Lab 18.2: Patching EXEs with Ollydbg

**Huynh Ngoc Quang – SE181838**

1. Patching an EXE

**Checking the Hash:**

A screenshot of a computer

Description automatically generated

SHA256: 1a3fcb290fcb80d136266bc94f4d5963578154147a91eb5d3866507a32dc0e59

**Running the EXE:**

A computer screen shot of a computer screen

Description automatically generated

**Examining the EXE with Ollydbg:**

A screenshot of a computer

Description automatically generated

**Modifying the EXE:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Saving the Modified File:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Running the Modified File**

A computer screen shot of a black screen

Description automatically generated

**Checking the Hash**

A screenshot of a computer

Description automatically generated

**CRC32: a1dccbf4**

1. Patching three EXEs:

**Checking the Hash:**

A screenshot of a computer

Description automatically generated

**SHA256: d8e696f0eabd6dac529222850ecd6b47a12237497cef7c910cc390f9144f96f5**

**Patch the Files: like above**

**Gather the Results:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Patching 19 EXEs:

I created a super.py script to automatically complete all step above.

1. import subprocess

2. import os

3. import binascii

4.

5. search\_start = b'\x3B\x05'

6. search\_end = b'\x75\x1E'

7. replace\_value = b'\x90\x90\x90\x90\x90\x90\x90\x90'

8.

9. # Change directory to the path where your executable files are located

10. os.chdir('C:/Users/Administrator/Documents/256')

11.

12. # Create a list to hold the results from all executable files

13. results = []

14.

15. # Loop through all files in the directory

16. for filename in os.listdir('.'):

17. if filename.endswith('.exe') or filename.endswith('.dll'):

18. # Open the file in binary mode

19. with open(filename, 'rb') as f:

20. # Read the file contents into a bytes object

21. file\_contents = f.read()

22.

23. # Loop through the file contents searching for the hex pattern

24. i = 0

25. while i < len(file\_contents):

26. if file\_contents[i:i+2] == search\_start:

27. j = i + 2

28. while j < len(file\_contents):

29. if file\_contents[j:j+2] == search\_end:

30. # Replace the hex pattern with the replacement value

31. file\_contents = (

32. file\_contents[:i] +

33. replace\_value +

34. file\_contents[j + 2:]

35. )

36. break

37. j += 1

38. i += 1

39.

40. # Write the modified file contents back to the file

41. with open(filename, 'wb') as f:

42. f.write(file\_contents)

43.

44. # Run the executable file with input "18" using subprocess

45. proc = subprocess.Popen([filename], stdin=subprocess.PIPE, stdout=subprocess.PIPE)

46.

47. # Send input to the subprocess

48. proc.stdin.write(b'18\n')

49. proc.stdin.close()

50.

51. # Wait for the subprocess to finish and get its output

52. result = proc.stdout.read()

53. print result

54.

55. # Extract the string inside the parentheses and add it to the results

56. result\_str = result.decode('utf-8')

57. start\_index = result\_str.find('(')

58. end\_index = result\_str.find(')')

59.

60. if start\_index != -1 and end\_index != -1:

61. results.append(result\_str[start\_index + 1:end\_index])

62.

63.

64. # Write the concatenated results to a text file

65. with open('results.txt', 'w') as f:

66. f.write(''.join(results))

67.

68. print('Results written to results.txt')

69.

1. Patching 256 EXEs:

Use the super.py code above to automatically patch, run program and collect the output into the result.txt

A screenshot of a computer

Description automatically generated

The content of result.txt:

Javascript: [][(]+[][+[]]+([![]]+[][[]][+!+[]+[+[]]]+(![]+[][!+[]+!+[]]+(!![]+[][+[]]+(!![]+[][!+[]+!+[]+!+[]]+(!![]+[][+!+[]]][([][(![]+[][+[]]+([![]]+[][[]][+!+[]+[+[]]]+(![]+[][!+[]+!+[]]+(!![]+[][+[]]+(!![]+[][!+[]+!+[]+!+[]]+(!![]+[][+!+

![A screenshot of a computer

Description automatically generated