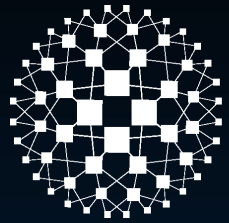


HAPROXY

Turning Exposed URIs Bulletproof

Securing APIs with HAProxy

Sébastien Gross
Solution Architect - HAProxy Technologies

booth #A09

apidays
Paris 2025

Agenda

1. What is HAProxy?
2. Broken designs context.
3. Fix broken design step by step.
4. Questions?

HAProxy Load Balancer

- Highly available load balancing for TCP, HTTP, QUIC
- Request routing
- TLS termination
- Rate limiting

High Performance Modules

- Next Gen WAF
- Bot Management
- Response policies
- Global Rate Limiting
- UDP Load Balancing
- SSO Offload
- Geolocation & Device Detection
- Dynamic Updates

Context

Digital Vault API Innovations

- Provides web site and API to users.
- Missioned you:

Your mission if you choose to accept it is to secure our web platform and API services. Should you fail, the SOC will disavow any knowledge of your action.

Damn Vulnerable API

- Developed by Payatu.

  https://github.com/payatu/dvapi/
- Uses top 10 OWASP vulnerabilities.
- CTF like game.
- Provides walkthrough.

This talk spoils the game solution.

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Disclaimer

- One solution is provided.
- May not fit all cases.
- Always check the documentation.
- Test your environment.
- Know your application.

https://github.com/sg-workshops/apidays-paris-2025

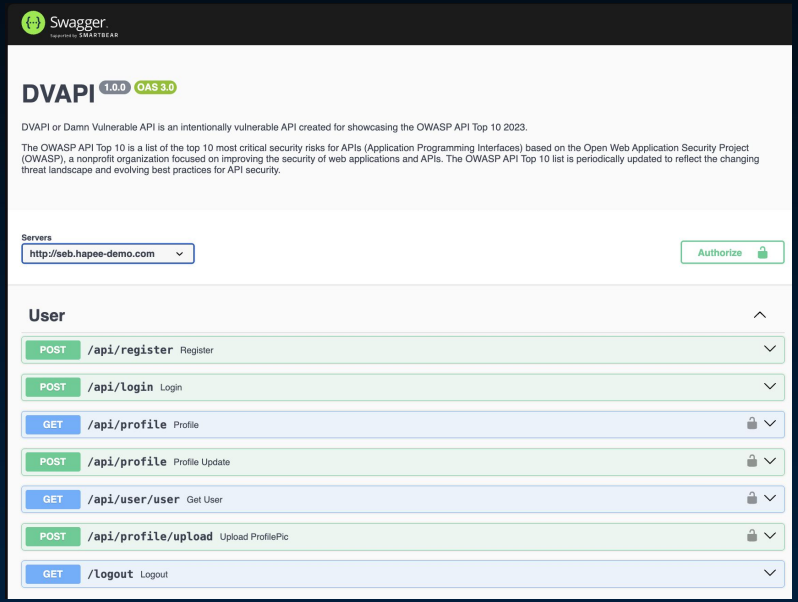
Minimal setup

API Endpoints

- Application has a Swagger page:

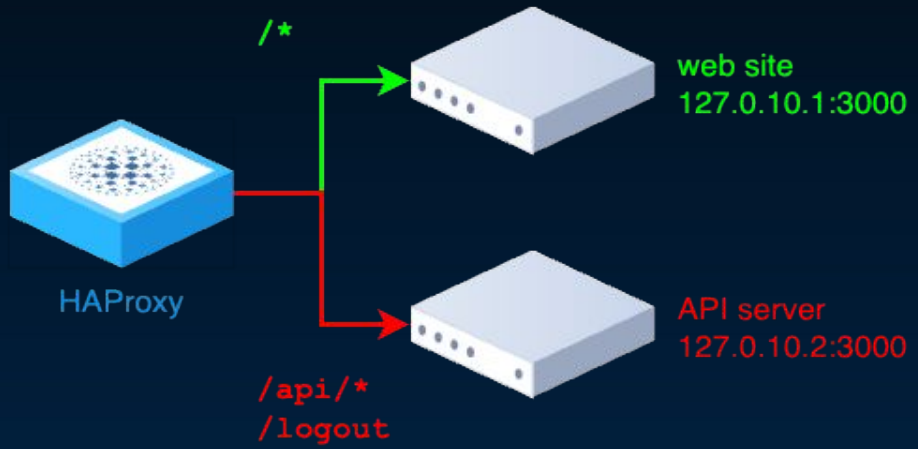
  http://dvapi/sagger

- Only accept request to a known endpoint.
  - Starts with `/api`
  - Is `/logout`
- Limitation should not impact website.



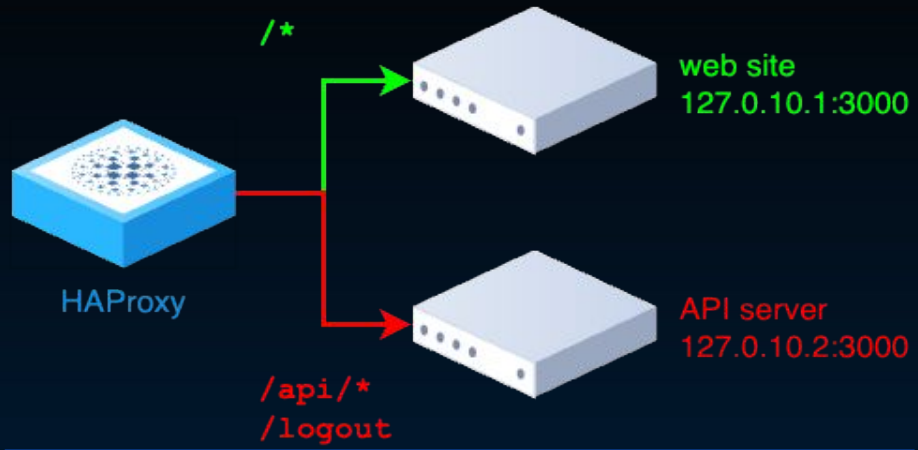
Initial setup

- 2 servers:
  - web site: `127.0.10.1:3000`
  - API server: `127.0.10.1:3000`
- HAProxy routes the traffic:
  - `/api/*` and `/login` → API server
  - `/*` → web site
- All nodes are on same host for simplicity.



HAProxy configuration

- Frontend: client facing side.
- Backend: server facing side.
- **use_backend**: select a specific backend
  - Can have a condition : API path is
    - starting with **/api/**: **-m beg /api/**
    - exactly **/logout**: **-m str**
  - Can be declared more than once.
  - Are tested sequentially
- **default_backend**:
  - Optional
  - Selected if all **use_backend** rules failed.
  - Can only be declared once.
- If no backend can be found: error **503**.




```
frontend dvapi-fe
  bind :80

  # Route API traffic
  use_backend api-be if { path -m str /login }
  use_backend api-be if { path -m beg /api/ }

  # Default traffic: web interface
  default_backend web-be

backend web-be
  server web 127.0.10.1:3000

backend api-be
  server api 127.0.10.2:3000
```

Traffic routing

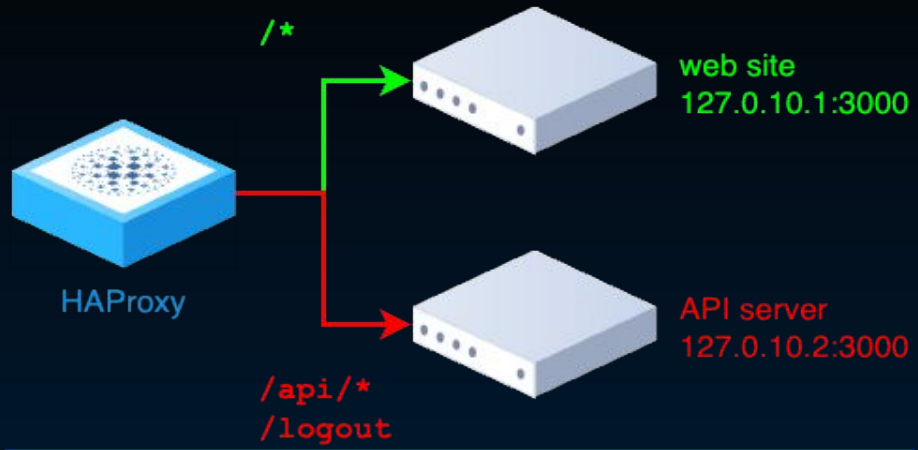
- ACL: `is_api` checks first component.
  - convertor: `word` splits `path` on `/`.
  - and takes 1st item:

```
/api/login
/api/user/john
/logout
```

- ACL are executed when called.

Caveat:

- `path -m beg /api/ /logout` matches:
  - `/logoutgarbage`
  - `/api/something`




```
frontend dvapi-fe
  bind :80

  # is_api is true if 1st component is either api or logout
  acl is_api path,word(1,/) -m str api logout
  use_backend api-be if is_api

  # Default traffic: web interface
  default_backend web-be

backend web-be
  server web 127.0.10.1:3000

backend api-be
  server api 127.0.10.2:3000
```

Server monitoring

- Activated with **`check`** directive.
  - Simple TCP check.
  - Opens connection to server.
  - Closes the connection.
- Add **`option httpchk`** to Layer 7 checks.
- Use **`http-check`** rules for advanced checks:
  - Specify method.
  - Target an URI.
  - Select HTTP version.
  - Add headers.
- See **`http-check`** rules in document for even more advanced setup.

```
backend web-be
  option httpchk
  http-check send meth HEAD uri /login ver HTTP/1.1 hdr host
web
  server web 127.0.10.1:3000 check


backend api-be
  option httpchk
  http-check send meth HEAD uri /logout ver HTTP/1.1 hdr
host api
  server api 127.0.10.2:3000 check
```

Preserve client IP address

- Send client IP through HTTP header:
  - `x-forwarded-*`: convention
  - `forwarded`: RFC 7239 (only in backends).
- Can be a security risk.
  - Don't trust external value
  - Remove existing header if unsure.
  - Or save them in alternate headers.
- Check web server documentation.

```
setup:
frontend dvapi-fe
  [...]
  http-request set-header x-forwarded-for %[src]
  http-request set-header x-forwarded-port %[src_port]
  http-request set-header x-forwarded-proto
%[ssl_fc,iif(https,http)]

Headers:
x-forwarded-for: 192.0.2.3
x-forwarded-port: 6312
x-forwarded-proto: http
```

```
setup:
backend api-be
  [...]
  option forwarded proto by by_port for for_port

Headers:
forwarded: proto=http;by="172.31.43.4:80";for="192.0.2.3:6312"
```

SSL EZ AS ABC

- HAProxy is an SSL offloader.
- Only 2 options required:
  - `ssl`: bind is over SSL.
  - `crt`: path to the certificate bundle.
  - `strict-sni`: optional only accept known SNI.
- `redirect` from http to HTTPS is connection is not over SSL.
  - Use code `307`, not `302`.
  - `307` force to keep the same method.
- `http-request` rules are executed in order of definition.
- Optional: activate HSTS if connection is over SSL.

```
frontend dvapi-fe
  bind :80
  bind :443 ssl crt seb.hapee-demo.com.pem strict-sni

  # Redirect http->https
  http-request redirect scheme https code 307 if !{ ssl_fc }

  [...]

  use_backend api-be if is_api
  default_backend web-be

  [...]

  # Activate HSTS if connection is over SSL
  http-after-response set-header strict-transport-security
"max-age=31536000" if { ssl_fc }
```

Status page and metrics

- Go to `192.0.2.10:8000/haproxy/`
- Login with `admin` / `haproxy`
- Plug Prometheus collector to `192.0.2.10:8000/metrics`
- Best practice:
  - Use dedicated management network
  - Add source IP address filtering.

```
userlist stats-auth
  group admin users admin,john
  user admin insecure-password haproxy
  user john insecure-password doe
  user operator insecure-password secret

frontend stats_fe
  bind 192.0.2.10:8000
  # only allow management network
  # tcp-request connection reject if { src 10.1.1.0/24 }
  use_backend stats_be if { path,word(1,/) -m str haproxy
metrics }


backend stats_be
  stats enable
  stats uri /haproxy/
  acl AUTH http_auth(stats-auth)
  acl AUTH_ADMIN http_auth_group(stats-auth) admin
  stats http-request auth realm "HAProxy Statistics" unless
AUTH
  stats admin if AUTH_ADMIN
  stats show-desc "short description"
  stats show-legends
  stats show-modules
  stats show-node

  http-request use-service prometheus-exporter if { path
/metrics }
```

HAProxy status page

dvapi-fe

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								Extra modules				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	ssl	h3	quic	h2	h1
Frontend				0	0	-	0	0	524 273	0			0	0	0	0	0					OPEN									ssl	h3	quic	h2	h1

web-be

		Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									Extra modules				
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	ssl	h3	quic	h2	h1
☐	web	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	6s UP	L7OK/200 in 2ms	1/1	Y	-	0	0	0s	-	ssl				
	Backend	0	0		0	0		0	0	52 428	0	0	?	0	0	0	0		0	0	0	0	6s UP		1/1	1	0		0	0s		ssl			h2	h1

Choose the action to perform on the checked servers : Apply

api-be

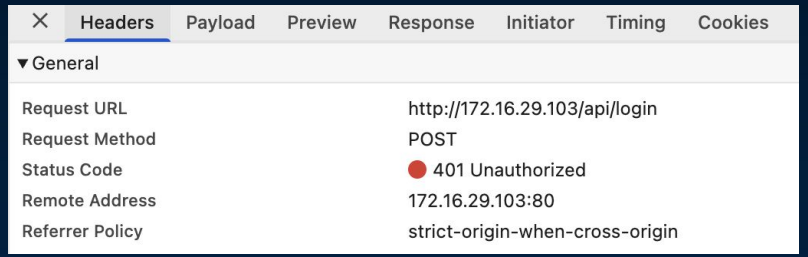
		Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									Extra modules				
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	ssl	h3	quic	h2	h1
☐	web	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	6s UP	L7OK/302 in 1ms	1/1	Y	-	0	0	0s	-	ssl				
	Backend	0	0		0	0		0	0	52 428	0	0	?	0	0	0	0		0	0	0	0	6s UP		1/1	1	0		0	0s		ssl			h2	h1

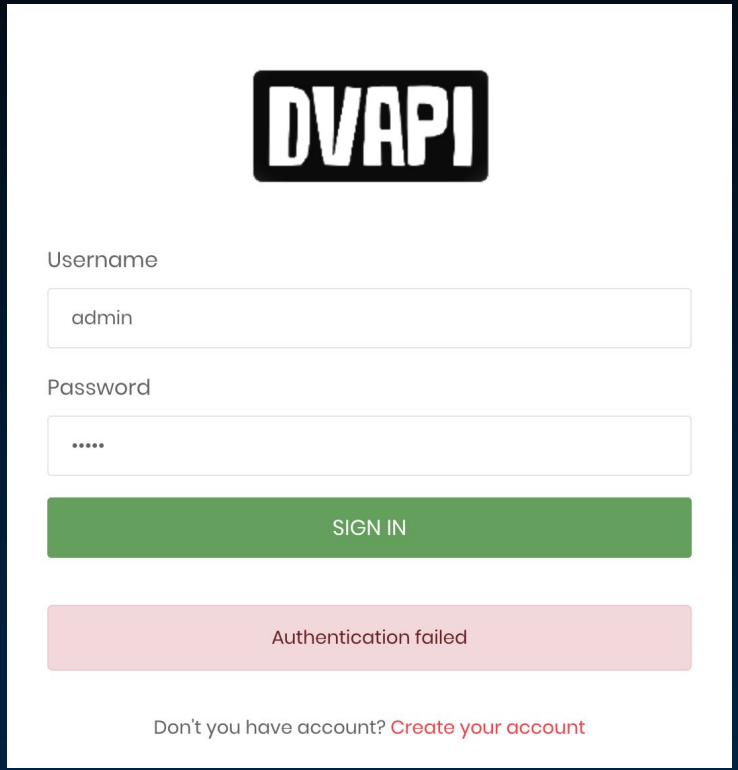
Choose the action to perform on the checked servers : Apply

Login protection

Login

- POSTs login data to `/api/login`.
- Returns 401 in case of failure.
- No limitation in case of failure.





Rate limit - stick tables

- Key / values storage.
- Key can be almost anything.
- Values can contain:
  - Connections count or rate.
  - Request count or rate.
  - HTTP error (4xx) or failure (5xx).
  - Transferred bytes.
  - General Purpose Counters.
  - etc...
- Entries can have an expiration time.
- Data can be automatically or manually updated.
- Small memory footprint.
- Data be aggregated using Global Profiling Engine (Enterprise module).

Rate limit - counts vs. rates

Counts:

- Incremental only values.
- Provides absolute value since entry creation.
- Useful for usage counter.

Value.

Rates:

- Computed over a sliding window.
- Dynamic information.
- Fine-tune rate limiting.
- Can detect spikes.

Value per unit of time.

Limit brute force

- Limitation must be by source IP.
- Detect HTTP response code
  - 401 - Unauthorized.
  - Store abuse count into `gpc0`.
- Block user after 3 unsuccessful attempts.
  - Arbitrary value
  - Depends on your policy.
- User will be able to connect again after a 5-minute pause.
- Tables are referred as `peer`/`table`.

```
peers mypeers
  table per_src_brute_force type ip size 1m expire 5m store
gpc0


frontend main_fe
  [...]
  acl is_http_abuser sc0_get_gpc0 -m int ge 3

  # track client as soon as possible
  tcp-request connection track-sc0 src table
mypeers/per_src_brute_force

  # Prevent abuser's future connections
  tcp-request connection reject if is_http_abuser
  # Reject abuser's current request
  http-request reject if is_http_abuser

  [...]


backend api-be
  [...]
  # increment General Purpose Counter if unauthorized
  http-after-response sc-inc-gpc0(0) if { status -m int 401
}
```

Protect endpoints

Validate endpoints

- Create a transaction variable `txn.path`.
- Variable will allow to normalize `path`.

Special case: `/api/user/{user}`

- path parameter: `user`.
- Normalize `txn.path`:
  - replace username with `{user}`.
  - `/api/user/johndoe` → `/api/user/{user}`.
  - Only if path starts with `/api/user/`.
- `deny` if path is unknown.
- Note: `deny` does not returns a body.
  - Check doc for embellishments.

```
backend api-be
  [...]
  # Save current path to allow normalisation
  http-request set-var(txn.path) path

  # Protect API routes. Deny any unknown route
  acl is_api var(txn.path),word(1,/) -m str api logout

  # /api/user/{user} is special. replace username
  # by {user} to normalize endpoint lookups
  http-request set-var-fmt(txn.path)
"/%[var(txn.path),word(1,/,2)]/{user}" if { path -m beg
/api/user/ }

  [...]

  # Reject unknown endpoints
  http-request deny status 404 if is_api !{ var(txn.path) -m
str -f endpoints.acl }
```

Validate endpoints (Cont'd)

- `Deny` any request if normalized path is not known.
- Known paths are looked up from a file: `endpoints.acl`.

`endpoints.acl`:

```
/api/register
/api/login
/api/profile
/api/user/{user}
/api/profile/upload
/logout
/api/addNoteWithLink
/api/getNote
/api/addNote
/api/flag/submit
/api/allChallenges
/api/getSolves
/api/scores
/api/addTicket
/api/getticket
```

```
backend api-be
  [...]
  # Save current path to allow normalisation
  http-request set-var(txn.path) path

  # Protect API routes. Deny any unknown route
  acl is_api var(txn.path),word(1,/) -m str api logout

  # /api/user/{user} is special. replace username
  # by {user} to normalize endpoint lookups
  http-request set-var-fmt(txn.path)
"/%[var(txn.path),word(1,/,2)]/{user}" if { path -m beg
/api/user/ }

  [...]

  # Reject unknown endpoints
  http-request deny status 404 if is_api !{ var(txn.path) -m
str -f endpoints.acl }
```

Endpoints method

Upgrade from ACL to MAP

- 1st column: lookup key:
  - normalized path
- 2nd column:
  - Value contains allowed methods.

```
# Path                  METHOD
/api/register           POST
/api/login              POST
/api/profile            POST,GET
/api/user/{user}        GET
/api/profile/upload     POST
/logout                 GET
/api/addNoteWithLink    POST
/api/getNote            GET
/api/addNote            POST
/api/flag/submit        POST
/api/allChallenges      POST
/api/getSolves          GET
/api/scores             GET
/api/addTicket          POST
/api/getticket          POST
```

ACL files vs. MAP files

- ACL is a list of pattern.

```
pattern1
pattern2
...
```

- HAProxy checks pattern presence.

```
# check path is in acl file
http-request deny status 404 if
  !{ var(txn.path) -m str -f endpoints.acl }
```

- MAP is a key / values file.

```
key1 value1
key2 value2 with spaces
...
```

- HAProxy gets value matching key.
- key presence can be checked with `-m found`.
- values can be splitted using `word()`.

```
# req.ep_data contains the value
http-request set-var(req.ep_data)
  var(txn.path),map(endpoints.map)

# check path is in map file
http-request deny status 404 if
  !{ var(txn.path),map(endpoints.map) -m found }
```

Check method

- Allowed method is read directly.
- Special case: `/api/profile`.
  - 2 allowed methods: `POST` and `GET`.
  - Map gives a string: `POST,GET`.
  - HAProxy can only compare single values.
  - Array comparison is not supported.
- HAProxy supports Lua script to extend its functionalities.
- Lua can help to compare multiple values:
  - Split method from map file on comma.
  - Compare each item against HTTP method.

```
# Path                  Methods
/api/register           POST
/api/login              POST
/api/profile            POST,GET
/api/user/{user}        GET
/api/profile/upload     POST
/logout                 GET
/api/addNoteWithLink    POST
/api/getNote            GET
/api/addNote            POST
/api/flag/submit        POST
/api/allChallenges      POST
/api/getSolves          GET
/api/scores             GET
/api/addTicket          POST
/api/getticket          POST
```

Lua can...

- Access to any HAProxy fetch:
  - `txn.f:method()`
  - `txn`: current transaction.
  - `f`: Access to fetch.
  - `method()`: the fetch.
- Access to variables:
  - `txn.get_var(txn,"req.ep_method")`
  - `txn`: current transaction.
  - `req.ep_method`: variable name.
- `check-method` action is called:

```
http-request lua.check-method
```

- Deny request if `req.invalid_method` is `true`:

```
http-request deny status 405 if {
var(req.invalid_method),bool }
```

```
--- check_method check that current transaction
--  method is one of defined in req.ep_method,
--  which is a comma separated items string.
--
--  If set req.invalid_method to true if method
--  is not allowed. Action to be taken in HAProxy
--  configuration:
--
--     http-request deny if { var(req.invalid_method),bool }
function check_method(txn)
  local method = txn.f:method()
  local allowed_methods = txn.get_var(txn,"req.ep_method")
  local methods = core.tokenize(allowed_methods,",",true)
  for _,v in ipairs(methods) do
    if v == method then
      return
    end
  end
  txn.set_var(txn,"req.invalid_method",true)
  return
end

-- register helpers
core.register_action("check-method", { "http-req" },
check_method)
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Validate JWT

JWT structure

- Use `jwt` tool to decode a token.
- Signed with `alg` and a shared secret.
- Can contain claims.
- Passed with bearer authorization.

```
jwt decode $TOKEN --ignore-exp --secret secret123

Token header
------------
{
  "typ": "JWT",
  "alg": "HS256"
}

Token claims
------------
{
  "iat": 1763385353,
  "isAdmin": "false",
  "userId": "691b1ff693b9f109bc7b84d6",
  "username": "seb"
}
```

Check token

- `http_auth_bearer` extracts the JWT from request authentication headers.
- `jwt_header_query` extracts data from a given JSON path.
- `jwt_verify` validate JWT signature with algorithm and secret.

```
acl has_auth http_auth_bearer -m found


http-request set-var(txn.bearer) http_auth_bearer

# Extract algorithm from JWT header
http-request set-var(txn.jwt_alg)
var(txn.bearer),jwt_header_query('$.alg')

# Check algorithm (as provided by application)
http-request deny if has_auth !{ var(txn.jwt_alg) -m str
"HS256" }

# Validate JWT with application secret
http-request deny if has_auth !{
var(txn.bearer),jwt_verify(txn.jwt_alg,"secret123") -m int 1
}
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

JWT forgery

- Upgrade `isAdmin` to `true`.
- `jwt` does not read payload from stdin.
  - use `printf` and `xargs` to generate correct command.
- We have now a valid forged JWT.
- Can be validated using a pipe:

```
jwt decode -j $TOKEN | \
  jq '.payload.isAdmin = "true"' | \
  printf "%s\0" "$(cat)" | \
  xargs -0 -I{} jwt encode --secret secret123 '{}' \
  | jwt  decode --secret secret123 --ignore-exp -
```

```
jwt decode -j $TOKEN
{
  "header": {
    "typ": "JWT",
    "alg": "HS256"
  },
  "payload": {
    "iat": 1763376850,
    "isAdmin": "false",
    "userId": "691739fe93b9f109bc7b833c",
    "username": "seb"
  }
}

# Forge the token
jwt decode -j $TOKEN | \
  jq '.payload.isAdmin = "true"' | \
  printf "%s\0" "$(cat)" | \
  xargs -0 -I{} jwt encode --secret secret123 '{}'

Token claims
------------
{
  "iat": 1763376850,
  "isAdmin": "true",
  "userId": "691739fe93b9f109bc7b833c",
  "username": "seb"
}
```

Check username

- Extract `username` into `txn.jwt_user` variable.
- Deny request if:
  - `username` is not in `admins.acl`.
  - and `isAdmin` is `true`.
- Protect `/api/getNote`:
  - `username` query parameter must match `txn.jwt_user`.

```
# Retrieve username from JWT token
http-request set-var(txn.jwt_user)
var(txn.bearer),jwt_payload_query('$.username')


# Only users declared in admins.acl are allowed to be admin.
http-request deny if !{ var(txn.jwt_user) -m str -f
admins.acl } {
var(txn.bearer),jwt_payload_query('$.isAdmin') -m str true
}


# Protect /api/user/USERNAME
http-request deny if METH_POST { path -m beg /api/user/ } !{
path,word(3,/),strcmp(txn.jwt_user) -m int eq 0 }

# Protect /api/getNote?username=USERNAME
http-request deny if { path -m str /api/getNote } !{
url_param(username),strcmp(txn.jwt_user) -m int eq 0 }
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Protect registration

URL anatomy

- Registration endpoint does not have rate limiting.
- Same goes with ticket creation.
- Limit access:
  - Per source IP.
  - and per endpoint.
- Some useful fetches, retrieve some values:
  - `req.hdr(host)` → `http://api.example.com:8080/api/endpoint?param=value`
  - `req.hdr(host),host_only` → `http://api.example.com:8080/api/endpoint?param=value`
  - `port` → `http://api.example.com:8080/api/endpoint?param=value`
  - `path` → `http://api.example.com:8080/api/endpoint?param=value`
  - `pathq` → `http://api.example.com:8080/api/endpoint?param=value`
  - `base` → `http://api.example.com:8080/api/endpoint?param=value`
  - `baseq` → `http://api.example.com:8080/api/endpoint?param=value`
  - `url_param(param)` → `http://api.example.com:8080/api/endpoint?param=value`
- `base32+src`:
  - Binary: `base` checksum and source IP address.
  - Ideal to rate limit per endpoint and source IP.

Rate limit sensitive endpoints

- Create a new `per_src_path_abuse` table:
  - tracks request rate over a 1 min window.

- In backend:
  - track request per `base32+src`.
  - only for:
    - `/api/register`
    - `/api/addTicket`
  - block the request if request rate is too high greater than 3/min.
  - Increment `gpc0` in case of abuse.

```
peers mypeers
  [...]
  # New table to track abuse per source and endpoint
  table per_src_path_abuse type binary len 28 size 1m expire
5m store http_req_rate(1m)


backend api-be
  [...]

  # track per source and endpoint access
  http-request track-sc1 base32+src table
mypeers/per_src_path_abuse if { path -m str /api/register
/api/addTicket }

  # Block request if client abuses
  acl is_endpoint_abuser sc1_http_req_rate -m int gt 3
  http-request deny status 429 if is_endpoint_abuser

  # Do not forget to increment gpc0 in case of abuse
  http-after-response sc-inc-gpc0(0) if is_api { status -m
int 401 403 404 429 }
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Upload size

Size matters

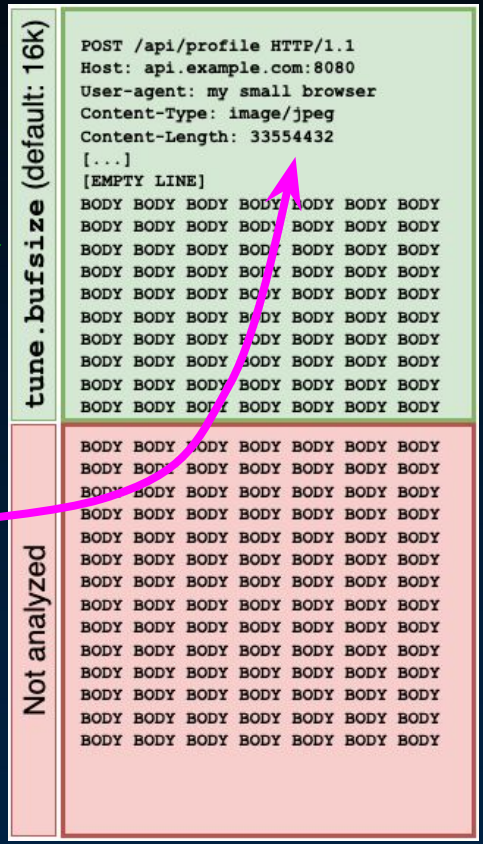
- No check is performed when uploading a profile picture.
- POST requests do not have any limitation.
- Endpoint map can be extended to add request body limitation:

```
/api/register           POST;5120
/api/login              POST;5120
/api/profile            POST,GET;5120
/api/user/{user}        GET;0
```

- Split method and size using `word()` converter:
  - `http-request set-var(req.ep_method) var(req.ep_data),word(1,;)`
  - `http-request set-var(req.ep_size) var(req.ep_data),word(2,;)`
- Notes:
  - GET should not have a body. ... but it can since not forbidden.

Notes on body sizes

- HAProxy only analyses 16k by default (see **`tune.bufsize`** option in global):
  - Headers.
  - Part of the body.
  - End of body may not be analyzed.
- Check body length:
  - **`req.body_len`**: available body size in bytes (in the request buffer, may be lower than advertised body size).
  - **`req.body_size`**: advertised body size from **`Content-Length`** header, or available data size if **`Transfer-Encoding`** is **`chunked`**.
- Rule of thumb: always check advertised size.



Check request size on HAProxy

- Retrieve the max allowed body size from map file (as seen before)
- Test the advertised **req.body_size**.

Caveats:

- HAProxy cannot compare a value (such as **req.body_size**) to a variable:

```
{ req.body_size -m int gt var(req.ep_post_size) }
```

- Need to subtract **req.ep_post_size** to **req.body_size**:
  - > 0 (**gt 0**): body is too large: reject.
  - = 0 (**eq 0**): body is the max size: OK.
  - < 0 (**gt 0**): body is less than max size: OK.

```
backend api-be
  [...]

  # Get max upload size for current endpoint
  http-request set-var(req.ep_post_size)
var(req.ep_data),word(2,;) if is_api

  # deny request if too large
  http-request deny status 413 if is_api {
req.body_size,sub(req.ep_post_size) -m int gt 0 }

  [...]

  # Do not forget to increment gpc0 if content is too large
  http-after-response sc-inc-gpc0(0) if is_api { status -m
int 401 403 404 413 429 }
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Miscellaneous fixes

Inject user attributes

- **score** and **solves** can be set when registering new user.
- HAProxy can validate body if:
  - Fits in **tune.bufsize** value (16k by default).
  - Body is buffered.
- Use **json_query** to get value from a json path.
- Bonus: block access to unreleased challenges.

```
{
  "username": "john",
  "password": "secret-password",
  "score": 123456,
  "solves": {
    "1":1,"2":1,"3":1,"4":1,"5":1,
    "6":1,"7":1,"8":1,"9":1,"10":1
  }
}
```

```
backend api-be
  [...]

  # Wait for body content
  option http-buffer-request

  [...]
  # Do not allow users to setup a score at registration time
  http-request deny if { path -m str /api/register } {
req.body,json_query('$.score') -m found }
  # Same with scores
  http-request deny if { path -m str /api/register } {
req.body,json_query('$.solves') -m found }


  # Unreleased challenges
  http-request deny if { path -m str /api/allChallenges } {
req.body,json_query('$.unreleased') -m found }
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

NoSQL injection

- Example of NoSQL injection:

```
{"username":"admin","password":{"$ne":null}}
```

- `json_path` fails if object is not a:
  - number (integer, float, boolean)
  - string
  - array
  - dictionaries are not return.
- Simply deny request if password cannot be retrieved for:
  - user registration.
  - user login.

```
backend api-be

  [...]

  # Deny invalid passwords
  http-request deny if { path -m str /api/register
/api/login } !{ req.body,json_query('$.password') -m found }
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Unfixable by design

- `/api/AddNoteWithLink` cannot be fixed:
  - Is meant to add a note with a link.
  - Fetches data from URL.
  - Broken design.
- Only solution: disable endpoint.
  - Remove `/api/AddNoteWithLink` from endpoint map.
  - Block endpoint with an explicit rule.

```
In endpoint map file:


/logout                     GET;0
# /api/addNoteWithLink        POST;5120
/api/getNote                GET;0
```

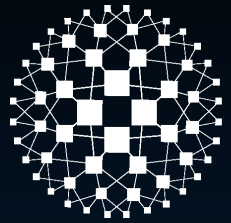
```
backend api-be
  [...]

  # Deny broken endpoints
  http-request deny if { path -m str /api/AddNoteWithLink }
```

OWASP

- `0xa1`: Broken Object Level Authorization
- `0xa2`: Broken Authentication
- `0xa3`: Broken Object Property Level Authorization
- `0xa4`: Unrestricted Resource Consumption
- `0xa5`: Broken Function Level Authorization
- `0xa6`: Server-Side Request Forgery
- `0xa7`: Security Misconfiguration
- `0xa8`: Lack of Protection from Automated Threats
- `0xa9`: Improper Assets Management
- `0xa10`: Unsafe Consumption of APIs

Thank you
for your attention



HAPROXY

Questions?

Visit us booth #A09

apidays

Paris 2025