bitblaze是由加州伯克利分校的一个团队研发 的linux平台下的一个二进制分析平台，它由静态分析组件vine和动态分析组建temu组成。vine的安装和配置相对简单一些，而temu的使用则需要qemu的支持，它是基于qemu来运行虚拟系统的。

qemu的版本现在已经到14.X 了好像，不过temu支持的qemu的版本为0.9.1，所以这里我们在安装qemu的时候不直接使用apt-get install在软件仓库中安装，只能自己下载0.9.1的版本进行安装，而该版本的qemu对应的加速器版本为kqemu1.3。

        此外，在此首先说明的是，本次安装我所选择的host系统为ubuntu10.04 LTS这个版本，其实在bitblaze的说明文档中开发者的开发平台是9.04，10.04这里我安装运行没有出现不兼容的问题，其实接用12.10安装也能正常使用。

# vine的安装与配置

## 需要的软家包

# For compiling C++ code:

sudo apt-get install g++

# For OCaml support:

sudo apt-get install ocaml ocaml-findlib libgdome2-ocaml-dev camlidl \

                                libextlib-ocaml-dev ocaml-native-compilers

# Ocamlgraph >= 0.99c is required; luckily the version in Ubuntu 9.04 is now new enough.

sudo apt-get install libocamlgraph-ocaml-dev

# For the BFD library:

sudo apt-get install binutils-dev

# For building documentation:

sudo apt-get install texlive texlive-latex-extra transfig hevea

上面的这段你可以直接在vine的源码主文件夹下/doc文件夹下找到，就在该文件夹下的install-vine-release.sh文件中。

**执行最后一步可能会遇到的问题：**

**E: Sub-process /usr/bin/dpkg returned an error code (1)错误**

**解决方法：**

sudo mv /var/lib/dpkg/info /var/lib/dpkg/info2

sudo mkdir /var/lib/dpkg/info

sudo apt-get update -f

sudo apt-get -f install

## 安装vine

tar xvzf /tmp/vine-1.0.tar.gz

(cd vine-1.0 && ./configure)

(cd vine-1.0 && make)

(cd vine-1.0/doc/howto && make doc)

## 注意

**make的时候会出错这时候需要通过修改vine-1.0/libasmir/src/disasm目录下 的asm\_program.cpp文件，在文件添加头文件部分中#include "libiberty.h"前面加入一行宏定义#define HAVE\_DECL\_BASENAME  1，保存文件后再进行编译就可以通过了。**

至此，vine的安装与配置就算是完成了，它可以读取动态组建temu生成的trace文件，vine提供了一个trace示例文件，可以用来检测你的vine的安装是否成功。具体步骤如下：

        cd bitblaze/vine-1.0

        $ ./trace\_utils/trace\_reader -trace examples/five.trace | grep T1

部分运行结果如下：



# temu的安装与配置

temu的使用是需要qemu支撑的，所以在安装temu之前我们首先需要安装qemu，同时也提到过，temu目前支持的qemu版本为0.9.1，所以这里需要安装0.9.1版本的qemu，而与该版本的qemu对应的加速器kqemu的版本为kqemu-1.3.Opre11。

       temu支持的gcc的版本是3.4版本的gcc，而我使用的ubuntu10.04默认的gcc版本是4.4的，所以在安装配置之前需要首先将gcc降级到3.4这个版本。

## 在Linux 下编译安装 QEMU

### [系统环境及版本](http://butteredcat.yo2.cn/articles/qemu-compilation.html" \l "top1)

Linux发行版本：ubuntu 8.04

安装QEMU-0.9.1

KQEMU-1.3.0(pre11)

### [准备工作](http://butteredcat.yo2.cn/articles/qemu-compilation.html#top2)

在[QEMU主页](http://bellard.org/qemu/" \t "_blank)下载[源代码(Source code)](http://bellard.org/qemu/qemu-0.9.1.tar.gz)，并解压到/tmp目录下（假设源代码保存在~/Setup）。

$ tar zvxf ~/Setup/qemu-0.9.1.tar.gz -C /tmp

### 建立编译环境（gcc）

$ sudo apt-get install build-essential

编译安装QEMU，最重要的工具就是[gcc](http://gcc.gnu.org/gcc)。在大多数主机，特别是x86架构的PC机上，gcc 4.x是不被QEMU所支持的。build-essential 中包含的是 gcc 4.x，为了正确编译，需要低版本的gcc 3.x(gcc32 或 gcc34)，低版本安装后，QEMU的配置脚本(configure script)在配置时会自动搜寻到。我安装的是gcc 3.4:

$ sudo apt-get install gcc-3.4 cpp-3.4

[**手动替换GCC版本**](http://blog.csdn.net/braveching/article/details/5735821)**，ubuntu默认为4.4，需要降低版本**

**http://blog.csdn.net/braveching/article/details/5735821**

当我们的系统里面会有2个以上版本的gcc时，系统会缺省的默认一个gcc版本，当然我们可以更改系统的默认配置，来降低gcc、g++的版本以满足不同的需求。

1.查看GCC的版本信息

**[c-sharp]** [view plaincopy](http://blog.csdn.net/braveching/article/details/5735821)

1. $ls /usr/bin/gcc\*
2. $/usr/bin/gcc  /usr/bin/gcc-4.1  /usr/bin/gcc-4.4  /usr/bin/gccbug-4.1

 可以看到，已经有了2个版本的gcc，安装成功。

2.增加gcc 4.1.2和gcc 4.4.1可选项

**[c-sharp]** [view plaincopy](http://blog.csdn.net/braveching/article/details/5735821)

1. $sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4 40

 update-alternatives: 使用 /usr/bin/gcc-4.1 来提供 /usr/bin/gcc (gcc)，于 自动模式中

**[c-sharp]** [view plaincopy](http://blog.csdn.net/braveching/article/details/5735821)

1. $sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.1 30

 update-alternatives: 使用 /usr/bin/gcc-4.4 来提供 /usr/bin/gcc (gcc)，于 自动模式中

3.设置默认编译器，如下:

**[c-sharp]** [view plaincopy](http://blog.csdn.net/braveching/article/details/5735821)

1. $sudo update-alternatives --config gcc

有 2 个选项可用于替换项 gcc (提供 /usr/bin/gcc)

**[c-sharp]** [view plaincopy](http://blog.csdn.net/braveching/article/details/5735821)

1. $
2. 选择       路径            优先级  状态
3. ------------------------------------------------------------
4. \* 0            /usr/bin/gcc-4.4   40        自动模式
5. 1            /usr/bin/gcc-4.1   30        手动模式
6. 2            /usr/bin/gcc-4.4   40        手动模式

9. 要维持当前值[\*]请按回车键，或者键入选择的编号：

键入你需要的版本代号，按回车即可

这时为了保险起见，用命令查询当前gcc版本号

**[c-sharp]** [view plaincopy](http://blog.csdn.net/braveching/article/details/5735821)

1. $gcc -v

这样就完成了安装和默认项配置，于是可以默认使用GCC 4.1.2编译程序了,其它版本的类似……

g++的版本变换同gcc

安装QEMU需要[SDL](http://www.libsdl.org/)开发库的支持，可以在其下载页面上下载源代码编译安装，但是推荐在apt中安装，不容易出问题：

$ sudo apt-get install libsdl1.2-dev

另外，还需要安装zlib和ESD：

sudo apt-get install zlib1g-dev zlib1g-dbg libesd0-dev

### [编译安装QEMU](http://butteredcat.yo2.cn/articles/qemu-compilation.html" \l "top3)

**建议在Ubuntu9.04 直接安装QEMU**

# and image development:

sudo apt-get install qemu

# Stuff needed to compile QEMU/TEMU:

sudo apt-get build-dep qemu

**Ubunt10，源码安装**

$ cd /tmp/qemu-0.9.1

$ ./configure --prefix=/usr/local/qemu

不加上prefix选项也是可以的。这个选项的作用是指定安装目录。如果不指定prefix，则可执行文件默认放在/usr/local/bin，库文件默认放在/usr/local/lib，配置文件默认放在/usr/local/etc。由于QEMU不能用make uninstall的方法卸载，指定安装目录可以便于卸载（其实就是删掉整个安装目录）。然后

$ make $ sudo make install

make比较费时，请耐心等待。

**不过在安装完所需的编译环境之后，我make的时候还是出现了错误，这里的解决办法是将/usr/include/目录下的dirent.h文件复制到/usr/include/linux目录下。**

至此，QEMU安装完成。

### [设置环境变量](http://butteredcat.yo2.cn/articles/qemu-compilation.html" \l "top4)

如果./configure时使用了prefix选项，还需要设置环境变量PATH，否则bash是找不到qemu的命令的（除非你输入该命令的具体路径，如/usr/local/qemu/bin/qemu）。编辑/etc/profile

$ sudo gedit /etc/profile

在文件中添加

PATH="$PATH:/usr/local/qemu/bin" export PATH

保存，退出，并用Ctrl+Alt+Backspace重新启动X Window。

### [编译安装KQEMU](http://butteredcat.yo2.cn/articles/qemu-compilation.html" \l "top5)

KQEMU是QEMU的加速器。在x86主机上虚拟PC时，使用KQEMU，可以大大提高运行速度。KQEMU支持QEMU-0.8.1及其以上的版本，并需要QEMU开启KQEMU支持(KQEMU support)(默认都是开启的)。

安装过程比较简单（假设源代码保存在~/Setup目录下）：

$ tar zxvf ~/Setup/kqemu-1.3.0pre11.tar.gz -C /tmp $ cd /tmp/kqemu-1.3.0pre11 $ ./configure $ make $ sudo make install

安装完后，需要载入KQEMU模块：

$ sudo modprobe kqemu

$ sudo mknod /dev/kqemu c 250 0 //此步骤不再需要

目前在9.04无法安装kqemu,貌似可以不需要

$ sudo chmod 666/dev/kqemu

  以上两步实现了将kqemu加速器模块加载到内核，这样qemu就可以使用了，但是这样每次机器重启都需要使用该方法将加速模块加载到内核，否则，qemu会提示找不到kqemu模块。如果不想每次重启机器都进行加载工作的话，可以在/etc/目录下的rc.local文件最后exit 0之前加入以下信息并保存：

 device="/dev/kqemu"

rm -f $device

mknod $device c 250 0

/sbin/modprobe kqemu

chmod 666 /dev/kqemu

  这样linux每次启动时都会自动将该模块信息加载到内核中，不用每次都手动加入了。

**运行表明这样还是不能加入，若提示错误，需要手动加载：**

至此，kqemu模块安装完成，如果使用的需要将其加载到内核，步骤为：

sudo modprobe kqemu

sudo chmod 666 /dev/kqemu

## qemu-0.9.1的安装与配置

同样，下载temu-1.0并将其解压缩放置到在home folder中建立的bitblaze目录下，temu的安装与配置过程如下：

(cd temu-1.0 && ./configure --target-list=i386-softmmu --proj-name=tracecap --cc=gcc-3.4 --prefix=$(pwd)/install)

**TEMU支持XP sp3**

Edit temu-1.0 the directory file **taintcheck.c**, to find taintcheck\_patch ()

1. **int** taintcheck\_patch()          //patch for keystroke propagation on Windows XP sp2
2. {
3. #ifndef NO\_PROPAGATE

**if** (cpu\_single\_env->eip != 0xbf8a4bde &&

    cpu\_single\_env->eip != 0xbf84a74f &&

**cpu\_single\_env->eip != 0xbf848d65 &&    //sp3eip**

**cpu\_single\_env->eip != 0xbf848d1c)      //sp3eip**

1. **return** 0;
3. **if**(!TEMU\_emulation\_started) **return** 0;
5. uint32\_t phys\_addr, addr, addr2, phys\_addr2;
6. addr = cpu\_single\_env->regs[R\_EBP] + 8;
7. phys\_addr = TEMU\_get\_phys\_addr(addr);
8. **if** (phys\_addr == -1)
9. **return** 0;
11. **if** (!taint\_mem\_check(phys\_addr, 1))
12. **return** 0;
14. addr2 = cpu\_single\_env->regs[R\_EBP] + 0x14;
15. **if** (TEMU\_read\_mem(addr2, 4,  &addr2) >= 0 &&
16. (phys\_addr2 = TEMU\_get\_phys\_addr(addr2)) != -1) {
17. taintcheck\_mem2reg\_nolookup(phys\_addr, addr, 1, R\_T0 \* 4);
18. taintcheck\_reg2mem(R\_T0 \* 4, 1, phys\_ram\_base + phys\_addr2);
19. }
20. #endif
21. **return** 0;
22. }

(cd temu-1.0 && make)

(cd temu-1.0 && make install)

上面的 过程在temu-1.0/docs/目录下的install-temu-release.sh文件中也可以找到，其中也会提示你需要使用3.4版本的gcc，而3.3和4.X版本的都不行。--prefix后面使用$(pwd)/install。

  到make install没有出现问题的话，temu就安装成功了，可以利用temu对系统及程序进行动态跟踪了。

使用temu运行的虚拟机的步骤如下：

cd temu-1.0 && ./tracecap/temu -snapshot -monitor stdio ../ubuntu.img

xp镜像不支持加速，显示错误：

Could not open '/dev/kqemu' - QEMU acceleration layer not activated: No such device or address

**解决办法：**

./tracecap/temu -no-kqemu -snapshot -monitor stdio ../windx.img

相对于使用qemu启动虚拟机，temu启动的要慢一些。

./tracecap/temu -no-kqemu -snapshot -monitor stdio ../windx.img

成功利用temu启动虚拟系统后就可以在上面显示的终端中使用temu提供的功能了，具体使用方法可以通过在终端中输入help命令进行查看。

如下所示：

