Initiale Analyse

Gegeben ist ein Programm `space_terminal` und eine Datei `secret_message`, welche höchstwahrscheinlich die verschlüsselte Flagge enthält.

Die Challenge Beschreibung deutet darauf hin, dass das Programm in der Lage ist, die Flagge zu entschlüsseln, sie allerdings nicht ausgeben kann. Das sehen wir auch, wenn wir das Programm ausführen:

```
$ ./space_terminal
Welcome Commander! Please choose one of the following options:
---------- Space Terminal ----------
[1] Send a message to your friends on Mars
[2] Load an encrypted message
[3] Decrypt the loaded message
[4] Display the decrypted message
> 2
---------- Space Terminal ----------
[1] Send a message to your friends on Mars
[2] Load an encrypted message
[3] Decrypt the loaded message
[4] Display the decrypted message
> 3
---------- Space Terminal ----------
[1] Send a message to your friends on Mars
[2] Load an encrypted message
[3] Decrypt the loaded message
[4] Display the decrypted message
>
```

Mit Option 2 die Flagge zu laden und mittels Option 3 zu entschlüsseln, scheint das Programm problemlos geschafft zu haben. Wollen wir die Flagge aber nun mit Option 4 ausgeben, stürzt das Programm leider ab. Also was können wir tun?

Lösung

Die Flagge befindet sich, nachdem wir Option 2 und 3 verwendet haben, unverschlüsselt in unserem Speicher. Wir müssen sie nur noch irgendwie auslesen. Um die Flagge aus dem Speicher des laufenden Prozesses auszulesen, gibt es viele Möglichkeiten. Der einfachste Weg ist es wahrscheinlich, einen Debugger wie GDB zu verwenden. Um zu lernen, wie man GDB benutzt, gibt es viele Quellen im Internet. Um zwei Beispiele zu nennen: 1, 2. Im Rahmen eurer Recherche werdet ihr wahrscheinlich über **Assembly** stoßen, was quasi die Maschinensprache ist, welche direkt vom Prozessor ausgeführt ist. Falls ihr darüber mehr lernen möchtet, kann ich folgendes empfehlen: 3.

Um die Aufgabe zu lösen, müssen wir das Programm mit GDB starten, das geht mit `gdb ./space_terminal`. Nun nutzen wir den Command `run`, um das Programm zu starten.

Dann nutzen wir Option 2 und 3 des Programms, um die Flagge zu entschlüsseln. Wenn wir das getan haben und wieder im Menü sind können wir `STRG + C` drücken, um wieder in GDB zu landen.

Nun geben wir `print secret` ein, um die Flagge auszulesen. `secret` ist hierbei der Name der Variable, welche die Flagge enthält. Diesen konnte man herausfinden, wenn man das Programm z. B. in Ghidra öffnet **oder** intensiver mit GDB analysiert.

Damit erhalten wir die Flagge `SSH{4tt4ch_4nd_r3v347_7h3_h1dd3n_06b8427af2f1b8f5b3b7ec65bdd183}`.

Bei offenen Fragen zu der Challenge, schreibt mir gerne auf Discord: **kampet**