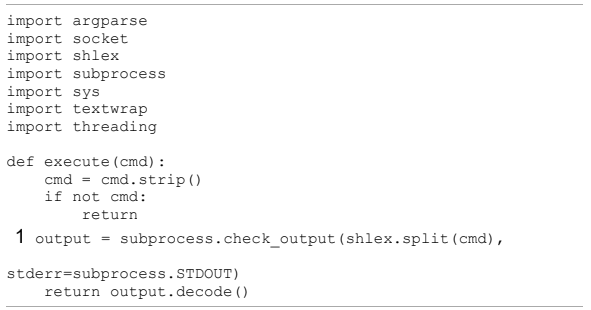
**Merancang Netcat**

Netcat adalah pisau utilitas jaringan, jadi tidak mengherankan jika administrator sistem yang cerdas menghapusnya dari sistem mereka. Alat yang berguna seperti itu akan menjadi aset yang cukup besar jika penyerang berhasil menemukan jalan masuk.

Dengan netcat, Anda dapat membaca dan menulis data di seluruh jaringan, artinya Anda dapat menggunakannya untuk menjalankan perintah jarak jauh, meneruskan file bolak-balik, atau bahkan membuka shell jarak jauh. Pada lebih dari satu kesempatan, kami telah menemukan server yang tidak menginstal netcat tetapi memiliki Python. Dalam kasus ini, akan berguna untuk membuat klien dan server jaringan sederhana yang dapat Anda gunakan untuk mendorong file, atau pemroses yang memberi Anda akses baris perintah. Jika Anda telah membobol aplikasi web, ada baiknya menjatuhkan Python callback untuk memberi Anda akses sekunder tanpa harus terlebih dahulu membakar salah satu trojan atau backdoor Anda. Membuat alat seperti ini juga merupakan latihan Python yang bagus, jadi mari kita mulai menulis netcat.py:



Di sini, kita mengimpor semua pustaka yang diperlukan dan mengatur fungsi execute, yang menerima perintah, menjalankannya, dan mengembalikan output sebagai string. Fungsi ini berisi pustaka baru yang belum kita bahas: pustaka subprocess. Pustaka ini menyediakan antarmuka pembuatan proses yang memberi Anda sejumlah cara untuk berinteraksi dengan program klien. Dalam kasus ini **1**, kita menggunakan metode check\_output, yang menjalankan perintah pada sistem operasi lokal dan kemudian mengembalikan output dari perintah itu. Sekarang mari kita buat blok utama kita yang bertanggung jawab untuk menangani argumen baris perintah dan memanggil fungsi kita yang lain:

A screenshot of a computer program

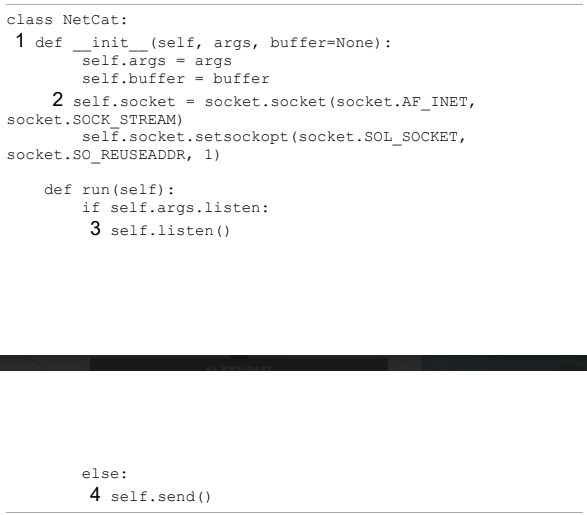
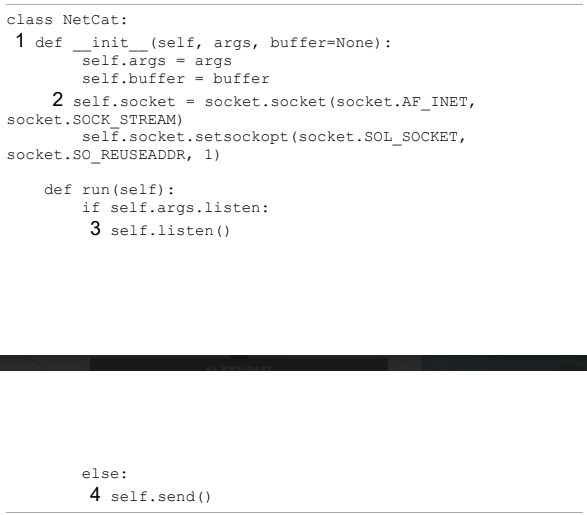
Description automatically generated

Kami menggunakan modul argparse dari pustaka standar untuk membuat antarmuka baris perintah **1**. Kami akan menyediakan argumen sehingga dapat dipanggil untuk mengunggah file, menjalankan perintah, atau memulai shell perintah.

Kami memberikan contoh penggunaan yang akan ditampilkan program ketika pengguna memanggilnya dengan –help **2** dan menambahkan enam argumen yang menentukan bagaimana kami ingin program berperilaku **3**. Argumen -c mengatur shell interaktif, argumen -e mengeksekusi satu perintah tertentu, argumen -l menunjukkan bahwa listener harus diatur, argumen -p menentukan port untuk berkomunikasi, argumen -t menentukan IP target, dan argumen -u menentukan nama file yang akan diunggah. Baik pengirim maupun penerima dapat menggunakan program ini, sehingga argumen menentukan apakah itu dipanggil untuk mengirim atau mendengarkan. Argumen -c, -e, dan -u menyiratkan argumen -l, karena argumen tersebut hanya berlaku untuk sisi listener. Sisi pengirim membuat koneksi ke listener, sehingga hanya perlu argumen -t dan -p untuk mendefinisikan target listener.

Jika kita mengaturnya sebagai listener **4**, kita memanggil objek NetCat dengan string buffer kosong. Jika tidak, kami mengirim konten buffer dari stdin. Akhirnya, kita memanggil metode run untuk memulainya

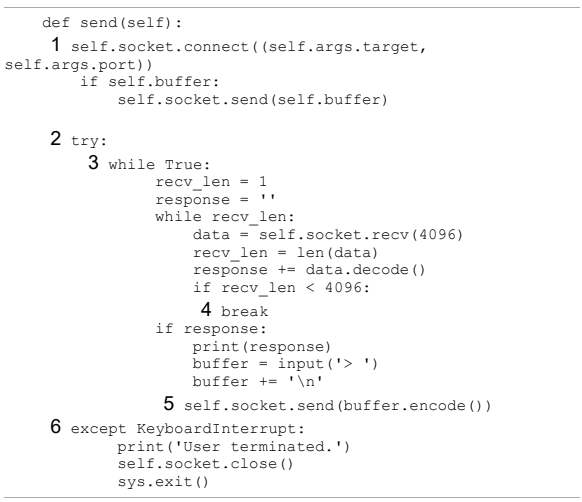
Sekarang mari kita mulai menempatkan pipa ledeng untuk beberapa fitur ini, dimulai dengan kode klien kita. Tambahkan kode berikut di atas blok utama:



Kami menginisialisasi objek NetCat dengan argumen dari baris perintah dan buffer **1** dan kemudian membuat objek soket **2**.

Metode run, yang merupakan titik masuk untuk mengelola objek NetCat, cukup sederhana: mendelegasikan eksekusi ke dua metode. Jika kita menyiapkan listener, kita memanggil metode listen **3**. Jika tidak, kita memanggil metode send **4**.

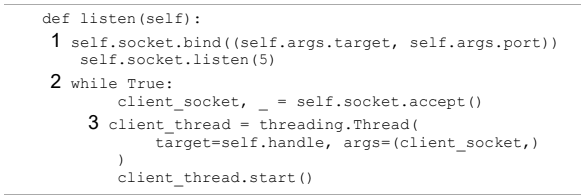
Sekarang mari kita tulis metode send itu:



Kami terhubung ke target dan port **1**, dan jika kami memiliki buffer, kami mengirimkannya ke target terlebih dahulu. Kemudian kita mengatur blok try/catch sehingga kita dapat menutup koneksi secara manual dengan CTRL-C **2**. Selanjutnya, kita memulai loop **3** untuk menerima data dari target. Jika tidak ada data lagi, kita keluar dari loop **4**. Jika tidak, kami mencetak data respons dan berhenti sejenak untuk mendapatkan input interaktif, mengirim input itu **5**, dan melanjutkan loop.

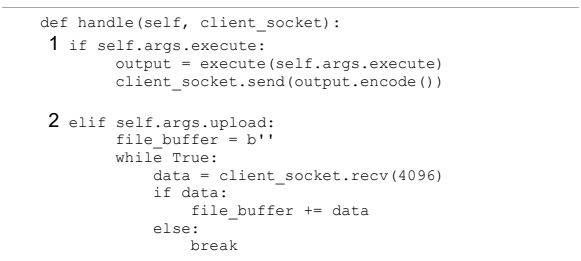
Loop akan berlanjut hingga KeyboardInterrupt terjadi (CTRL- C) **6**, yang akan menutup soket.

Sekarang mari kita tulis metode yang dieksekusi ketika program berjalan sebagai listener:



Metode listen mengikat target dan port **1** dan mulai mendengarkan dalam loop **2**, meneruskan soket yang terhubung ke metode handle **3**.

Sekarang mari kita terapkan logika untuk melakukan unggahan file, menjalankan perintah, dan membuat shell interaktif. Program dapat melakukan tugas-tugas ini saat beroperasi sebagai listener.



A computer code with black text

Description automatically generated

Metode handle mengeksekusi tugas yang sesuai dengan argumen baris perintah yang diterimanya: menjalankan perintah, mengunggah file, atau memulai shell. Jika perintah harus dieksekusi **1**, metode handle meneruskan perintah tersebut ke fungsi execute dan mengirim output kembali ke soket. Jika file harus diunggah **2**, kami menyiapkan loop untuk mendengarkan konten pada soket pendengar dan menerima data sampai tidak ada lagi data yang masuk. Kemudian kita menulis yang terakumulasi content ke file yang ditentukan. Akhirnya, jika shell akan dibuat **3**, kita mengatur loop, mengirim prompt ke pengirim, dan menunggu string perintah kembali. Kami kemudian menjalankan perintah dengan menggunakan fungsi execute dan mengembalikan output perintah ke pengirim.

Anda akan melihat bahwa shell memindai karakter baris baru untuk menentukan kapan harus memproses perintah, yang membuatnya ramah netcat. Artinya, Anda dapat menggunakan program ini di sisi listener dan menggunakan netcat sendiri di sisi pengirim. Namun, jika Anda menyulap klien Python untuk berbicara dengannya, ingatlah untuk menambahkan karakter baris baru. Dalam metode send, Anda dapat melihat kita menambahkan karakter baris baru setelah kita mendapatkan input dari konsol.