B1ll_Gat35

PicoCTF 2019 - Reverse Engineering

Challenge Information

Author: Alex Bushkin

Category: Reverse Engineering

Points: 400

Description: Can you reverse this Windows Binary?

File Analysis

The challenge provides a Windows PE32 executable file. Initial file analysis:

```
$ file win-exec-1.exe
 win-exec-1.exe: PE32 executable for MS Windows, Intel i386, 6 sections
```

String Extraction

Extracting readable strings from the binary reveals the program flow:

```
$ strings win-exec-1.exe | grep -E "(key|flag|vault)" -i
```

Key strings identified:

```
Input a number between 1 and 5 digits:
 The key is:
 Enter the correct key to get the access codes:
 PICOCTF{These are the access codes to the vault:
```

Solution

The program contains a critical vulnerability: it prints the validation key before requesting user input. This allows for direct extraction of the correct key.

Execution steps:

1. Execute the binary
2. Input number: **`1`**
3. Program displays: "`The key is: 4253360`"
4. Input the displayed key: **`4253360`**

Program output:

```
Input a number between 1 and 5 digits: 1
 Initializing...
 Enter the correct key to get the access codes: The key is: 4253360
 Correct input. Printing flag:
 PICOCTF{These are the access codes to the vault: 1063340}
```

Flag

`PICOCTF{These are the access codes to the vault: 1063340}`

Technical Analysis

Program Logic:

The binary implements a key generation algorithm based on user input. However, the implementation contains a debug statement that outputs the generated key to stdout before validating user input. This information disclosure allows an attacker to bypass the intended validation mechanism.

Vulnerability Type: Information Disclosure

Root Cause: Debug print statement left in production code

Tools Used

`file` - Binary file type identification

`strings` - Extract printable strings from binary

`grep` - Pattern matching and filtering

Conclusion

This challenge demonstrates the security risks of leaving debug code in production environments. The inadvertent disclosure of the validation key completely bypasses the intended security mechanism, allowing trivial exploitation.