使用 OllyDbg 从零开始 Cracking

第三章

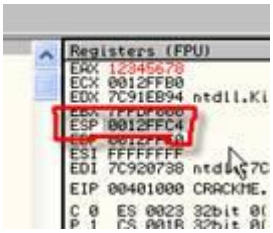
（翻译：BGCoder）

什么是寄存器，有什么作用

寄存器用来做什么，什么是寄存器？

处理器在执行程序时需要一个助手。当执行一条指令时，例如将两个内存单元中存放的内容相加，处理器需要先把其中一个的内容置入寄存器，然后再把另一个内容置入，这是使用寄存器的一个例子（译注 1）。

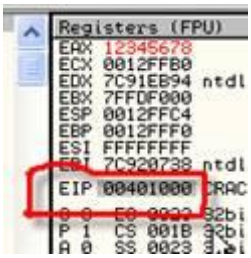
ESP 指向堆栈最顶端的地址，现在来看一下这个 CrueHead'a CrackMe（译注 2）。



ESP 为 12FFC4，如果你看一下 OllyDbg 的堆栈窗口，

我们看到寄存器显示了在我们的堆栈最上方的值，打个比方，它就是一堆信件最上方的那一封。

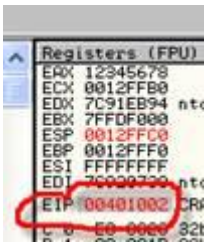
EIP – 另一个非常重要的寄存器，它指向当前将要执行的指令。



我们在下面截图中看到 CrackMe 第一条将执行的指令的地址为 401000，很明显，这正是 EIP

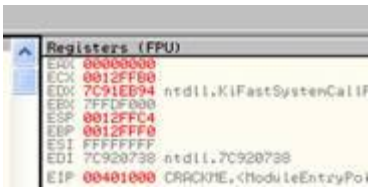
所指向的值。

如果你按下 F7，那么将执行第一条指令，然后切到第二条将执行的指令。



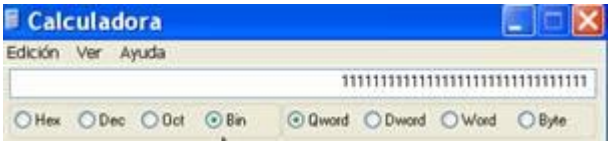
EIP 现在为 401002，在反汇编窗口第一条指令已经走过，现在位于第二条上。

其它寄存器包含有不同的值来为帮助处理器执行指令提供服务。
记住 OllyDbg 在哪里显示这些寄存器。



在这里，显示有 EAX，ECX,，EDX,，EBX， ESP， EBP,，ESI,，EDI 和 EIP 等
它们都被称为 32 位寄存器。

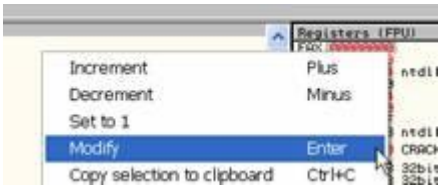
在 OllyDbg 中，它们的内容以十六进制显示。例如，EAX 的最小值为 00000000，最大值为 FFFFFFFF，用二进制表示将是 11111111111111111111111111111111。



我们看到它为 32 位，每一位可被设为 0 或 1，所以这些寄存器被称为是 32 位的。

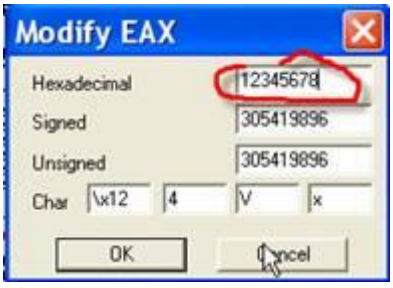
你可以在汇编语言教材中查阅参考这些 32 位寄存器。

现在，在 OllyDbg 中完成一个使用寄存器的例子，来获得一些实践经验。
打开 OllyDbg，加载 CrackMe（也可以加载其它的程序）。把 EAX 更改为我们需要的值，这里假如为 12345678

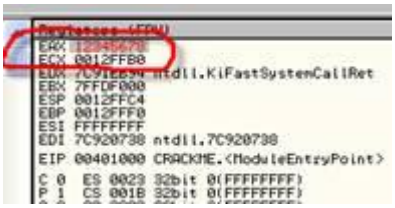


在打开的窗口的 Hexdecimal 处填入 12345678。

就在这里：



然后点击 OK。



现在 EAX 变为了我们的期望值，OllyDbg 将变化的值用红色高亮显示。

如果你要用到 EAX 寄存器的一部分，在这个例子中，AX 是 EAX 的一部分，是 16 位寄存器，例如，在上述例子中，它的值为 5678，我们在 CommandBar 中进行输入也可以看到。

？ AX（问号也可用于查询寄存器的值）

当你按下回车键后，

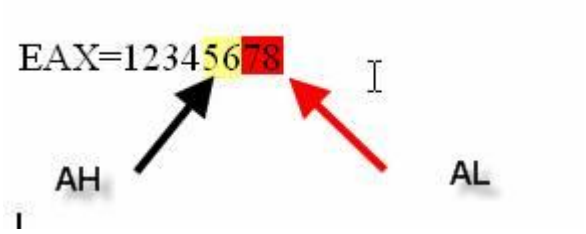
看到了 5678，AX 包含的值为 EAX 的后 4 位数字。还可继续分为 AL 和 AH（译注：16 位寄存器 AX 的低八位和高八位），它们的值在 OllyDbg 中同样能够观察到。

？ AL

？ AH

或者这样更直观，如果 EAX = 12345678，那么 AX 就是它的后四位数字。

AH 就是数字 5 和 6 的组合，AL 就是最后两位。

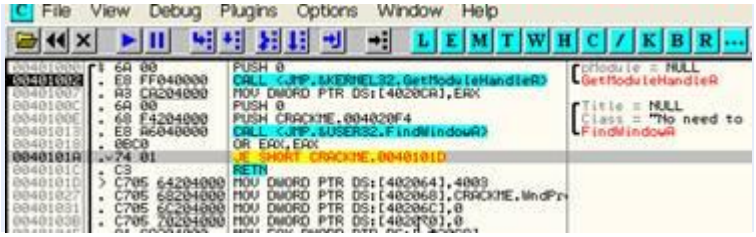


同样的，EBX 可被分为 BX，BL 和 BH。几乎所有其它寄存器都可以如此分割（译注 3）。

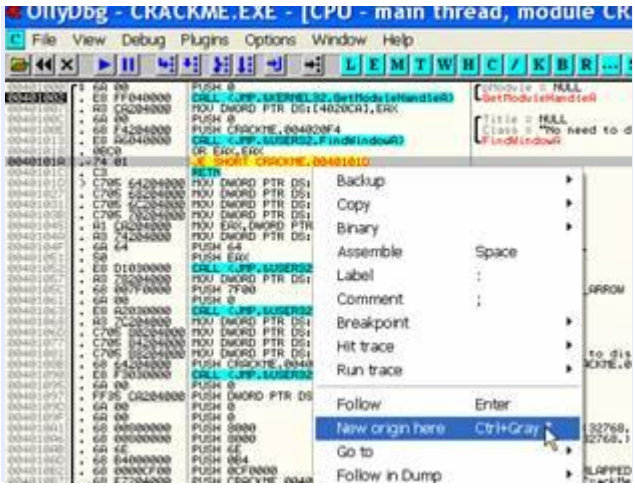
如何更改寄存器的值

我们已经看到了，OllyDbg 可以更改寄存器的值。我们在 EAX 上进行的一切操作同样适用于其它寄存器：检查寄存器，看哪一个是你想要更改的，然后右键点击它选择 Modify。但 EIP 是唯一一个例外的，它指向下一条将要执行的指令。
要改变头的值，需要如下操作。

EIP 指向将要执行的指令，只需简单的在反汇编窗口中选择新的指令起始点



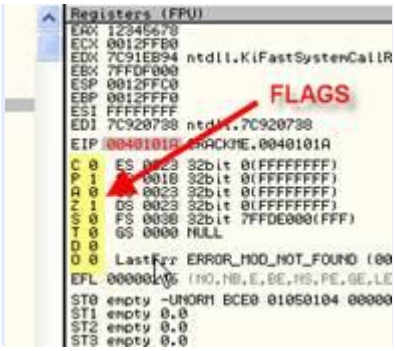
一旦选择，例如 40101A，在其上点击鼠标右键，选择 New origin here（汉化版翻译为：此处为新的 EIP），EIP 就会改变为 40101A，这样，程序就将会从这条指令执行。



在这里你会看到，EIP 指向了 40101A。

什么是标志寄存器

就像我们在第一章看到的，在 OllyDbg 寄存器信息的下方显示的就是标志寄存器。



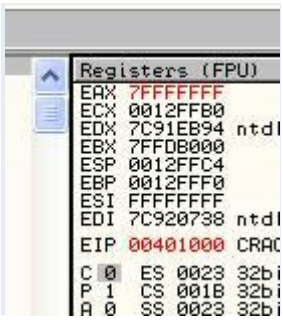
我们看到，这里的标志分为 C，P，A，Z，S，T，D 和 O。
我们还看到，它们只能是两个数字值，0 和 1。某一具体指令的执行可以改变它们的含义。

我们一起来看看这些标志：

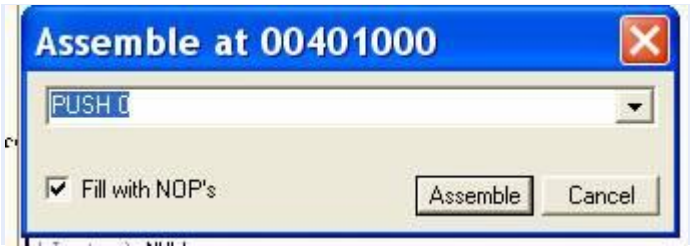
1） O 标志（溢出标志）

溢出标志在当操作改变了符号位，返回错误值时被设置（译注 4）。

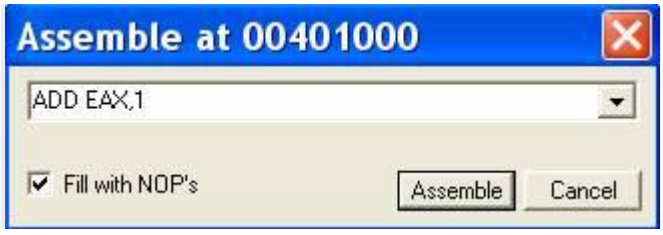
看一下以下在 OllyDbg 中的例子，同样使用 CrueHead'a 的 CrackMe。
我们按照前面讲述的方法将 EAX 的值改为 7FFFFFFF，即最大的正数。



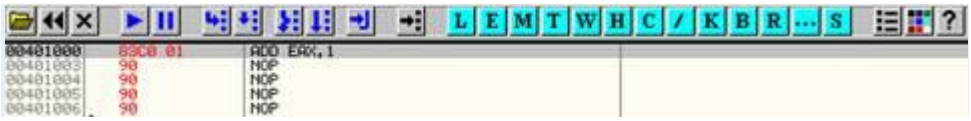
现在使其加 1，其和将超过最大正数，我们还知道，80000000 对应的是一个负数。
这需要打开一个能够写入指令的对话框（译注：本例中请在反汇编窗口的 00401000 指令上按空格键，或在反汇编代码那一列的指令上双击）。



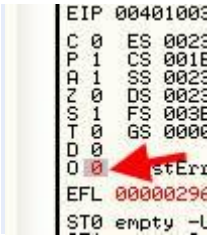
在里面写入：ADD EAX，1。



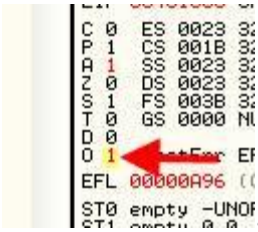
写完后点击 Assemble，就可以看到原来在 00401000 处的指令变为了我们输入的指令。



ADD EAX，1，执行这条指令会发生什么？它将 EAX 加 1，并将值返回到 EAX。
F7 将完成这条指令，O 标志现在等于 0



在 F7 后，看看发生了什么，EAX 变为 80000000，其数字符号更改。
O 标志被设置为 1，该标志的目的：当指令的结果超出了它可能存取的最大值，将被设置



2） A 标志（辅助进位标志）

完成操作后，用其它的某种形式对其进行记录。目前我们不需要关心此标志。

3） P 标志（奇偶标志）

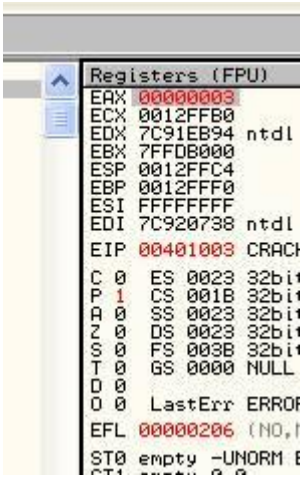
如果指令的结果用二进制表示，该二进制数中的的 1 的总个数为偶数时，P 标志被设置。例如：1010，1100，1111000.

为做个试验，我们已经在 OllyDbg 中设置了指令：ADD EAX，1（译注 5），再一次执行该操作。选择 401000 行，右键点击选择 New orgin here，如果按下 F7，将执行该指令。

现在，EAX 中包含的值为 00000000，P 标志等于 1，（这里是先前指令的结果），让我们看一下，当向 EAX 中加入 1 时发生了什么。
按下 F7

我们看到，P 标志变为了 0，EAX 中值用二进制表示为 1，其二进制格式含有 1 的个数是一个，是奇数。
再次返回，选择 ADD EAX，1 这行，右键点击该行，选择 New origin here，按 F7，再次加 1。

我们看到，EAX 先前为 1，现在为 2，其二进制为 10，1 的个数为奇数，P 标志未被设置。
重复上述操作，再加 1，



现在 EAX 包含的值为 3，二进制为 11，现在它的结果包含偶数个 1，奇偶标志被设置。
从这里，我们可以考到该标志的设置基于以下事实：当指令结果的二进制格式含有偶数个 1 时，被设置。

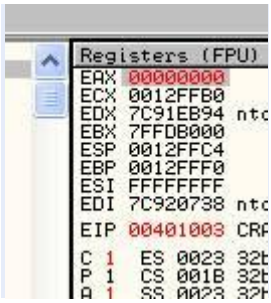
4） Z 标志（零标志）

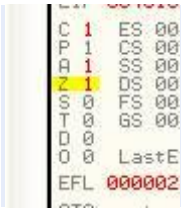
这是在 Cracking 过程中最著名最有用的一个标志。当运算产生的结果为 0 时被设置。

让我们返回到 401000 处的指令：ADD EAX，1。（右键点击 New origin here），先设置 EAX 的值为 FFFFFFFF，即十进制-1，当按下 F7 运行指令 ADD EAX，1 使，结果为-1+1，等于 0，零

标志被设置。

我们看到按下 F7 后，EAX 的值为 0，所以，如果操作的结果为 0，Z 标志被设置为 1。





我想现在应该清楚了，当指令的结果为 0 时，该标志被设置。

5） S 标志（符号标志）

这个标志在运算结果为负时设置为 1。来看一下它是如何运作的，改变 EAX 的数值为 FFFFFFF8，它等于十进制-8

再次按上述操作选择 New origin here，然后 F7 再次执行 ADD EAX，1。结果为 FFFFFFF9，等于十进制-7，是个负数，所以符号标志位被设置。

按 F7 执行指令，S 标志位被设置，标志位为 1，很清楚：负结果导致 S 标志被设置。

6） C 标志进位标志

（无符号运算的结果）在超过最大数值时设置，可能是寄存器的值，例如，将 EAX 设为 FFFFFFFF，然后加 1，我们会看到，进位标志位设为 1.

7） T，D 标志和其它

我现在还不打算解释它们的用途，这是一个相对复杂的话题。我们对它们也不太感兴趣。所以目前可以先着手相对更简单问题，此话题将留到以后探讨。

这样，我们已经对寄存器和标志位有了一定的理解。接下来，你可以逐个回顾一下其它指令，尽管到现在为止，我们仅看到了一个指令：ADD

如果看完本文后，你还存有疑惑，请用指令 ADD EAX，1 跟随本文的操作（以改变标志位的值）来进行实践。

对基础知识的深刻理解是非常重要的，所以请不要倦于实践操作，对待本文同样如此。

译注 1
寄存器是 CPU 内部的高速存储单元，访问速度比常规内存快很多。

译注 2
此 CrackMe 仍然来自第一章，因本章讲述的是寄存器基础知识，所以您可以用其它程序代替。此文件随本文附带。

译注 3
仅用于 EAX，EBX，ECX，EDX，其它寄存器可含有低 16 位寄存器，但不能进一步再分。

译注 4
标志被设置，意思是说使其等于 1，被清除，则使其等于 0。

译注 5
1. 更改寄存器 EAX 为 00000000。

2. 将 4010000 处改为 ADD EAX，1。

随文附件

1. CrackMe：ollydbg01-Crackme.zip

翻译说明：

该系列教程目前官方已更新到第 47 章。本文原文为俄语，译者不才，斗胆翻译，采用了能用的所有手段。虽经本人严加审校，但难免讹误。有些词句加入了译者的理解，所以部分内容可能与原文有所出入。翻译本文也是译者学习的过程，所以错误在所难免。如发现错误，敬请指正，以免误人子弟。

该系列教程链接：http://wasm.ru/series.php?sid=17

本文原文链接：http://wasm.ru/article.php?article=ollydbg03