

Einführung

Ich habe Python nur wegen des Hackings gelernt – und ich behaupte, dass das auch für eine ganze Reihe anderer Leute gilt. Ich habe viel Zeit damit verbracht, eine Sprache zu suchen, die sich für das Hacking und das Reverse Engineering eignet, und vor ein paar Jahren wurde mir klar, dass sich Python zum Anführer in der Liga der Hacking-Programmiersprachen entwickeln würde. Das Problematische an der Sache war, dass es keine Anleitung gab, wie man Python für eine Vielzahl von Hacking-Aufgaben nutzen konnte. Man musste sich durch Forum-Postings und Manpages kämpfen und viel Zeit damit verbringen, den Code immer wieder durchzugehen, damit die Dinge richtig funktionierten. Dieses Buch möchte diese Lücke schließen und Ihnen zeigen, wie man Python auf unterschiedlichste Art und Weise für das Hacking und das Reverse Engineering nutzen kann.

Das Buch vermittelt Ihnen die Theorie zu vielen Hacking-Tools und -Techniken, wie Debuggern, Hintertüren, Fuzzern, Emulatoren und Code-Injection, und erläutert ihnen gleichzeitig, wie Sie sich vorgefertigte Python-Tools zunutze machen können, wenn eigene Lösungen unnötig sind. Und Sie lernen nicht nur, wie man Python-basierte Tools nutzt, sondern wie man solche Tools in Python *entwickelt*. Doch seien Sie gewarnt: Dies ist keine allumfassende Referenz! Es gibt sehr viele in Python geschriebene Infosec-Tools (»Information Security«), die hier nicht behandelt werden. Dennoch erlaubt es Ihnen dieses Buch, Ihr Wissen auf viele weitere Anwendungen zu übertragen, sodass Sie jedes Python-Tool Ihrer Wahl nutzen, debuggen, erweitern und anpassen können.

Sie können dieses Buch auf verschiedene Arten durcharbeiten. Wenn Sie mit Python oder der Entwicklung von Hacking-Tools noch nicht vertraut sind, sollten Sie es vom Anfang bis zum Ende lesen. Sie werden die erforderliche Theorie kennen lernen, Unmengen an Python-Code entwickeln und ein solides Verständnis dafür erwerben, wie eine Vielzahl von Hacking- und Reverse-Engineering-Aufgaben anzupacken sind. Wenn Sie bereits mit Python vertraut sind und die Python-Bibliothek ctypes schon kennen, dann können Sie direkt mit Kapitel 2 loslegen. Diejenigen, die sich schon länger mit der Materie beschäftigen, können sich nach Gutdünken innerhalb des

Buches bewegen und Codefragmente oder bestimmte Abschnitte so nutzen, wie es ihre tägliche Arbeit erfordert.

Ich widme einen Großteil der Zeit den Debuggern, angefangen mit der Debugger-Theorie in Kapitel 2 bis hin zum Immunity Debugger in Kapitel 5. Debugger sind ein wichtiges Werkzeug für jeden Hacker und ich betone ausdrücklich, dass ich sie umfassend behandle. Danach lernen Sie in den Kapiteln 6 und 7 einige Hooking- und Injection-Techniken, die einen Teil der Debugging-Konzepte zur Programmkontrolle und Speicherverarbeitung erweitern.

Der nächste Abschnitt des Buches zeigt, wie man Anwendungen mithilfe sogenannter Fuzzer knackt. In Kapitel 8 führen wir in das Fuzzing ein, und Sie werden einen eigenen einfachen Datei-Fuzzer entwickeln. In Kapitel 9 nutzen wir das leistungsfähige Sulley Fuzzing-Framework, um einen echten FTP-Daemon zu knacken, und in Kapitel 10 werden Sie lernen, wie man einen Fuzzer entwickelt, der Windows-Treiber knackt.

In Kapitel 11 schließlich lernen Sie, wie man Aufgaben der statischen Analyse unter IDA Pro (einem populären Tool zur binären statischen Analyse) automatisiert. Wir runden das Buch in Kapitel 12 mit PyEmu, dem Python-basierten Emulator, ab.

Ich habe versucht, die Codelistings kurz zu halten, wobei ich an bestimmten Stellen umfangreiche Kommentare eingefügt habe, die die Funktionsweise des Codes erläutern. Ein Teil des Erlernens einer neuen Sprache oder einer neuen Bibliothek besteht in der schweißtreibenden Arbeit, den Code tatsächlich »herunterzuschreiben« und die eigenen Fehler zu entdecken. Ich empfehle Ihnen, den Code wirklich einzugeben, auch wenn der gesamte Quellcode auf www.dpunkt.de/python-hacking abgelegt ist, damit Sie ihn bequem herunterladen können.

Lassen Sie uns beginnen!