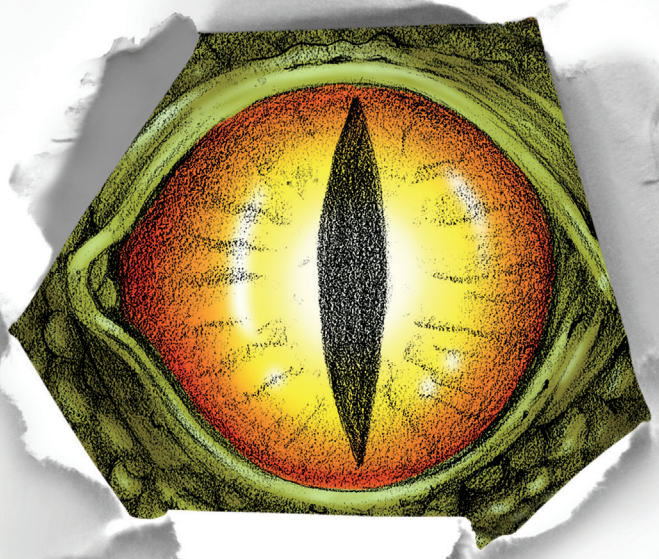
- Justin Seitz -

PYTHON

per hacker

Tecniche offensive black hat



Scalare i privilegi di Windows >>

Aggirare i meccanismi difensivi di sandboxing >>

Creare trojan command-and-control con GitHub >>

Iniettare codice di shell nelle macchine virtuali >>

*pro DigitalLifeStyle

*pro
Digital*LifeStyle

PYTHON

per hacker

Tecniche offensive black hat

Justin Seitz

EDIZIONI
LSWR

Titolo originale: Black Hat Python
ISBN: 978-1-59327-590-7
Published by No Starch Press, Inc.
245 8th Street, San Francisco, CA 94103
phone: 415.863.9900; info@nostarch.com
www.nostarch.com
Cover Illustration: Garry Booth

Edizione italiana:
Python per hacker | Tecniche offensive black hat

Traduzione di: Marco Buttu

Collana: *pro DigitalLifeStyle

Editor in Chief: Marco Aleotti
Progetto grafico: Roberta Venturieri

ISBN: 978-88-6895-247-1

EDIZIONI LSWR

Via G. Spadolini, 7
20141 Milano (MI)
Tel. 02 881841
www.edizionilswr.it

Printed in Italy

Finito di stampare nel mese di ottobre 2015 presso "LegoDigit" Srl, Lavis (TN)

(*) Edizioni Lswr è un marchio di La Tribuna Srl. La Tribuna Srl fa parte di LSWR GROUP.

Sommario

Prefazione

Python continua a essere il linguaggio dominante nel mondo dell'information security, anche se in realtà le conversazioni in merito al linguaggio preferito a volte somigliano a delle guerre di religione. I tool basati su Python includono ogni tipo di fuzzer, proxy e talvolta anche degli exploit. I framework di exploit come CANVAS sono scritti in Python e così pure alcuni tool più oscuri come PyEmu o Sulley.
Praticamente ogni fuzzer o exploit che ho scritto, l'ho implementato in Python. La ricerca sull'hacking automobilistico che ho fatto di recente con Chris Valasek contiene una libreria che serve per iniettare messaggi CAN sulla rete della vostra automobile, usando Python!
Se siete interessati al campo dell'information security, allora spendere del tempo per imparare Python è veramente un ottimo investimento, visto che sono disponibili numerose librerie per il reverse engineering e l'exploitation. Se gli sviluppatori di Metasploit passassero da Ruby a Python, la nostra community sarebbe unita.
In questo libro Justin copre un ampio spettro di argomenti che un giovane hacker intraprendente dovrebbe affrontare. Questi argomenti includono il saper leggere e scrivere pacchetti di rete, sapere sniffare la rete e ogni altra cosa che vi potrebbe servire per analizzare applicazioni web e realizzare attacchi. Justin spende parecchio tempo per mostrare come scrivere codice per indirizzare specifici attacchi ai sistemi Windows. In generale, *Python per hacker | Tecniche offensive black hat* è una lettura divertente, e anche se non riuscisse a farvi diventare un super hacker come me, certamente vi farà iniziare nel modo giusto. Ricordate che la differenza tra scrivere script per gioco e script professionali è pari alla differenza tra usare solamente tool scritti da altri e scrivere i tool di mano vostra.

Charlie Miller
St. Louis, Missouri

Introduzione

Python e *hacker*. Queste sono le due parole che dovreste veramente usare per descrivermi. A Immunity ho la fortuna di lavorare con persone che sanno realmente come scrivere codice Python. Non sono una di queste persone. Spendo la maggior parte del mio tempo facendo penetration testing, e questo necessita di dover scrivere velocemente dei tool in Python, focalizzandomi sul raggiungimento del risultato (non necessariamente bellezza del codice, ottimizzazione o stabilità). Attraverso questo libro imparerete che questo è il modo in cui scrivo il codice, e credo che sia anche uno dei motivi che fa di me un valido pentetration tester. Spero che questa filosofia e stile di lavoro aiutino anche voi a diventarlo.

Man mano che progredirete avanzando nella lettura del libro, vi renderete anche conto che non scendo troppo nei dettagli di ogni singolo argomento. Questo è voluto, perché intendo fornirvi le informazioni basilari, con qualche piccolo condimento, in modo che abbiate le conoscenze fondamentali. Avendo questo in mente, nel libro ho distribuito le idee e i compiti da realizzare a casa in modo che possiate iniziare autonomamente a fare le vostre prove e documentarvi in modo più approfondito. Vi incoraggio a esplorare queste idee, e mi piacerebbe sentire da voi come avete realizzato le vostre implementazioni personali, i vostri tool e come avete risolto i compiti che vi ho assegnato.

Come per ogni libro tecnico, i lettori con differenti livelli di conoscenza di Python (o più in generale, di information security) affronteranno e apprezzeranno il libro in maniera differente. Alcuni di voi semplicemente prenderanno il libro e andranno a leggere i capitoli pertinenti con il loro lavoro o con ciò che maggiormente gli interessa, mentre altri lo leggeranno da cima a fondo. Vorrei raccomandare a chi di voi ha un livello di Python che si pone tra il principiante e l'intermedio, di cominciare dall'inizio del libro e leggerlo da lì in avanti seguendo l'ordine prestabilito degli argomenti. Strada facendo acquisirete le competenze che vi occorrono.

Per iniziare, nel Capitolo 2 introduciamo alcune fondamentali conoscenze di networking, quindi lentamente progrediamo studiando i socket raw nel Capitolo 3 e usando

Scapy nel Capitolo 4, in modo da poter implementare interessanti tool di rete. Le successive sezioni del libro si occupano di hacking con applicazioni web, iniziando nel Capitolo 5 con la realizzazione di un tool custom, per poi estendere la popolare Burp Suite nel Capitolo 6. Da qui spenderemo una grande parte del tempo a parlare di trojan iniziando dal Capitolo 7, dove parleremo di command e control con GitHub, sino al Capitolo 10, dove vedremo alcuni trucchi per scalare i privilegi su Windows. Il capitolo finale si occupa dell'utilizzo di Volatility per automatizzare alcune tecniche forensi per scopi offensivi.

Ho provato a mantenere il codice semplice, breve e mirato, e lo stesso vale per le spiegazioni. Se siete alle prime armi con Python, vi incoraggio a soffermarvi su ogni linea di codice, così sarà più facile tenere a mente le cose e capire le successive parti del libro. Il codice di tutti gli esempi del libro è disponibile su http://nostarch.com/blackhatpython/. Siamo pronti per iniziare!

Ringraziamenti

Vorrei ringraziare la mia famiglia (la mia magnifica moglie Clare e i miei cinque figli, Emily, Carter, Cohen, Brady e Mason) per tutti gli incoraggiamenti e le pazienza durante l'anno e mezzo della mia vita dedicato alla scrittura di questo libro. I miei fratelli, mia sorella, mia madre, mio padre e Paulette, per avermi motivato ad andare avanti a prescindere da tutto. Vi adoro tutti quanti.

A tutti i miei colleghi di Immunity (mi sarebbe piaciuto elencare qui ciascuno di voi, se ne avessi avuto lo spazio): grazie per sopportarmi quotidianamente. Siete veramente un gruppo incredibile con il quale lavorare. Al team di No Starch (Tyler, Bill, Serena e Leigh): grazie tantissimo per il duro lavoro che avete investito in questo libro e nel resto della vostra collana. Lo apprezziamo tutti.

Mi piacerebbe ringraziare anche i miei revisori tecnici, Dan Frisch e Cliff Janzen. Questi ragazzi hanno digitato e criticato ogni singola linea di codice, scritto codice di supporto, fatto editing e fornito un aiuto incredibile attraverso l'intero processo. Chiunque stia scrivendo un libro sulla sicurezza informatica dovrebbe veramente prendere questi ragazzi in squadra; sono stati meravigliosi, e anche di più.

Per il resto di voi furfanti con cui ho condiviso bevute, risate e GChats: grazie per avermi consentito di lamentarmi con voi in merito alla scrittura di questo libro.

L'autore

Justin Seitz lavora per Immunity, Inc. come ricercatore senior della sicurezza, e passa il suo tempo ricercando bug, facendo reverse engineering, scrivendo exploit e codice Python. È l'autore di *Gray Hat Python*, il primo libro su Python rivolto ad analisti della sicurezza.

I revisori tecnici

Dan Frisch ha più di dieci anni di esperienza nel campo dell'information security. Attualmente lavora come analista senior della sicurezza per un'agenzia canadese che si occupa di aspetti legislativi. Precedentemente, ha lavorato come consulente per la sicurezza per società finanziarie e dell'ICT del Nord America. Poiché è ossessionato dalle tecnologie e ha un terzo dan di cintura nera, potete presumere (correttamente) che la sua intera vita sia basata attorno a "The Matrix".
Sin dalle origini del Commodore PET e VIC-20, le tecnologie sono state una costante compagna (e a volte un'ossessione!) di *Cliff Janzen*. Cliff ha scoperto la passione per questo ambito lavorativo quando cambiò attività nel 2008, occupandosi di information security dopo più di una decade spesa nel settore dell'IT. Negli ultimi anni Cliff ha felicemente svolto le mansioni di consulente per la sicurezza, facendo ogni cosa, dalle policy review ai penetration test, e si sente fortunato di lavorare su tematiche che costituiscono anche il suo hobby preferito.

1

Configurare l'ambiente Python

Questa è la parte meno divertente (ma comunque molto importante) del libro, nella quale ci occuperemo di configurare l'ambiente dove scrivere e testare il codice Python. Vedremo rapidamente come ottenere una macchina virtuale che ospita Kali Linux e come installare un IDE, in modo da avere tutto ciò che ci occorre per lo sviluppo. Alla fine del capitolo dovreste essere pronti per affrontare gli esercizi e il codice di esempio della restante parte del libro.

Prima di iniziare, scarichiamo e installiamo VMWare Player dal sito web http://www.vmware.com/. Inoltre vi raccomando di avere a disposizione anche delle macchine virtuali con installati Windows XP e Windows 7, entrambe preferibilmente a 32-bit.

Installare Kali Linux

Kali è il successore della distribuzione Linux BackTrack ed è stato progettato da Offensive Security mirando sin da subito a ottenere un sistema operativo finalizzato al penetration testing. Ha già diversi tool preinstallati ed è basato su Linux Debian, per cui sarete in grado di installare un'ampia varietà di tool e librerie addizionali, oltre a quelli presenti di default. Prima di tutto, recuperate un'immagine

della Kali VM dalla seguente URL: https://www.offensive-security.com/kali-linux-vmware-arm-image-download/.

NOTA

Se volete avere una lista dei link indicati in questo capitolo, sui quali poter cliccare, visitate http://nostarch.com/blackhatpython/.

Scaricate e decomprimete l'immagine, poi fate doppio clic su di essa in modo che VMWare Player la avvii. Lo username di default è *root* e la password è *toor*. Una volta fatto il login, dovreste ritrovarvi all'interno dell'ambiente desktop di Kali, come mostrato in Figura 1.1.

Figura 1.1 - Il desktop di Kali Linux.

La prima cosa che faremo è assicurarci che sia installata la corretta versione di Python. In questo libro viene usata la 2.7. Nella shell (**Applications -> Accessories -> Terminal**), eseguite il seguente comando:

```
root@kali:~# python --version
Python 2.7.3
root@kali:~#
```

Se avete scaricato l'immagine che vi ho raccomandato qualche riga sopra, Python 2.7 sarà già installato. Usando una differente versione di Python, è possibile che alcuni esempi riportati nel libro non funzionino. Siete quindi avvertiti.
A questo punto installiamo *easy_install* e *pip*. Questi sono molto simili al package manager *apt*, poiché consentono di installare direttamente delle librerie Python senza doverle scaricare, spacchettare e installare manualmente. Installiamo entrambi i gestori di pacchetto eseguendo il seguente comando:

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

Una volta che i package sono installati, possiamo fare un test veloce che consiste nell'installare un modulo che useremo nel Capitolo 7 per creare un trojan basato su GitHub. Eseguite il seguente comando nel terminale:

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

L'output del terminale dovrebbe indicare che la libreria viene scaricata e installata. A questo punto aprite una shell Python e verificate che sia realmente installata correttamente:

```
root@kali:~#: python
Python 2.7.3 (default, Mar 14 2014, 11:57:14)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import github3
>>> exit()
```

Se i risultati non sono identici a questi, allora c'è un problema di configurazione nel vostro ambiente Python. In questo caso assicuratevi di aver seguito i passi riportati sopra e di avere la versione corretta di Kali. Tenete a mente che, per la maggior parte degli esempi di questo libro, potete sviluppare il vostro codice in una varietà di sistemi operativi, inclusi Mac, Linux e Windows. Ci sono alcuni capitoli che sono specifici per Windows e in questi casi lo indicherò all'inizio del capitolo stesso.
Adesso che abbiamo messo in piedi la nostra macchina virtuale per fare hacking, installiamo un IDE Python per lo sviluppo.

WingIDE

In genere non sono un gran sostenitore di prodotti software commerciali, ma in questo caso devo dire che WingIDE è il miglior IDE che abbia usato negli ultimi sette anni a Immunity. WingIDE fornisce tutte le funzionalità base di un IDE, come l'autocompletamento e la descrizione dei parametri di una funzione, ma si contraddistingue dagli altri IDE per le sue capacità di debugging. Vi mostrerò brevemente un riassunto di ciò che si ha a disposizione con la versione commerciale di WingIDE, ma sentitevi liberi di scegliete la versione che fa al caso vostro.

> **NOTA** Per una comparazione tra le diverse versioni, visitate: https://wingware.com/wingide/features/.

Potete ottenere WingIDE da http://www.wingware.com/: vi raccomando di installare la versione di prova, in modo che possiate provare alcune delle funzionalità disponibili nella versione commerciale.
Potete sviluppare sulla piattaforma che volete, ma la scelta migliore, almeno per iniziare, probabilmente è quella di installare WingIDE nella vostra Kali VM. Assicuratevi di aver scaricato il pacchetto .deb a 32-bit di WingIDE e di averlo salvato nella vostra directory utente.
Fatto ciò, aprite un terminale ed eseguite il seguente comando:

```
root@kali:~# dpkg –i wingide5_5.0.9–1_i386.deb
```

Questo dovrebbe installare WingIDE. Se ottenete qualche errore di installazione, potrebbe esserci un problema di dipendenze non soddisfatte. In questo caso, eseguite semplicemente:

```
root@kali:~# apt–get –f install
```

Questo dovrebbe risolvere il problema delle dipendenze mancanti e installare WingIDE. Per verificare di averlo installato correttamente, assicuratevi di potervi accedere come mostrato in Figura 1.2.
Avviate WingIDE e aprite un nuovo file Python. Fatto ciò, la schermata dovrebbe somigliare a quella mostrata in Figura 1.3, con l'area di editing principale nella parte in alto a sinistra e un insieme di tab sulla parte bassa.

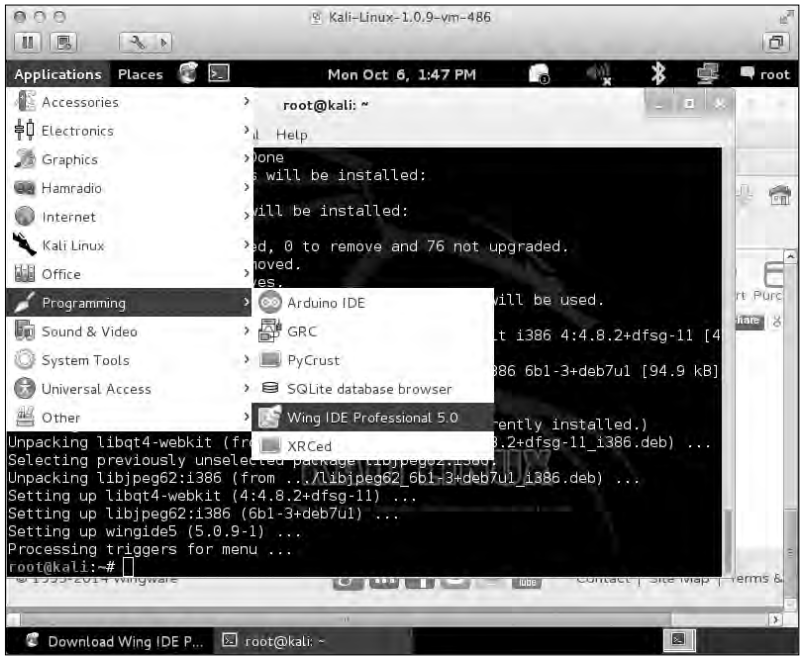


Figura 1.2 - Accesso a WingIDE dal desktop di Kali.

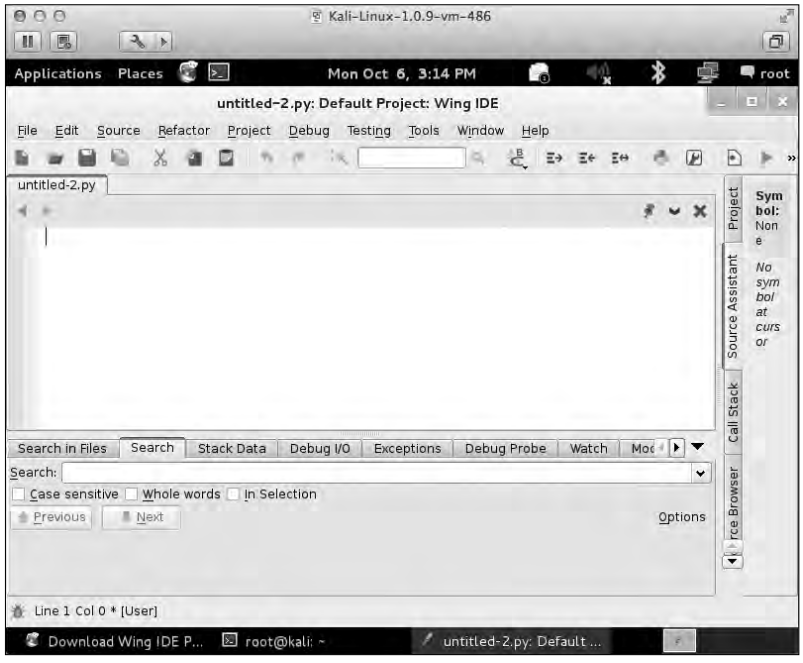


Figura 1.3 - Layout della finestra principale di WingIDE.

Scriviamo un semplice codice che illustra alcune utili funzionalità di WingIDE, incluso il *Debug Probe* e il tab dello *Stack Data*. Copiate il seguente codice nell'editor:

```
def 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

def convert_integer(number_string):
    converted_integer = int(number_string)
    return converted_integer

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

Questo è un esempio banale, ma è anche un'eccellente dimostrazione di come WingIDE vi può rendere semplice la vita. Salvate il file con il nome che preferite, cliccate sulla voce **Debug** del menu e selezionate l'opzione **Select Current as Main Debug File**, come mostrato in Figura 1.4.

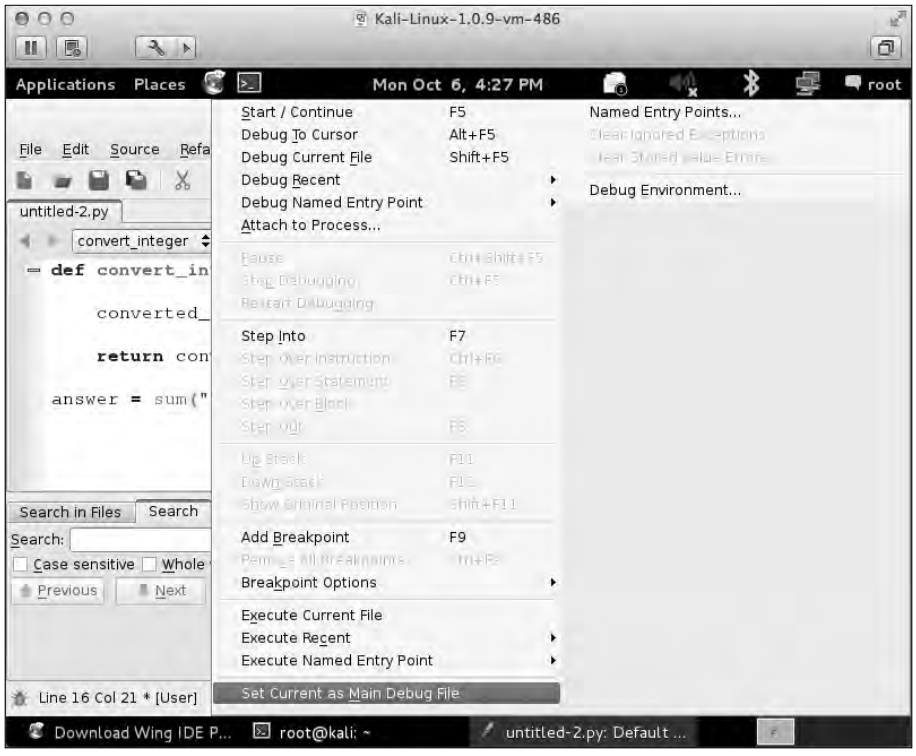


Figura 1.4 - Impostare il debugging per lo script Python.

Ora impostate un *breakpoint* sulla seguente linea di codice:

```
return converted_integer
```

Potete farlo cliccando sul margine sinistro o premendo il tasto **F9**. Dovreste vedere un piccolo punto rosso apparire nel margine. Ora eseguite lo script premendo il tasto **F5**: l'esecuzione dovrebbe fermarsi al breakpoint. Cliccando sul tab **Stack Data** dovreste vedere una schermata simile a quella di Figura 1.5.

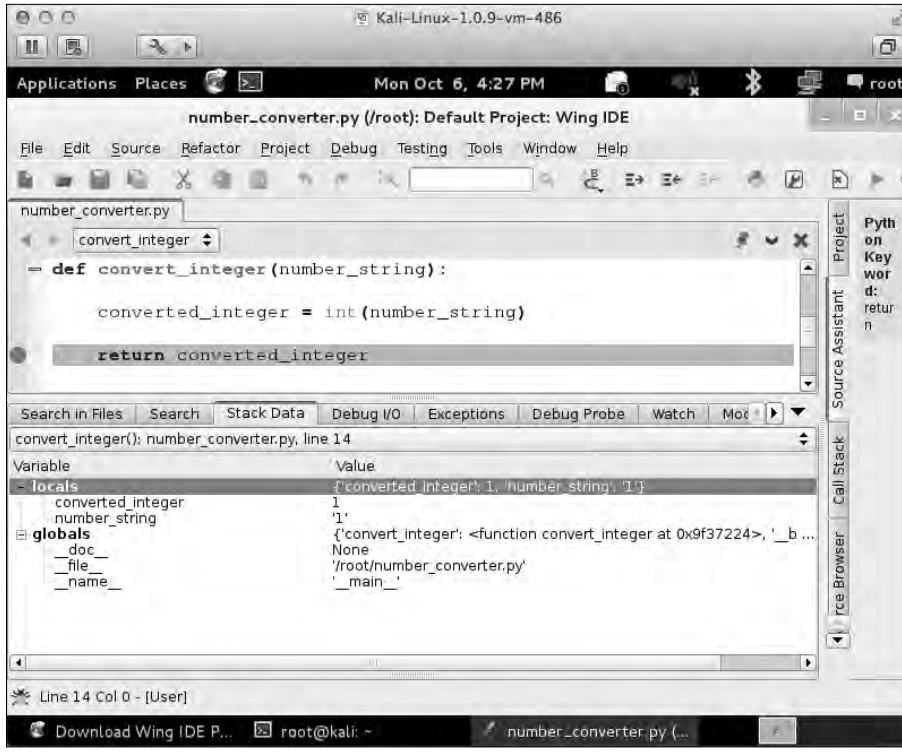


Figura 1.5 - Il tab Stack Data al raggiungimento del breakpoint.

Il tab **Stack Data** ci mostrerà alcune utili informazioni, come lo stato assunto delle variabili locali e globali nel momento in cui il nostro breakpoint viene raggiunto. Quando dovete ispezionare le variabili durante l'esecuzione, ad esempio per individuare dei bug, vi renderete conto che questo strumento vi consente di fare un debug del codice veramente efficace. Se cliccate sulla barra di scorrimento, potete vedere anche lo stack della chiamata corrente. Per vedere lo stack trace, date uno sguardo alla Figura 1.6.

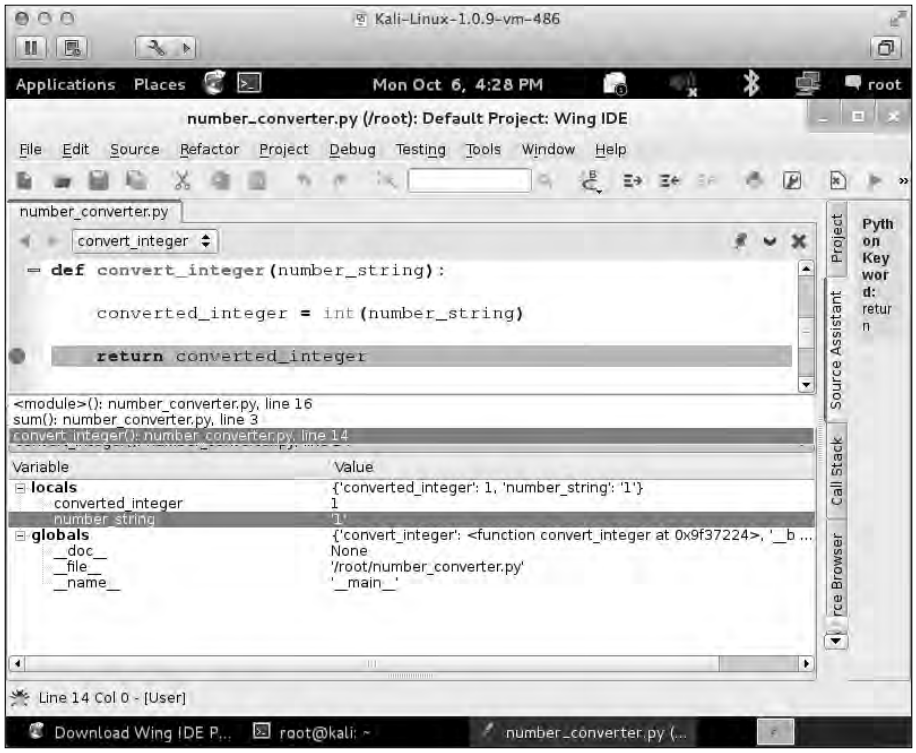


Figura 1.6 - Lo stack trace corrente.

Possiamo vedere che `convert_integer` è stata chiamata dalla funzione `sum` alla linea 3 del nostro script Python. Questo diventa molto utile se avete delle chiamate a una funzione ricorsiva o a una funzione che viene chiamata potenzialmente da differenti punti del programma. L'utilizzo del tab **Stack Data** vi tornerà molto utile nella vostra carriera di sviluppatori Python!

La prossima funzionalità importante che vedremo è il tab **Debug Probe**. Questo vi consente di passare velocemente a una shell Python nella quale avete a disposizione il contesto nel momento esatto in cui il breakpoint viene raggiunto, consentendovi quindi di ispezionare e modificare variabili, così come di scrivere piccole porzioni di codice di test, in modo da provare nuove idee o cercare di individuare dei problemi. La Figura 1.7 mostra come ispezionare la variabile `converted_integer` e cambiare il suo valore.

Dopo che avete fatto alcune modifiche, potete riprendere l'esecuzione dello script premendo il tasto **F5**. Anche se questo è un esempio molto semplice, consente di mostrare alcune delle più utili funzionalità di WingIDE per sviluppare e debuggare programmi Python.

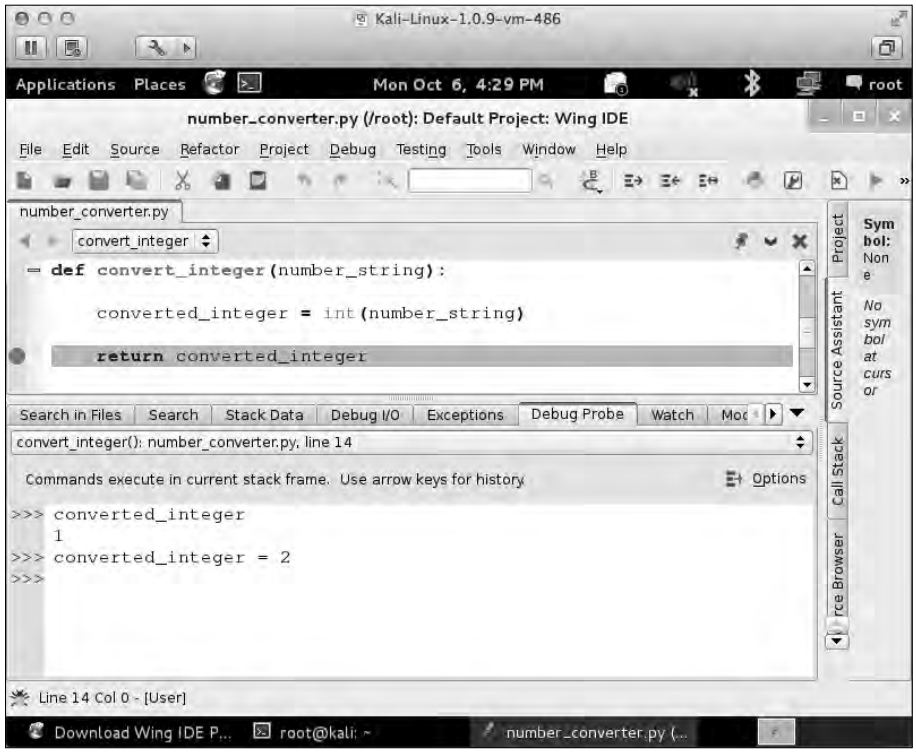


Figura 1.7 - Utilizzo del Debug Probe per ispezionare e modificare variabili locali.

NOTA

Se usate già un IDE che ha funzionalità comparabili con WingIDE, per cortesia inviatemi una mail o un tweet perché mi piacerebbe tanto conoscere il vostro parere al riguardo!

Questo è tutto ciò di cui avete bisogno per poter scrivere, eseguire e modificare il codice che vedrete nel resto del libro. Non dimenticate di creare una macchina virtuale che faccia da target nei capitoli specifici su Windows, anche se ovviamente potete utilizzare dell'hardware reale, senza alcun problema. Ora iniziamo con il vero divertimento!