

Web Application Penetration Testing eXtreme

v2

Encod%69ng and /^Filtering\$/

Section 01 | Module 01

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

Table of Contents

MODULE 01 | ENCODING AND /[^]FILTERING\$/

1.1 Data Encoding Basics

1.2 Filtering Basics

Learning Objectives

In this module we will talk about different types of **data encoding**.

We will see how to recognize, encode, and decode several different formats as well as discuss **filters** and how they work.



WAP

Data Encoding Basics



1.1 Data Encoding Basics

Even though web applications have different purposes, technologies, etc., the use of data encoding is something that cannot be neglected.

From a penetration testing point of view, understanding what kind of data encoding is being used and how it works is fundamental in ensuring that the tests are performed as intended.

```
20 def initialize(candidate):
21     self.candidate = candidate
22     self.score = 0
23     self.experiment = Experiment(self.candidate)
24     self.observations = []
25     self.control = Control()
26
27 def initialize(experiment, observations = [], control = None):
28     self.experiment = experiment
29     self.observations = observations
30     self.control = control
31     self.candidates = self.experiment.candidates
32     self.evaluate_candidates()
33
34 def evaluate_candidates():
35     # Evaluate the candidates' scores
36     for candidate in self.candidates:
37         self.score = self.experiment.score(candidate)
38         self.observations.append(candidate)
39         self.control.evaluate(candidate)
40
41 def match():
42     # Match the results of the experiment
43     # and the control
44     self.score = self.experiment.score(self.candidates)
45     self.observations = self.experiment.observations
46     self.control.evaluate(self.candidates)
47
48 def match():
49     # Match the results of the experiment
50     # and the control
51     self.score = self.experiment.score(self.candidates)
52     self.observations = self.experiment.observations
53     self.control.evaluate(self.candidates)
```



1.1.1 Dissecting Encoding Types

Let's briefly analyze the main types of data encoding used in web-oriented applications:

- **URL** encoding
- **HTML** encoding
- **Base (32|64)** encoding
- **Unicode** encoding

```
34 def _create_experiment(self, experiment_name, observations, control=None):
35     """Create a new experiment with the given name and observations.
36     """
37     # Create the experiment
38     experiment = Experiment(experiment_name)
39     # Create the observations
40     observations = self._create_observations(experiment, observations)
41     # Create the control
42     control = self._create_control(experiment, control)
43     # Create the candidates
44     candidates = self._create_candidates(experiment, observations, control)
45     # Evaluate the candidates
46     self._evaluate_candidates(experiment, candidates)
47     # Freeze the experiment
48     self._freeze_experiment(experiment)
49     # Return the experiment
50     return experiment
51
52 def _create_observations(self, experiment, observations):
53     """Create the observations for the given experiment.
54     """
55     # Create the observations
56     observations = []
57     for observation in observations:
58         observation = self._create_observation(experiment, observation)
59         observations.append(observation)
60     return observations
61
62 def _create_control(self, experiment, control):
63     """Create the control for the given experiment.
64     """
65     # Create the control
66     control = {}
67     for key in control.keys():
68         control[key] = self._create_control_value(experiment, key, control[key])
69     return control
70
71 def _create_candidates(self, experiment, observations, control):
72     """Create the candidates for the given experiment.
73     """
74     # Create the candidates
75     candidates = []
76     for observation in observations:
77         candidates.append(self._create_candidate(experiment, observation, control))
78     return candidates
79
80 def _evaluate_candidates(self, experiment, candidates):
81     """Evaluate the candidates for the given experiment.
82     """
83     # Evaluate the candidates
84     results = {}
85     for candidate in candidates:
86         results[candidate] = self._evaluate_candidate(experiment, candidate)
87     return results
88
89 def _freeze_experiment(self, experiment):
90     """Freeze the experiment.
91     """
92     # Freeze the experiment
93     experiment.freeze()
94     return experiment
95
96 def _create_observation(self, experiment, observation):
97     """Create the observation for the given experiment.
98     """
99     # Create the observation
100     observation = Observation(experiment, observation)
101     return observation
102
103 def _create_control_value(self, experiment, key, value):
104     """Create the control value for the given experiment.
105     """
106     # Create the control value
107     value = self._create_control_value(experiment, key, value)
108     return value
109
110 def _create_candidate(self, experiment, observation, control):
111     """Create the candidate for the given experiment.
112     """
113     # Create the candidate
114     candidate = Candidate(experiment, observation, control)
115     return candidate
116
117 def _evaluate_candidate(self, experiment, candidate):
118     """Evaluate the candidate for the given experiment.
119     """
120     # Evaluate the candidate
121     result = self._evaluate_candidate(experiment, candidate)
122     return result
```


1.1.1.1 URL Encoding

As stated in [RFC 3986](#), URLs sent over the Internet must contain characters in the range of the US-ASCII code character set. If unsafe characters are present in a URL, encoding them is required.

The URL-encoding, or **percent-encoding**, replaces characters outside the allowed set with a "%" followed by the two hexadecimal digits representing the numeric value of the octet.

1.1.1.1 URL Encoding

The table shown [here](#) is a simple character encoding chart that is useful in explaining which characters are “safe” and which characters should be encoded in URLs.

CLASSIFICATION	INCLUDED CHARACTERS	ENCODING REQUIRED?
Safe characters	Alphanumeric [0-9a-zA-Z], special characters \$ _ . ! * ' () , and reserved characters used for their reserved purposes (e.g., question mark used to denote a query string)	NO
ASCII Control characters	Includes the ISO-8859-1 (ISO-Latin) character ranges 00-1F hex (0-31 decimal) and 7F (127 decimal.)	YES
Non-ASCII characters	Includes the entire “top half” of the ISO-Latin set 80-FF hex (128-255 decimal.)	YES
Reserved characters	\$ & + , / : ; = ? @ (not including blank space)	YES*
Unsafe characters	Includes the blank/empty space and " < > # % { } \ ^ ~ [] `	YES

*** NOTE:** Reserved characters only need encoding when not used for their defined, reserved purposes.

1.1.1.1 URL Encoding

Some commonly encoded characters are:

CHARACTER	PURPOSE IN URI	ENCODING
#	Separate anchors	%23
?	Separate query string	%3F
&	Separate query elements	%26
%	Indicates an encoded character	%25
/	Separate domain and directories	%2F
+	Indicates a space	%2B
<space>	Not recommended	%20 or +

1.1.1.2 HTML Encoding

Even in HTML, it is important to consider the information integrity of the URL's and ensure that user agents (browsers & co.) display data correctly.

There are two main issues to address: inform the user agent on which character encoding is going to be used in the document and preserve the real meaning of some characters that have special significance.

1.1.1.2 HTML Encoding

In order to generate potential attacks and test cases, you should not only know how this kind of encoding works, but also know how the decoding mechanism work.

1.1.1.2.1 Document Character Encoding

There are several ways to instruct the user agent on which character encoding has been used in a given document.

These methods use the HTTP protocol and/or HTML directives.

```
38 def initialize(experiment, observations = [], control = nil)
39   @experiment = experiment
40   @observations = observations
41   @control = control
42   @candidates = observations + [control]
43   evaluate_candidates
44
45   freeze
46
47   @context =
48     experiment.context
49   end
50
51   # Fetch the name of the experiment
52   def experiment_name
53     @experiment.name
54   end
55
56   # Fetch the result a match between an experiment and a control
57   def matched?
58     @experiment.result == @control
59   end
60
61   @experiment.result == 1.1
```



1.1.1.2.1 Document Character Encoding

Define character encoding using HTTP

According to [HTTP 1.1 RFC](#), documents transmitted via **HTTP** can send a charset parameter in the header to specify the character encoding of the document sent. This is the **HTTP** header: **Content-Type**.

If this header is sent, we will see something like this:

Content-Type: text/html; charset=utf-8

1.1.1.2.1 Document Character Encoding

Define character encoding using HTTP

The **Content-Type** header indicates the media type of the body sent to the recipient. In the case of the **HEAD** method, it indicates the media type that would have been sent if the request had been a **GET**. If not defined, the RFC defines as the default charset the **ISO-8859-1**.

"8-bit single-byte coded graphic character sets" aka Latin 1

1.1.1.2.1 Document Character Encoding

Define character encoding using HTTP

To make the server send out the appropriate charset information, it is possible to change the server settings or use the server-side scripting language.

Let's look at some examples in different programming languages.

```
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
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2
```

1.1.1.2.1 Document Character Encoding

Define character encoding using HTTP

PHP> Uses the header() function to send a raw **HTTP** header:
header('Content-type: text/html; charset=utf-8');

ASP.Net> Uses the response object:
<%Response.charset="utf-8"%>

JSP> Uses the page directive:
<%@ page contentType="text/html; charset=UTF-8" %>

```
22 * @param $experiment - the experiment data result to be evaluated
23 * @param $observations - an array of observations, in which each
24 * @param $control - the control observation
25
26
27
28 def skillRank(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - @control
33   evaluate_candidates
34
35   freeze
36   end
37
38 * @param $context - the experiment's context
39 def context
40   @experiment.context
41 end
42
43 * @param $name - the name of the experiment
44 def experiment_name
45   @experiment.name
46 end
47
48 * @param $result - the result of a match between an experiment
49 def matches
50   ...
51 end
52
53 def result
54   ...
55 end
```

1.1.1.2.1 Document Character Encoding

Define character encoding using HTTP

It is also possible to set the character encoding using the **HTML** directive **META**. For example, this code is useful in specifying the character encoding of the current document to **UTF-8**:

```
<meta http-equiv="Content-Type" Content="text/html; charset=utf-8">
```

With **HTML5**, is also possible to write: `<meta charset="utf-8">`

1.1.1.2.2 Character References

In **HTML**, there are some special characters that can have multiple meanings. For example, the character **<** can represent the following:

- the beginning of a tag element

Hello

- a comparison operator in JavaScript

if (x < 7) {

- part of a text message

*"...less-than 3 would be written as **<** 3..."*

```
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
```

1.1.1.2.2 Character References

To preserve the real meaning of characters, the HTML specification provides a way to escape these special characters so that they are not "confused" as HTML or other codes. The following links will take you to the respective HTML4 and HTML5 specifications.



[CHR. REFERENCES](http://www.w3.org/TR/1998/REC-html40-19980424/charset.html#h-5.3)



[CHR. REFERENCES](http://www.w3.org/TR/html5/single-page.html#character-references)

1.1.1.2.2 Character References

As the standard states, character references must start with a **U+0026 AMPERSAND** character (&) and following this, there are multiple ways to represent character references.

Let's see some examples.

```
def initialize(experiment, address, name, context, candidates)
  @experiment = experiment
  @address = address
  @name = name
  @context = context
  @candidates = candidates
end

def freeze
  end

# Returns the experiment's context
def context
  @experiment.context
end

# Returns the name of the experiment
def experiment_name
  @experiment.name
end

# Returns whether the result is a match between an experiment and a result
def matched?
  !!@experiment.result.sub
end
```


1.1.1.2.2 Character References

We want to encode the character < (*less-than sign*):

Character Reference	Rule	Encoded character
Named entity	& + <u>named character references</u> + ;	<
Numeric Decimal	& + # + D + ; D = a decimal number	<
Numeric Hexadecimal	& + #x + H + ; H = an hexadecimal number (case-insensitive)	< <

1.1.1.2.2 Character References

Here we can see some interesting variations:

Character Reference	Variation	Encoded character
Numeric Decimal	No terminator (;)	&#60
	One or more zeroes before code	&#060 &#00000060
Numeric Hexadecimal	No terminator (;)	&#x3c
	One or more zeroes before code	&#0x3c &#00000x3c

1.1.1.3 Base (36|64) Encoding

Everyday, we see hexadecimal (aka Base16) numbers in network MAC addresses, X.509 certificates, Unicode characters, CSS colors, etc. This encoding scheme is frequently used in computer science.

In addition to Base16 encoding, there are other interesting **binary-to-text** encoding schemes: **Base36** and **Base64**

1.1.1.3.1 Base 36

Base 36 Encoding Scheme

Base36 is an interesting encoding scheme to play. It is the most compact, **case-insensitive**, alphanumeric numeral system using ASCII characters. In fact, the scheme's alphabet contains all digits [0-9] and Latin letters [A-Z].

The table on the next slide contains the conversions and comparisons with other well-known bases.

1.1.1.3.1 Base 36

Base 36 Encoding Scheme

binary	dec	hex	36
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	8
1001	9	9	9
1010	10	a	a
1011	11	b	b
1100	12	c	c

binary	dec	hex	36
1101	13	d	d
1110	14	e	e
1111	15	f	f
10000	16	10	g
10001	17	11	h
10010	18	12	i
10011	19	13	j
10100	20	14	k
10101	21	15	l
10110	22	16	m
10111	23	17	n
11000	24	18	o
11001	25	19	p

binary	dec	hex	36
11010	26	1a	q
11011	27	1b	r
11100	28	1c	s
11101	29	1d	t
11110	30	1e	u
11111	31	1f	v
100000	32	20	w
100001	33	21	x
100010	34	22	y
100011	35	23	z
100100	36	24	10

1.1.1.3.1 Base 36

Base 36 Encoding Scheme

For example, the number **1294870408610** in **Base10** is represented in **Base36** as **GIUSEPPE**.

Remember that it is case-insensitive. So, for example, the terms **XSS**, **xss**, **XsS**, etc... have the same representation in **Base10**, **43804**.

```
def skill_size(experiment, observations = [], control = null)
  # skill_size - the experiment
  # observations - an array of Observations, or null
  # control - the control observation
  # skill_size(experiment, observations = [], control = null)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - [control]
  evaluate_candidates

  freeze
end

def context
  experiment.context
end

def experiment_name
  experiment.name
end

def match?
  # check whether the result is a match between all
  # candidates
  @candidates.all? { |c| c.matches?(@control) }
end
```



1.1.1.3.1 Base 36

Base 36 Encoding Scheme

Now the question is, "why should we know this encoding scheme?" The answer is easy, because Base36 is used in many real-world scenarios.

For example, **Reddit** uses it for identifying both post's and comments, while some URL shortening services like **TinyURL** use **Base36** integer as compact, alphanumeric identifiers.



<http://tinyurl.com/jfvqr>

1.1.1.3.1 Base 36

Base 36 Encoding Scheme

Another example is if we want to convert **OHPE** from **Base36** to decimal, there are different implementations in many programming languages.

Let's see how to do this with **PHP** and **JavaScript**.

1.1.1.3.1 Base 36

Base 36 Encoding Scheme: PHP

PHP uses the [base_convert\(\)](#) function to convert numbers:

OHPE in Base 10 is `<?=base_convert("OHPE",36,10);?>`

1.1.1.3.1 Base 36

Base 36 Encoding Scheme: JavaScript

JavaScript uses two functions:

- **(1142690).toString(36)**
- **1142690..toString(36)**
parseInt("ohpe",36)

// encode

// decode

```
1 // ...
2 // ...
3 // ...
4 // ...
5 // ...
6 // ...
7 // ...
8 // ...
9 // ...
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 // ...
```

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

Base64 is one of the most widespread binary-to-text encoding schemes to date. It was designed to allow binary data to be represented as **ASCII** string text.

It is an encoding scheme, not an encryption one. This is not clear to many developers who use **Base64** instead of encryption to store or transmit sensitive information.

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

The alphabet of the **Base64** encoding scheme is composed of digits [0-9] and Latin letters, both upper and lower case [a-zA-Z], for a total of 62 values. To complete the character set to **64** there are the plus (+) and slash (/) characters.

Different implementations, however, may use other values for the latest two characters and the one used for padding (=). For a complete list look [here](#).

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

To encode a message in **Base 64**, the algorithm divides the message into groups of **6 bits*** and then converts each group, with the respective **ASCII** character, following the conversion table.

**That's why the allowed characters are 64 ($2^6 = 64$).*

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

Binary (dec)	Base 64	Binary (dec)	Base 64	Binary (dec)	Base 64	Binary (dec)	Base 64
000000 (0)	A	010000 (16)	Q	100000 (32)	g	110000 (48)	w
000001 (1)	B	010001 (17)	R	100001 (33)	h	110001 (49)	x
000010 (2)	C	010010 (18)	S	100010 (34)	i	110010 (50)	y
000011 (3)	D	010011 (19)	T	100011 (35)	j	110011 (51)	z
000100 (4)	E	010100 (20)	U	100100 (36)	k	110100 (52)	0
000101 (5)	F	010101 (21)	V	100101 (37)	l	110101 (53)	1
000110 (6)	G	010110 (22)	W	100110 (38)	m	110110 (54)	2
000111 (7)	H	010111 (23)	X	100111 (39)	n	110111 (55)	3
001000 (8)	I	011000 (24)	Y	101000 (40)	o	111000 (56)	4
001001 (9)	J	011001 (25)	Z	101001 (41)	p	111001 (57)	5
001010 (10)	K	011010 (26)	a	101010 (42)	q	111010 (58)	6
001011 (11)	L	011011 (27)	b	101011 (43)	r	111011 (59)	7
001100 (12)	M	011100 (28)	c	101100 (44)	s	111100 (60)	8
001101 (13)	N	011101 (29)	d	101101 (45)	t	111101 (61)	9
001110 (14)	O	011110 (30)	e	101110 (46)	u	111110 (62)	+
001111 (15)	P	011111 (31)	f	101111 (47)	v	111111 (63)	/

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

If the total number of bits is not a multiple of 6, then null bits need to be added until the total is both a multiple of 6 and the result length a multiple of 4.

Then, if the **latest** group is 'null' (000000), the respective encoding value is = but, if the trailing "null groups" are two they will be encoded as ==.

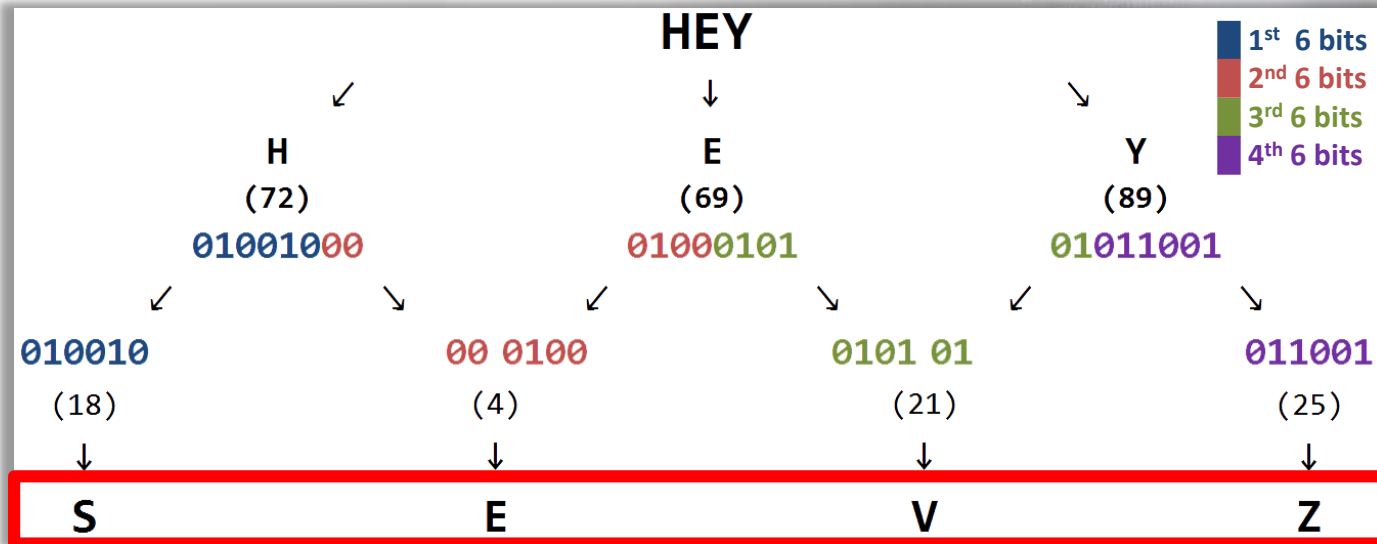
Let's check out some examples.

```
41 def _null_control = "000000"
42
43 # Note: Don't use the experiment title directly in the
44 # observations - an array of observations, in this case
45 # control - see control observation
46
47 def _null_observation(experiment, observations = [], control = null)
48   @experiment = experiment
49   @observations = observations
50   @control = control
51   @candidates = observations + [control]
52
53   experiment.context
54   nil
55
56 # Build the name of the experiment
57 def experiment_name
58   experiment.name
59   nil
60
61 # Build the result a match between all
62 def match
63   nil
64
65 # Return result
66 def result
67   nil
```

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

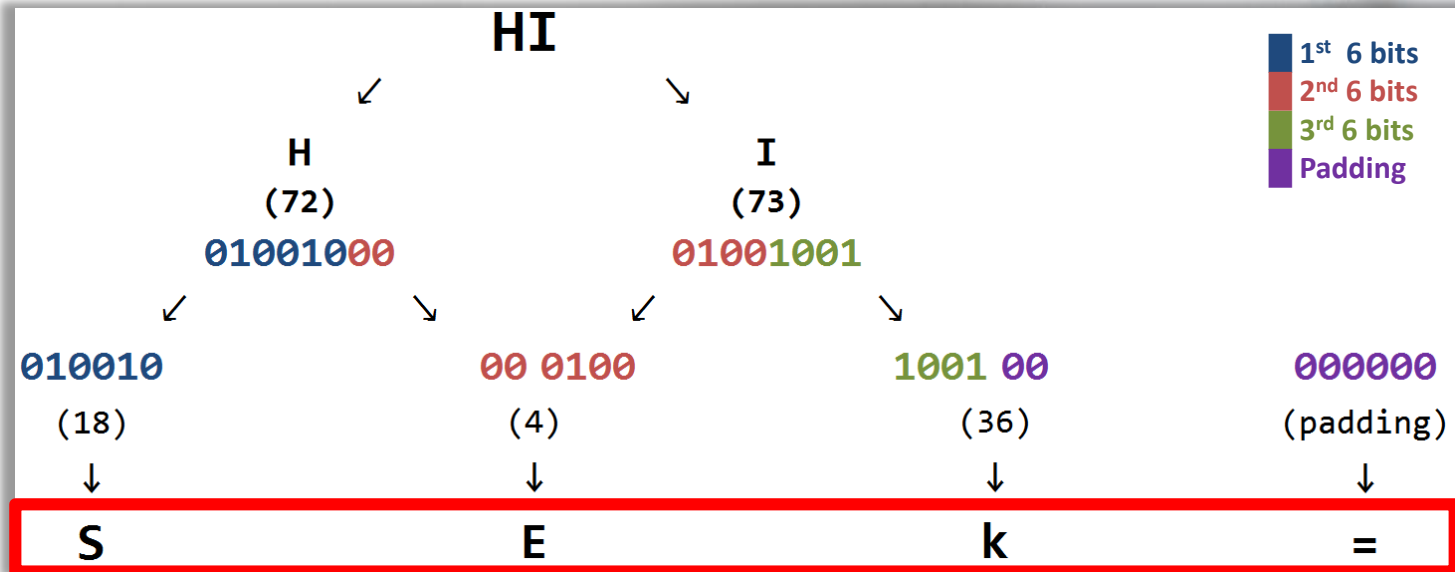
To encode the term "HEY" we have:



1.1.1.3.2 Base 64

Base 64 Encoding Scheme

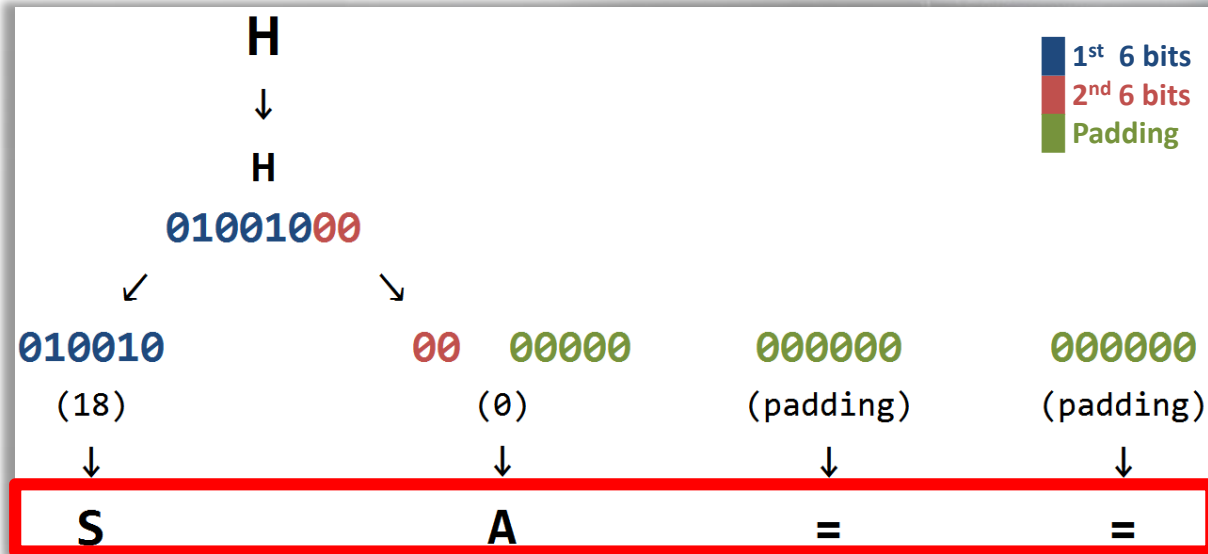
To encode the term "HI" we have:



1.1.1.3.2 Base 64

Base 64 Encoding Scheme

To encode the char "H" we have:



1.1.1.3.2 Base 64

Base 64 Encoding Scheme

Naturally, due to its popularity, there are many encoding / decoding implementations of **Base64** in a variety of different programming languages.

Let's see some of them.

```
def encode_base64(data):
    """Encode data to base64"""
    import base64
    return base64.b64encode(data).decode()

def decode_base64(data):
    """Decode data from base64"""
    import base64
    return base64.b64decode(data).decode()

# Example usage
data = "Hello, World!"
encoded = encode_base64(data)
decoded = decode_base64(encoded)
print(encoded)  # SGVsbG8sIFdvb2Rs!
```



1.1.1.3.2 Base 64

Base 64 Encoding Scheme: PHP

PHP uses [base64_encode](#) and [base64_decode](#) functions to encode/decode data based on **MIME Base 64** implementation:

```
<?=base64_encode('encode this string')?> //Encode
```

```
<?=base64_decode('ZW5jb2RlIHRobaXMgc3RyaW5n')?>  
//Decode
```

1.1.1.3.2 Base 64

Base 64 Encoding Scheme: JavaScript

Many browsers can handle **Base64** natively through functions **btoa** and **atob**:

`window.btoa('encode this string');` //Encode

`window.atob('ZW5jb2RlIHRobXMgc3RyaW5n');` //Decode

1.1.1.3.2 Base 64

Base 64 Encoding Scheme

It is important to notice that if we want to handle **Unicode** strings, then we should encode them before using **Base64** functions. For example, in **JavaScript** this is possible as follows:

The escapes and encodings are required to avoid exceptions with characters out of range. Learn more [here](#).

```
1 function utf8_to_b64( str ) {  
2     return window.btoa(encodeURIComponent( escape( str ) ));  
3 }  
4  
5 function b64_to_utf8( str ) {  
6     return unescape(decodeURIComponent(window.atob( str ) ));  
7 }  
8  
9 // Usage:  
10 utf8_to_b64('✓ à la mode'); // JTI1dTl3MTMlMjUyMCUyNUUwJTl1MjBsYSUyNTIwbn9kZQ==  
11 b64_to_utf8('JTI1dTl3MTMlMjUyMCUyNUUwJTl1MjBsYSUyNTIwbn9kZQ=='); // "✓ à la mode"  
12  
13 utf8_to_b64('I \u2661 Unicode!'); // SSUyNTIwJTl1dTl2NjElMjUyMFVuaWVhZGULMjUyMQ==  
14 b64_to_utf8('SSUyNTIwJTl1dTl2NjElMjUyMFVuaWVhZGULMjUyMQ=='); // "I ♥ Unicode!"
```

1.1.1.4 Unicode Encoding



Unicode (aka **ISO/IEC 10646** Universal Character Set) is the character encoding standard created to enable people around the world to use computers in any language. It supports all the world's writing systems.

Because **Unicode** contains such a large number of characters, glyphs, numbers, etc., from a security point of view, it is fascinating because incorrect usage can expose web applications to possible security attacks. One such example, is that it can be useful to bypass filters.

1.1.1.4 Unicode Encoding

We are not going to cover the Unicode specifics, but if you want to have a better background on the argument, character sets and related topics, the following link is a great starting point:

<http://www.joelonsoftware.com/articles/Unicode.html>

1.1.1.4 Unicode Encoding

There are three ways to map Unicode character points:

- **UTF-8**
- **UTF-16**
- **UTF-32**

UTF means **Unicode Transformation Format** and the trailing number indicates the number of bits to represent code points.

[illegible]

1.1.1.4 Unicode Encoding

Thus, each UTF has a different representation and it is important to understand how to handle these in our tests. The following table shows a sample message encoded in the three different UTF formats.

CHARACTER	REPRESENTATION			
	Unicode	UTF-8 code point	UTF-16 code point	UTF-32 code point
I	U+0049	49	0049	00000049
♥	U+2665	E2 99 A5	2665	00002665
📄	U+1F37B	F0 9F 8D BB	D83C DF7B	0001F37B

1.1.1.4 Unicode Encoding

It is also useful to know how Unicode characters are handled through different implementations like URLs, HTML, JavaScript, etc. We can see some of them below.

CHARACTER	REPRESENTATION					
	Unicode Code Point	URL-Encoding (UTF-8)	Named	HTML Entity Decimal	Hexadecimal	JavaScript, JSON, Java (UTF-16)
I	U+0049 (hex 49, dec 73)	%49	-	I	I	\u0049
♥	U+2665 (hex 2665, dec 9829)	%E2%99%A5	♥	♥	♥	\u2665
📄	U+1F37B (hex 1F37B, dec 127867)	%F0%9F%8D%BB	-	🍻	🍻	\uD83C\uDF7B

1.1.1.4 Unicode Encoding

Besides the representation of Unicode characters in multiple encoding types, another interesting aspect is the interpretation that humans and different implementations give to some characters.

```
42 # Define the experiment's context
43 def context
44   experiment.context
45 end
46
47 # Define the name of the experiment
48 def experiment_name
49   experiment.name
50 end
51
52 # Define whether the result is a match between two samples
53 def matched?
54   # ...
55 end
56
57 # Define the result of the experiment
58 def result
59   # ...
60 end
```



1.1.1.4 Unicode Encoding

Homoglyph | Visual Spoofing

*"In typography, a **Homoglyph** is one or two or more characters, or glyphs, with shapes that either appear identical or cannot be differentiated by quick visual inspection."* [[Wikipedia](https://en.wikipedia.org/wiki/Homoglyph)]

An additional classification is:

HOMOGRAPH > a **word** that looks the same as another word

HOMOGLIPH > a look-alike **character** used to create homographs

1.1.1.4 Unicode Encoding

Homoglyph | Visual Spoofing

One of the possible attacks with Unicode is called:

Visual Spoofing



U+006F
LATIN SMALL
LETTER O

U+03BF
GREEK SMALL
LETTER OMICRON

1.1.1.4 Unicode Encoding

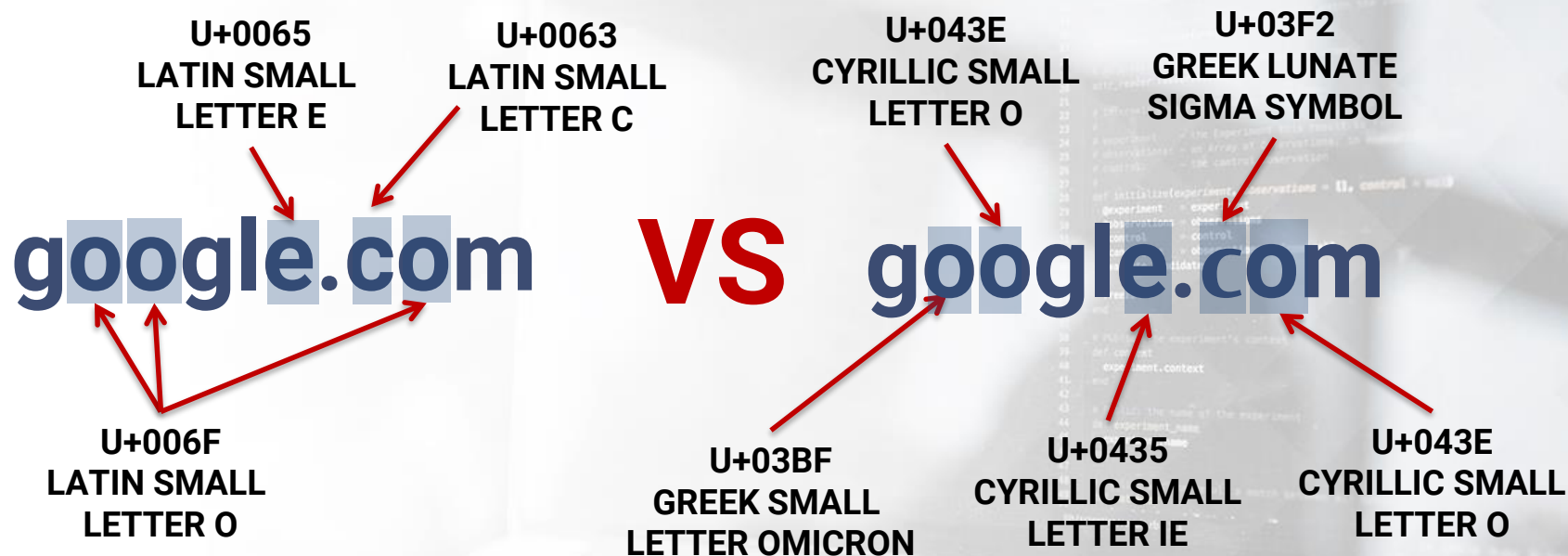
Homoglyph | Visual Spoofing

If we analyze the characters code points of the string, the differences between the **o** and the **o** are evident, but for a human this is not so obvious.

These kind of characters, also known as a **confusable**, received special attention from the Unicode Consortium (**TR39**). So much so, that they have provided a **utility**, whereby given an input string you can see the combinations that are confusable with it.

1.1.1.4 Unicode Encoding

Homoglyph | Visual Spoofing - Example: google.com



1.1.1.4 Unicode Encoding

Homoglyph | Visual Spoofing

To speed the homoglyphs generation, rather than searching for look-alike characters in Unicode, there is an interesting application made by Adrian “Irongeek” Crenshaw:

Homoglyph Attack Generator

<http://www.irongeek.com/homoglyph-attack-generator.php>

1.1.1.4 Unicode Encoding

Homoglyph | Visual Spoofing

This attack generator tool is part of a really interesting paper where the author explains the abuse of Unicode characters in order to obfuscate phishing attacks through the use of Homoglyph and Punycode.

Punycode and Homoglyph Attacks to Obfuscate URLs for Phishing

<http://www.irongeek.com/i.php?page=security/out-of-character-use-of-punycode-and-homoglyph-attacks-to-obfuscate-urls-for-phishing>

1.1.1.4 Unicode Encoding

Computer Interpretations

Another interesting aspect is related to string and character "evolutions," which occur during normal software processes transformations.

An example of this is upper and lower casing transformations, which are described in the upcoming slides.

```
21 def initialize_experiment(experiment, observations = {}, control = null)
22   @experiment = experiment
23   @observations = observations
24   @control = control
25   @candidates = initialize_candidates
26   evaluate_candidates
27
28   freeze
29 end
21
22 # Returns the experiment's context
23 def context
24   @context
25 end
26
27 # Returns the name of the experiment
28 def name
29   @name
30 end
31
32 # Returns whether the results a match between an experiment
33 # and a control
34 def match?
35   @match
36 end
37
38 # Returns the result of the experiment
39 def result
40   @result
41 end
```

1.1.1.4 Unicode Encoding

Computer Interpretations - Example: Censored Feedback

In a feedback page, the application layer performs a censorship check before storing data in a DB.

There is an input filter that blocks the term **EVIL**, then transform the string to lowercase and store it in DB.

1.1.1.4 Unicode Encoding

Computer Interpretations - Example: Censored Feedback

The input flow could be as follow:

➤ A user sends the following message:

Evil intent, as usual!

➤ The filter checks for evil strings, but without success.

U+0130 (İ)
LATIN CAPITAL LETTER I
WITH DOT ABOVE

Evİl != evil



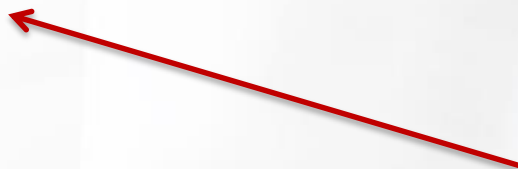
```
17 def __init__(self, experiment_name, observations, control, candidates):
18     self.experiment_name = experiment_name
19     self.observations = observations
20     self.control = control
21     self.candidates = candidates
22
23     # The experiment will result in a
24     # observations = an array of observations, in which
25     # the control observation
26
27     self.experiment, observations = self, control
28     self.observations = observations
29     self.control = control
30     self.candidates = candidates
31     self.evaluate_candidates()
32
33     freeze
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
```

1.1.1.4 Unicode Encoding

Computer Interpretations - Example: Censored Feedback

➤ The casing operation is performed [to lowercase]:

evil intent, as usual!



U+0130(İ) to lowercase is

U+0069

LATIN SMALL LETTER I

➤ CENSURED BYPASSED

1.1.1.4 Unicode Encoding

Computer Interpretations - Example: Censored Feedback

This happened because a casing operation is performed in the application flow **AFTER** a security check.

Of course, it also works by upper casing characters like this:

U+017F
LATIN SMALL LETTER LONG S

↗

ſ to upper case is S ↖

U+0053
LATIN CAPITAL LETTER S

1.1.1.4 Unicode Encoding

Computer Interpretations- Example: Censored Feedback

It turns out, that this type of vulnerable implementation may allow an attacker to bypass filters. For example, they can bypass anti cross-site scripting and SQL injection filters and so forth.

These are things that ~~never~~ happen in real world!
Check them out here:

Creative usernames and Spotify account hijacking

<http://labs.spotify.com/2013/06/18/creative-usernames/>

1.1.1.4 Unicode Encoding

Computer Interpretations

There are other ways in which characters and strings can be transformed by software processes, such as normalization, canonicalization, best fit mapping, etc.

These are brilliantly summarized and explained by Chris Weber in his:

Unicode Security Guide

<http://websec.github.io/unicode-security-guide/>

1.1.1.4 Unicode Encoding

Computer Interpretations: Mixed Examples

Normalization:

d r o p t a b l e **becomes** drop table

Canonicalization:

$\left. \begin{array}{l} < \text{ (U+2039)} \\ < \text{ (U+FE64)} \\ < \text{ (U+ff1c)} \end{array} \right\} \text{ becomes } < \text{ (U+003C)}$

```
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
```

1.1.1.4 Unicode Encoding

If you want to play a bit with Unicode characters, you can visit [Unicode utilities](#) to get information about a character or search confusable characters.

In addition to this tool, there are other interesting resources such as [codepoints.net](#), [txtn.us](#) and [Unicode Text Converter](#).

<http://unicode.org/cldr/utility/>

<http://codepoints.net/>

<http://txtn.us/>

<http://www.panix.com/~eli/unicode/convert.cgi>

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

1.1.2 Multiple (De|En) Codings

As we have just seen in the previous slides, data encoding is fundamental for communication in web applications.

Sometimes, encoding is required by the chosen channel. Other times it is used intentionally by developers but, sometimes it is used **multiple times**, intentionally and not. It is also common to **abuse** multiple encodings to bypass security measures



1.1.2 Multiple (De|En) Codings

Example: URL-Encoding > URL

A simple scenario could be a URL sent over URL, like the common URL redirects we see daily for surfing web sites:

`http://mywebsite/login.php?redirectURL=FORW-URL?is_ok=yes`

In this case, the forwarding URL will be URL-encoded in order to respect the URL rules:

`http://mywebsite/login.php?redirectURL=FORW-URL%3Fis_ok%3Dyes`

1.1.2 Multiple (De|En) Codings

Example: URL-Encoding > URL

Of course, even if a parameter sent is not a URL, encoding is still required:

`http://mywebsite/login.php?param=I`  

In this case, the parameter contains Unicode characters, hence the URL-encoding will be:

`http://mywebsite/login.php?param=I`  

1.1.2 Multiple (De|En) Codings

Example: URL-Encoding > URL

Multiple encodings may also occur if the parameter sent is previously encoded, like the following:

`http://mywebsite/login.php?param=Rk9SVy1VUkw/Y2F0PWnsb3ducw==`

In this case, we have a Base64 data encoded to send over URL; thus, the result will be:

`http://mywebsite/login.php?param=Rk9SVy1VUkw%2FY2F0PWnsb3ducw%3D%3D`

1.1.2 Multiple (De|En) Codings

Example: URL-Encoding > URL

Sometimes, a simple parameter can be a structured parameter. For example, the following cookie value:

SESSION = dXN1cm5hbWU6Y2xvd247cGFzc3dvcmQ6dGh1Q2xvd24h

...may appear like a 'random' value used to identify the user session, but it decodes to:

SESSION = username:clown;password:theClown!

1.1.2 Multiple (De|En) Codings

It turns out that we can construct several examples based on multiple encoding/decoding scenarios, but the topic here is to understand that identifying different data encoding types is an important skill that may help you to detect and exploit multiple scenarios.

All in all, in order to respect the application requirements and properly test a web application, we must detect and consider multiple encoding and decoding operations.

You've been studying quite intently. We recommend taking a quick break and come back refreshed. ^_^

Filtering Basics



1.2 Filtering Basics

A common, yet often recommended, best practice to protect web applications against malicious attacks is the use of specific **input filtering** and **output encoding** controls.

These kinds of controls may range from naive blacklists to experienced and highly restrictive whitelists. What about in the real world? We are somewhere in the middle!

1.2 Filtering Basics

Controls can be implemented at different layers in a web application. They can be represented as either **libraries and APIs** (by naive developers) or, in the best case, by internal specialists or external organizations, like **ESAPI by OWASP**.

Security controls are also inside most common browsers.

1.2 Filtering Basics

Sometimes, because of a multitude of complications and standards, it is not possible, or it is too problematic to implement internal solutions. These solutions may be in the form of libraries or APIs; however, the key is in the adoption of external solutions.

Generally, these solutions fall into the **IDS** and **IPS** world, but for web applications, the most chosen are the **Web Application Firewall (WAFs)**.

```
def initialize(experiment, observations = list(), control = None):
    @experiment = experiment
    @observations = observations
    @control = control
    @candidates = observations + [control]
    evaluate_candidates

    freeze
    end

# Returns the experiment's context
def context
    @experiment.context
end

def experiment_name
    @experiment_name
end

def metadata
    @metadata
end

@experiment.resultub 1.1
```



1.2 Filtering Basics

Even with WAFs, it is possible to choose between several implementations.

These range from not only commercial and very expensive, but also free and open source solutions like the well known **ModSecurity**.

1.2 Filtering Basics

The instructions on what a WAF, or generally, what a filter must block/allow are defined as **rules** (also referred as **filters**).

Before we analyze different filter implementations, let's see the *de facto* standard used to write rules.

right

wrong

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

1.2 Filtering Basics

Regular Expressions (RE or RegEx) represents the official way used to define the filter rules. Mastering RegEx is fundamental to understand how to bypass filters because RE are extremely powerful!

The upcoming slides contain a brief introduction to the subject.

```
18 def initialize(experiment, observations = [], candidates = [])
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations - @control
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 # Returns the result a match between an observation and a candidate
34 def match?
35   %r{science/result: 1.1}
```



1.2.1 Regular Expressions

NOTE:

This is a brief introduction to Regular Expressions. For a comprehensive introduction and much more, in the references you will find some interesting resources.

1.2.1 Regular Expressions

A regular expression is a special sequence of characters used for describing a **search pattern**.

For the sake of clarity, in the next slides we will see the following notation:

- regular expression > regex
- pattern matched > match

1.2.1 Regular Expressions

Many programming languages, text processors, etc., support regular expressions. The implementation system of regex functionality is often called **regular expression engine**. Basically, a regex "engine" tries to match the pattern (regex) to the given string.

There are two main types of regex **engines**: **DFA** and **NFA**, also referred to as **text-directed** and **regex-directed** engines. The key difference between the two engines is a notational convenience in the construction of the FA (Finite Automaton).

1.2.1 Regular Expressions

The DFA engine is faster than NFA because of its deterministic approach, but it does not support useful features like **lazy quantifiers** and **backreferences**. Additionally, NFA works the way humans tend to do: "regex-directed". It's no surprise that the NFA engine is more popular.

Here is a table of notable programs that use DFA or NFA engines:

ENGINE	PROGRAM
DFA	awk, egrep, MySQL, Procmail
NFA	.NET languages, Java, Perl, PHP, Python, Ruby, PCRE library, vi, grep, less, more

```
23 # new_obs - the experiment's new observations
24 # observations - an array of Observations, in ascending
25 # order
26 # control - the control observation
27
28 def initialize(experiment, observations = [], control = nil)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = initialize_candidates
33   evaluate_candidates
34
35   freeze
36 end
37
38 # Find all the experiment's contexts
39 def context
40   @context ||= experiment.context
41 end
42
```


1.2.1 Regular Expressions

The syntax and behavior of a particular engine is called a **regular expression flavor**. Since the engine is a piece of software, there are different versions and of course they are not fully compatible with each other.

Thus, expect to find multiple flavor based on the library you are using/testing!

```
def initialize(experiment, observations = [], control = None):
    @experiment = experiment
    @observations = observations
    @control = control
    @candidates = observations + (control if control else [])
    evaluate_candidates

    freeze

    end

# Returns the experiment's context
def context
    experiment.context

    {
      experiment_name:
        experiment.name
    }
end

# Returns whether the result is a match between all
def match?
    @candidates == @observations
end

# Returns the result of the match
def result
    @match ? @candidates : @observations
end
```

1.2.1 Regular Expressions

For a complete comparison list of regular expression engines/flavors visit the following links:

[Comparison of regular expression engines](http://en.wikipedia.org/wiki/Comparison_of_regular_expression_engines)

- http://en.wikipedia.org/wiki/Comparison_of_regular_expression_engines

[Regular Expression Flavor Comparison](http://www.regular-expressions.info/reflavors.html)

- <http://www.regular-expressions.info/reflavors.html>

1.2.1 Regular Expressions

Since regular expression is a "symbolic language", to master this tool we must know the symbols of the language. They are few and have specific meanings.

Let's look at some tables that collect these symbols and their meanings.



1.2.1.1 Metacharacters

Char	Name	Meaning
^	Caret	The position at the START of the line.
\$	Dollar	The position at the END of the line.
()	Opening/Closing parenthesis	Start/close a characters group.
[]	Opening/Closing square bracket	Start/close a characters class.
?	Question mark	One or zero (optional) of the immediately-preceding item (char or group).
+	Plus	One or more of the immediately-preceding item (char or group).
*	Star or asterisk	Any number, including none, of the immediately-preceding item (char or group).
.	Period or dot	Shorthand for a character class that matches any character.
\	Backslash	Escape special characters.
	Vertical bar or pipe	It means OR. Combines multiple expressions in one that matches any of single ones.
{}	Opening/Closing curly brace	Start/close repetitions of a characters class.

1.2.1.2 Shorthand Character Classes

Since there are some character classes frequently used, there are also related shorthand classes that are useful to decrease the size and increase the readability of a regex.

The table on the next slide has the most common shorthand classes.

1.2.1.2 Shorthand Character Classes

SHORTHAND	NAME	MEANING
<code>^</code>	Caret	If at the beginning of the character class, it means to reverse the matching for the class.
<code>\d</code>	Digit	Matches any digit character. The same as <code>[0-9]</code>
<code>\D</code>	Non-digit	The complement of <code>\d</code> . The same as <code>[^\d]</code>
<code>\w</code>	Part-of-word character	Matches any alphanumeric character or an underscore. The same as <code>[a-zA-z0-9_]</code> In some flavors the underscore is omitted.
<code>\W</code>	Non-word character	The complement of <code>\w</code> . The same as <code>[^\w]</code>
<code>\s</code>	Whitespace character	Matches any whitespace character. The same as <code>[\f\n\r\t\v]</code>
<code>\S</code>	Non-whitespace character	The complement of <code>\s</code> . The same as <code>[^\s]</code>

1.2.1.3 Non-Printing Characters

Oftentimes, to evade bad filters and obfuscate the payload it is common to use **non-printing characters**. These are control characters used mainly to control the format of displayed/printed information.

The most used characters are represented in the table on the next slide.

1.2.1.3 Non-Printing Characters

SHORTHAND	NAME (Symbol, Unicode)	MEANING
\0	NUL (🔍 U+0000)	NUL Byte, in many programming languages marks the end of a string.
\b	Backspace (🔍 U+0008)	Within a character class represent the backspace character, while outside \b matches a word boundary.
\t	Horizontal tab (🔍 U+0009)	Generated by the Tab key on a standard keyboard.
\n	Line feed (🔍 U+000A)	New line.
\v	Vertical tab (🔍 U+000B)	Vertical tabulation.
\f	Form feed (🔍 U+000C)	Form feed.
\r	Carriage return (🔍 U+000D)	In HTTP, the \r\n sequence is used as the end-of-line marker.
\e	Escape (🔍 U+001B)	Escape character (Only for GNU Compiler Collection).


1.2.1.4 Unicode

Regular expression flavors that work with Unicode use specific meta-sequences to match code points.

The sequence is `\ucode-point`, where *code-point* is the hexadecimal number of the character to match. There are regex flavors like PCRE that do not support the former notation, but use an alternative sequence `\x{code-point}` in its place.

1.2.1.4 Unicode

Match Unicode Code Point

For example, the regex `\u2603` matches the snowman character  in .NET, Java, JavaScript and Python.

If we want to match the same character to the PCRE library in Apache and PHP, we must use the other notation:

`\x{2603}`

1.2.1.4 Unicode

Instead of matching a single Unicode code point, it is possible to match if a character has a specific "quality". This is possible with the use of **Unicode properties**.

Unicode defines for each character, properties or qualities, such as: *"is this character uppercase"* or *"is this character a punctuation"* and so on, in order to match these qualities with regex exists specific meta-sequences.

1.2.1.4 Unicode

The characters that have a specific quality are matched with the meta-sequence `\p{quality-id}`. To match the characters that do not have the quality, the meta-sequence is `\P{quality-id}`.

Some general Unicode character qualities are reported in the table on the next slide.

```
18 def initialize(experiment, observations = nil)
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations - @control
23   evaluate_candidates
24
25   freeze
26 end
27
28 # PARSING the experiment's context
29 def context
30   experiment.context
31 end
32
33 # EXPERIMENT_NAME
34 def experiment_name
35   experiment.name
36 end
37
38 # MATCHING the results a match between all
39 def match?
40   ...
41 end
42
43 # SCIENTIST/RESULT
44 def scientist/result
45   1.1
```

1.2.1.4 Unicode

CHARACTER QUALITY	DESCRIPTION
<code>\p{L}</code> or <code>\p{Letter}</code>	All the letters, from any language.
<code>\p{Li}</code> or <code>\p{Lowercase_Letter}</code>	Lowercase letters that have the respective uppercase quality.
<code>\p{Z}</code> or <code>\p{Separator}</code>	Characters used to separate, but without visual representation.
<code>\p{S}</code> or <code>\p{Symbol}</code>	Currency symbols, math symbols, etc...
<code>\p{N}</code> or <code>\p{Number}</code>	All the numeric characters.
<code>\p{Nd}</code> or <code>\p{Decimal_Digit_Number}</code>	Numbers from zero to nine in multiple scripts except Chinese, Japanese, and Korean.
<code>\p{P}</code> or <code>\p{Punctuation}</code>	All the punctuation characters.

1.2.1.4 Unicode

Match Unicode Category

For example, to match the lowercase characters in this string:

Ğ ĩ ũ Ș ê p P Ë

the regex is `\p{Ll}` and the characters matched are

ĩ ũ ê p

1.2.1.4 Unicode

Match Unicode Category

To match the string with all the case variations (lower, upper and title), this regex does the job:

```
[ \p{Ll} \p{Lu} \p{Lt} ]
```

As a shorthand, some regex flavors implement this solution:

```
\p{L&}
```

1.2.2 Web Application Firewall

We have briefly seen Regular Expressions, which is the main method used to define the rules of how a WAF should behave. They define which input is good or bad for the web application and respectively what the WAF should allow or block.

The meaning given to the rules defines the mode by which a WAF should behave. It can be whitelisted or blacklisted. Basically, a WAF in whitelisting mode allows only what is explicitly defined in the rules; however, in contrast, blacklisting mode allows anything except what is explicitly denied in the rules.

1.2.2 Web Application Firewall

It turns out that whitelisting mode is the best solution to protect a web application; on the other hand, to customize the rules is not an easy task. This requires a deep knowledge of the application to protect and, obviously, of the WAF solution.

Furthermore, the whitelisting mode is prone to false positives, which is the reason it is very common to find WAFs deployed in blacklisting mode rather than whitelisting mode.

1.2.2 Web Application Firewall

The main problem with this kind of approach is that there are multiple ways to reach the same goal; therefore, every small change of the attack payload must be added to the blacklist, otherwise you have a WAF bypass!

Predicting or keeping track of each payload tweak is very hard, that's why we frequently read the expression:

All WAFs can be bypassed!

1.2.2 Web Application Firewall

Over the years, security researchers have discovered several "*alternative vectors*" (i.e. **WAF bypasses**) and a lot of well-known names were involved.

The following examples are just simple rules to follow to deceive ingenuous WAFs.

1.2.2.1 Simple Rules to Bypass WAFs

Cross-Site Scripting

Instead of using:

- ▼ **alert('xss')**
- ▼ **alert(1)**

DO



The best choice is:

- ▲ **prompt('xss')**
- ▲ **prompt(8)**
- ▲ **confirm('xss')**
- ▲ **confirm(8)**
- ▲ **alert(/xss/.source)**
- ▲ **window[/alert/.source](8)**

1.2.2.1 Simple Rules to Bypass WAFs

Cross-Site Scripting

Instead of using:

▼ `alert(document.cookie)`

DO



The best choice is:

- ▲ `with(document)alert(cookie)`
- ▲ `alert(document['cookie'])`
- ▲ `alert(document[/cookie/.source])`
- ▲ `alert(document[/coo/.source+/kie/.source])`

1.2.2.1 Simple Rules to Bypass WAFs

Cross-Site Scripting

Instead of using:

▲ ``

▲ `javascript:alert(document.cookie)`

DO



The best choice is:

▲ `<svg/onload=alert(1)>`

▲ `<video src=x onerror=alert(1);>`

▲ `<audio src=x onerror=alert(1);>`

▲ `data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4=`

1.2.2.1 Simple Rules to Bypass WAFs

Blind SQL Injection

Instead of using:

▼ ' or $1=1$

The best choice is:

▲ ' or 6=6

▲ ' or 0x47=0x47

▲ or char(32)='i'

▲ or 6 is not null

DO

1.2.2.1 Simple Rules to Bypass WAFs

SQL Injection

Instead of using:

▼ **UNION SELECT**

The best choice is:

▲ **UNION ALL SELECT**

DO



```
24 def skillsize(experiment, observations = [], control = null)
25   # observations - all the observations, in ascending order
26   # control      - the control observation
27   #
28   # skillsize(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36   end
37
38 # Returns the experiment's context
39 def context
40   @experiment.context
41 end
42
43 # Returns the name of the experiment
44 def experiment_name
45   @experiment.name
46 end
47
48 # Returns whether the result is a match between all names
49 def matched?
50   # ...
51   @experiment.result[0]
52 end
```

1.2.2.1 Simple Rules to Bypass WAFs

Directory Traversal

Instead of using:

▼ **/etc/passwd**

DO



The best choice is:

▲ **/too/../../etc/far/../../passwd**

▲ **/etc//passwd**

▲ **/etc/ignore/../../passwd**

▲ **/etc/passwd.....**

1.2.2.1 Simple Rules to Bypass WAFs

Web Shell

Instead of using:

- ▲ **c99.php**
- ▲ **r57.php**
- ▲ **shell.aspx**
- ▲ **cmd.jsp**
- ▲ **CmdAsp.asp**

DO



The best choice is:

- ▲ **augh.php**

1.2.2.2 WAF Detection and Fingerprinting

Usually WAFs work in **passive** mode, **reactive** mode, or sometimes both. It depends on the period at which they are installed. For example, once deployed, they can be in passive mode, reducing the number of false positives and avoiding blocking the application; however, once in production, most are reactive.

Before testing a web application, it is extremely useful to know if there is a WAF on the other side and what kind it is. WAF systems leave several footprints of their presence, which allow us to detect which WAF is in place. Let's check out some techniques.

1.2.2.2 WAF Detection and Fingerprinting

Cookie Values

Some WAF systems reveal their presence through cookies. They release their own cookie during the HTTP communications.

[illegible]

1.2.2.2 WAF Detection and Fingerprinting

Cookie Values:

Citrix Netscaler uses some different cookies in the HTTP responses like **ns_af** or **citrix_ns_id** or **NSC_**

F5 BIG-IP ASM (Application Security Manager) uses cookies starting with **TS** and followed with a string that respect the following regex:

```
^TS[a-zA-Z0-9]{3,6}
```

1.2.2.2 WAF Detection and Fingerprinting

Cookie Values:

Barracuda uses two cookies **barra_counter_session** and **BNI__BARRACUDA_LB_COOKIE**.



```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 8543
Content-Type: text/html
Expires: Tue, 08 Apr 2014 08:56:45 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: Tue, 08 Apr 2014 08:57:44 GMT
Set-Cookie: BNI__BARRACUDA_LB_COOKIE=000000000000000000000000c40000a0000bb20; Path=/
```

1.2.2.2 WAF Detection and Fingerprinting

Header Rewrite

Some WAFs rewrite the HTTP headers. Usually these modify the `server` header to deceive the attackers.

For example, they either rewrite the header if the request is malicious or, depending on the malicious request, remove the HTTP server header in the response.

1.2.2.2 WAF Detection and Fingerprinting

Header Rewrite: Example – Rewrite Server Header

HTTP response for **non-hostile** request

```
HTTP/1.1 200 OK
Date: Mon, 7 Apr 2014 10:10:50 GMT
Server: Apache (Unix)
Content-Type: text/html
Content-Length: 2506
```

HTTP response for **hostile** request

```
HTTP/1.1 404 Not Found
Date: Mon, 7 Apr 2014 10:11:06 GMT
Server: Netscape-Enterprise/6.1
Content-Type: text/html;
Content-Length: 158
```

1.2.2.2 WAF Detection and Fingerprinting

HTTP Response Code

Some WAFs modify the HTTP response codes if the request is hostile; for example:

mod_security >

406 Not Acceptable

AQTRONIX WebKnight >

999 No Hacking

1.2.2.2 WAF Detection and Fingerprinting

HTTP Response Body

It is also possible to detect the presence of the WAF plainly in the response body.

[illegible]

1.2.2.2 WAF Detection and Fingerprinting

HTTP Response Body: mod_security



```
HTTP/1.1 406 Not Acceptable
Date: Mon, 7 Apr 2014 11:10:50 GMT
Server: Apache
Content-Length: 226
Keep-Alive: timeout=10, max=30
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
<head><title>Not Acceptable!</title></head><body><h1>Not Acceptable!</h1>
<p>An appropriate representation of the requested resource could not be found on this server.
This error was generated by Mod_Security.</p></body></html>
```

1.2.2.2 WAF Detection and Fingerprinting

HTTP Response Body: AQTRONIX WebKnight

WebKnight Application Firewall Alert

Your request triggered an alert! If you feel that you have received this page in error, please contact the administrator of this web site.

What is WebKnight?

AQTRONIX WebKnight is an application firewall for web servers and is released under the GNU General Public License. It is an ISAPI filter for securing web servers by blocking certain requests. If an alert is triggered WebKnight will take over and protect the web server.

For more information on WebKnight:
<http://www.aqtronix.com/WebKnight/>

AQTRONIX WebKnight

```
def __init__(self, experiment, observations = [], control = null)
@experiment = experiment
@observations = observations
@control = control
@candidates = observations - @control
evaluate_candidates

freeze
end

# Returns the experiment's context
def context
  experiment.context
end

# Returns the name of the experiment
def experiment_name
  experiment.name
end

# Returns whether the result is a match between an experiment and a control
def matches
  @experiment.result == @control
end
```

1.2.2.2 WAF Detection and Fingerprinting

HTTP Response Body: dotDefender

03-Feb-14

dotDefender Blocked Your Request

Please contact the site administrator, and provide the following
Reference ID:

FAB8-125C-151A-2F07

1.2.2.2 WAF Detection and Fingerprinting

Close Connection

An interesting feature supported by some WAFs is **close connection**.

It is useful in dropping the connection in the case the WAF detects a malicious request.

```
10 def initialize_experiment(experiment_name, observations, control)
11   @experiment_name = experiment_name
12   @observations = observations
13   @control = control
14   @candidates = []
15   evaluate_candidates
16 end
17
18 def initialize_experiment(experiment_name, observations = [], control = nil)
19   @experiment_name = experiment_name
20   @observations = observations
21   @control = control
22   @candidates = []
23   evaluate_candidates
24 end
25
26 def freeze
27   @candidates.freeze
28 end
29
30 def WAF?
31   # WAF? the experiment's context
32   @context = WAF?
33 end
34
35 def WAF?
36   # WAF? the name of the experiment
37   @experiment_name
38 end
39
40 def WAF?
41   # WAF? the result a match between an observation and the control
42   @match = nil
43 end
44
45 def WAF?
46   # WAF? the result a match between an observation and the control
47   @match = nil
48 end
49
50 def WAF?
51   # WAF? the result a match between an observation and the control
52   @match = nil
53 end
54
55 def WAF?
56   # WAF? the result a match between an observation and the control
57   @match = nil
58 end
59
60 def WAF?
61   # WAF? the result a match between an observation and the control
62   @match = nil
63 end
64
65 def WAF?
66   # WAF? the result a match between an observation and the control
67   @match = nil
68 end
69
70 def WAF?
71   # WAF? the result a match between an observation and the control
72   @match = nil
73 end
74
75 def WAF?
76   # WAF? the result a match between an observation and the control
77   @match = nil
78 end
79
80 def WAF?
81   # WAF? the result a match between an observation and the control
82   @match = nil
83 end
84
85 def WAF?
86   # WAF? the result a match between an observation and the control
87   @match = nil
88 end
89
90 def WAF?
91   # WAF? the result a match between an observation and the control
92   @match = nil
93 end
94
95 def WAF?
96   # WAF? the result a match between an observation and the control
97   @match = nil
98 end
99
100 def WAF?
101   # WAF? the result a match between an observation and the control
102   @match = nil
103 end
```



1.2.2.2 WAF Detection and Fingerprinting

Close Connection: mod_security

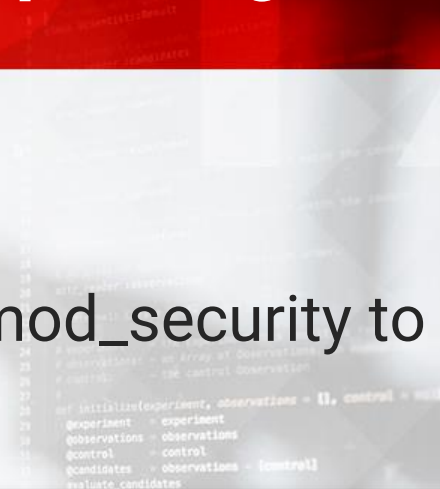
Here is a possible implementation with mod_security to detect a brute force attack:



```
SecAction phase:1,id:109,initcol:ip=%{REMOTE_ADDR},nolog

SecRule ARGS:login "!^$"
"nolog,phase:1,id:110,setvar:ip.auth_attempt=+1,deprecatevar:ip.auth_attempt=20/120"

SecRule IP:AUTH_ATTEMPT "@gt 25"
"log,drop,phase:1,id:111,msg:'Possible Brute Force Attack'"
```

A blurred screenshot of a code editor showing various lines of code, likely related to the mod_security configuration.

1.2.2.2 WAF Detection and Fingerprinting

Many penetration testing tools have features to detect the presence of a WAF. These features are both used as a first step to understand how to craft payloads and if it is needed.

An example would be to obfuscate the attack vector or use a specific bypass.

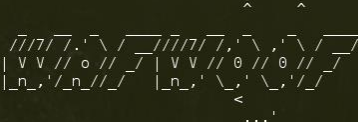
```
24 def initialize(experiment, observations = [], control = nil)
25   @experiment = experiment
26   @observations = observations
27   @control = control
28   @candidates = observations + [control]
29   evaluate_candidates
30
31   freeze
32 end
33
34 # Returns the experiment's context
35 def context
36   experiment.context
37 end
38
39 # Returns the experiment's name
40 def experiment_name
41   experiment.name
42 end
43
44 # Returns the result a match between an observation and the control
45 def match?
46   @observations[0] == @control
47 end
48
49 # Returns the result a match between an observation and the control
50 def result
51   @observations[0] == @control
52 end
```

1.2.2.2 WAF Detection and Fingerprinting

The most well-known tool made by Sandro Gauci and Wendel G. Henrique is called wafw00f.

Wafw00f is a tool written in python that can detect up to 20 different WAF products.

```
ohpe@kali:~$ wafw00f -l
```



```
WAFW00F - Web Application Firewall Detection Tool

By Sandro Gauci && Wendel G. Henrique

Can test for these WAFs:

Profense
NetContinuum
Incapsula
Barracuda
HyperGuard
BinarySec
Teros
F5 Trafficshield
F5 ASM
Airlock
Citrix NetScaler
ModSecurity
IBM Web Application Security
IBM DataPower
DenyALL
dotDefender
webApp.secure
BIG-IP
URLScan
WebKnight
SecureIIS
Imperva
ISA Server
```


1.2.2.2 WAF Detection and Fingerprinting

The techniques used to detect a WAF are similar to those we have seen previously:

1. Cookies
2. Server Cloaking
3. Response Codes
4. Drop Action
5. Pre-Built-In Rules

1.2.2.2 WAF Detection and Fingerprinting

Scanning a website with wafw00f is very simple, and the following image confirms it:

```
ohpe@kali:~$ wafw00f www.imperva.com
```

^ ^
//_/_/_/'_/_/_/_/_/_/_/_/_/_/_/_/_/_/_'
| V V // o // | V V // 0 // 0 //
| _n_, '/ _n_// | _n_, '\ , '\ , '/' //

<
...'

WAFW00F - Web Application Firewall Detection Tool
By Sandro Gauci & Wendel G. Henrique

```
Checking http://www.imperva.com
The site http://www.imperva.com is behind a Incapsula
Number of requests: 2
```

1.2.2.2 WAF Detection and Fingerprinting

As an addition to wafw00f you might want to use Nmap. It contains a script that tries to detect the presence of a web application firewall, its type and version.

The script file is <http-waf-fingerprint> and is authored by Hani Benhabiles.

```
def skillsize(experiment, observations = [], control = null)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - !metrell
  evaluate_candidates

  freeze
end

# WAF: the experiment's context
# ...

# Define the name of the experiment
def experiment_name
  experiment.name
end

# Define the result a match between an ...
def match?
  ...
end

# ... result
result = 1.1
```

1.2.2.2 WAF Detection and Fingerprinting

Scanning a website with nmap is as simple as running wafw00f. We just require the script name to be in the command:

```
ohpe@kali:/$ nmap --script=http-waf-fingerprint www.imperva.com -p 80
```

```
Starting Nmap 6.40 ( http://nmap.org ) at 2014-04-08 14:20 CEST
Nmap scan report for www.imperva.com (185.11.125.104)
Host is up (0.045s latency).
PORT      STATE SERVICE
80/tcp    open  http
| http-waf-fingerprint:
|   Detected WAF
|_   Incapsula WAF

Nmap done: 1 IP address (1 host up) scanned in 11.57 seconds
```

1.2.2.2 WAF Detection and Fingerprinting

Another interesting resource is [imperva-detect](#) by Lamar Spells. This utility is 100% focused on the detection of an Imperva WAF and it runs 6 tests, one baseline and five additional:

```
# Test 0 - Baseline to establish expected behavior
# Test 1 - "Web Leech" blocking
# Test 2 - "E-mail Robot" blocking
# Test 3 - BlueCoat Proxy Manipulation blocking
# Test 4 - Web Worm blocking
# Test 5 - XSS blocking
```


1.2.2.2 WAF Detection and Fingerprinting

The following image is an example of how to run imperva-detect test scripts:

```
ohpe@kali:/$ imperva-detect.sh blog.imperva.com

--- Testing [blog.imperva.com] for presence of application firewall ---

Test 0 - Good User Agent...
  -- HTTP Return Code = 200
  -- Content Size Downloaded = 79749
Test 1 - Web Leech User Agent...
  -- HTTP Return Code = 200 & downloaded content size is the same -- application firewall not detected
Test 2 - E-mail Collector Robot User Agent Blocking...
  -- HTTP Return Code = 200 & downloaded content size is the same -- application firewall not detected
Test 3 - BlueCoat Proxy Manipulation Blocking...
  -- HTTP Return Code = 302 -- expected 404 -- application firewall possibly present
Test 4 - Web Worm Blocking...
  -- HTTP Return Code [404] encountered - application firewall possibly present
Test 5 - XSS Blocking...
  -- HTTP Return Code = 302 -- while checking XSS blocking

--- Tests Finished on [blog.imperva.com] -- 3 out of 5 tests indicate Imperva application firewall present ---
```

1.2.3 Client-Side Filters

Web Application Firewalls and libraries are filtering solutions used to block web attacks, **server-side** at the heart of web applications. Over the years, this has become the "classic" and consolidated approach.

However, in the last ten years another approach has arisen. The concept is to block web attacks **client-side** within web browsers. These browsers are the primary mean used to address attacks.

1.2.3 Client-Side Filters

The goal of client-side defenses is to protect users against vulnerabilities in web applications. Of course this approach is not simple, and defenses need to be generic enough to always be enabled. If they are not, they can become a blocker for the browsers themselves and to their respective users.

From an attacker's point of view, we want to understand these mechanisms and how to bypass them. Our aim is the target users who would otherwise be protected.

1.2.3.1 Browser Add-ons

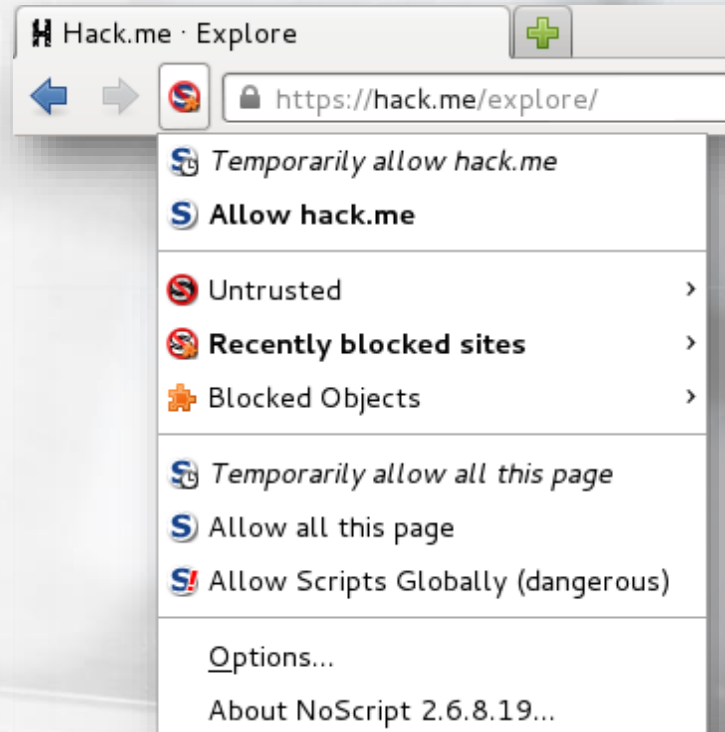
The first browser protection began in the open source community.

The pioneer of the first valid solution was Giorgio Maone, in late 2005, with the **NoScript Security Suite** extension for Firefox.



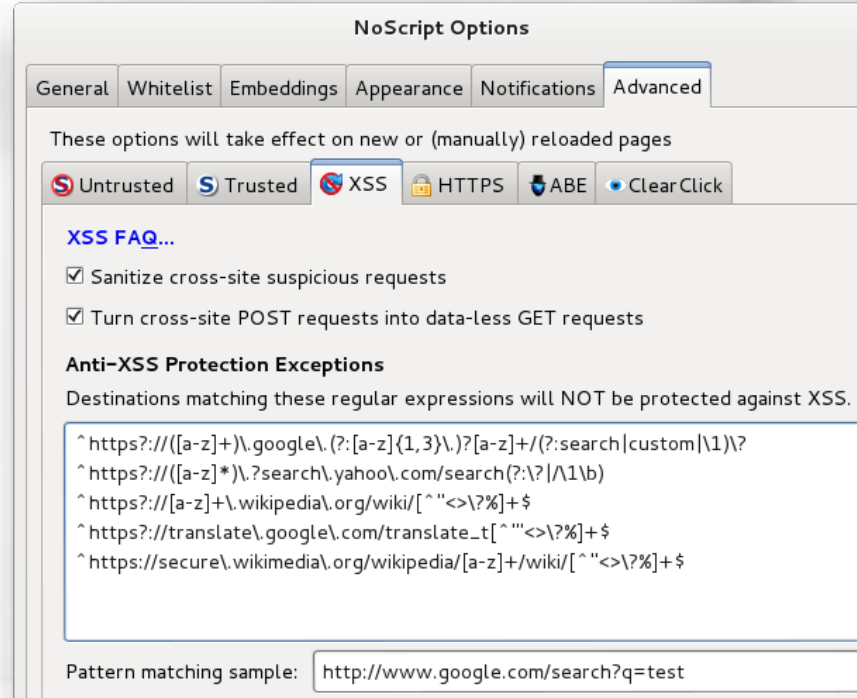
1.2.3.1 Browser Add-ons

NoScript is a whitelist-based security tool that basically disables all the executable web content (JavaScript, Java, Flash, Silverlight, ...) and lets the user choose which sites are "trusted", thus allowing the use of these technologies.



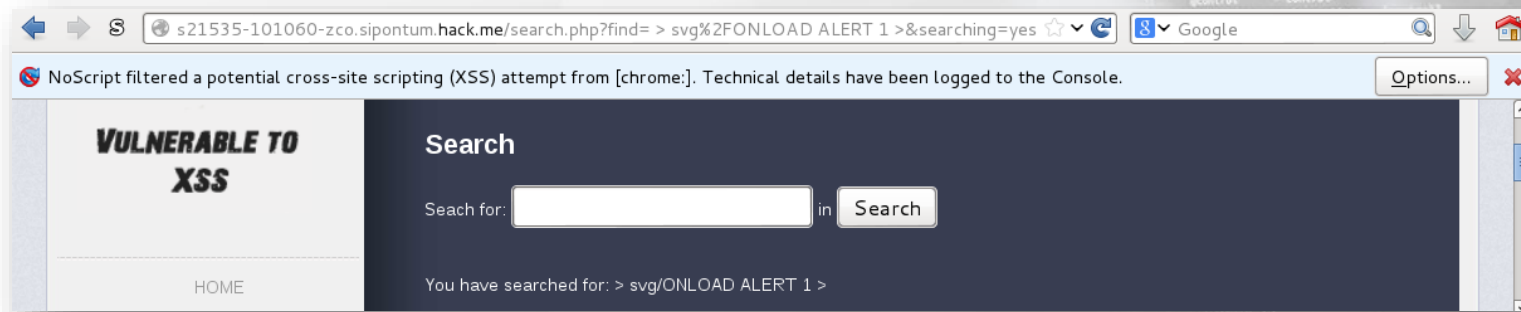
1.2.3.1 Browser Add-ons

NoScript is easy enough to use; however, the strongest point of this extension is the extensive list of security features supported.



1.2.3.1 Browser Add-ons

Among the features, the strong and powerful **anti-XSS protection** is probably one of the most effective browser-based solutions to prevent targeted malicious Web attacks.



1.2.3.2 Native Browser Filters

History

The first attempt at blocking malicious requests "*natively*" (i.e. internally in the browser), was made by Microsoft and introduced in Internet Explorer 8 as **XSS Filter**.

This filter attempts to block reflected XSS attacks by applying regular expressions to response data.

1.2.3.2 Native Browser Filters

History

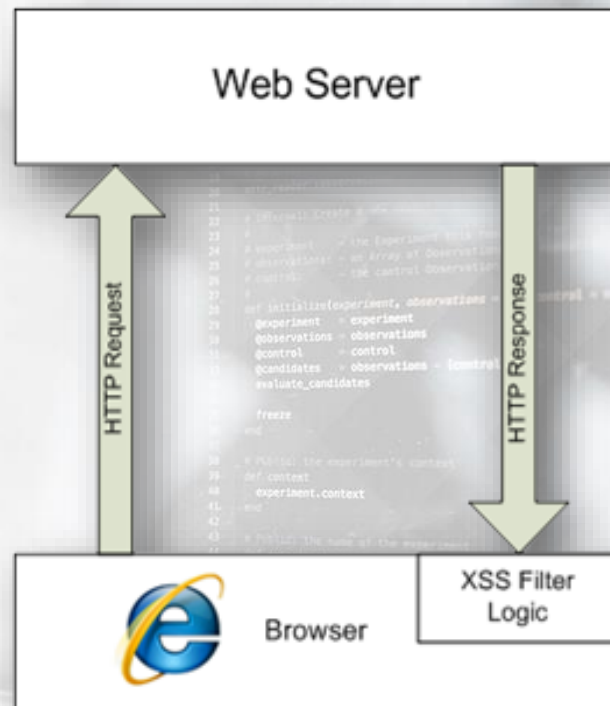
After Microsoft, Google Chrome introduced their own cross-site scripting filters, **XSS Auditor**.

This filter is slightly different from IE's XSS Filter and NoScript. Instead of being layered on top of the browser, it is integrated into WebKit/Blink, which are the rendering engines that support XSS Auditor.

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer)


The architecture and implementation of XSS Filter in Internet Explorer is explained in this [blog post](#).



1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer)

The XSS Filter rules are **hardcoded** in the `c:\windows\system32\mshtml.dll` library. We have multiple ways to inspect them using the following:

- Hex editors like WinHex, or Notepad++ with TextFX plugin
- IDAPro
- MS-DOS commands!  **Faster solution!**

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer)

The command is:

```
findstr /C:"sc{r}" \WINDOWS\SYSTEM32\mshtml.dll | find "{"
```

If you want a more "human-readable" version, then export the result to a file and use a text editor to read the content.

```
findstr /C:"sc{r}" \WINDOWS\SYSTEM32\mshtml.dll | find "{" > savepath
```

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer)

Here are few extracted rules in Internet Explorer 11:

```
{<EM{B}ED[ /+\t].*?((src)|(type)).*?=}  
{[ /+\t\"'`]{0}n\\c\\c\\c+?[ +\t]*?=.  
{[i]?f{r}ame.*?[ /+\t]*?src[ /+\t]*=}  
{<fo{r}m.*?>}  
{<sc{r}ipt.*?[ /+\t]*?((src)|(xlink:href)|(href))[ /+\t]*=}  
{<BA{S}E[ /+\t].*?href[ /+\t]*=}  
{<LI{N}K[ /+\t].*?href[ /+\t]*=}  
{<ME{T}A[ /+\t].*?http-equiv[ /+\t]*=}
```

```
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  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
21
```

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer): Filter Examination

This filter detects the string **javascript:**

```
((j|(&#x?0*((74)|(4A)|(106)|(6A));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(a|(&#x?0*((65)|(41)|(97)|(61));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(v|(&#x?0*((86)|(56)|(118)|(76));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(a|(&#x?0*((65)|(41)|(97)|(61));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(s|(&#x?0*((83)|(53)|(115)|(73));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(c|(&#x?0*((67)|(43)|(99)|(63));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*{(r|(&#x?0*((82)|(52)|(114)|(72));?))}([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(i|(&#x?0*((73)|(49)|(105)|(69));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(p|(&#x?0*((80)|(50)|(112)|(70));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(t|(&#x?0*((84)|(54)|(116)|(74));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(:|(&(#x?0*((58)|(3A));?)|(\colon;))))}.
```


1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer): Filter Examination

This filter detects the string **vbscript**:

```
{(v|(&#x?0*((86)|(56)|(118)|(76));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;)))*)*(b|(&#x?0*((66)|(42)|(98)|(62));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(s|(&#x?0*((83)|(53)|(115)|(73));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(c|(&#x?0*((67)|(43)|(99)|(63));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*{(r|(&#x?0*((82)|(52)|(114)|(72));?))}([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(i|(&#x?0*((73)|(49)|(105)|(69));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(p|(&#x?0*((80)|(50)|(112)|(70));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(t|(&#x?0*((84)|(54)|(116)|(74));?))([\t]|(&(#x?0*(9|(13)|(10)|A|D);?)|(\tab;)|(\newline;))))*(:(|(&(#x?0*((58)|(3A));?)|(\colon;))))}.
```

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

In the previous slides, you probably noticed the **red** characters highlighted between curly braces. They are also known as '**neutering**' characters.

The use of neutering characters is the approach that the IE team decided to use to neutralize detected attacks, which is a kind of trade-off between usability and effectiveness.

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

Basically, once a malicious injection is detected, the XSS Filter modifies the evil part of the payload by adding the # (*pound*) character in place of the neuter character, defined in the rules.

evil $\xrightarrow{\text{ev\{i\}l}}$ ev#l

Let's look at some examples.

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer): Neutering in Action

The XSS attack:

`<svg/onload=alert(1)>`

is transformed to:

`<svg/#nload=alert(1)>`

`{[/+\t\"'`]{0}n\c\c\c+?[+\t]*?=.}`

Filter rule

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer): Neutering in Action

The XSS attack:

`<isindex/onmouseover=alert(1)>`

is transformed to:

`<is#index/#nmouseover=alert(1)>`

`{<is{i}ndex[/+\t>]}`

`{[/+\t\"'`]{o}n\c\c\c+?[+\t]*?=.}`

Filter rules

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

The XSS Filter has a few rules, just 25, but are well constructed and difficult to attack.

Over the years, several bypasses have been discovered; however, the latest versions 'seem' stronger than past versions (unless you have a 0-day 😊).

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

XSS Filter is enabled by default in the Internet, Trusted, and Restricted security zones, but an interesting feature was introduced to disable the filter.

The main reason was because some sites may depend on the reflected values that the filter searches for.

1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

Web sites that chose to opt-out of this protection can use the HTTP response header:

X-XSS-Protection: 0

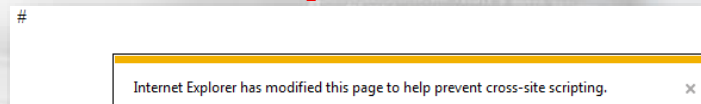
1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

Later, the IE team added support to a new token in the X-XSS-Protection header:

X-XSS-Protection: 1; **mode=block**

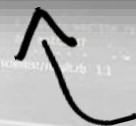
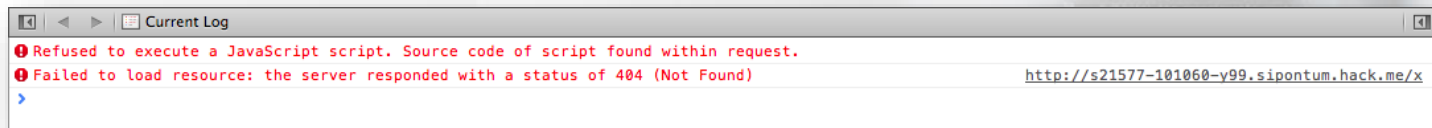
With this token, if a potential reflected XSS attack is detected, the browser, rather than attempting to sanitize the page, will render a simple #. Here is a **simple test**.



1.2.3.2 Native Browser Filters

XSS Filter (Internet Explorer):

Even if the x-XSS-Protection header was initially introduced by Internet Explorer, today other browsers based on WebKit and Blink support it.



Safari

1.2.3.2 Native Browser Filters

XSSAuditor (WebKit/Blink)

In the footsteps of Internet Explorer, some researchers developed their own set of client-side XSS filters, also known as **XSSAuditor**.

The implementation is only for the **Blink/WebKit** rendering engines. This is enabled by default in browsers such as Google Chrome, Opera and Safari.

1.2.3.2 Native Browser Filters

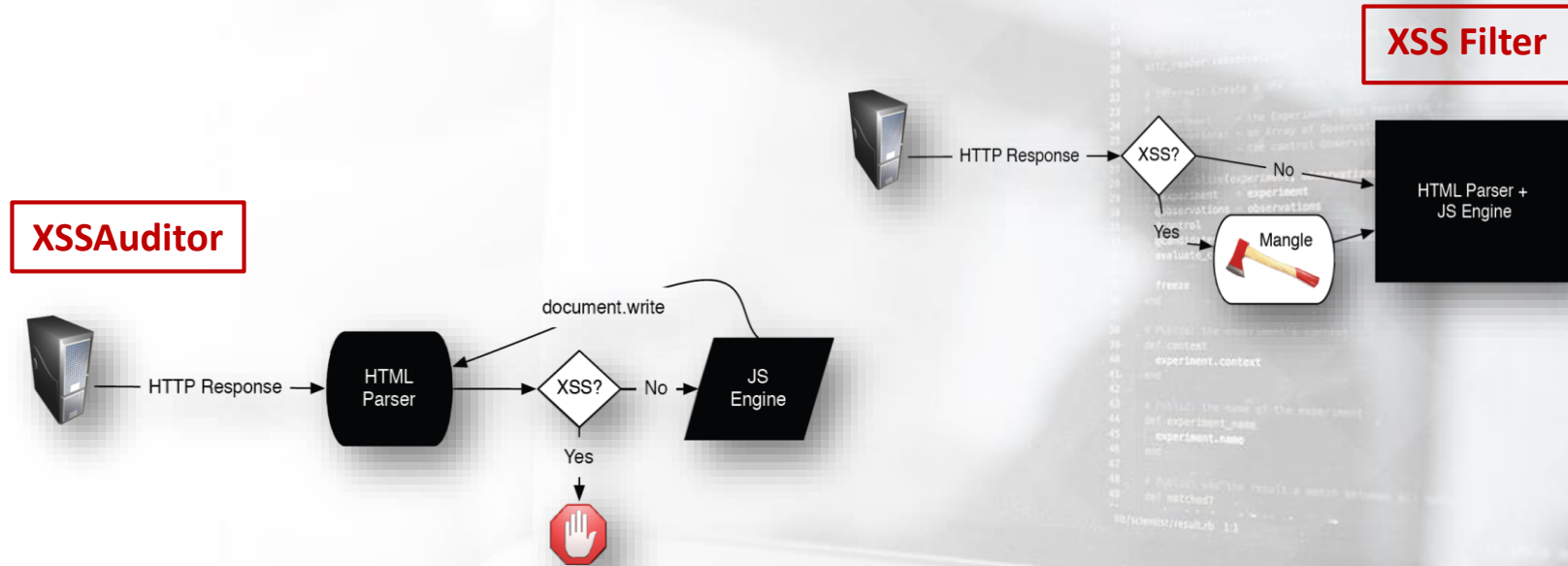
XSSAuditor (WebKit/Blink)

Despite the IE XSS Filter, XSSAuditor adopts a different approach to the problem. As the authors claims, the new filter design is both effective and highly precise. To do this, they placed XSSAuditor in between the HTML Parser and JS engine.

The image on the following slide will clarify the differences.

1.2.3.2 Native Browser Filters

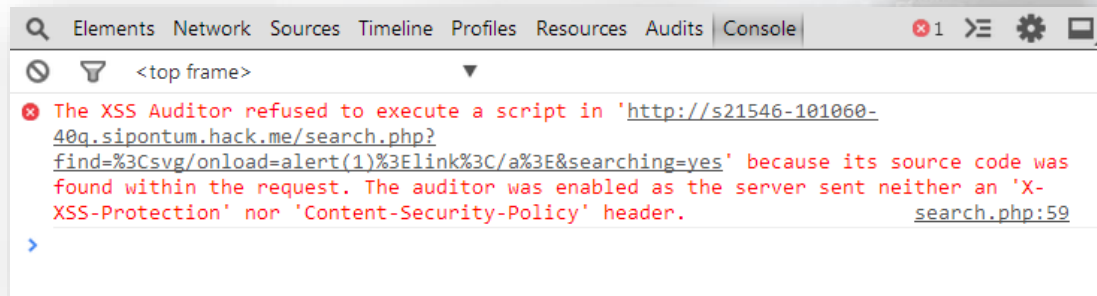
XSSAuditor (WebKit/Blink): XSS Filter vs XSSAuditor



1.2.3.2 Native Browser Filters

XSSAuditor (WebKit/Blink)

The filter analyzes both the inbound requests and the outbound. If, in the parsed HTML data, it finds executable code within the response, then it stops the script and generates a console alert similar to the following:



1.2.3.2 Native Browser Filters

XSSAuditor (WebKit/Blink)

Over the years, even with XSS Auditor, security researchers found multiple bypasses...Oh yes, they did!

A simple search on google about xss Auditor returns more information on bypasses than on the filter itself.

WAP

References



References

[RFC 3986](http://tools.ietf.org/html/rfc3986#section-2.1)

<http://tools.ietf.org/html/rfc3986#section-2.1>

[\(Please\) Stop Using Unsafe Characters in URLs](http://perishablepress.com/stop-using-unsafe-characters-in-urls/): Character Encoding Chart

<http://perishablepress.com/stop-using-unsafe-characters-in-urls/>

[RFC 2616](https://tools.ietf.org/html/rfc2616)

<https://tools.ietf.org/html/rfc2616>

[ISO/IEC 8859-1](http://en.wikipedia.org/wiki/ISO/IEC_8859-1)

http://en.wikipedia.org/wiki/ISO/IEC_8859-1



References

PHP header()

<http://www.php.net/header>

HttpResponse Class

<http://msdn.microsoft.com/en-us/library/system.web.httpresponse>

HTML Document Representation: 5.3 Character References

<http://www.w3.org/TR/1998/REC-html40-19980424/charset.html#h-5.3>

HTML Standard: 12.1.4 Character References

<http://www.w3.org/TR/html5/single-page.html#character-references>



References

Character entity references in HTML

http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references#Character_entity_references_in_HTML

Reddit

<http://www.reddit.com/>

TinyURL

<http://tinyurl.com/>

base_convert

<http://www.php.net/manual/en/function.base-convert.php>



References

[Base64: Implementations and History](http://en.wikipedia.org/wiki/Base64#Implementations_and_history)

http://en.wikipedia.org/wiki/Base64#Implementations_and_history

[base64_encode](https://www.php.net/base64_encode)

https://www.php.net/base64_encode

[base64_decode](https://www.php.net/base64_decode)

https://www.php.net/base64_decode

[WindowOrWorkerGlobalScope.btoa\(\)](https://developer.mozilla.org/en-US/docs/Web/API/Window.btoa)

<https://developer.mozilla.org/en-US/docs/Web/API/Window.btoa>



References

[The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets \(No Excuses!\)](http://www.joelonsoftware.com/articles/Unicode.html)

<http://www.joelonsoftware.com/articles/Unicode.html>

[Homoglyph](http://en.wikipedia.org/wiki/Homoglyph)

<http://en.wikipedia.org/wiki/Homoglyph>

[Unicode® Technical Standard #39 - UNICODE SECURITY MECHANISMS](http://www.unicode.org/reports/tr39/)

<http://www.unicode.org/reports/tr39/>

[Unicode Utilities: Confusables](http://unicode.org/cldr/utility/confusables.jsp)

<http://unicode.org/cldr/utility/confusables.jsp>



References

Homoglyph Attack Generator

<http://www.irongeek.com/homoglyph-attack-generator.php>

Out of Character: Use of Punycode and Homoglyph Attacks to Obfuscate URLs for Phishing

<http://www.irongeek.com/i.php?page=security/out-of-character-use-of-punycode-and-homoglyph-attacks-to-obfuscate-urls-for-phishing>

Creative usernames and Spotify account hijacking

<http://labs.spotify.com/2013/06/18/creative-usernames/>

Unicode Utilities: Description and Index

<http://unicode.org/cldr/utility/>



References



[Codepoints](http://codepoints.net/)

<http://codepoints.net/>



[Transformation tools for Unicode text](http://txtn.us/)

<http://txtn.us/>



[Unicode Text Converter](http://www.panix.com/~eli/unicode/convert.cgi)

<http://www.panix.com/~eli/unicode/convert.cgi>



[OWASP Enterprise Security API](https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API)

https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API



References



[ModSecurity](http://www.modsecurity.org/)

<http://www.modsecurity.org/>



[Deterministic finite automaton](http://en.wikipedia.org/wiki/Deterministic_finite_automaton)

http://en.wikipedia.org/wiki/Deterministic_finite_automaton



[Nondeterministic finite automaton](http://en.wikipedia.org/wiki/Nondeterministic_finite_automaton)

http://en.wikipedia.org/wiki/Nondeterministic_finite_automaton



[Control character](http://en.wikipedia.org/wiki/Control_character)

http://en.wikipedia.org/wiki/Control_character



References

[GitHub: SpiderLabs / ModSecurity Documentation](https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#drop)

<https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#drop>

[GitHub: EnableSecurity / wafw00f](https://code.google.com/p/waffit/)

<https://code.google.com/p/waffit/>

[File http-waf-fingerprint](http://nmap.org/nsedoc/scripts/http-waf-fingerprint.html)

<http://nmap.org/nsedoc/scripts/http-waf-fingerprint.html>

[imperva-detect](https://code.google.com/p/imperva-detect/)

<https://code.google.com/p/imperva-detect/>



References



NoScript Security Suite

<https://addons.mozilla.org/en-US/firefox/addon/noscript/>



NoScript: Anti-XSS protection

<http://noscript.net/features#xss>



IE8 Security Part IV: The XSS Filter

<http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx>



IE 8 XSS Filter Architecture / Implementation

<http://blogs.technet.com/b/srd/archive/2008/08/18/ie-8-xss-filter-architecture-implementation.aspx>



References

Event 1046 - Cross-Site Scripting Filter: Remediation

[http://msdn.microsoft.com/en-us/library/dd565647\(v=vs.85\).aspx#remediation](http://msdn.microsoft.com/en-us/library/dd565647(v=vs.85).aspx#remediation)

Regular Expressions Considered Harmful in Client-Side XSS Filters

<http://www.adambarth.com/papers/2010/bates-barth-jackson.pdf>

The META element

<https://www.w3.org/TR/html401/struct/global.html#h-7.4.4.2>

Unicode Security Guide

<http://websec.github.io/unicode-security-guide/>



References

[Comparison of regular expression engines](https://en.wikipedia.org/wiki/Comparison_of_regular_expression_engines)

https://en.wikipedia.org/wiki/Comparison_of_regular_expression_engines

[Regular Expressions Reference Table of Contents](https://www.regular-expressions.info/refflavors.html)

<https://www.regular-expressions.info/refflavors.html>

[Base 36 as senary compression](http://tinyurl.com/jfvqr)

<http://tinyurl.com/jfvqr>

