逆向平台Binary Ninja介绍

翻译

小蓝人敌法 于 2018-01-17 20:01:22 发布 2873 收藏 2

分类专栏： Android 文章标签： 二进制 Binary Ninja 逆向工程


Android 专栏收录该内容

25 篇文章 3 订阅

订阅专栏

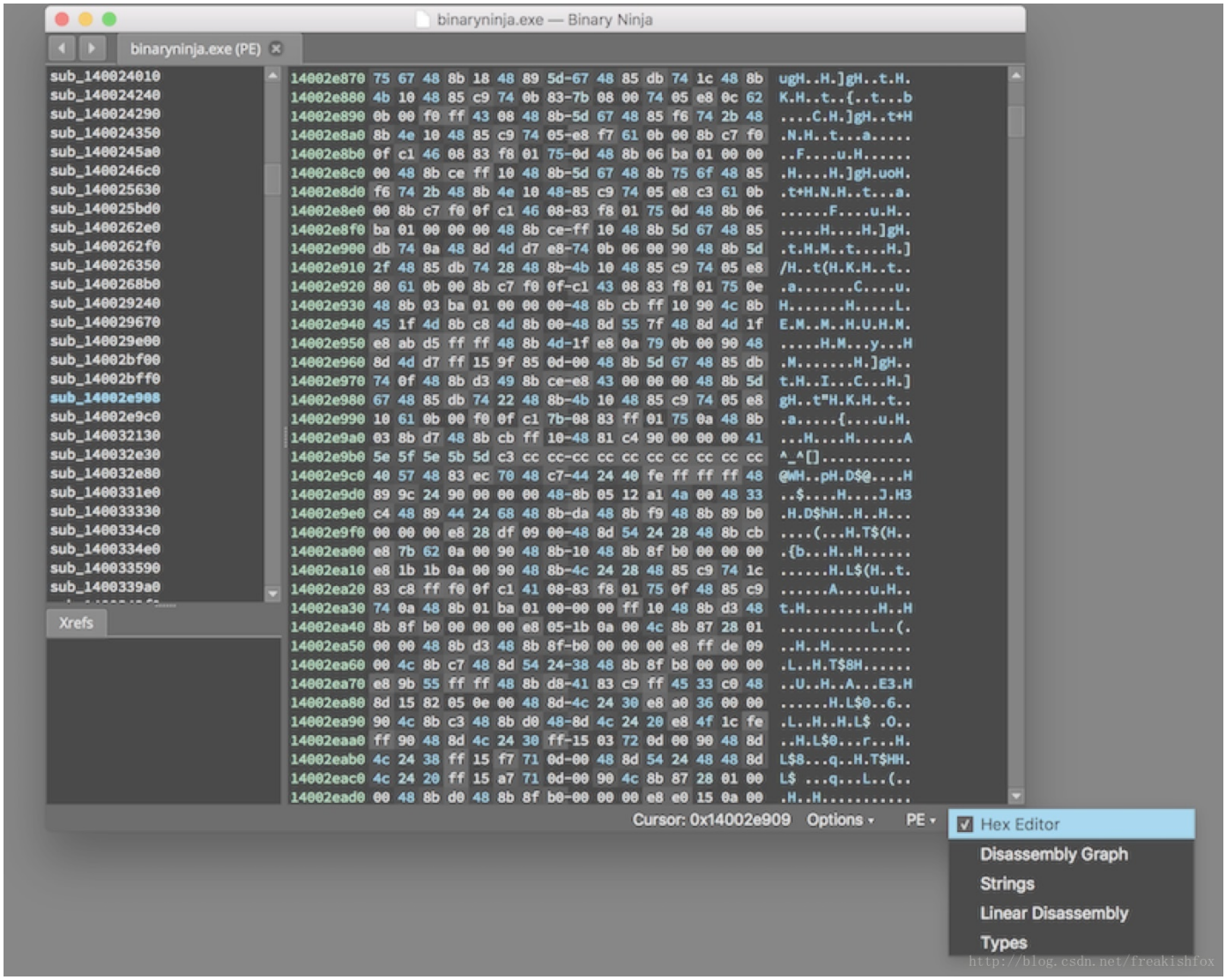
逆向平台Binary Ninja介绍

概述

- **我们经常会收到这样的问题，比如：**
    1. Binary Ninja比xx软件好在哪里呢？
    2. Binary Ninja这个软件最主要的作用是什么呢？
- **对于上面的两个问题，简要的回答起来是这样的：**
    1. Binary Ninja相对于其他类似软件来说， 能够更快、更便捷的修改二进制文件
    2. Binary Ninja最主要的作用是提供了一个可供操作二进制文件的平台，你甚至还可以在平台的基础上基于API来编写更方便的脚本和插件
- **那么我们为什么要修改现有的二进制文件呢？总结起来大致有一下几种情况：**
    1. 无需长时间的等待编译完毕，可以快速的测试修改结果
    2. 对程序进行黑盒评测
    3. 维护或者升级老程序（一般是没有源码对情况）
    4. 通过修改二进制文件学习现有的一个软件系统或者修复软件问题
    5. 修复第三方库的安全问题
    6. 反混淆
- 在Binary Ninja这个平台上，提供了许多的方式来修改二进制文件，大致可以分为低级和高级模式两种， 其中低级模式主要就是原始码的十六进制编辑和汇编模式， 高级模式可以使用内置的C编译器直接书写C代码来进行操作。

HEX编辑模式

- 先来一睹为快，大致操作界面如下：



- 这个是最直观的修改方式，也是最简易的一种修改方式。如上图所示，可以通过右下角的菜单按钮或者热键H打开(^ + H)，这里要注意一下，这里介绍的热键如果没有效果，可以去参考 用户文档.
- 这种编辑模式下，可以一边修改，一边实时的看修改结果，从而到达比较高的修改测试效率，一次修改过程大致的操作步骤如下：

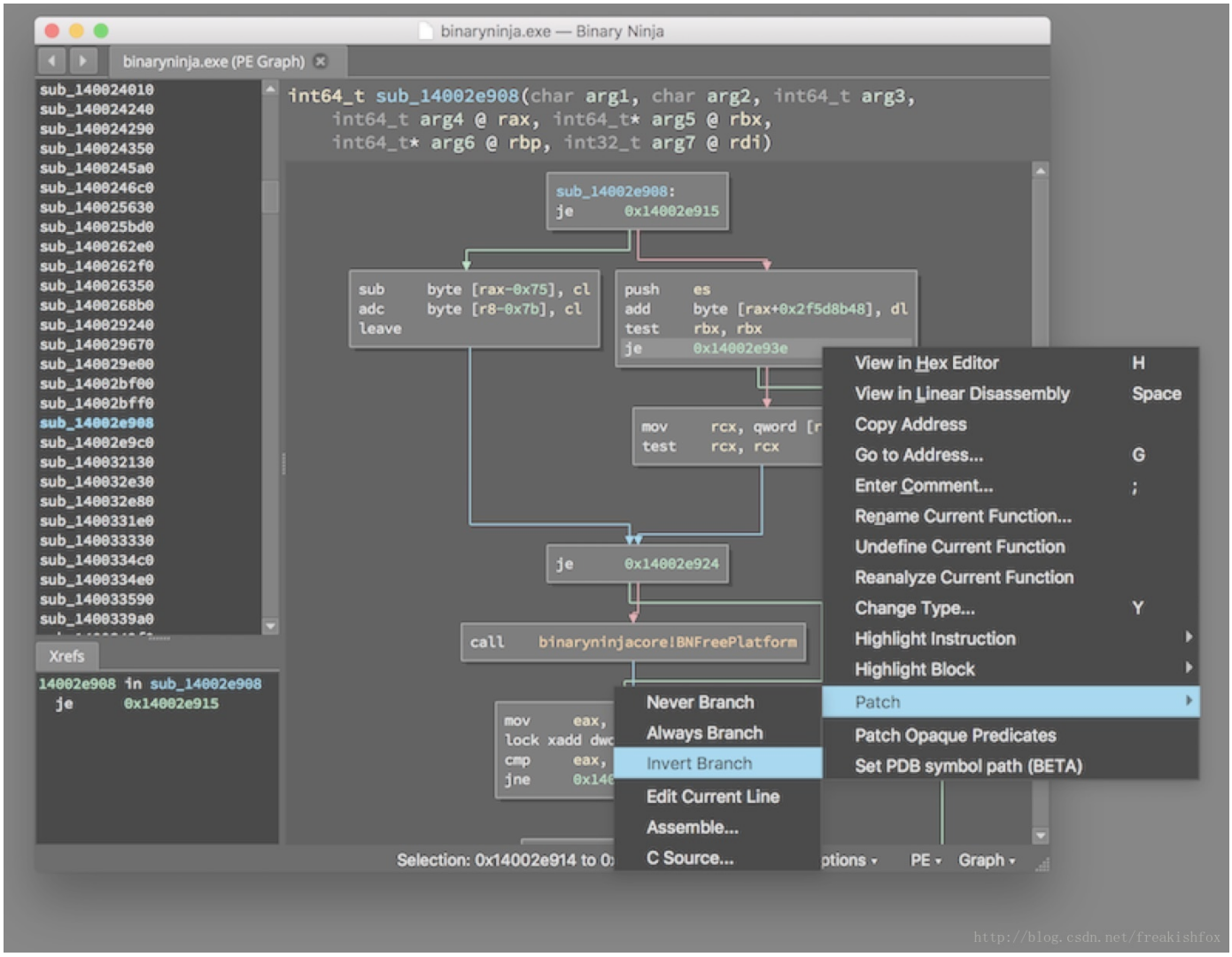
    1. 先在图形视图或者线性汇编试图中找一个准备修改的函数
    2. 使用菜单 view -> Split进行一次屏幕视图拆分
    3. 现在可以随意调整拆分的视图，看着舒服就行，把其中一个视图切换到hex编辑模式，并修改一下上面选定的函数。你这边修改，另一个视图中就会看到同步的修改结果展示，一般情况下，如果修改一个比较大的函数的时候，操作反应可能会慢一点。编辑器也支持大块的复制粘贴操作。

结构体实时修改

- 这个实时预览功能说起来比反汇编字节码有用多了， 这里给出一个实时修改结构体并查看结构体解码结果的的演示视频：
- 这里本来是一段演示视频，但是是youtube的视频源，没有办法观看，请在原文链接中翻墙观看（sorry)，视频地址是：http://www.youtube.com/embed/sCKiG_xdNSc

点两下鼠标就搞定

- 上面介绍的方法，需要你对二进制的内容甚至是系统架构有所了解，对于大多人来说可能还有点困难，但是在Binary Ninja这个平台上，你甚至不必了解那么多的知识也可以通过平台来完成二进制文件的修改工作。比如，专有的 右键修改菜单，有了这个菜单的帮助，你只需要点两下鼠标就可以了：

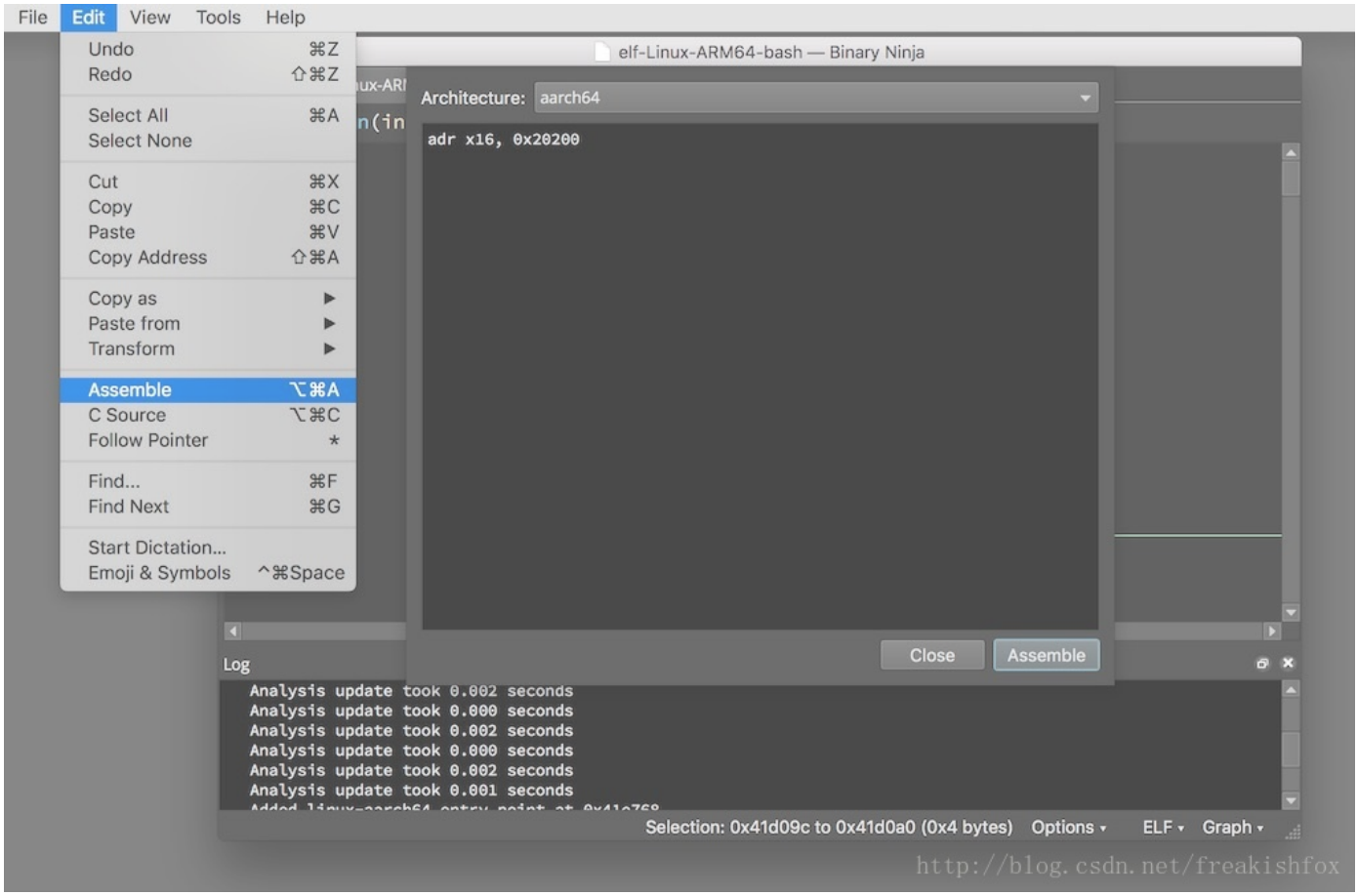


- 这个弹出菜单里面有不少专门用来**修改条件分支跳转**的功能，通过这些选项你可以强制性的把一个条件分支改成 永远跳转或者永不跳转。使用 Invert Brach功能，可以进行条件反转，这些功能使用起来很简单命令，但是确实能很方便快速的实现你想要的结果。
- 还有其他的一些有用的功能，比如把一些**汇编指令给Nop掉**，也就是说，你选中一条指令，然后用这个功能来进行Nop指令替换，这里系统还会自动的进行剩余空间填充，比如原先的指令占4个直接，你直接把这条指令给Nop调用，那么剩下的3个字节系统会自动的给你也填充成Nop指令，这个功能对于那种变长指令系统还是很实用的。
- **这里需要注意的一点是，** 如果你把鼠标放到一个条件分支上，这个时候菜单里面可能没有 Convert to NOP这个选项，因为和 as Never功能是一样的，软件显示一个就够了。

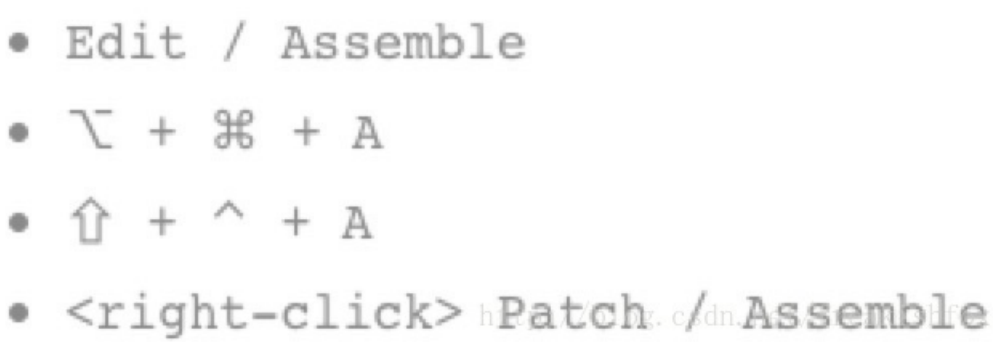
单行编辑

- 在**Patch**这个菜单中，还有一些隐藏的菜单项，比如单行编辑功能。使用单行编辑功能，你可以快速的以汇编形式修改单行指令。选择一条需要修改的指令，按快捷键e或者使用右键菜单Patch/Edit current line选项，然后就可以看到选中那条汇编指令就变成了可编辑状态，编辑完成之后按回车即可，Binary Ninja会自动把修改结果应用到二进制文件中去。
- 如果编辑汇编的时候，新指令比原先的指令长度小的话，工具会自动填充余下的空间为NOP指令，比如在 x86 x64这些变长指令系统上。
- 附加说明：工具会尽最大努力保证汇编和反汇编的正确性， 如果遇到指令错误或者不支持的指令，可以联系我们，把这些特殊的情况跟我们沟通一下

In-Depth汇编

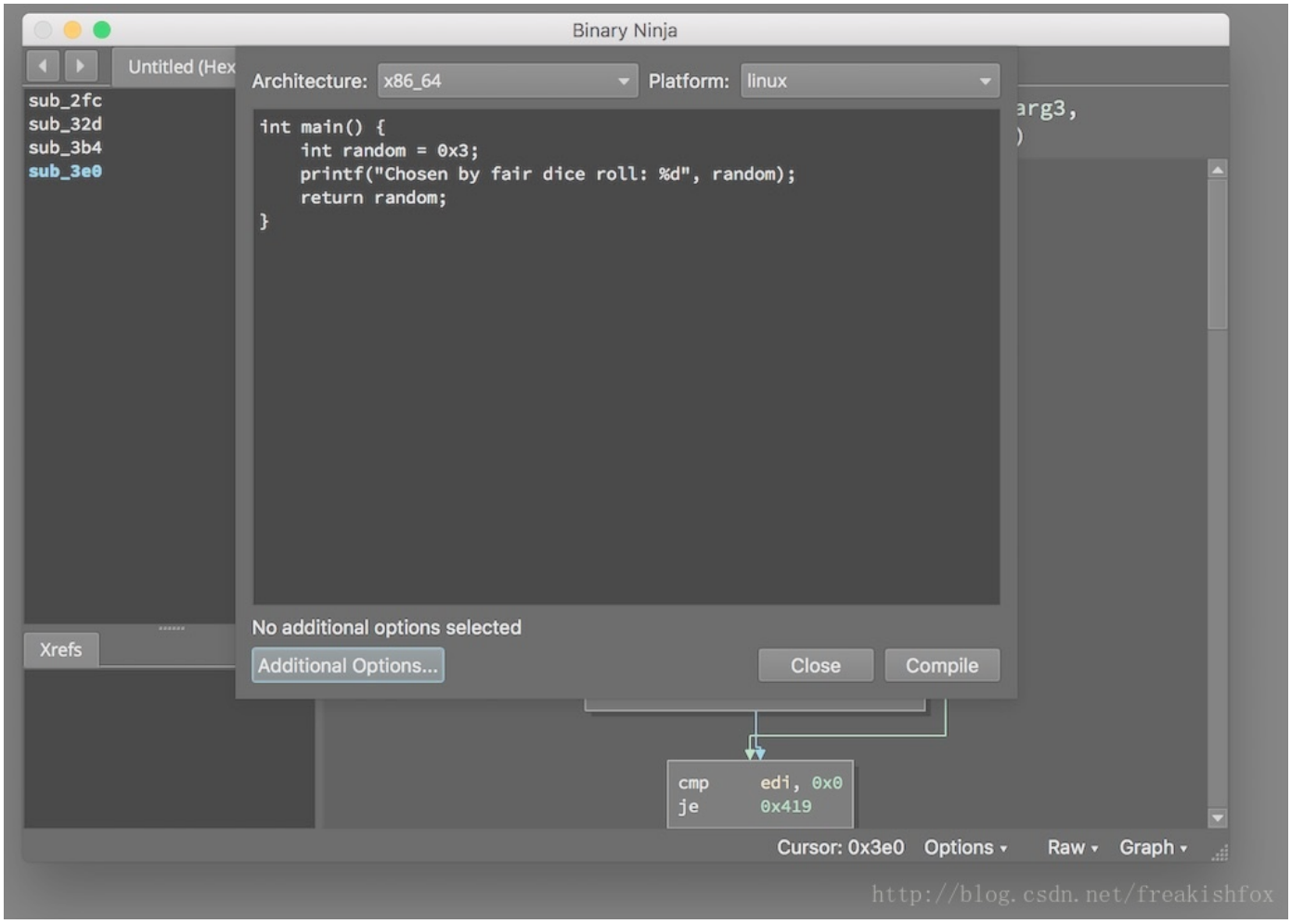


- 这种方式可以简单的理解为多行汇编的操作入口，操作路径是：



- 目前，对于非x86/x64平台使用我们自定义的LLVM进行汇编，x86/x64使用yasm引擎

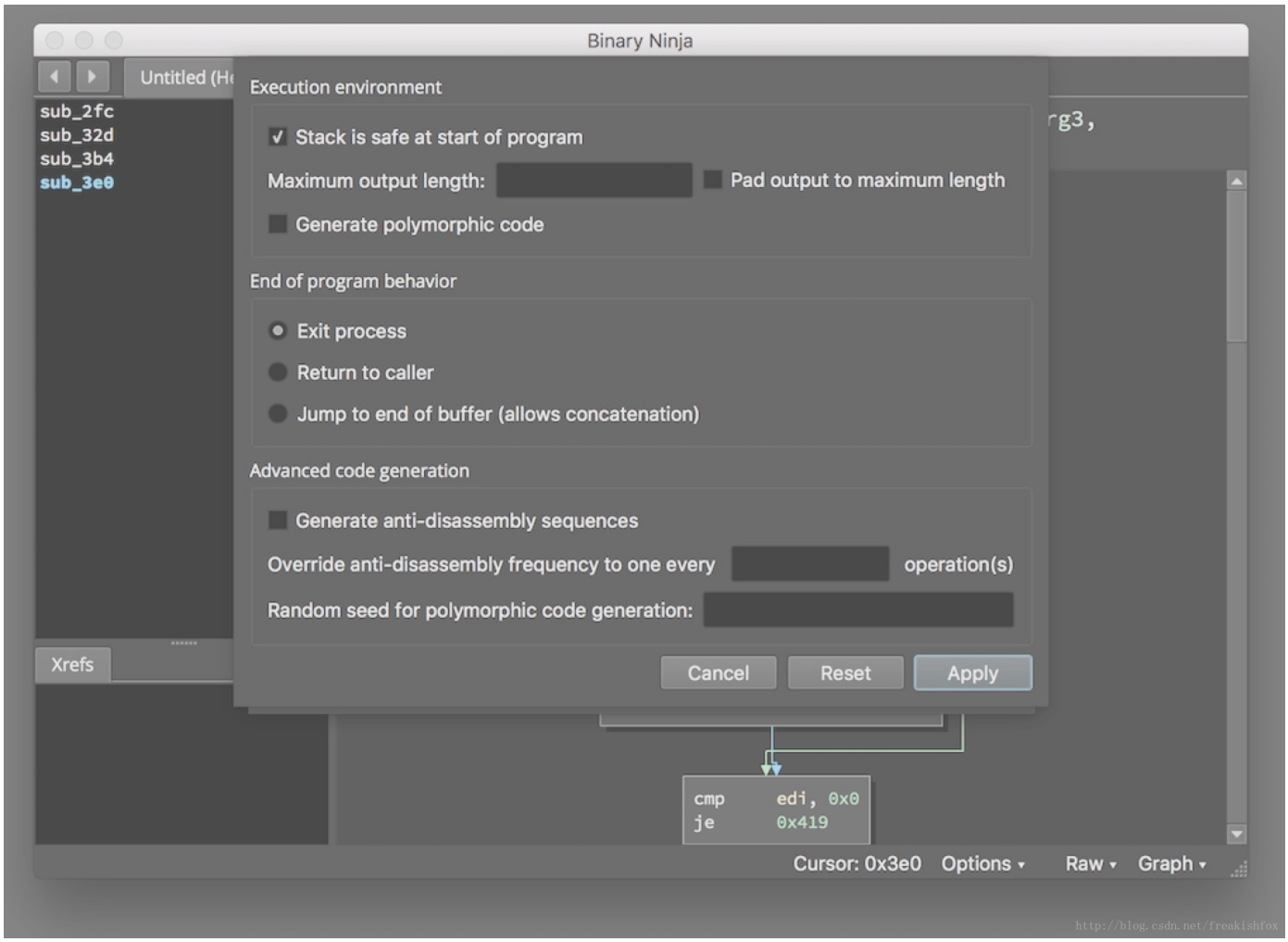
居然还可以直接用C语言？



- Binary Ninja平台上的秘密武器，Shellcode Compiler算是一个，我们简称它为SCC, SCC是一个迷你C编译器，目前这个编译器支持多种系统平台架构，这个小编译器在编写小型可注入代码（一般认为是那种位置无关的代码）或者小范围修改代码的时候很有用，在这些场景下用这个编译器来写代码比写汇编方便多了。
- SCC是一个自定义的从零开始构建的C编译器，目前支持 MacOS, Windows, Linux 和 FreeBSD这些系统，CPU架构支持x86,x86_x64, PPC32, MIPS32, ARM(不包含Thumb)以及quark(一种自定义的CPU架构)。SCC在编译的时候会保留函数的原型， 然后会对编译的结果进行基础块打乱和函数内寄存器随机打乱来模糊函数原型，同时也会加入一些anti反汇编的字节（注：不确定这么理解对不对，这里给出原文：SCC can generate code with signature and analysis evasions built in. It will randomize placement of basic blocks and randomize its register allocator to evade naive signatures. It will also add anti-disassembly sequences.）
- 在Windows是平台上，Binary Ninja显示出了更高的工作效率。比如这样一个场景，如果你想在编译器中编写一些代码，代码中需要使用到一些API函数，如果用汇编，你就要知道函数的地址然后才能去调用，但是Windows系统版本不同的话，这个API地址又是不固定的，所以会很烦。但是现在又了这个C编译器就不一样了，你直接调用API就可以了， 编译器会自动给你解决符号问题，比如你可以像下面这样写代码：

```
1  void* __stdcall ShellExecute(void* hwnd,
2                               const char* op,
3                               const char* filename,
4                               const char* parameters,
5                               const char* directory,
6                               int showcmd) __import("shell32");
7  void main() {
8      MessageBoxA(NULL, "Backdoor Loading", "Loading up your backdoor now!", 0);
9      ShellExecute(NULL, NULL, "notepad.exe", NULL, NULL, 0);
10 }
```

- 你看，不需要任何定义，你就可以直接使用MessageBoxA. 具体可以调用的所有内建函数，详细可以看这个列表, 对于不在这个列表里面的函数，只需要模仿ShellExecute函数在上面声明一下就可以使用了。
- 毫无疑问，SCC这个功能使用起来那是相当的让人震撼（译者注：感觉也没多牛逼），比手工汇编方便太多了，另外还有一些属性可以自己挖掘下。由于这个C编译器本来的设计目的是用来写小型的shellcode用的，因此很多常规编译器的功能还没有，下面展示一下配置界面选项：



不足之处

对于目前的Binary Ninja版本来说，有2个需要提高的地方：

1. 智能Patch机制
2. 编译器和汇编器 对符号/库 的上下文信息感知

这里所说的智能Patch机制实际上是说，当你使用一大段Patch代码的时候，不需要去担心这么一大段代码会不会把其他的代码给覆盖掉， 而是工具内部应该提供一种机制，在老代码和Patch代码之间使用一种代码桥接机制， 就是老代码先跳到代码桥，再跳到你的大段Patch代码，甚至，智能机制还能够实现多个segment的修改。目前 BNHook 已经实现了一部分上述特性， 关心的话可以去看看。

- 如果能能够感知二进制文件中的类型信息和符号信息，那么编译器和汇编器的能力又会大大的提升，因为有了这些信息，在写代码的时候就有大量的符号信息可以直接在代码中使用，而不用像以前那样写死地址信息了。拥有这样高级特性的的编译器和汇编器我们还没开发出来，但是请继续关注我们，Binary Ninja说不定哪天就支持了。

保存修改

- 在Binary Ninja可以保存两部分内容：
    1. Binary Ninja对打开的文件等分析结果，还包括你的修改，注释信息等，你可以保存成分析数据库（注：就像IDA里面的 .idb），操作路径是 File/Save Analysis Database, 这个把上述信息保存到一个后缀是 .bndb的sqlite数据库里面（自定义修改过的sqlite数据库）
    2. 另一个就是保存修改到原始的文件，操作路径是 File/Save Contents

综述

- Binary Ninja对于经常需要修改二进制文件的人来说却是是一个不错的选择，既有快速方便的UI操作界面，还有内置强大的C编译引擎的支撑。欢迎大家给我们留言或者加入slack或者其他的方式和我们联系。

译者

- 原文地址：https://binary.ninja/2017/12/15/change-is-in-the-air.html
- 部分无法确定的部分，我采用了原文信息，请参考
- 感谢 看雪 **@Edlie** 提供原文
- 本文 由看雪 **敌法@freakish** 提供翻译