



UMCS

**UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE**

Wydział Matematyki, Fizyki i Informatyki

Kierunek: Informatyka

Piotr Jasina

nr albumu: 279183

Identyfikacja inteligentnych kontraktów w sieci Ethereum

Ethereum smart contracts identification

Praca licencjacka

napisana w Zakładzie Cyberbezpieczeństwa

pod kierunkiem dr. Damiana Rusinka

Lublin rok 2019

Spis treści

Wstęp	5
1 Ethereum	7
1.1 Historia	7
1.2 Opis platformy	7
1.3 Ethereum Virtual Machine	7
1.4 Inteligentne kontrakty	7
2 Solidity	9
2.1 Sygnatura funkcji	9
2.2 Selektor funkcji	9
2.3 Generowanie akcesorów podczas kompilacji	9
3 Projekt Aplikacji	11
3.1 Funkcjonalność	11
3.1.1 Identyfikacja inteligentnych kontraktów	12
3.1.2 Dodanie kodu źródłowego kontraktu do aplikacji	14
3.1.3 Interfejs programistyczny aplikacji	15
3.2 Architektura	20
3.2.1 Wyszukiwanie sygnatur funkcji w kodzie źródłowym	20
3.2.2 Wyszukiwanie selektorów funkcji w kodzie bajtowym	20

Wstęp

...

Rozdział 1

Ethereum

1.1 Historia

Literatura: [2, ?]. TODO

1.2 Opis platformy

Literatura: [2, ?]. TODO

1.3 Ethereum Virtual Machine

Literatura: [2, ?]. TODO

1.4 Inteligentne kontrakty

Literatura: [2, ?]. TODO

Rozdział 2

Solidity

2.1 Sygnatura funkcji

Literatura: [2, ?]. TODO

2.2 Selektor funkcji

Literatura: [2, ?]. TODO

2.3 Generowanie akcesorów podczas kompilacji

Literatura: [2, ?]. TODO

Rozdział 3

Projekt Aplikacji

Celem mojej pracy licencjackiej było stworzenie aplikacji internetowej umożliwiającej identyfikację inteligentnych kontraktów wykorzystywanych w sieci Ethereum. Dzięki aplikacji użytkownik po wprowadzeniu na stronie kodu bajtowego kontraktu jest w stanie otrzymać najbardziej prawdopodobną implementację kontraktu napisana w języku Solidity bazując na bazie danych aplikacji.

Poniżej zostało opisane działanie aplikacji wraz ze szczegółowym opisem funkcjonalności, architektury oraz wykorzystanych technologii.

3.1 Funkcjonalność

Podczas korzystania z aplikacji użytkownik ma dostępne trzy funkcjonalności: identyfikację inteligentnych kontraktu, wprowadzanie plików źródłowych kontraktów do aplikacji oraz dokumentacje API aplikacji. Na stronie głównej poniżej menu znajduje się opis aplikacji wraz z aktualna liczba kodów źródłowych znajdujących się w bazie danych aplikacji oraz podstawowymi definicjami związanymi z aplikacją.

3.1.1 Identyfikacja intelligentnych kontraktów

Pierwsza opcją dostępną w aplikacji jest identyfikacja intelligentnych kontraktów. Identyfikacje kontraktów rozpocząć będąc na stronie głównej lub za pomocą podstrony do której można się dostać klikając przycisk **Identify bytecode** znajdujący się w menu. W przypadku wprowadzenia kodu bajtowego na stronie głównej zostaniemy przekierowani na podstronę na której pojawią się wyniki identyfikacji kontraktu. Jeśli użytkownik wykorzysta opcje z menu to będzie musiał na podstronie wprowadzić kod bajtowy swojego kontraktu. W obydwu przypadkach, zarówno na stronie głównej, jak i podstronie dedykowanej tej funkcjonalności, kod bajtowy należy wprowadzać w szesnastkowym systemie liczbowym.

Podczas wprowadzania kodu istnieje możliwość wprowadzenia kodu z prefiksem **0x** lub bez tego prefixu. Jeśli użytkownik poda kod z prefiksem to aplikacja podczas przetwarzania tego kodu zignoruje ten prefiks. Takie rozwiązanie zostało zastosowane w celu zapewnienia użytkownikowi większej wygody oraz komfortu w korzystaniu z aplikacji. Użytkownik nie będzie musiał zastanawiać się czy kopując kod bajtowy z dowolnego źródła, jest on z prefiksem czy nie, ponieważ obie opcje są wspierane.

Pomyślne wprowadzone dane wykorzystywane do identyfikacji zatwierdzamy przyciskiem "Submit", a następnie po stronie serwerowej aplikacji rozpoczynany jest proces analizy wprowadzonego kodu bajtowego oraz wyszukiwane są najbardziej prawdopodobne implementacje. W rezultacie jak możemy zobaczyć na rysunku 3.1 użytkownik otrzymuje listę wyszukanych dziesięciu najbardziej prawdopodobnych implementacji posortowanych malejąco według współczynnika dopasowania. Mimo że domyślnie jest wyświetlanych tylko dziesięć najbardziej prawdopodobnych implementacji istnieje też możliwość pobrania wszystkich wyników identyfikacji kontraktu używając przycisku **Get all**. Jak widać na rysunku 3.1, przycisk **Get all** znajduje się pod pierwszą dziesiątką wyników. Po naciśnięciu

przycisku strona zostanie załadowana ponownie oraz zostanie wyświetlona pełna lista identyfikatorów plików wraz ze współczynnikiem dopasowania do kodu bajtowego.

Po naciśnięciu w jedną z wyświetlanych implementacji, użytkownicy w nowej karcie przeglądarki otworzy się specjalna podstrona umożliwiająca podgląd implementacji. Jak widać na rysunku 3.2 kod źródłowy kontraktu jest wyświetlany z podświetleniem składni przygotowanym dla języka Solidity, natomiast po lewej stronie można zobaczyć specjalnie przygotowaną numerację wierszy która ma za zadanie ułatwić poruszanie się po kodzie źródłowym na stronie. Rozwiążanie z numerowaniem linii zostało zaimplementowane w taki sposób, aby podczas kopiowania kodu źródłowego ze strony nie zostawały kopiowane wraz z nim liczby identyfikujące konkretną linie w kodzie.

The screenshot shows the interface of the Ethereum Smart Contract Identifier. At the top, there's a dark header bar with the title "Ethereum Smart Contract Identifier" on the left and two buttons on the right: "Identify bytecode" and "Upload solidity". Below the header is a text input field with placeholder text "Paste your bytecode below if you want to identify your contract" and a sample hex string "0xd0db...". Underneath the input field is a blue "Identify" button. To the right of the input field, there's some small text about the percentage of matching contracts. The main content area is titled "Top ten the most matching files" and contains a table with 10 rows of data. The table has three columns: "ID", "File Hash", and "%". The data is as follows:

ID	File Hash	%
1	0x06c61b8e505d7e4070f9e91bdf8085560e90a133c77db32bd32e686f6a8d52	66.67%
2	0xb25a734b6b296b3bcfafe0d248a3cb8e88b28a791483571a1bcfd3e2116c398	42.86%
3	0x720388d3126d49859644d84d87581b6ece88ad52c62f24e344a22089857bf18	35.29%
4	0x268944d5bffe69a06a2cdceef20d7f1fc14403cce080fcad378006566bc2a849	33.33%
5	0xfd7745e0092c6a03c21c410f24e871dfc2d131115e014675284c58c782dbb04	30.00%
6	0x6922463e7e17e961f6b72ac685524e855ead2150f2e1338f3428fe4112336f61	28.57%
7	0xfb83bfff670ef81c79d749ff76d368ccce094e0fc36787ff806d4ecfb9b01ed1	27.27%
8	0xd2d06108e6977437f4d9e0b2b490001ec18bc601e28a085994311541d29e0b6	26.09%
9	0x9bc154aa3130150f674043e4dc87d9613540a32ca35b9b2c5b9058890c064e3b	25.00%
10	0x1b5985f2cb0c51c03e1bf8ba4e328f56c366621d5e3d0bf70462c430edf3f772	25.00%

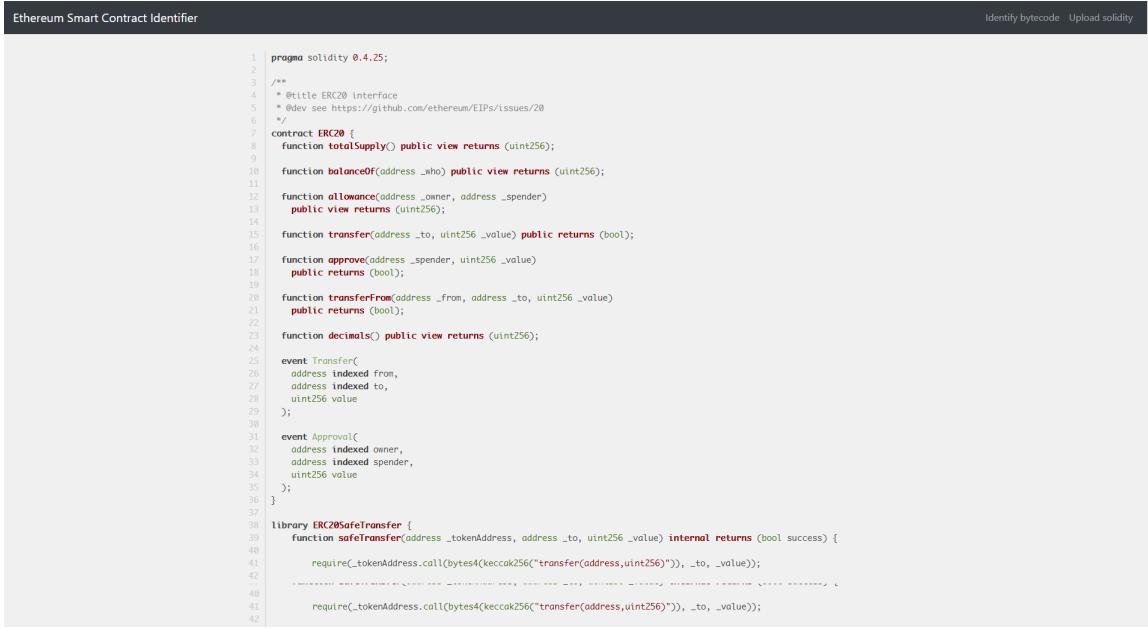
At the bottom of the table is a blue "Get all" button.

Rysunek 3.1: Wynik identyfikacji inteligentnego kontraktu

3.1.2 Dodanie kodu źródłowego kontraktu do aplikacji

Kolejna funkcjonalnością dostępną dla użytkownika jest możliwość dodania własnego kodu źródłowego kontraktu napisanego w języku Solidity. Opcja ta umożliwia użytkownikowi uzupełnienie aktualnej bazy danych o kolejne kody źródłowe inteligentnych kontraktów, w wyniku takiego działania wszyscy pozostali użytkownicy mają większą szansę na precyzyjną identyfikację kontraktu. W celu wykorzystania tej funkcjonalności użytkownik musi użyć przycisku **Upload solidity**, po czym zostanie on przekierowany na podstronę na której będzie mógł wprowadzić kod źródłowy, który zostanie przesłany do serwera aplikacji. Zostały utworzone dwie możliwości wprowadzania kodów źródłowych.

Pierwszą sposobem jest przesyłanie do aplikacji pliku zawierającego implementacje kontraktu napisana w języku Solidity. W tym przypadku użytkownik powinien kliknąć przycisk **Browse**, który umożliwi mu wybranie za pomocą prze-



```

Ethereum Smart Contract Identifier
Identify bytecode Upload solidity

1 pragma solidity 0.4.25;
2
3 /**
4  * @title ERC20 interface
5  * @dev see https://github.com/ethereum/EIPs/issues/20
6 */
7 contract ERC20 {
8     function totalSupply() public view returns (uint256);
9
10    function balanceOf(address _who) public view returns (uint256);
11
12    function allowance(address _owner, address _spender)
13        public view returns (uint256);
14
15    function transfer(address _to, uint256 _value) public returns (bool);
16
17    function approve(address _spender, uint256 _value)
18        public returns (bool);
19
20    function transferFrom(address _from, address _to, uint256 _value)
21        public returns (bool);
22
23    function decimals() public view returns (uint256);
24
25    event Transfer(
26        address indexed from,
27        address indexed to,
28        uint256 value
29    );
30
31    event Approval(
32        address indexed owner,
33        address indexed spender,
34        uint256 value
35    );
36}
37
38 library ERC20SafeTransfer {
39     function safeTransfer(address _tokenAddress, address _to, uint256 _value) internal returns (bool success) {
40         require(_tokenAddress.call(bytes4(keccak256("transfer(address,uint256)")), _to, _value));
41
42         // ... (remaining code for the library)
43
44         require(_tokenAddress.call(bytes4(keccak256("transfer(address,uint256)")), _to, _value));
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
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
307
308
309
309
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
13
```

glądarki internetowej konkretnego pliku znajdującego się na dysku lokalnym, a następnie zatwierdzić go przyciskiem **Upload** znajdującym się obok wcześniej wspomnianego przycisku.

Innym sposobem na przesłanie kodu źródłowego do aplikacji jest wklejenie kodu źródłowego bezpośrednio do formularza znajdującego się po prawej części strony internetowej. Ta opcja została utworzona w celu zapewnienia użytkownikowi większej elastyczności i komfortu w korzystaniu z aplikacji. Przykładowo podczas korzystania z aplikacji, użytkownik może bezpośrednio skopiować kod źródłowy, który jest w dowolnym innym źródle tekstowym np. innej stronie internetowej i wkleić go bezpośrednio do aplikacji bez konieczności tworzenia pliku tymczasowego.

3.1.3 Interfejs programistyczny aplikacji

Trzecią funkcjonalnością aplikacji jest interfejs programistyczny. Umożliwia on tworzenie własnego oprogramowania oraz skryptów, bazując na utworzonej przez zemnie aplikacji, przez innych programistów. Dzięki temu można wykorzystać mechanizmy zaimplementowane w aplikacji w celu rozszerzenia ich w innej aplikacji lub w celu zautomatyzowania niektórych procesów bez wykorzystania GUI (ang. graphical user interface) aplikacji.

W celu skorzystania z interfejsu programistycznego należy utworzyć specjalne żądanie HTTP. Za pomocą interfejsu aplikacji użytkownik ma możliwość dostarczenia na serwer nowego kodu źródłowego, pobrania kodu źródłowego na podstawie na podstawie hasza wcześniej przesłanego pliku źródłowego oraz identyfikacji kontraktu wykorzystując.

Przykładowym zastosowaniem API (ang. Application programming interface) może być utworzenie skryptu umożliwiającego zautomatyzowanie wysyłania kodów źródłowych do aplikacji, bez konieczności korzystania z interfejsu graficz-

nego aplikacji. W tym przypadku użytkownik, który chce przesłać nowy plik do aplikacji musi najpierw przejść proces autoryzacji, natomiast pozostałe funkcje API nie wymagają uwierzytelniania użytkownika przez aplikacje.

Listing 3.1: Przykładowe żądanie HTTP

```
1 GET /api/solidityFiles/sourceCode?fileHash=0
    x06c61b8e505d7a407af9a91bdff8085560e90a133c77ab32bde32e686f6a8d52
    HTTP/1.1
2 Host: contractmy.herokuapp.com
3 Connection: keep-alive
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/72.0.3626.96 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
   image/webp,image/apng,*/*;q=0.8
8 Accept-Encoding: gzip, deflate, br
9 Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
```

Pobieranie informacji o kodzie zródłowym z API

Jeśli użytkownik chce pobrać z aplikacji informacje o konkretnym kodzie źródłowym danego pliku, w tym celu powinien utworzyć żądanie HTTP. W przypadku tej aplikacji do wykonania tej czynności nie jest potrzebne uwierzytelnienie użytkownika. Na listingu 3.1 można zaobserwować przykładowe żądanie HTTP, które jest wykorzystane w celu pobrania informacji o konkretnym kodzie źródłowym.

Użytkownik podczas korzystania z API powinien zwrócić szczególną uwagę na wykorzystywaną metodę w nagłówku HTTP, ponieważ interfejs aplikacji wspiera tylko wybrane metody. W tym przypadku w pierwszej linii na listingu 3.1 widać, że używaną metodą jest **GET**, która jest jedną z kilku dostępnych metod w

standardzie *HTTP/1.1*, istnieją też między innymi metody: *POST*, *HEAD*, *PUT* czy *DELETE*. Metody HTTP zostały opisane szczegółowo w sekcji 9.2 dokumentu RFC2616 [1].

Po metodzie GET, jak widać na listingu 3.1, znajduje się adres pod którym zostanie wysyłane żądanie. Na końcu adresu po znaku zapytania podawany jest atrybut o nazwie *fileHash* ze wskazaną wartością hasza pliku. Przykładową odpowiedz w formacie JSON (w skróconej formie) możemy zobaczyć na listingu 3.2:

Identyfikacja inteligentnego kontraktu za pomocą API

Podczas identyfikacji inteligentnego kontraktu, za pomocą interfejsu programistycznego, żądanie wygląda nieco inaczej niż poprzednio. Podczas identyfikacji wykorzystywana jest metoda POST. W nagłówku HTTP należy więc wprowadzić tą metodę, a następnie podać ścieżkę */api/bytocode*, pod którą należy wysłać żądanie. W przypadku metody POST opisanej w sekcji 9.5 dokumentu RFC2616 [1] podajemy poniżej nagłówka zawartość, która będzie przesłana do serwera. W odróżnieniu od wykorzystanej poprzednio metody GET, tutaj zawartość zostaje podana poniżej nagłówka zamiast w adresie.

API pod adresem apibetcode wymaga od użytkownika dwóch atrybutów o nazwach: bytecode oraz allFiles. Poniżej nagłówka HTTP w żądaniu należy wprowadzić odpowiednio zakodowane atrybuty. W celu zakodowania listy atrybutów, każdy atrybut zapisuje się w postaci:

NAZWA_ATRYBUTU=WARTOSC_ATRYBUTU

łącząc je wszystkie ampersandem. Poprawnie zakodowana zawartość w żądaniu HTTP w tym przypadku powinna wyglądać następująco:

bytecode=KOD_BAJTOWY&allFiles=WARTOSC_LOGICZNA.

W tym przypadku wartość KOD_BAJTOWY użytkownik musi zamienić na swój kod bajtowy, który chce przeanalizować oraz WARTOSC_LOGICZNA należy zamienić na napis *true* lub *false*, w zależności czy chcemy w rezultacie otrzymać wszystkie pliki, które brały udział, czy chcemy otrzymać tylko pierwsze dziesięć najbardziej pasujących plików. Dodatkowo w nagłówku należy wprowadzić parametr **Content-Length**, dla którego należy podać długość wysyłanych. Przykładowo dla zawartości:

```
bytecode=234a12b45&allFiles=true
```

fragment nagłówka powinien wyglądać następująco:

```
Content-Length: 32
```

W rezultacie zapytania użytkownik otrzyma listę składającą się z haszu pliku oraz współczynnika dopasowania danego pliku w formacie JSON.

Przesyłanie nowego kodu źródłowego za pomocą API

Podczas przesyłania nowego kodu źródłowego przy wykorzystaniu API istnieje możliwość przesłania kodu źródłowego. W odróżnieniu od sposobu opisanego powyżej, teraz nie będą używane atrybuty z nazwami, tylko zostanie wysłane żądanie HTTP w którym pod nagłówkiem będzie wprowadzony tylko kod źródłowy.

W celu wysłania takiego żądania najpierw użytkownik musi zostać uwierzyteliony, w innym przypadku próba dodania nowego kodu źródłowego do aplikacji ukończy się niepowodzeniem. Zakładając, że użytkownik został uwierzyteliony to bazując na przykładowym żądaniu HTTP z listingu 3.1 należy wprowadzić poniżej nagłówka cały kod źródłowy kontraktu.

Należy pamiętać, że kod źródłowy musi być zaimplementowany w języku Solidity, w innym wypadku otrzymamy fałszywe informacje.

W żądaniu HTTP użytkownik musi ustawić metodę POST, wskazać ścieżkę na /api/solidityFiles oraz analogicznie do sekcji wyżej, ustawić wartość *Content-Length* w zależności od długości przesyłanych danych.

Po pomyślnym przesłaniu kontraktu, w odpowiedzi od serwera utrzymujemy status HTTP 200, informujący nas, że wszystko poszło pomyślnie. Następnie w informacji zwrotnej użytkownik otrzymuje przesłany przez niego kod źródłowy, hasz stworzony na podstawie kodu źródłowego oraz listę znalezionych w nim funkcji wraz z akcesoriami generowanymi podczas komplikacji. Na listingu 3.2 można zaobserwować przykładowe dane zwracane przez API po dodaniu nowego kodu źródłowego.

Listing 3.2: Przykładowa odpowiedź w formacie JSON(w skróconej formie)

```
1 {
2     "sourceCodeHash": "0
x80b739cbf3e89eee1a96d9cfef0567ddfef2af82eb14d2c5f97862f71e56265
",
3     "sourceCode": "pragma solidity ^0.4.25; \n\ncontract Hello { \n\n    \tstring public message; \n    \tconstructor(string initialMessage) public { \n        \tmessage = initialMessage; \n    } \n    \tfunction setMessage(string newMessage) public { \n        \tmessage = newMessage; \n    } \n} \n",
4     "solidityFunctions": [
5         {
6             "selector": "e21f37ce",
7             "signature": "message()"
8         },
9         {
10            "selector": "368b8772",
11            "signature": "setMessage(string)"
12        }
13    ]
14 }
```

3.2 Architektura

Literatura: [2, ?]. TODO

3.2.1 Wyszukiwanie sygnatur funkcji w kodzie źródłowym

Literatura: [2, ?]. TODO

3.2.2 Wyszukiwanie selektorów funkcji w kodzie bajtowym

Literatura: [2, ?]. TODO

3.2.3 Szukanie implementacji na podstawie kodu bajtowego

Literatura: [2, ?]. TODO

3.3 Wykorzystane technologie

Literatura: [2, ?]. TODO

Bibliografia

[1] Network Working Group <https://www.ietf.org/rfc/rfc2616.txt>.

[2] Bibliografia 2. *Nazwa*.

Spis tabel

Spis rysunków

3.1 Wynik identyfikacji inteligentnego kontraktu	13
3.2 Podgląd implementacji	14

Spis listingów

3.1 Przykładowe żądanie HTTP	16
3.2 Przykładowa odpowiedz w formacie JSON(w skróconej formie) . .	19

