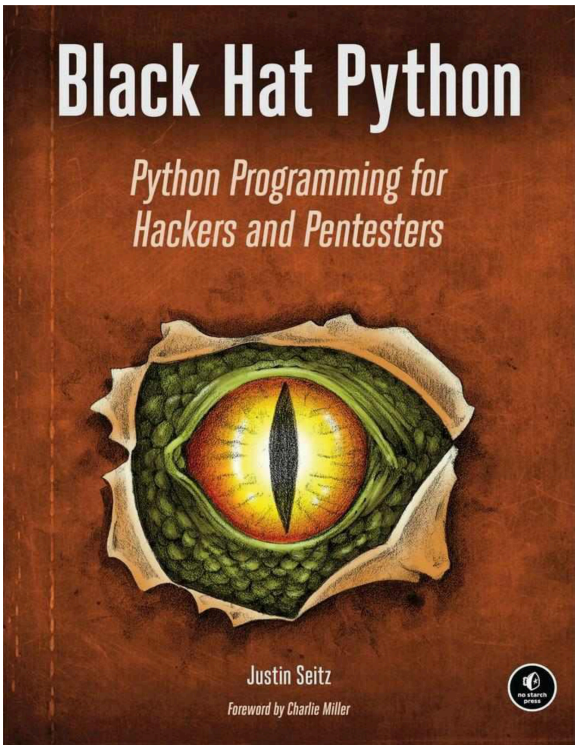


ebookbig.com

or scan the QR code to access the digital study portal.

n is still the dominant language in the world of information security, even if the conversatio
your language of choice sometimes looks more like a religious war. Python-based tools in
anner of fuzzers, proxies, and even the occasional exploit. Exploit frameworks like CANVA
ritten in Python as are more obscure tools like PyEmu or Sulley.

bout every fuzzer or exploit I have written has been in Python. In fact, the automotive hacki
rch that Chris Valasek and I recently performed contained a library to inject CAN messages
automotive network using Python!

are interested in tinkering with information security tasks, Python is a great language to lea
se of the large number of reverse engineering and exploitation libraries available for your
if only the Metasploit developers would come to their senses and switch from Ruby to Pyth
ommunity would be united.

s new book, Justin covers a large range of topics that an enterprising young hacker would n
off the ground. He includes walkthroughs of how to read and write network packets, how t
the network, as well as anything you might need for web application auditing and attacking.
pends significant time diving into how to write code to address specifics with attacking
ows systems. In general, *Black Hat Python* is a fun read, and while it might not turn you int
stunt hacker like myself, it can certainly get you started down the path. Remember, the
ence between script kiddies and professionals is the difference between merely using other
e's tools and writing your own.

ie Miller

uis, Missouri

mber 2014

n hacker. Those are two words you really could use to describe me. At Immunity, I am luck h to work with people who actually, really, know how to code Python. I am not one of thos e. I spend a great deal of my time penetration testing, and that requires rapid Python tool opment, with a focus on execution and delivering results (not necessarily on prettiness, ization, or even stability). Throughout this book you will learn that this is how I code, but I s though it is part of what makes me a strong pentester. I hope that this philosophy and style you as well.

u progress through the book, you will also realize that I don't take deep dives on any single . This is by design. I want to give you the bare minimum, with a little flavor, so that you hav foundational knowledge. With that in mind, I've sprinkled ideas and homework assignment ghout the book to kickstart you in your own direction. I encourage you to explore these idea ld love to hear back any of your own implementations, tooling, or homework assignments t ave done.

ith any technical book, readers at different skill levels with Python (or information security al) will experience this book differently. Some of you may simply grab it and nab chapters ertinent to a consulting gig you are on, while others may read it cover to cover. I would mend that if you are a novice to intermediate Python programmer that you start at the begin book and read it straight through in order. You will pick up some good building blocks alo ay.

rt, I lay down some networking fundamentals in Chapter 2 and slowly work our way throu ockets in Chapter 3 and using Scapy in Chapter 4 for some more interesting network toolin ext section of the book deals with hacking web applications, starting with your own custom g in Chapter 5 and then extending the popular Burp Suite in Chapter 6. From there we will a great deal of time talking about trojans, starting with GitHub command and control in ter 7, all the way through Chapter 10 where we will cover some Windows privilege escala . The final chapter is about using Volatility for automating some offensive memory forensic iques.

o keep the code samples short and to the point, and the same goes for the explanations. If yo vely new to Python I encourage you to punch out every line to get that coding muscle memor . All of the source code examples from this book are available at */nostarch.com/blackhatpython/*.

we go!

ld like to thank my family — my beautiful wife, Clare, and my five children, Emily, Carter, n, Brady, and Mason — for all of the encouragement and tolerance while I spent a year and f my life writing this book. My brothers, sister, Mom, Dad, and Paulette have also given m motivation to keep pushing through no matter what. I love you all.

l my folks at Immunity (I would list each of you here if I had the room): thanks for tolerating ay-to-day basis. You are truly an amazing crew to work with. To the team at No Starch — Serena, and Leigh — thanks so much for all of the hard work you put into this book and the r collection. We all appreciate it.

ld also like to thank my technical reviewers, Dan Frisch and Cliff Janzen. These guys typed ritiqued every single line of code, wrote supporting code, made edits, and provided absolu ing support throughout the whole process. Anyone who is writing an infosec book should re ese guys on board; they were amazing and then some.

e rest of you ruffians that share drinks, laughs and GChats: thanks for letting me piss and m about writing this book.

vironment

is the least fun — but nevertheless critical — part of the book, where we walk through setti vironment in which to write and test Python. We are going to do a crash course in setting up Linux virtual machine (VM) and installing a nice IDE so that you have everything you need op code. By the end of this chapter, you should be ready to tackle the exercises and code ples in the remainder of the book.

e you get started, go ahead and download and install VMWare Player.[1] I also recommend ave some Windows VMs at the ready as well, including Windows XP and Windows 7, rably 32-bit in both cases.

is the successor to the BackTrack Linux distribution, designed by Offensive Security from t d up as a penetration testing operating system. It comes with a number of tools preinstalled ed on Debian Linux, so you'll also be able to install a wide variety of additional tools and ies beyond what's on the OS to start.

grab a Kali VM image from the following URL: *http://images.offensive-security.com/kali -1.0.9-vm-i486.7z*.[2] Download and decompress the image, and then double-click it to mak are Player fire it up. The default username is *root* and the password is *toor*. This should ge he full Kali desktop environment as shown in Figure 1-1.



Figure 1-1. The Kali Linux desktop

irst thing we are going to do is ensure that the correct version of Python is installed. This b se Python 2.7 throughout. In the shell (**Applications▸Accessories▸Terminal**), execute the ing:

```
t@kali:~# python --version
hon 2.7.3
t@kali:~#
```

downloaded the exact image that I recommended above, Python 2.7 will be automatically

ies, without having to manually download, unpack, and install them. Let's install both of th
ge managers by issuing the following commands:

```
t@kali:~#: apt-get install python-setuptools python-pip
```

the packages are installed, we can do a quick test and install the module that we'll use in
ter 7 to build a GitHub-based trojan. Enter the following into your terminal:

```
t@kali:~#: pip install github3.py
```

hould see output in your terminal indicating that the library is being downloaded and install

drop into a Python shell and validate that it was installed correctly:

```
t@kali:~#: python
hon 2.7.3 (default, Mar 14 2014, 11:57:14)
C 4.7.2] on linux2
e "help", "copyright", "credits" or "license" for more information.
 import github3
 exit()
```

r results are not identical to these, then there is a "misconfiguration" in your Python enviro
ou have brought great shame to our Python dojo! In this case, make sure that you followed a
above and that you have the correct version of Kali.

in mind that for most examples throughout this book, you can develop your code in a variet
onments, including Mac, Linux, and Windows. There are some chapters that are Windows-
fic, and I'll make sure to let you know at the beginning of the chapter.

that we have our hacking virtual machine set up, let's install a Python IDE for development.

I typically don’t advocate commercial software products, WingIDE is the best IDE that I’ in the past seven years at Immunity. WingIDE provides all the basic IDE functionality like a letion and explanation of function parameters, but its debugging capabilities are what set it other IDEs. I will give you a quick rundown of the commercial version of WingIDE, but of e you should choose whichever version is best for you.[3]

an grab WingIDE from *http://www.wingware.com/*, and I recommend that you install the tri ou can experience firsthand some of the features available in the commercial version.

an do your development on any platform you wish, but it might be best to install WingIDE Kali VM at least to get started. If you’ve followed along with my instructions so far, make s ou download the 32-bit `.deb` package for WingIDE, and save it to your user directory. The into a terminal and run the following:

```
t@kali:~# dpkg -i wingide5_5.0.9-1_i386.deb
```

should install WingIDE as planned. If you get any installation errors, there might be unmet dencies. In this case, simply run:

```
t@kali:~# apt-get -f install
```

should fix any missing dependencies and install WingIDE. To verify that you’ve installed it rly, make sure you can access it as shown in Figure 1-2.

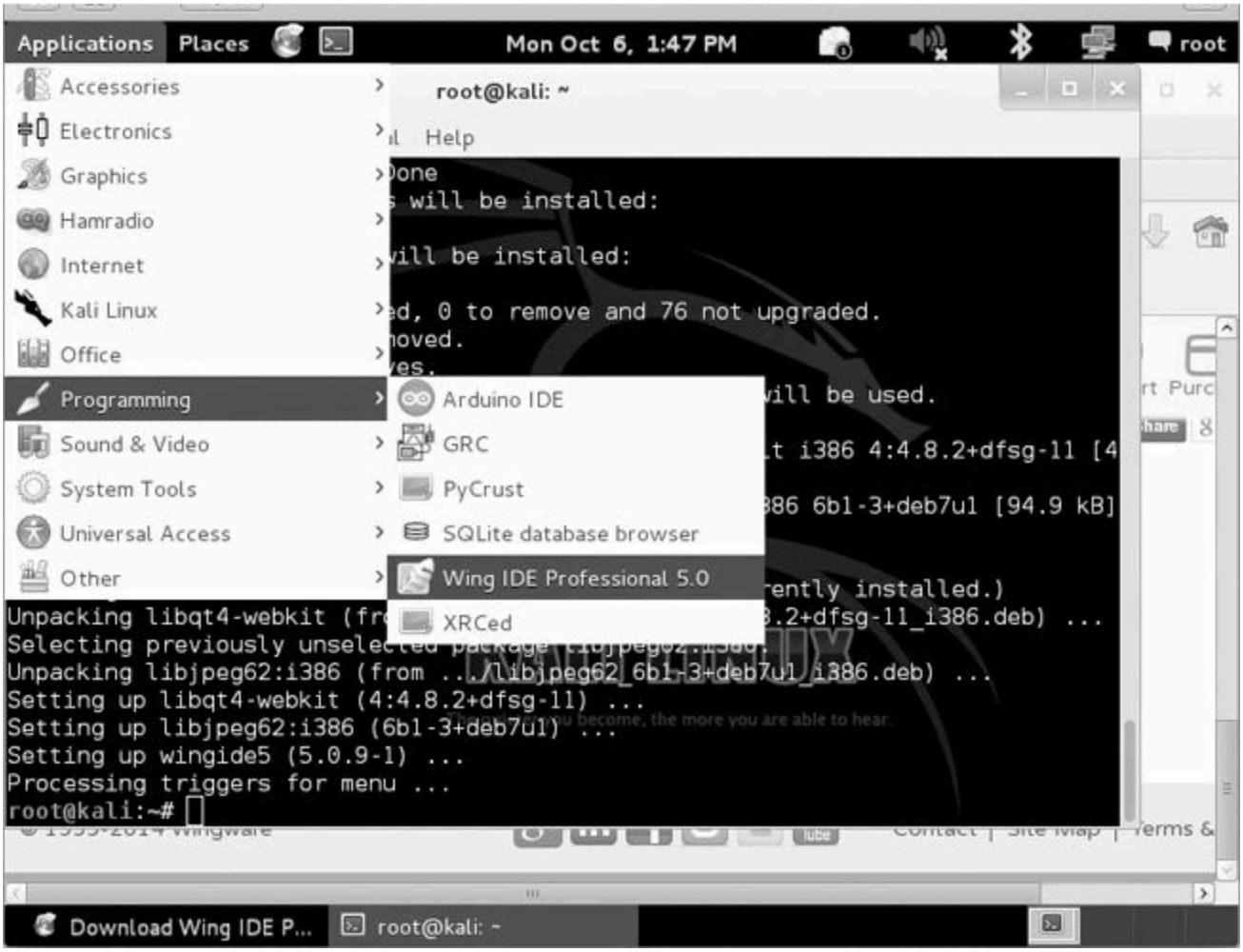


Figure 1-2. Accessing WingIDE from the Kali desktop

p WingIDE and open a new, blank Python file. Then follow along as I give you a quick run me useful features. For starters, your screen should look like Figure 1-3, with your main co g area in the top left and a set of tabs on the bottom.

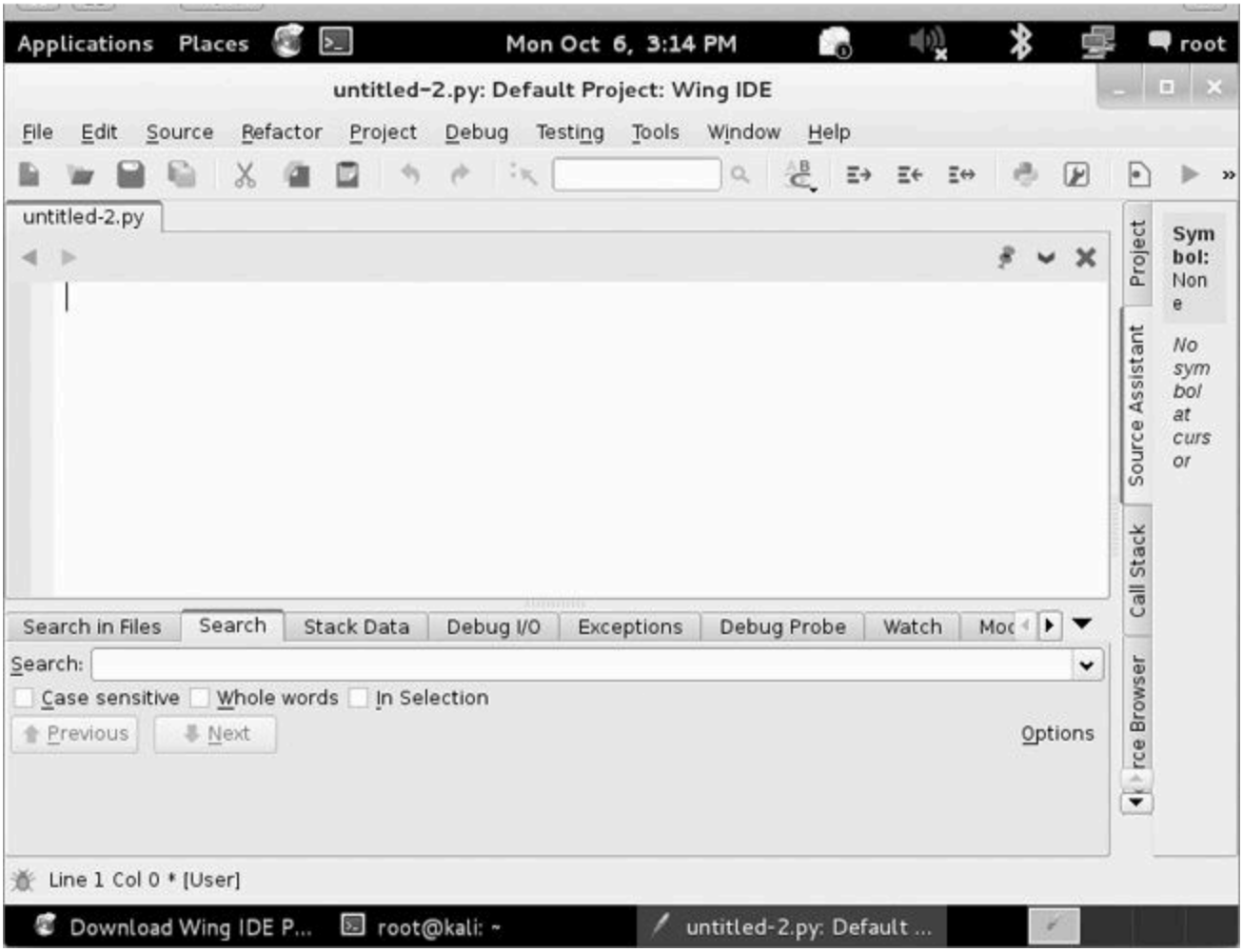


Figure 1-3. Main WingIDE window layout

write some simple code to illustrate some of the useful functions of WingIDE, including the g Probe and Stack Data tabs. Punch the following code into the editor:

```
 sum(number_one,number_two):
 number_one_int = convert_integer(number_one)
 number_two_int = convert_integer(number_two)

 result = number_one_int + number_two_int

 return result

 convert_integer(number_string):

 converted_integer = int(number_string)
 return converted_integer

wer = sum("1","2")
```

is a very contrived example, but it is an excellent demonstration of how to make your life e WingIDE. Save it with any filename you want, click the **Debug** menu item, and select the **Se ent as Main Debug File** option, as shown in Figure 1-4.