BINARY

Reverse engineering challenge

[Publish Date]

European Cyber Security Challenge 2019
Bucharest, Romania

Version	**Name**	**Comments**	**Date**
0.1	Csaba Virág		26/08/2019
0.2	Adrian Belmonte	Review	10/09/2019

1. Initial Write-Up

There is a given a C# windows binary program and a serial number generator algorithm can be found in the application. The application is using obfuscation to make reverse engineering harder. Your task is to analyse the binary file and try to find a valid serial number.

2. Challenge specifications

- Category: reverse engineering
- Difficulty : easy/medium
- Expected time to solve: 1 hour to solve (aprx)

3. Technical specifications

1. Binary file is provided
2. Participant shall have environment for reverse engineering

4. Questions and answers

CTF Specific questions:

Question:

What is a valid serial number?

Answer:

SERIAL-TNOU-P3XD-QDOU-8YAR

Question:

What type of obfuscation has been used?

Answer:

.NET Reactor

Question:

What is the name of the process validating serial numbers?

Answer:

IsSerialValid

5. Attack Scenario

There is a given a C# windows binary program and a serial number generator algorithm can be found in the application. The application is using obfuscation to make reverse engineering harder. Your task is to analyse the binary file and try to find a valid serial number.

6. Installation instructions

Setup for the organizers:

Distribute the attached binary to participants.

7. Tools needed

Description:

Tools needed for the solution of the challenge:

- Linux and programming knowledge
- C# knowledge

8. Artifacts Provided

Description:

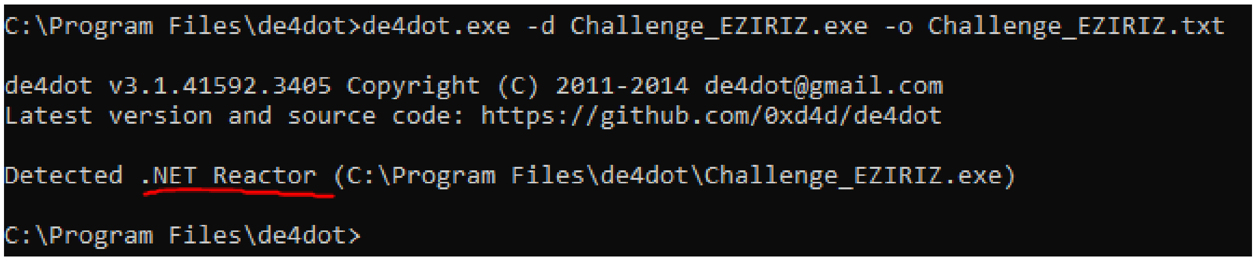
List of artifacts provided with checksums.

Example:

Name	**Format**	**Comment**	**Checksum (SHA256)**
binary.exe	binary		ab215760bfb3d37c570f5b5ab441031e82d042efec30b3d08b462c08ce2c14b7
Binary_generator.php	php		11b9b3407094899ee65624e3e489ceeac90fdd9bb6ccf472430ca9a43258d849

9. Walkthrough (writeup)

1. The source code of the file should be gained with de4dot.exe and JustDecompile (or similar applications).



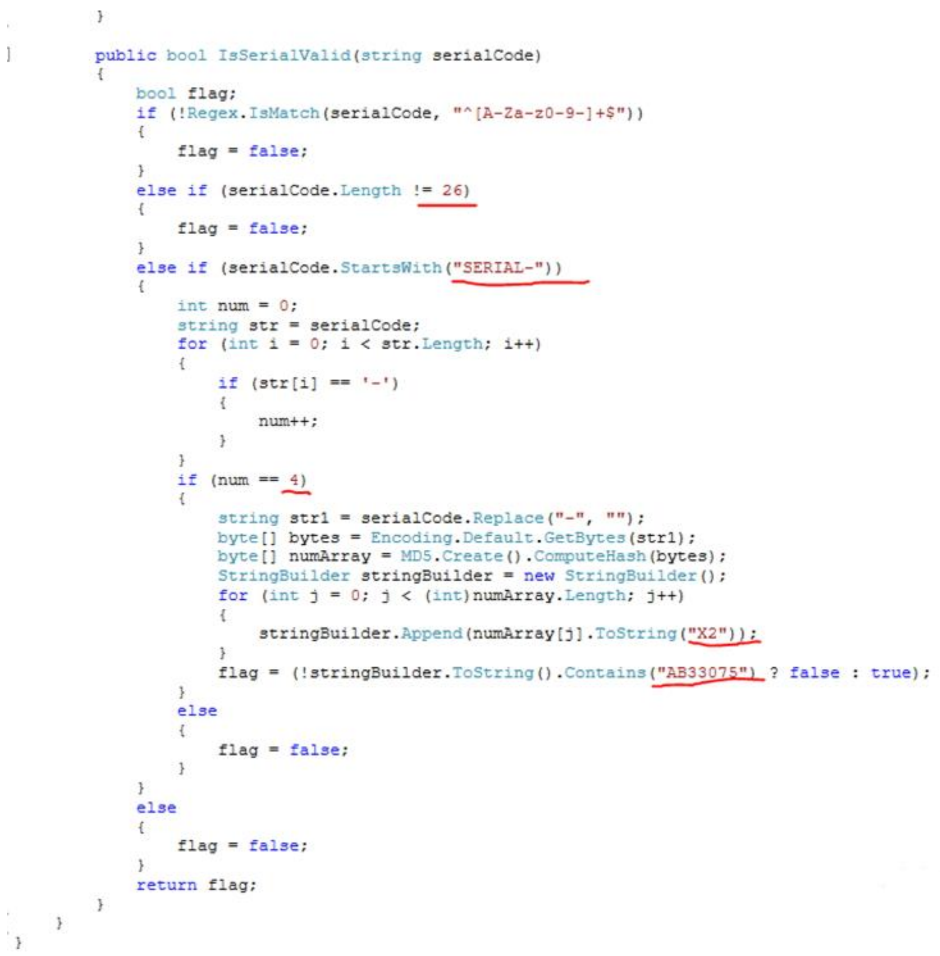
```
C:\Program Files\de4dot>de4dot.exe -d Challenge_EZIRIZ.exe -o Challenge_EZIRIZ.txt

de4dot v3.1.41592.3405 Copyright (C) 2011-2014 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected .NET Reactor (C:\Program Files\de4dot\Challenge_EZIRIZ.exe)

C:\Program Files\de4dot>
```

2.,
Analysing the source code, the function can be found that validates the serial number



```
        }

        public bool IsSerialValid(string serialCode)
        {
            bool flag;
            if (!Regex.IsMatch(serialCode, "^[A-Za-z0-9-]+$"))
            {
                flag = false;
            }
            else if (serialCode.Length != 26)
            {
                flag = false;
            }
            else if (serialCode.StartsWith("SERIAL-"))
            {
                int num = 0;
                string str = serialCode;
                for (int i = 0; i < str.Length; i++)
                {
                    if (str[i] == '-')
                    {
                        num++;
                    }
                }
                if (num == 4)
                {
                    string str1 = serialCode.Replace("-", "");
                    byte[] bytes = Encoding.Default.GetBytes(str1);
                    byte[] numArray = MD5.Create().ComputeHash(bytes);
                    StringBuilder stringBuilder = new StringBuilder();
                    for (int j = 0; j < (int)numArray.Length; j++)
                    {
                        stringBuilder.Append(numArray[j].ToString("X2"));
                    }
                    flag = (!stringBuilder.ToString().Contains("AB33075") ? false : true);
                }
                else
                {
                    flag = false;
                }
            }
            else
            {
                flag = false;
            }
            return flag;
        }
    }
}
```

3., IsSerialValid function runs multiple checks and reveals the serial number is 26 characters and has 4 dashes and starts with "SERIAL".

4., The validator script can be used as solution to generate valid serial numbers, using the reverse logic of what the binary is willing to accept as valid code.