

## 第 21 章 构建检测分析环境

在开始介绍检测工程的概念之前，首先需要配置一个相关的分析环境，以便读者能够使用在接下来的章节中提到的各种分析工具与练习。本章中，将会配置 Linux 与 Windows 各一个虚拟机（一个新的 Linux 虚拟机），就足以完成本书中相关的所有练习。

### 虚拟机

由于需要使用 Linux 分析虚拟机来构建与使用两个分析工具：Suricata 与 APIMiner，首先需要 Linux 分析虚拟机。为了构建 Suricata，需要在一个 Linux 发行版上进行编译与运行。而对于 APIMiner，在 Windows 系统中可以使用编译后的二进制文件。也可以在 Linux 系统中使用 mingw64 软件包，自行交叉编译 APIMiner 的源代码以构建 Windows 系统可用的可执行文件。因此，这两个分析工具都可以在 Linux 上使用源码构建。

一般来说，读者可以自行选择任意 Linux 发行版，本书选择的发行版是 Ubuntu 16.04。读者也可以尝试使用 Ubuntu 18.04 或者更新版本的 Ubuntu，甚至是其他发行版。只要相关依赖包都能够正确安装，并且能够正常完成代码编译即可。想要安装 Ubuntu 16.04 可以仿照第 2 章中创建 Windows 分析虚拟机的步骤，创建一个 Linux 分析虚拟机。该虚拟机的硬件配置如图 21- 1 所示，读者可以参考该配置进行虚拟机的设置，也可以根据物理机的资源条件自行调整。

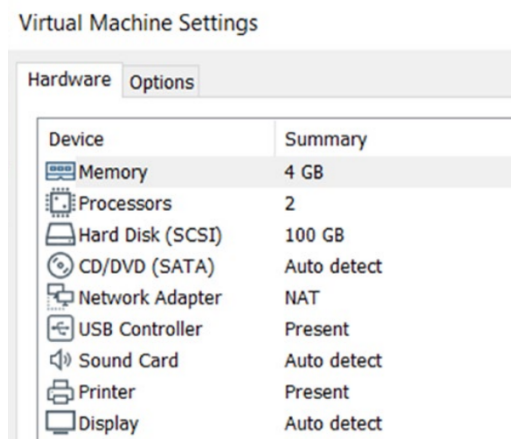


图 21- 1

请注意，与第 2 章中相同，此处仍使用 VMWare Workstation 作为虚拟机管理平台。读者可以自行选择要使用的虚拟机管理平台，例如 VirtualBox。必须强调的是，使用的虚拟机管理平台或者仿真程序必须具有创建快照与恢复快照的能力。但并非所有软件都提供类似的功能，在选择时需要加以甄别。快照不仅在分析恶意软件时有用，在部署新的分析工具或者进行新的环境配置时，一旦发现分析环境因此而受到影响，就可以恢复到之前保存的、可用的快照状态。本书不会具体介绍 Linux 虚拟机的安装过程，这并不是本书的重点，并且读者可以参考互联网上的各种资源来完成这一步。只要确保虚拟机硬件配置正确即可，如图 21- 1 所示。在进行后续步骤前，希望读者能够先正确安装好 Linux 分析虚拟机。在安装完成后，可能需要进行各种测试，例如是否可以正确连接到互联网等。确认安装完成后，读者可以创建一个快照，以备日后可以恢复使用。

## Suricata 配置

Suricata 是下一代入侵检测与防御系统的代表。本书将在第 23 章中详细介绍 IDS 与 IPS 以及其内部构成，包括典型代表 Suricata。现在需要配置使用 Suricata 所需要的各种环境，来编译、构建与安装 Suricata。

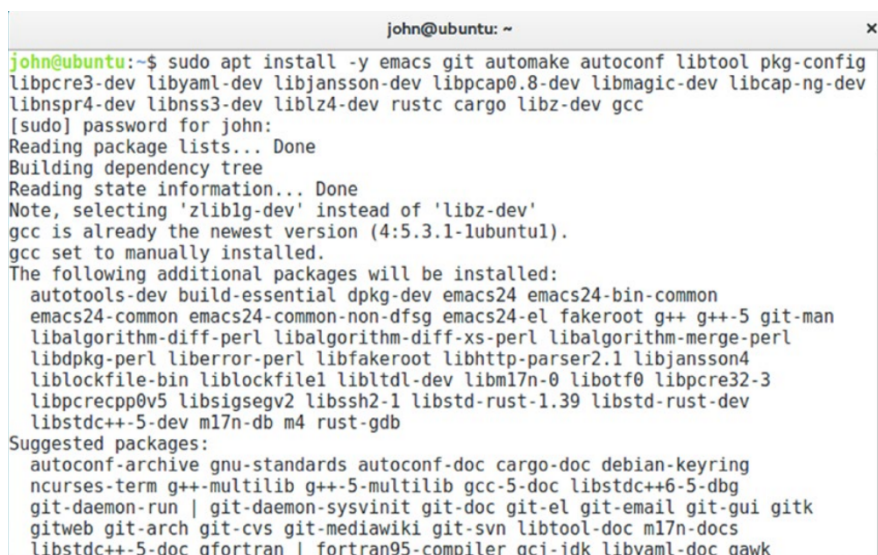
本书使用的 Suricata 的版本号为 5.0.2，这是撰写本书时最新发布的稳定版本。用户可以按

照官方使用手册中的要求安装对应的各种依赖，后续各种操作步骤差别不大，这些内容基本上可以扩展至后续更新版本的 Suricata。也可以使用 Suricata 的源码进行编译、安装，其源代码可以通过官方 GitHub 仓库 (<https://github.com/OISF/suricata>) 获取。在下载与编译 Suricata 之前，必须首先安装各种依赖。读者可以在 Ubuntu 虚拟机中打开终端并运行代码 21- 1 中所示的命令：

代码 21- 1

```
$ sudo apt install -y emacs git automake autoconf libtool pkg-config libpcap-dev libyaml-dev libjansson-dev libpcap0.8-dev libmagic-dev libcap-ng-dev libnspr4-dev libnss3-dev liblz4-dev rustc cargo libz-dev gcc
```

图 21- 2 显示了命令执行的结果。由于有相当多的依赖包需要安装，且受到互联网下载速度与服务器负载的影响，命令可能需要执行相当长的时间。有时由于系统上未能下载获取包信息，命令执行可能会失败。在这种情况下，可以先执行命令 `sudo apt update`，再重新执行代码 21- 1 中的命令。



```
john@ubuntu: ~  
john@ubuntu:~$ sudo apt install -y emacs git automake autoconf libtool pkg-config  
libpcap-dev libyaml-dev libjansson-dev libpcap0.8-dev libmagic-dev libcap-ng-dev  
libnspr4-dev libnss3-dev liblz4-dev rustc cargo libz-dev gcc  
[sudo] password for john:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Note, selecting 'zlib1g-dev' instead of 'libz-dev'  
gcc is already the newest version (4:5.3.1-ubuntu1).  
gcc set to manually installed.  
The following additional packages will be installed:  
  autotools-dev build-essential dpkg-dev emacs24 emacs24-bin-common  
  emacs24-common emacs24-common-non-dfsg emacs24-el fakeroot g++ g++-5 git-man  
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl  
  libdpkg-perl liberror-perl libfakeroot libhttp-parser2.1 libjansson4  
  liblockfile-bin liblockfile1 libltdl-dev libm17n-0 libotf0 libpcap32-3  
  libpcrcpp0v5 libsigsegv2 libssh2-1 libstd-rust-1.39 libstd-rust-dev  
  libstdc++-5-dev m17n-db m4 rust-gdb  
Suggested packages:  
  autoconf-archive gnu-stardards autoconf-doc cargo-doc debian-keyring  
  ncurses-term g++-multilib g++-5-multilib gcc-5-doc libstdc++6-5-dbg  
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk  
  gitweb git-arch git-cvs git-mediawiki git-svn libtool-doc m17n-docs  
  libstdc++-5-doc gfortran | fortran95-compiler gcj-jdk libyaml-doc gawk
```

图 21- 2

在成功安装了依赖后，就可以开始下载、构建与安装 Suricata。首先通过 [www.openinfosecfoundation.org/download/suricata-5.0.2.tar.gz](http://www.openinfosecfoundation.org/download/suricata-5.0.2.tar.gz) 下载 5.0.2 版本的 Suricata，再利用代码 21- 2 中的两个命令将其解压缩。请注意，该链接地址在撰写本书

时有效。但如果该软件包移动到其他位置，可以利用搜索引擎进行检索，通过官方网站进行下载。

代码 21- 2

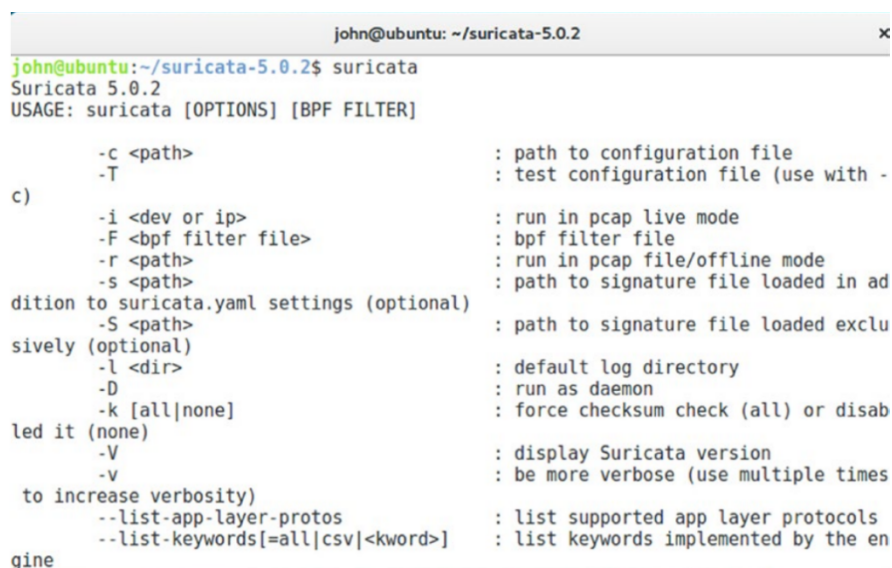
```
$ wget https://www.openinfosecfoundation.org/download/suricata-5.0.2.tar.gz
$ tar -xvzf suricata-5.0.2.tar.gz
```

后续可以使用 cd 命令进入解压后的 Suricata-5.0.2 文件夹，并且使用代码 21- 3 中的命令来构建、安装 Suricata。

代码 21- 3

```
$ cd suricata-5.0.2
$ ./configure
$ make -j
$ sudo make install
$ sudo ldconfig
```

执行后就可以尝试通过终端执行 suricata 命令来验证 Suricata 是否已经被正确安装。正常情况下应该输出 Suricata 的帮助信息，如图 21- 3 所示。



```
john@ubuntu: ~/suricata-5.0.2
john@ubuntu:~/suricata-5.0.2$ suricata
Suricata 5.0.2
USAGE: suricata [OPTIONS] [BPF FILTER]

    -c <path>                : path to configuration file
    -T                        : test configuration file (use with -c)
    -i <dev or ip>            : run in pcap live mode
    -F <bpf filter file>      : bpf filter file
    -r <path>                 : run in pcap file/offline mode
    -s <path>                 : path to signature file loaded in addition to suricata.yaml settings (optional)
    -S <path>                 : path to signature file loaded exclusively (optional)
    -l <dir>                  : default log directory
    -D                        : run as daemon
    -k [all|none]             : force checksum check (all) or disabled it (none)
    -V                        : display Suricata version
    -v                        : be more verbose (use multiple times to increase verbosity)
    --list-app-layer-protos   : list supported app layer protocols
    --list-keywords[=all|csv|<keyword>] : list keywords implemented by the engine
```

图 21- 3

这样就基本完成了，后续在第 23 章时还需要对配置文件 suricata.yaml 进行一些细微的调整。而下一节中，将会介绍构建 APIMiner 与 Cuckoo Monitor 所需的依赖。

## APIMiner 与 Cuckoo Monitor

构建 APIMiner 与 Cuckoo Monitor 所需要的依赖会简单很多，在安装了 Ubuntu 系统的 Linux 开发虚拟机中执行代码 21- 4 中的命令即可。

代码 21- 4

```
sudo apt-get install -y mingw-w64 python-pip nasm  
sudo pip install --upgrade pip  
sudo pip install sphinx docutils pyyaml
```

读者可以使用浏览器，通过 <https://github.com/poona/APIMiner> 从 GitHub 下载 APIMiner 的源代码。或者，也可以使用 git clone 命令来获取源代码。随后，读者可以通过执行 cd 命令进入 APIMiner 源代码的根目录并执行代码 21- 5 中的命令来进行构建。

代码 21- 5

```
$ make
```

执行此命令后，APIMiner 的二进制文件会被构建并输出至名为 bin 的文件夹中，该文件夹位于执行 make 命令的 APIMiner 根文件夹中。

构建 Cuckoo Monitor 所需的依赖与 APIMiner 所需的依赖项相同，可以通过类似的方式利用 git clone <https://github.com/cuckoosandbox/monitor.git> 命令获取 Cuckoo Monitor 的源代码。随后，同样可以使用代码 21- 5 中的命令来进行构建，在 bin 文件夹下会输出二进制文件。

## Windows 开发虚拟机

本节将会介绍如何配置 Windows 开发虚拟机，并且使用其来构建本书要使用到的各种分析工具，例如二进制插桩的工具（详见第 25 章）。

首先需要创建一个新的 Windows 虚拟机，该虚拟机使用 32 位 Windows 7 操作系统，并且被称为 Windows 开发虚拟机。创建虚拟机时，可以使用与图 21- 1 相同的硬件设置。

在安装虚拟机的过程中不需要任何特殊操作，与第 2 章中创建分析虚拟机时相同即可。安

装完成后，需要将操作系统更新到微软提供的最新版本。

在 Windows 开发虚拟机准备就绪时，就可以配置 Visual Studio 编译器及其 SDK。请注意，这里使用的是 Visual Studio 编译器而不是 Visual Studio 这个 IDE 本身。很多时候，初学者最好不要使用 IDE，因为 IDE 抽象并且隐藏了许多有关构建与链接源代码的细节。一旦使用 IDE，就简化为点击按钮即可！多么神奇！这对于初学入门可能并不是有利的。相反，使用简单的文本编辑器编写源代码，然后使用编译器 `cl.exe` 在命令行中进行编译，则是了解各种外部依赖与链接过程等内部细节的更好方式。

不仅如此，在 Windows 的命令行环境（如 Cygwin）中也提供了其他实用程序，例如 `make` 命令可以帮助用户使用 `makefile` 自动执行源代码构建的过程。

## Visual Studio

Visual Studio 有付费版与社区版之分。读者可以通过 <https://visualstudio.microsoft.com> 下载 VS Studio 社区版的安装程序。撰写本书时，作者使用的是 Visual Studio 社区版 2019 (Visual Studio Community 2019)。但读者可以自行选择其他版本，不论是付费版还是社区版都可以。只需要执行 Visual Studio 社区版 2019 的安装程序就可以在引导下完成安装，在安装过程中可以选择想要安装的各种组件。安装时必须选择的组件为 Desktop development with C++，如图 21- 4 所示。

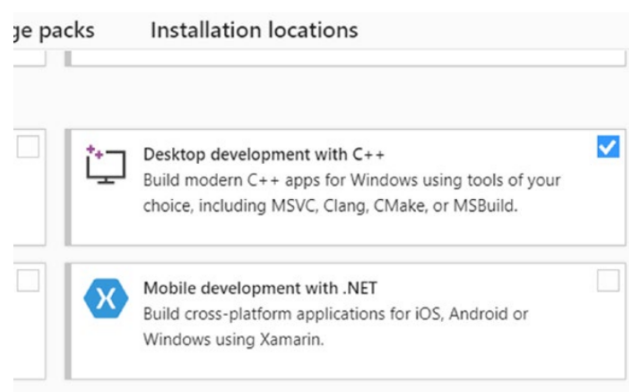


图 21- 4

安装完成后，可能需要重新启动计算机。用户可以通过打开 Visual Studio 2019 的开发人员命令行工具来测试该软件已经完成安装，如图 21- 5 所示。

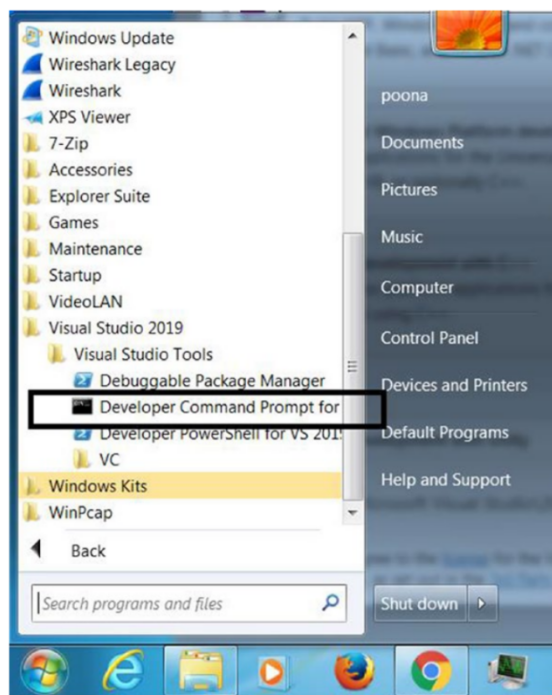


图 21- 5

如图 21- 5 所示, 打开开发人员命令行工具可以打开终端, 读者可以在其中执行命令 `cl.exe` 来测试 Visual Studio 编译器是否已经安装成功并且能够正常使用, 如图 21- 6 所示。

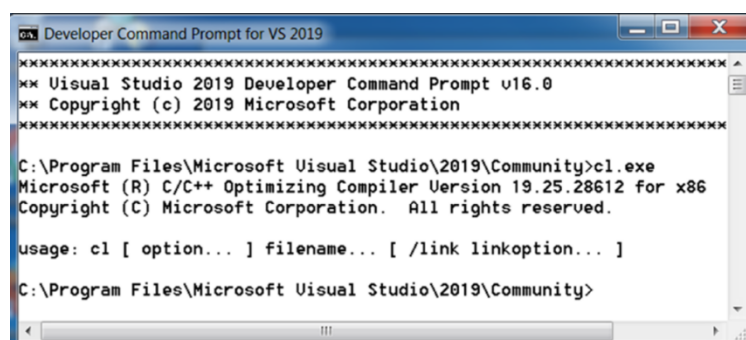


图 21- 6

## Cygwin

Cygwin 是一个 POSIX 兼容工具，通过控制台为 Windows 提供各种类 Unix 应用程序。

Cygwin 一直广受业界好评，利用它可以支持在 Windows 上使用 Makefile 等工具来自动

构建大型项目的源代码。现在重要的是将 Cygwin 与上一节中的 Visual Studio 相结合，以便可以在 Cygwin 的控制台中使用 Visual Studio 环境(包括 Visual Studio 的编译器 cl.exe)。在这之前，需要安装 Cygwin。通过 <https://cygwin.com> 即可下载并运行 Cygwin 的安装程序，请确保下载 32 位的 Windows 安装程序。安装过程中会提供要安装的列表，用户可以自行选择要安装的部分。在本书中，只需要安装表 21- 1 中的包。

表 21- 1

| 上层包名    | 要安装的包 |
|---------|-------|
| Shells  | 全部安装  |
| Base    | 全部安装  |
| Devel   | make  |
| Archive | unzip |
| 其他      | 全部不安装 |

用户需要有选择地安装表格中列出的软件包。首先，通过选择 “Uninstall for All” 取消选择所有软件包。随后只选择表格中列出的软件包进行安装，如图 21- 7 所示。对于 make、unzip 这类特定名称的软件包，可以在顶部的搜索框中按名称进行搜索，再选择这些软件包对应要安装的版本。选择完成后就可以开始安装,但等待下载和安装全部完成需要一些时间。

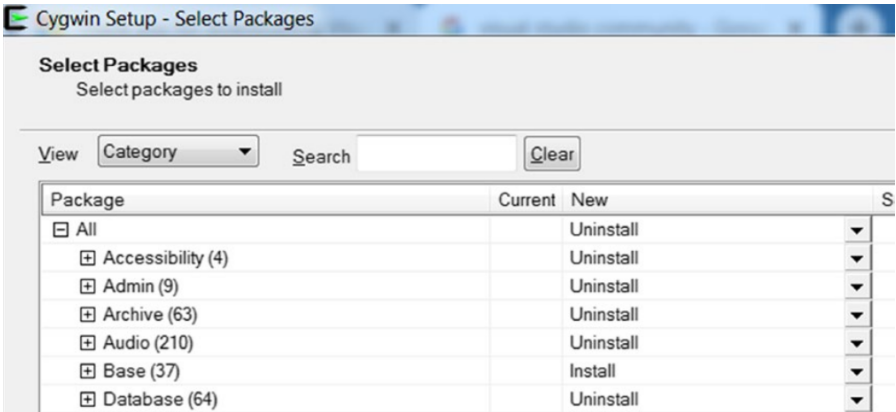


图 21- 7



## Cygwin+Visual Studio

现在已经完成了 Cygwin 的安装，接下来要配置在 Cygwin 控制台中使用 Visual Studio 工具链。为此，必须在 Cygwin 中对 Visual Studio 的环境进行配置。为了帮助用户解决这个问题，Visual Studio 通过一个批处理文件提供了工具链的使用环境。在 Visual Studio 2015 及以前的版本中，该批处理文件被命名为 vsvars32.bat。而在更新的版本中，该批处理脚本叫做 VsDevCmd.bat。对于本书使用的 Visual Studio 2019 社区版，批处理文件位于 C:\Program Files\Microsoft Visual Studio\2019\Community\Common7\Tools\文件夹中。读者可以在 Cygwin 安装目录（默认为 C:\cygwin\bin\）下的 bin 文件夹中，创建一个名为 cygwin.bat 的文件，具体如代码 21- 6 所示。

代码 21- 6

```
@echo off
@REM Select the latest VS Tools
# Below is one full long line. Might look folded here due to
# length. Unfold when you type it into your cygwin.bat
CALL "C:\Program Files\Microsoft Visual Studio\2019\Community\Co
mmon7\Tools\VsDevCmd.bat"
C:
chdir C:\cygwin\bin
START mintty.exe -i /Cygwin-Terminal.ico -
```

如果希望将 cygwin.bat 文件添加为桌面快捷方式方便使用，就可以使用此处的这段代码。

可以通过双击该文件运行来进行测试，尝试使用 Visual Studio 的编译器 cl.exe，正常情况下如图 21- 8 所示。

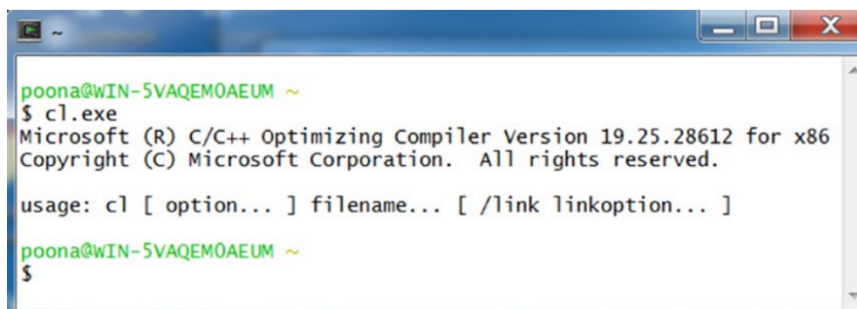


图 21- 8

## 其他工具

在 Windows 开发虚拟机中还需要使用另外几个分析工具：Yara、BinText、Wireshark 与 IDA Pro。在第 2 章中曾经介绍过这些分析工具的安装步骤，这次如果要重新装开发虚拟机的话，也要遵循相同的步骤再安装一次。又或者，直接使用在第 2 章中安装好的分析虚拟机也可以。与分析虚拟机一样，开发虚拟机中也会安装各种工具，在开发时为分析人员提供帮助。永远不要在开发虚拟机中运行任何恶意软件，请使用分析虚拟机。当分析人员要尝试使用新的分析工具或者开发新的分析工具时，可能必须要在这些虚拟机中安装更多依赖、框架与软件。保留这两个开发虚拟机（Linux 与 Windows）是非常有用且便利的，分析人员也可以在需要时增加新的分析工具。另外，一定要确保为这两个开发虚拟机都创建原始状态的快照。这样每当虚拟机的环境或者设置出现问题且无法恢复时，都可以通过快照恢复到原始状态。

## 总结

学习检测工程的第一步是确保拥有正确的开发环境与配置，可以帮助读者修改与构建检测工具。本章通过配置两个分析虚拟机（一个 Linux 分析虚拟机、一个 Windows 分析虚拟机），来实现这一点。本章介绍了如何在两个分析虚拟机上安装与配置各种分析工具，这都有助于构建后续章节中的各种检测工具。