



Threat Research

The FLARE On Challenge Solutions: Part 1 of 2

November 18, 2014 | by [Richard Wartell](#), [Mike Sikorski](#)

EXPLOITS THREAT RESEARCH

In July, the FireEye Labs Advanced Reverse Engineering (FLARE) team created and released the first [FLARE On Challenge](#) to the community. A total of 7,140 people participated and showed off their skills, and 226 people completed the challenge. Everyone who finished the challenge received a challenge coin to commemorate their success.



The coveted challenge coin

We are releasing the challenge solutions to help those who didn't finish improve their skills. There are many different ways to complete each challenge, so we waited to see what solutions people devised. We found the following solutions posted online and recommend taking a look at these as well to see how the later challenges can be solved in different ways.

<http://www.ghettoforensics.com/2014/09/a-walkthrough-for-flare-re-challenges.html>

<https://www.codeandsec.com/Detailed-Solutions-to-FireEye-FLARE-Challenge>

In this initial issue of the blog series, we focus on the first five challenges, which were easier. We hope that by starting with easier solutions, people who have never reversed before can still benefit. Below is a write up of how to solve the first five challenges, written as if we'd never seen them before. The goal of each challenge is to find a key in the form of an email address that



Promotion



Subscribe



Share



Recent



RSS



Stay tuned for Part 2 where we show two different and interesting ways of solving Challenge 6.

Challenge 1: Bob Doge

The first challenge starts out pretty easy. When we drop the binary into CFF Explorer (or equivalent PE tool), it informs us that we're dealing with a PE 32-bit .NET Assembly, so we can run it in an x86 Windows VM and see what happens. A decode button appears to have two functions: transforming Bob Ross into Bob Doge, and converting the top label into some unprintable strings. We drop the binary into ILSpy (or equivalent .NET decompiler) to get an idea what this decode button is doing. The decompiled code is shown in the top of Figure 1.

```
// XXXXXXXXXXXXXXXX.Form1
private void btnDecode_Click(object sender, EventArgs e)
{
    this.pbRoge.Image = Resources.bob_roge;
    byte[] dat_secret = Resources.dat_secret;
    string text = "";
    for (int i = 0; i < dat_secret.Length; i++)
    {
        byte b = dat_secret[i];
        text += (char)((b >> 4 | ((int)b << 4 & 240)) ^ 41);
    }
    text += "\0";
    string text2 = "";
    for (int j = 0; j < text.Length; j += 2)
    {
        text2 += text[j + 1];
        text2 += text[j];
    }
    string text3 = "";
    for (int k = 0; k < text2.Length; k++)
    {
        char arg_B6_0 = text2[k];
        text3 += (char)((byte)text2[k] ^ 102);
    }
    this.lbl_title.Text = text3;
}
```



Promotion



Subscribe



Share



Recent



RSS



```

text = ""
for i in range(len(dat_secret)):
    b = ord(dat_secret[i])
    text += chr(((b >> 4) | ((b << 4) & 240)) ^ 41)
text += "\0"
print "Text 1: %s" % text

text2 = ""
for j in range(len(text)/2):
    text2 += text[j*2 + 1]
    text2 += text[j*2]
print "Text 2: %s" % text2

text3 = ""
for k in range(len(text2)):
    text3 += chr(ord(text2[k]) ^ 102)
print "Text 3: %s" % text3

```

Figure 1: Decode button code in ILSpy (top) and re-implemented in Python (bottom)

The button changes the image to Bob Doge, and encodes a resource string twice and sets the label text to the result. If we save out the resource that is being manipulated, we can play around with it. The Python code in the right side of Figure 1 is the decode button re-implement to help us figure out what we are dealing with. When this Python code is run, the following is printed out showing the solution to Challenge 1 as "Text 1" in the output.

```

Text 1: 3rmahg3rd.b0b.d0ge@flare-on.com
Text 2: r3amghr3.d0b.b0degf@alero-.noc m
Text 3: 11U80J11UH0U♦H♦U0v0 &

```

Figure 2: Challenge 1 result

Challenge 2: Javascrap

The next challenge (to the bane of some of our players) is not a Windows PE file. Instead we have a version of the website www.flare-on.com. There has to be something special about this version of the website. So we look at page source of *home.html*. If we compare this version with the live challenge website, one line in particular stands out:

```
<?php include "img/flare-on.png" ?>
```

