Lab Assignment L2: Defusing a Binary Bomb

一、引言

邪恶的 Dr. Evil 已经在我们的机器上种下了一大堆“二进制炸弹”。炸弹是一个由一系列关卡组成的程序,每个关卡都希望你在 stdin 上输入一个特定的字符串。如果你输入正确的字符串,那么这个关卡被解除,炸弹会进入下一个关卡。否则,炸弹爆炸打印“BOOM!!!”然后终止。当所有关卡都通过后,这个炸弹就被拆解了。

我们要处理的炸弹太多,所以我们给每个学生一颗炸弹来化解。你的任务是在到期之前成功拆解你的炸弹。祝你好运,欢迎来到炸弹小队!

Step1:获取炸弹

将 bomb.tar 文件保存到你计划执行工作的(受保护的)目录中。然后给出命令:

tar -xvf bomb.tar

这将使用以下文件创建一个名为./bomb 的目录:

•README:标识炸弹及其所有者。

•bomb::可执行的二进制炸弹。

•bomb.c:炸弹主要程序的源文件,以及来自 Dr. Evil 的友好问候。

Step2:消除炸弹

本次实验你的任务是拆解你的炸弹。

你可以使用许多工具来帮助你拆解炸弹。详细信息请看提示部分的一些提示和想法，最好的方法是使用调试器来反编译二进制文件。

前四个关卡每一个关卡的分数都为 10 分。关卡 5 和关卡 6 有点困难，所以他们每个都值 15 分，因此你可以得到的最高分是 70 分。

虽然关卡的难度逐渐提升，但是当你从低难度关卡转向高难度关卡时，你可以利用你获得的专业知识来解决这个难题。

如果你用命令行参数运行你的炸弹，

linux> ./bomb psol.txt

那么它将从 psol.txt 读取输入行，并且忽略空白的输入行，直到达到 EOF（文件结束），然后切换到 stdin。利用这个功能，你不必在每次运行程序时都重新输入你已经解决的关卡的答案。

为了避免意外引爆炸弹，你将需要学习如何单步执行汇编代码以及如何设置断点，还需要了解如何检查寄存器和内存状态。本次实验的一个很好的作用是你将会非常擅长使用调试器，这是一项至关重要的技能。

二、提示

有很多方法可以化解你的炸弹。例如你可以仔细查看它的汇编代码，而无需运行程序，就可以找出答案到底是什么。这是一个有用的技术，但并不总是很容易做到。你也可以在调试器下运行程序，逐步运行并且观察它的运行情况，并使用这些信息来拆解炸弹，这个可能是解除它的最快方法。

本次实验有一个重要要求——请不要使用暴力拆解！

你可以写一个程序，尝试每一个可能的答案最终总是能找到正确的答案。但

这对于你而言并不是好事，这也不是本次实验的目的。我们没有告诉你这些字符串有多长，也没有告诉你字符串中有什么数据。甚至如果你做了错误的假设，那么在任务到期之前，你仅仅使用暴力拆解将无法得到答案。

有许多工具可以帮助你了解程序是如何工作的，以及当它不正常工作时是因为什么问题。以下是你在分析炸弹时可能会使用到的一些工具列表：

•gdb

GNU 调试器，这是几乎每个平台都可用的命令行调试器工具。你可以通过程序逐行追踪，查看内存和寄存器，同时查看源代码和汇编代码（我们不给你大部分炸弹的源代码），设置断点，设置内存观察点，并编写脚本。

- 为了防止每次输入错误的输入时炸弹爆炸，你要学习如何设置断点。

- 对于联机文档，请在 gdb 命令提示符处键入“help”，或键入“man gdb”或在 Unix 提示符下输入“info gdb”。有些人也喜欢在 gdb-mode 下运行 gdb emacs 的。

想了解更多关于 gdb 命令的详细信息，请查看文档 gdbnotes.pdf。

•objdump -t

这将打印出炸弹的符号表。符号表包含所有函数的名称和炸弹中的全局变量，炸弹所调用的所有函数的名称，以及它们的地址。你可以通过查看函数名称来学习一些东西！

•objdump -d

使用这个来反汇编炸弹中的所有代码，你也可以看看个别的功能。阅读汇编代码可以告诉你炸弹如何工作。尽管 objdump -d 给你提供了很多信息，但并不能告诉你整个故事。调用系统级函数会以隐秘的形式显示。例如，汇编代码中

可能会出现对 sscanf 的调用，例如：

8048c36：e8 99 fc ff ff call 80488d4 <_init + 0x1a0>

要确定调用 sscanf，你需要在 gdb 中反汇编。

- **strings**

  这条命令将在你的炸弹中显示可打印的字符串。

如果你对本次实验有任何疑问，请随时向你的老师寻求帮助。