> **一些要说的话**
>
> 本人是在VMWare虚拟机中执行的Peach编译步骤，所使用的原始镜像为；
> *ubuntu-16.04.6-server-i386.iso*
> Peach的编译所需部件较多，请在尝试编译时保持耐心
>
> 本文所有步骤参考自下列网址：
> [1]. Gitlab官方仓库：*https://gitlab.com/gitlab-org/security-products/protocol-fuzzer-ce.git*
> [2]. BruceWayne09的博客：ubuntu 16.04 安装peach - BruceWayne09 - 博客园 (cnblogs.com)

前置条件

关于Linux编译执行前的环境，Peach官方给出了下图中所示的要求，这些步骤所指代的工具在linux中的安装步骤不分先后，但个人建议初学者（本人是初学者qwq）在成功安装所给出的8条工具后（除了*mono-complete* 之外，从 *Ubuntu16.04* 到 *Intel Pin* 一共8条）就*拍摄系统快照*，以免在编译的最后一步切换 *mono-complete* 版本时出了差错等等原因导致无法正确退回前一步。

Linux Build Prerequisites:

- Ubuntu 16.04 recommended
- gcc and g++
- g++-multilib (for x86 cross compiling)
- python 2.7
- ruby 2.3
- doxygen, java, xmllint, xsltproc
- mono-complete v4.8.1
- nodejs and tsc v2.8
- Download Intel Pin (see 3rdParty/pin/README.md)

具体步骤

> **Attention**

> 下述将会频繁用到apt命令，而原始apt来源下载过慢，请参照下述博客更换为对应系统版本的文件
> [ubuntu 提升下载速度—更换阿里源 - 知乎 (zhihu.com)]

1. 第一步：安装 *gcc* 和 *g++*

   zqb@ubuntu:/$ **sudo apt install gcc**
   zqb@ubuntu:/$ **sudo apt install g++**

2. 第二步：安装 *g++-multilib*

   zqb@ubuntu:/$ **sudo apt install g++-multilib**

3. 第三步：安装 *python2.7*

   zqb@ubuntu:/$ **sudo apt install python2.7 zqb@ubuntu:/$** which python2.7**zqb@ubuntu:/$**sudo ln -s /usr/bin/python2.7 /usr/bin/python`

4. 第四步：安装 *ruby2.3*

   zqb@ubuntu:/$ **sudo apt install ruby2.3**

5. 第五步：安装 *doxygen、java、xmllint、xsltproc*

   zqb@ubuntu:/$ **sudo apt install doxygen**
   zqb@ubuntu:/$ **sudo apt install openjdk-8-jdk**
   zqb@ubuntu:/$ **sudo apt install libxml2-utils**
   zqb@ubuntu:/$ **sudo apt install xsltproc**

6. 第六步：安装 *nodejs、tsc*

   zqb@ubuntu:/$ **sudo apt install nodejs**
   zqb@ubuntu:/$ **sudo apt install npm zqb@ubuntu:/$** sudo npm install -g typescript@2.8**zqb@ubuntu:/$**sudo ln -s /usr/bin/nodejs /usr/bin/node`

7. 第七步：安装 *Intel Pin*

   在windows中点击该链接下载Intel Pin工具包linux版，之后可以用文件传输工具发往你的linux主机中，并解压：
   zqb@ubuntu:~$ **tar -zxvf pin-3.19-98425-gd666b2bee-gcc-linux.tar.gz**

   更改解压后的文件名：
   zqb@ubuntu:~$ **mv pin-3.19-98425-gd666b2bee-gcc-linux pin-3.19-98425-gcc-linux**

8. 第八步：*下载peach项目*

将peach项目下载并拷贝到linux中，之后将第七步解压后的 **pin-3.19-98425-gcc-linux** 拷贝到peach项目中 **3rdParty/pin/** 目录下，命令如下（请根据你自己的路径更改命令）：

zqb@ubuntu:~$ **mv pin-3.19-98425-gcc-linux/ protocol-fuzzer-ce-main/3rdParty/pin/**

更改peach项目文件 **protocol-fuzzer-ce-main/core/BasicBlocks/bblocks.cpp** ，添加一行宏定义：

```
zqb@ubuntu:~/protocol-fuzzer-ce-main/core/BasicBlocks$ cat bblocks.cpp

// Authors:
//   Michael Eddington (mike@dejavusecurity.com)

//
// PIN Tool to find all basic blocks a program hits.
//  This PIN Tool is intended for use with Peach.
//
//  Code based on examples from PIN documentation.
//

#define UNUSED_ARG(x) x;

#define STATIC_ASSERT(expr) typedef char __static_assert[expr ? 1 : -1] __attribute__((__unused__));

#if defined(_MSC_VER)
#pragma warning(disable: 4127) // Conditional expression is constant
```

- [x] **建议保存一次系统快照**

9. 第九步：安装 *mono-complete*

   （下述指令参照[这个网址](Download - Stable | Mono (mono-project.com)))

   zqb@ubuntu:/$ **sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF**

   zqb@ubuntu:/$ **sudo apt install apt-transport-https ca-certificates**

   zqb@ubuntu:/$ **echo "deb https://download.mono-project.com/repo/ubuntu stable-xenial main" | sudo tee /etc/apt/sources.list.d/mono-official-stable.list**

   zqb@ubuntu:/$ **sudo apt update**

10. 第十步：**开始编译**（先进入你自己的peach目录下，即第八步中的 **protocol-fuzzer-ce-main** ）

    zqb@ubuntu:~/protocol-fuzzer-ce-main$ **sudo python waf configure**

    效果如下（只要有一个linux开头的variant是available就可以）：

```
zqb@ubuntu:~/protocol-fuzzer-ce-main$ sudo python waf configure
Configuring variant doc                  : Available - Missing Features: asciidoctor-pdf,webhelp
Configuring variant linux_x86            : Available
Configuring variant linux_x86_64         : Available
Configuring variant osx                  : Not Available - Unsupported build host
Configuring variant win_x86              : Not Available - Unsupported build host
Configuring variant win_x64              : Not Available - Unsupported build host
'configure' finished successfully (1m7.532s)
```

    zqb@ubuntu:~/protocol-fuzzer-ce-main$ **sudo python waf build**

效果如下：

```
Waf: Leaving directory `/home/zqb/protocol-fuzzer-ce-main/slag/linux_x86_64_release'
Waf: Leaving directory `/home/zqb/protocol-fuzzer-ce-main/slag'
'build' finished successfully (2m17.620s)
zqb@ubuntu:~/protocol-fuzzer-ce-main$
```

删除第九步中下载的mono-complete v6.12，参照下述指令安装mono v4.8.1:

mono 卸载mono6

```
sudo apt-get autoremove mono-devel -y
```

C# 复制 全屏

编译安装mono4.8.1

```
wget https://download.mono-project.com/sources/mono/mono-4.8.1.0.tar.bz2
tar -jxvf mono-4.8.1.0.tar.bz2
cd mono-4.8.1
```

按照README.md操作

```
./autogen.sh
make get-monolite-latest
sudo make
sudo make install
```

检查mono版本

```
mono -V
Mono JIT compiler version 4.8.1 (Stable 4.8.1.0/22a39d7 2021年 07月 14日 星期三 14:30:01 CST)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-project.com
        TLS:           __thread
        SIGSEGV:       altstack
        Notifications: epoll
        Architecture:  amd64
        Disabled:      none
        Misc:          softdebug
        LLVM:          supported, not enabled.
        GC:            sgen
```

由于我的电脑缺少一些工具，执行完其中的 **./autogen.sh** 命令后，会出现下面的打印信息，根据你自己的打印信息，用 **sudo apt install xxx** 安装缺少的组件即可：

```
zqb@ubuntu:~/mono-4.8.1$ ./autogen.sh

**Error**: You must have `autoconf' installed to compile Mono.
Download the appropriate package for your distribution,
or get the source tarball at ftp://ftp.gnu.org/pub/gnu/

**Error**: You must have `libtoolize' installed to compile Mono.
Get ftp://ftp.gnu.org/gnu/libtool/libtool-1.2.tar.gz
(or a newer version if it is available)

**Error**: You must have `automake' installed to compile Mono.
Get ftp://ftp.gnu.org/pub/gnu/automake-1.3.tar.gz
(or a newer version if it is available)
```

安装完成后再次执行 `./autogen.sh` 命令，打印信息如下：

```
./configure: line 27410: libtool: command not found
checking for the soname of libX11.so... configure: WARNING:
checking if the tls_model attribute is supported... yes
checking malloc.h usability... yes
checking malloc.h presence... yes
checking for malloc.h... yes
checking for cmake... no
configure: error: "cmake not found"
```

发现还缺少两个组件，再次安装：`sudo apt install libtool-bin`、`sudo apt install cmake`

再次执行 `./autogen.sh` 命令，成功信息如下：

```
config.status: executing default commands

        mcs source:    mcs

   Engine:
        Host:          i686-pc-linux-gnu
        Target:        i686-pc-linux-gnu
        GC:            sgen and Included Boehm GC with typed GC and parallel mark
        TLS:           __thread
        SIGALTSTACK:   yes
        Engine:        Building and using the JIT
        BigArrays:     no
        DTrace:        no
        LLVM Back End: no (dynamically loaded: no)

   Libraries:
        .NET 4.x:        yes
        Xamarin.Android: no
        Xamarin.iOS:     no
        Xamarin.WatchOS: no
        Xamarin.TVOS:    no
        Xamarin.Mac:     no
        mobile_static:   no
        JNI support:     IKVM Native
        libgdiplus:      assumed to be installed
        zlib:            system zlib
        BTLS:            yes (i386)


Now type `make' to compile
zqb@ubuntu:~/mono-4.8.1$
```

依次执行 `sudo make` 和 `sudo make install` ，等待时间较长，结果如图：

```
zqb@ubuntu:~/mono-4.8.1$ mono -V
Mono JIT compiler version 4.8.1 (Stable 4.8.1.0/22a39d7 Fri Feb  2 17:38:46 PST 2024)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-project.com
        TLS:           __thread
        SIGSEGV:       altstack
        Notifications: epoll
        Architecture:  x86
        Disabled:      none
        Misc:          softdebug
        LLVM:          supported, not enabled.
        GC:            sgen
```

之后再次进入peach项目根目录下，执行 **sudo python waf install**

```
linux_x86_64_release | Install | Peach.Pro.Test.WebApi.exe.mdb | bin/Peach.Pro.Test.WebApi.exe.mdb
linux_x86_64_release | Install | CrashTestDummy.exe | bin/CrashTestDummy.exe
linux_x86_64_release | Install | BouncyCastle.Crypto.dll | bin/BouncyCastle.Crypto.dll
linux_x86_64_release | Install | PeachService.exe.mdb | bin/PeachService.exe.mdb
linux_x86_64_release | Install | PeachService.exe | bin/PeachService.exe
linux_x86_64_release | Install | Peach.Pro.ComContainer.exe | bin/Peach.Pro.ComContainer.exe
linux_x86_64_release | Install | BouncyCastle.Crypto.Tests.dll.mdb | bin/BouncyCastle.Crypto.Tests.dll.mdb
linux_x86_64_release | Install | Peach.Pro.Test.dll | bin/Peach.Pro.Test.dll
linux_x86_64_release | Install | BouncyCastle.Crypto.Tests.dll | bin/BouncyCastle.Crypto.Tests.dll
Waf: Leaving directory `/home/zqb/protocol-fuzzer-ce-main/slag/linux_x86_64_release'
Waf: Leaving directory `/home/zqb/protocol-fuzzer-ce-main/slag'
'install' finished successfully (22.312s)
zqb@ubuntu:~/protocol-fuzzer-ce-main$
```

成功编译！目录如下：

```
zqb@ubuntu:~/protocol-fuzzer-ce-main$ ll
total 144
drwxrwxrwx 11 zqb  zqb   4096 Feb  2 17:52 ./
drwxr-xr-x 10 zqb  zqb   4096 Feb  2 16:48 ../
drwxrwxr-x 19 zqb  zqb   4096 Feb  1 17:25 3rdParty/
drwxrwxr-x  8 zqb  zqb   4096 Feb  1 17:25 build/
-rw-rw-r--  1 zqb  zqb   2209 Feb  1 17:25 CONTRIBUTING.md
drwxrwxr-x  7 zqb  zqb   4096 Feb  1 17:26 core/
drwxrwxr-x  7 zqb  zqb   4096 Feb  1 17:26 docs/
-rw-rw-r--  1 zqb  zqb   2092 Feb  1 17:22 .gitignore
-rw-rw-r--  1 zqb  zqb   1087 Feb  1 17:26 LICENSE
-rw-r--r--  1 root root  2639 Feb  2 08:06 .lock-waf_linux2_bui
-rw-rw-r--  1 zqb  zqb  42916 Feb  1 17:26 NOTICE
drwxrwxr-x  7 zqb  zqb   4096 Feb  2 17:53 output/
drwxrwxr-x  6 zqb  zqb   4096 Feb  2 08:06 paket/
-rw-rw-r--  1 zqb  zqb     52 Feb  1 17:26 paket.cmd
-rw-rw-r--  1 zqb  zqb     50 Feb  1 17:26 paket.sh
-rw-rw-r--  1 zqb  zqb     49 Feb  1 17:26 peach-pro.properties
drwxrwxr-x 17 zqb  zqb   4096 Feb  1 17:29 pro/
-rw-rw-r--  1 zqb  zqb   9301 Feb  1 17:26 README.adoc
drwxr-xr-x  8 root root  4096 Feb  2 08:19 slag/
drwxrwxr-x  3 zqb  zqb   4096 Feb  1 17:29 tools/
-rwxrwxrwx  1 zqb  zqb    852 Feb  1 17:29 waf*
-rw-rw-r--  1 zqb  zqb     93 Feb  1 17:29 waf.bat
-rw-rw-r--  1 zqb  zqb    445 Feb  1 17:29 win_x64_debug.nunit
-rw-rw-r--  1 zqb  zqb    837 Feb  1 17:29 wscript
zqb@ubuntu:~/protocol-fuzzer-ce-main$
```

编译好的peach项目就在output目录下，自由选择使用。