Reverse Engineering Challenge Write-up: Tiempo Oculto

Abinash Bastola | Hakimuddin Lokhandwala

September 28, 2025

1 Introduction

The challenge consists of a Linux executable that prompts the user to enter a key. If the input matches the expected key, the program accepts the solution and indicates success. The goal of this write-up is to analyze the binary and recover the correct key. The analysis was performed exclusively using Ghidra.

The original CrackMe challenge can be found here: bingus

2 Challenge Description

"Don't let Bingus explode. Please don't patch, pretty please." — thedollar

3 Analysis

3.1 Decompiling the Binary

The compiled file is called `bingus`.
When Ghidra decompiles the binary, there is an `entry` function, this is where the program starts. In some compilations, there is a call that looks like:

```
__libc_start_main(FUN_00101149,param_2,&stack0x00000008,0,0,param_1,
    auStack_8);
```

Here, `FUN_00101149` is the program's `main` function.

3.2 The 'main' Function

```
undefined8 FUN_00101149(int param_1,long param_2)
{
  size_t sVar1;
  int local_20;
  int local_1c;

  if (((param_1 != 2) || (**(char **)(param_2 + 8) != *(char *)(*(
     long *)(param_2 + 8) + 1))) ||
     (sVar1 = strlen(*(char **)(param_2 + 8)), sVar1 != 2)) {
    puts("Bingus exploded");
```

```c
    return 1;
  }
  local_1c = 0x66;
  for (local_20 = 0; sVar1 = strlen("This is a red herring"), (ulong
      )(long)local_20 < sVar1;
      local_20 = local_20 + 1) {
    local_1c = local_1c + "This is a red herring"[local_20];
  }
  if (local_1c + (int)*(char *)(*(long *)(param_2 + 8) + 1) + (int)
      **(char **)(param_2 + 8) != 0x8c5
      ) {
    puts("Bingus exploded");
    return 1;
  }
  puts("Bingus survived");
  return 0;
}
```

3.3 Identifying Key Variables and Statements

- `param_1`: This is typically known as `argc`. In this case, it needs to be 2, meaning there must be one command-line argument in addition to the program name.
- `**(char **)(param_2 + 8)`: The first character of the second command-line argument.
- `*(char *)(*(long *)(param_2 + 8) + 1)`: The second character of the second command-line argument.
- `local_1c`: Initially set to 102 (0x66). During the loop, the integer values of each character in the string `"This is a red herring"` are added to this value. The total after the loop is 2021.
- `local_1c + (int)*(char *)(*(long *)(param_2 + 8) + 1) + (int)**(char **)(param_2 + 8) != 0x8c5`: The sum of 2021 and the first and second characters of the second command-line argument must equal 2245 (0x8c5).

3.4 Understanding C Command-line Arguments

In standard C, the `main` function is often defined as:

```c
int main(int argc, char *argv[]);
```

where:

- `argc` (argument count) holds the number of arguments passed to the program, including the program name itself.
- `argv` (argument vector) is an array of strings (`char *`) where:
  - `argv[0]` is the name of the program.
  - `argv[1]` is the first user-provided argument.
  - and so on...

For example, if the program is run as:

```
./bingus pp
```

Then:

- `argc = 2`
- `argv[0] = "./bingus"`
- `argv[1] = "pp"`

In the decompiled code:

- `param_1` corresponds to `argc`.
- `param_2` corresponds to `argv` but represented as a `long` pointer for internal calling conventions.
- `param_2 + 8` points to `argv[1]` (on 64-bit systems, pointers are 8 bytes each).

This is why expressions like `**(char **)(param_2 + 8)` and `*(char *)(*(long *)(param_2 + 8) + 1)` are used to access the first and second characters of the second command-line argument, since the first command-line arguement is the name of the code.

4 Solution

There can only be two command-line arguments: the name of the program and the key that the program accepts. The second argument can only contain two characters.

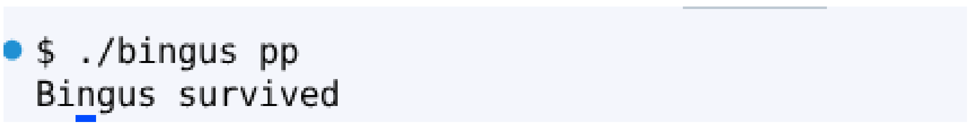
The first and second characters of the second command-line argument must be identical. Additionally, their combined ASCII values plus 2021 must equal 2245.
Therefore:

$$2245 - 2021 = 224 \quad \Rightarrow \quad \frac{224}{2} = 112$$

The ASCII value 112 corresponds to the character `'p'`.
Thus, any command-line argument whose first two characters are `p` will satisfy the condition.

The only solution is: `pp`

4.1 Test Solution



```
$ ./bingus pp
Bingus survived
```