
gpc0,gpc0_rate(30s),http_err_rate(5m)

http-request deny if { sc_http_err_rate(0) gt 10 }

http-request sc-inc-gpc0(0) if { sc_http_err_rate(0) eq 1 }
!exceeds_limit
```

- honeypot backend idealan, moguće je transparentno prosljeđivati

```
use_backend be_honeypot if { sc_http_err_rate(0) gt 10 }
```

Geolocation

- EE: MaxMind i Digital Element

```
http-request set-header x-geoip-country
%[src,maxmind-lookup(COUNTRY,country,iso_code)]  # MaxMind

http-request set-header x-geoip-country %[src,netacuity-lookup-ipv4
("pulse-two-letter-country")]  # Digital Element

use_backend be_honeypot if { sc_http_err_rate(0) gt 5 } {
req.hdr(x-geoip-country) CN }
```

Fingerprinting

- EE feature
- **identifikacija** klijenata i botova i unatoč lažnom User-Agent zaglavlju
- svaki klijent dobiva **jedinstvenu oznaku**: `y-eafbg-ba-m-m-w-6782e119-c40cb9ee-00000000-b91db5cc-61e4f601-y-x-1.0-u-y-y-n`
- 20ak karakterističnih detalja iz upita
  - CRLF header termination
  - Content zaglavlje: sadržaj i redoslijed
  - Connection zaglavlje: sadržaj i redoslijed
  - način pisanja velikog i malog slova u zaglavljima
  - Hashevi Accept-encoding, User-Agent, Accept-language itd. Zaglavlja
  - razmaci i ostali prazan prostor u zaglavljima
  - HTTP verzije
  - način pisanja metode
  - način zapisivanja cookiesa…
  - itd.

WAF

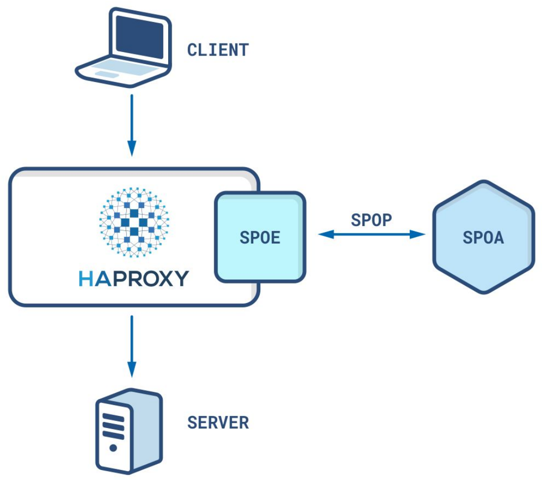
Kratki pregled

- tri različita native (high-perf) modula ovisno o potrebama
- različite performanse, način rada, HW zahtjevi te količina potrebne konfiguracije
  - Simple SQLi/XSS
    - jednostavan deployment, praktički nikakvo održavanje
    - isključivo za SQL injection i Cross-Site Scripting
    - ima ugrađeni set fingerprintova (moguće dinamičko nadograđivanje)
    - najbrzi od svih
  - ModSecurity
    - najsporiji, najkompleksniji
    - custom fork (ModSecurity v3) sa sigurnosnim izmjenama i poboljšanim performansama (RE2 engine, PCRE fallback, report blocking I/O, hard transaction timeouts, JSON/XML processing limits, memory pools...) - u stanju raditi i tijekom DDoSa
    - 100% kompatibilan sa OWASP ModSecurity Core Rule Setom
    - detektira SQLi, XSS, RCE..
    - procesira i upite i odgovore te cijeli body
  - Zero-Trust Mode / Advanced WAF: whitelist/blacklist
    - vrlo restriktivni blacklist sa known-good application-specific whitelist
    - learning mode i logiranje violationa
    - pravila naliče na Naxsi ruleset
    - vrlo zahtjevni deployment, nakon toga uglavnom low-medium maintenance
    - detektira SQLi, XSS, Remote File Inclusion, Directory Traversal, Evasion Tricks…
    - whitelist po pravilu, request pathu, varijablama te kombinacijama istih

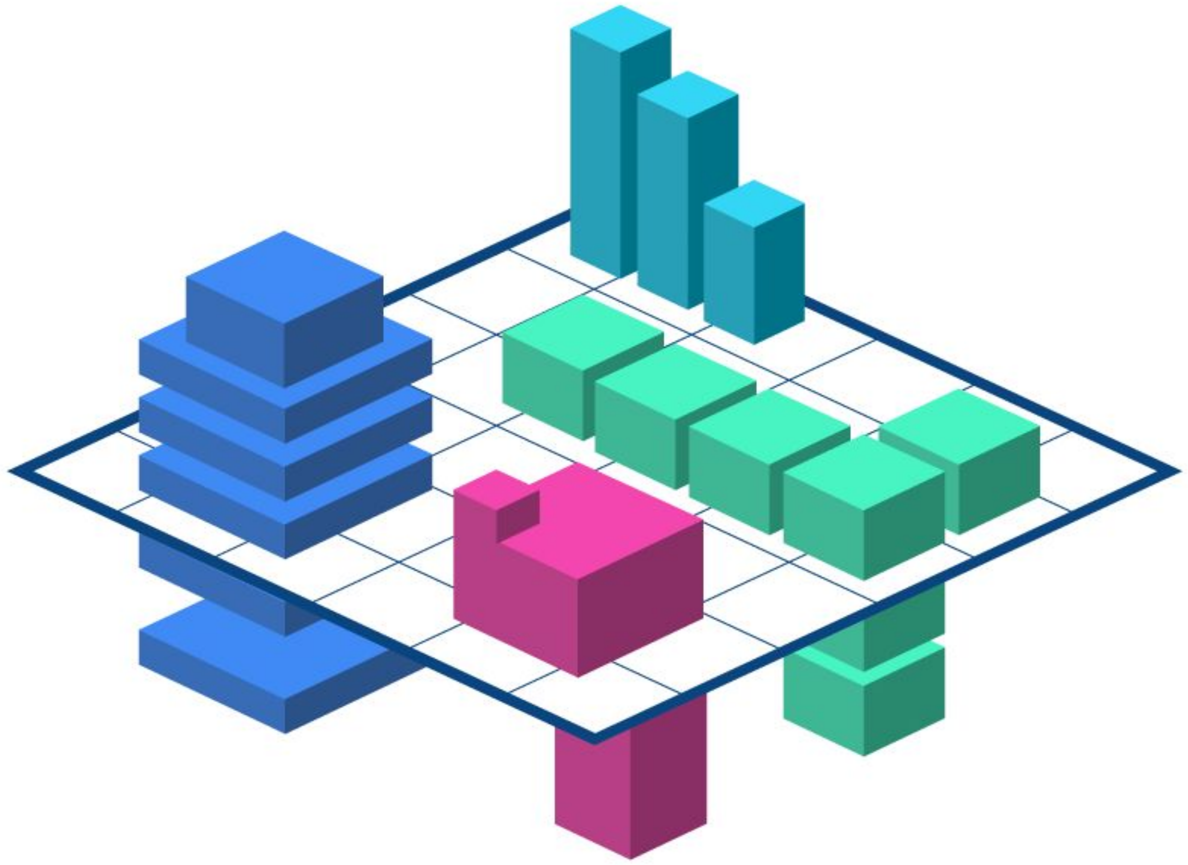
SPOE

SPOE

- Stream Processing Offload Engine (**SPOE**)
- šalje se promet **eksternim programima** (Agent odn. **SPOA**) koji procesiraju podatke out-of-band
- minimalna dodatna latencija i dobre performanse
- SPOA:
  - nema kompleksnost native HAProxy modula
  - podržani jezici: C, .NET Core, Golang, Lua, Python
  - primjeri: IP reputation reporting agent, ModSecurity (OSS, v2 library), DataDome...



QUESTIONS & ANSWERS



Where To Find Us

HAProxy Technologies
www.haproxy.com

HAProxy Technologies Support
support@haproxy.com
(844) 222-4340 (option 3)

HAProxy User Spotlight Series
www.haproxy.com/user-spotlight-series/

Twitter
@HAProxy

Slack
slack.haproxy.org