

FLARE-On 13 Challenge 2: The Ghost Stream

By Deepak Rajput (@deepakrajput)

Overview

This challenge consists of two interconnected Windows PE executables: a parent process and a partner process embedded in the resource section. It comes packaged with a `PolyjuicePotion.txt` file that contains a lore-based hint, as well as a secret key hidden within an Alternate Data Stream (ADS).

When executed, the parent reads the key file, performs various anti-analysis, anti-debugging, and time-based checks. If all checks pass, it sends the key to the partner process over a Named Pipe. The partner process then uses a modified RC4 algorithm to decrypt the final flag:

`y0u_kn0w_n07h1n6_j0n_5n0w@flare-on.com.`

If you run the parent executable normally, you will immediately notice that it prints a cryptic message and exits:

"The ghost won't reveal its secrets to the naked eye. If you want the flag, fire up your debugger..."

Furthermore, attempting to rely solely on basic static analysis will lead you down several "rabbit holes," including a fake network backdoor listening on `localhost`. The core challenge requires bypassing anti-debugging techniques, tracking asynchronous OS-level callbacks, identifying an inline hook on a standard C library function, and reversing a custom RC4 implementation to extract the flag.

Execution Flow & The Polyjuice Potion Theme

The challenge relies heavily on a Harry Potter theme. The provided `PolyjuicePotion.txt` file contains the following message:

"During their second year, Harry, Ron, and Hermione brewed Polyjuice Potion to sneak into the Slytherin common room. But this time... bloody hell! Ron took the initiative and added far too much boomslang skin to the cauldron. The potion has destabilized, and the transformation effect is now stuck for exactly 7 days."

If the executable is launched directly, it prints the warning message and terminates execution:

"The ghost won't reveal its secrets to the naked eye. If you want the flag, fire up your debugger..."

If the program detects it is being debugged, the warning message is suppressed, but execution is still stealthily terminated using the undocumented `NtSetInformationThread` API at `0x1400021FA`.

During execution, two threads are created:

- **The Decoy Thread:** Contains fake logic designed to look like it writes bytes to the key buffer.

- **The Real Thread:** Reads the initial key `AlbusDumbledore` from the Alternate Data Stream `PolyjuicePotion.txt:LordVoldemort`, provided a specific execution flag is set.

After reading the initial key, additional decryption is required. This logic is hidden inside a Windows GUI callback function and only triggers if the process has been running for more than 7 days. If the time check fails, the user is presented with:

“The Polyjuice Potion is still effective, preventing the true identity from being revealed.”

Once the 7-day check is bypassed, the key is sent to the partner process. Within the partner process, an inline hook placed on the standard `strlen` function serves two purposes: decrypting the secondary key `"Piertotum Locomotor"` and returning an inaccurate length for the string `"Wingardium Leviosa"`. Once the analyst identifies this and passes the correct length into the custom RC4 algorithm, the final flag is revealed.

Reversing Hurdles

This challenge layers multiple anti-analysis techniques to slow down the reverse engineer:

Static Analysis Obfuscation

Strings are encrypted and API calls are dynamically resolved at runtime to hide the program's true intent from tools like IDA Pro and Ghidra.

Disabling the Clipboard

Early in the execution, `OpenClipboard` is called and locked. This frustrates dynamic analysis by preventing the analyst from easily copy-pasting dynamically decrypted strings, keys, or memory addresses out of the debugger.

Decoy Code

Fake logic is strategically placed to confuse the analyst. For example, a decoy thread binds a TCP socket to `127.0.0.1` on port `31337`, creating the false impression of a remote backdoor that expects a key override.

Anti-Debug

The API `NtSetInformationThread` is called with the `ThreadHideFromDebugger (0x11)` parameter to detach the debugger and terminate execution.

Alternate Data Stream (ADS)

The actual starting key is hidden inside an ADS. The code responsible for reading it is gated by a flag and mixed with decoy thread logic.

Time Bomb and Asynchronous Callback

The key modification routine is protected by a time check requiring the process to run for 7 days. Furthermore, this logic is smuggled into the `WM_CREATE` callback of a hidden `CreateWindowEx` call, breaking linear control-flow analysis.

API-Hooking

Inside the partner process, the C standard library function `strlen` is hooked in memory to return an incorrect string length. This spoofed length is then fed directly into the decryption loop.

Custom RC4

A modified RC4 algorithm prevents standard crypto-solvers from decrypting the flag directly, forcing the analyst to understand and reverse the mathematical operations.

Solution Steps

Step 1: Find and Extract the ADS Key

Locate the code in the second thread responsible for querying the Alternate Data Stream path (`PolyjuicePotion.txt:LordVoldemort`). You must trigger the execution of this code block, either by manually changing the instruction pointer in your debugger or by toggling the flag value at `0x140001A8C` that gates the branch:

```
83 3D D5 4B 00 00 01    cmp     cs:dword_140006668, 1
75 2B                   jnz     short loc_140001AC0
4C 8B 47 10             mov     r8, [rdi+10h]
48 8B 57 08             mov     rdx, [rdi+8]
48 8B 0F               mov     rcx, [rdi]
E8 8B F6 FF FF         call    sub_140001130
4C 8B 4F 20             mov     r9, [rdi+20h]
4C 8B 47 10             mov     r8, [rdi+10h] ; Format
48 8B 57 28             mov     rdx, [rdi+28h] ; BufferCount
48 8B 4F 18             mov     rcx, [rdi+18h] ; Buffer
E8 16 F6 FF FF         call    sub_1400010D0
```

Step 2: Bypass ThreadHideFromDebugger

To prevent the process from terminating while being debugged, you must intercept and bypass the dynamically resolved call to `NtSetInformationThread`. This can be done by patching the call with `NOP`

instructions or by altering the `0x11` (`ThreadHideFromDebugger`) parameter before execution or bypassing the API call at `0x1400021FA`.

Step 3: Bypass the Time Bomb

Identify the `CreateWindowEx` API call. Recognize that the application passes a structure pointer via the `lpParam` argument, which is then processed synchronously by the window's callback function.

Inside the callback, bypass the 7-day uptime check (`cmp [r9+10h], rax`) at `0x140001BAA`. Doing so triggers an additional dynamic XOR decryption based on the sequence of OS window messages, followed by the static block of XOR instructions below. This two-stage decryption effectively neutralizes the Polyjuice Potion effect, transforming the initial key "`AlbusDumbledore`" into the final expected key: "`HarryPotter`".

```
80 32 8B      xor     byte ptr [rdx], 8Bh
80 72 01 8F    xor     byte ptr [rdx+1], 8Fh
80 72 02 92    xor     byte ptr [rdx+2], 92h
80 72 03 85    xor     byte ptr [rdx+3], 85h
80 72 04 88    xor     byte ptr [rdx+4], 88h
80 72 05 96    xor     byte ptr [rdx+5], 96h
80 72 06 98    xor     byte ptr [rdx+6], 98h
80 72 07 9B    xor     byte ptr [rdx+7], 9Bh
80 72 08 94    xor     byte ptr [rdx+8], 94h
80 72 09 8B    xor     byte ptr [rdx+9], 8Bh
80 72 0A 95    xor     byte ptr [rdx+0Ah], 95h
80 72 0B E6    xor     byte ptr [rdx+0Bh], 0E6h
80 72 0C ED    xor     byte ptr [rdx+0Ch], 0EDh
80 72 0D F0    xor     byte ptr [rdx+0Dh], 0F0h
80 72 0E E7    xor     byte ptr [rdx+0Eh], 0E7h
EB 50         jmp     short loc_7FF77E571C84
```

Step 4: Calculate the Correct Length and Decrypt

After transferring the key across the pipe to the partner process, the binary calls `strlen` to measure "`Wingardium Leviosa`". However, `strlen` has been hooked with shellcode that serves a dual purpose: it returns an artificially inflated string length, and it dynamically decrypts a secondary key "`Piertotum Locomotor`" in memory via an index-based rolling XOR algorithm.

Because generating the secondary key relies on the shellcode logic, decryption will fail if you try to unhook `strlen` by restoring its original bytes or if you craft a Python decryption script that ignores the hook.

To overcome this and solve the challenge, you must manually pass the genuine string length (`18`) into the custom RC4 decryption routine to reveal the flag: `y0u_kn0w_n07h1n6_j0n_5n0w@flare-on.com`.