

Python przyjacielem administratora

WAŻNA USŁUGACH

Jedną z największych zalet Pythona jest jego prostota.

Pokażemy, w jaki sposób administratorzy

i programiści mogą wykorzystać ten język w pracy z adresami IP i nazwami domenowymi.

ADNS

Konwersja adresów IP znajdujących się w logach serwerów na nazwy domenowe jest wykonywana automatycznie, najczęściej przez jeden z typowych skryptów korzystających ze standardowych bibliotek systemowych. Jest to jednak rozwiązanie najgorsze z możliwych, bowiem *gethostbyname()* może za jednym razem obsłużyć tylko jedno zapytanie i cały proces trwa bardzo długo, zwłaszcza jeśli mamy do przerobienia dziesiątki czy nawet setki tysięcy adresów.

Znacznie lepiej w tej roli sprawdza się biblioteka asynchroniczna GNU ADNS [1]. Ma ona szereg zalet, z których najważniejszą jest możliwość przetwarzania wielu zapytań równocześnie ze względu na nieblokujący sposób działania. Biblioteka ta idealnie sprawdza się w typowych zastosowaniach, na przykład przy sprawdzaniu nazw hostów czy poszczególnych rekordów DNS. Wraz z biblioteką rozpowszechnianych jest kilka programów narzędziowych, które pozwalają docenić wydajność ADNS.

Bibliotekę i narzędzia instalujemy w Debianie (i klonach, takich jak Ubuntu) poleceniem:

```
apt-get install adns-tools
```

W rezultacie w systemie zainstalowana zostaje sama biblioteka wraz z przykładami: *adnslogres*, ultraszybka wersja standardowego *logresolve* rozprowadzanego wraz z Apache, *adnsresfilter*, filtr przepisujący wejście z podmianą adresów IP na nazwy domenowe i *ad-*

nshost, wersja standardowego programu *host*, przeznaczona zwłaszcza do wykorzystania w skryptach. Prosty przykład użycia *adnsresfilter* znajduje się na Listingu 1.

Z kolei pythonowe dowiązania do biblioteki ADNS instalujemy poleceniem:

```
apt-get install python-adns
```

Zobaczmy, jak w praktyce wygląda korzystanie z interfejsu do naszej biblioteki; używamy w tym celu interpretera Pythona. Najpierw importujemy moduł i inicjujemy bibliotekę:

```
import adns
s = adns.init()
```

Możemy teraz utworzyć zapytanie. Odpowiedź zawsze otrzymujemy w postaci krotki (*status, CNAME, przedawienie, odpowiedź właściwa*). *Status* oznacza kod błędu, *CNAME* to nazwa kanoniczna (jeśli zapytanie nie odnosi się do niej, zwracana jest wartość *None*), *przedawienie* to czas, kiedy odpowiedź traci wartość, zaś *odpowiedź właściwa* zawiera krotkę z rekordami, których format zależy od zapytania.

Weźmy prosty przykład:

```
s.synchronous("linux-magazine.pl",
adns.rr.A)
```

Oznacza on, że dokonujemy synchronicznego zapytania o rekord DNS A dla domeny *linux-magazine.pl*. Oto, co otrzymamy w rezultacie:

```
(0, None, 1225824153,
('80.237.227.144',))
```

Początkowe zero oznacza, że zapytanie powiodło się. *None* wskazuje, że zapytanie nie dotyczyło bądź nie zawiera formy kanonicznej. Na ostatniej pozycji mamy krotkę z adresem IP. Z kolei czas odpowiadający wartości *1225824153* łatwo sprawdzamy za pomocą poniższej sekwencji:

```
import time
time.ctime(1225824153)
```

W rezultacie otrzymujemy dokładną datę i godzinę utraty ważności naszego zapytania.

W ten sposób możemy łatwo skonstruować zapytania dotyczące innych typów rekordów, w tym PTR. Aby jednak odwzorować adresy IP na nazwy domenowe, musimy pamiętać o odwrotnej notacji z *in-addr.arpa*. Przykładowo, jeśli chcemy sprawdzić, czy faktycznie *127.0.0.1* zostanie odwzorowany jako *localhost*, wydajemy polecenie:

```
s.synchronous("1.0.0.127.
in-addr.arpa", adns.rr.PTR)
```

W ten sposób możemy łatwo napisać skrypt sprawdzający nazwy hostów w danej sieci. Przyjrzyjmy się Listingowi 2. Sprawdza on po kolei nazwy wszystkich hostów z sieci *213.180.130.** i jeśli znajdzie jakiś adres IP posiadający nazwę domenową, wyświetli daną parę na standardowym wyjściu. Pierwszych kilka wierszy możemy zobaczyć na Listingu 3.

GeoIP

Geolokalizacja umożliwia – do pewnego stopnia i w przybliżeniu – określenie fizycznego umiejscowienia danego hosta. Bardzo

Listing 1: Prosty przykład użycia filtra *adnsresfilter*

```
$ echo Wykryto skanowanie portów z hosta 217.74.65.119 | adnsfilter
Wykryto skanowanie portów z hosta other.interia.pl
```

Listing 2: Prosty skrypt sprawdzający nazwy hostów w danej sieci

```
import adns

s = adns. init ()
for i in range (255):
    ip = str (i) + ". 130.180.213. in-addr. arpa"
    a = s. synchronous (ip, adns. rr. PTR)
    if a[3]:
        print (ip + "==" + a[3][0])
```

Listing 3: Fragment wyniku działania skryptu z Listingu 2

```
12.130.180.213. in-addr. arpa ==> spubserv. onet. pl
13.130.180.213. in-addr. arpa ==> z3virt. onet. pl
14.130.180.213. in-addr. arpa ==> ckxvirt. onet. pl
17.130.180.213. in-addr. arpa ==> newsgate1. onet. pl
18.130.180.213. in-addr. arpa ==> newsgate. onet. pl
27.130.180.213. in-addr. arpa ==> rtsp. onet. pl
28.130.180.213. in-addr. arpa ==> smtp4. poczta. onet. pl
29.130.180.213. in-addr. arpa ==> smtp3. poczta. onet. pl
30.130.180.213. in-addr. arpa ==> smtp2. poczta. onet. pl
31.130.180.213. in-addr. arpa ==> smtp1. poczta. onet. pl
32.130.180.213. in-addr. arpa ==> smtp5. poczta. onet. pl
33.130.180.213. in-addr. arpa ==> smtp-marlin. poczta. onet. pl
```

często wykorzystują to firmy reklamujące swoje produkty i usługi w Internecie dla odbiorców w różnych krajach: nawet jeśli potencjalny klient odwiedza stronę angielskojęzyczną, jego uwagę przyciągnie ogłoszenie w ojczystym języku.

Administratorzy spotykają się z geolokalizacją choćby poprzez skrypty generujące statystyki na podstawie logów serwera WWW. Dane te mogą być bardzo istotne, zwłaszcza dla firm działających globalnie. Jeśli okaże się, że dynamicznie rośnie liczba gości z danego kraju, jest to silny argument za stworzeniem wersji witryny w tym właśnie języku. Interesująca może być również analiza zmian zachowań klientów z różnych krajów na przestrzeni czasu.

Listing 3

```
{'city': 'Szczecin', 'region_name':
'Zachodniopomorskie', 'region': '87',
'area_code': 0, 'time_zone': 'Europe/Warsaw', 'longitude':
14.58329963684082, 'country_code3':
'POL', 'latitude':
53.416698455810547, 'postal_code':
None, 'dma_code': 0, 'country_code':
'PL', 'country_name': 'Poland'}
```

Chcąc efektywnie skorzystać z mechanizmów geolokalizacji, potrzebujemy przede wszystkim dobrego narzędzia. Firma MaxMind publikuje bazy geolokalizacyjne w wersji darmowej i komercyjnej, jak również zestaw narzędzi, bibliotek i dowiązań dla różnych języków – w tym dla Pythona.

Przede wszystkim powinniśmy pobrać darmowe bazy ze strony [1]. Pliki z bazami rozpakowujemy i umieszczamy w katalogu `/usr/local/share/GeoIP/`. Wystarczy nam jedynie plik `GeoLiteCity.dat` – zawiera on bardziej szczegółowe dane niż `GeoLiteCountry.dat`. Oczywiście bardzo zależy nam na dużej dokładności, możemy wykupić dostęp do baz komercyjnych, jednak różnice wynoszą kilka dziesiątych procenta.

Następnie instalujemy odpowiedni pakiet. W Debianie robimy to na przykład za pomocą:

```
apt-get install python-geoip geoip-bin
```

W ten sposób instalujemy zarówno narzędzia do sprawdzania hostów, jak i dowiązania do Pythona. Narzędzie `geoiplookup` pozwala sprawdzić, w jakim kraju znajduje się dany host:

```
$ geoiplookup mon. gov. pl
GeoIP Country Edition: PL, Poland
```

Jednak za pomocą bazy `GeoLiteCity` i dowiązań do Pythona uzyskujemy znacznie więcej informacji niewiele większym wysiłkiem. Przede wszystkim importujemy odpowiedni moduł:

```
import GeoIP
```

Następnie podajemy ścieżkę do bazy, którą pobraliśmy ze strony firmy MaxMind:

```
geoip_lib = '/usr/local/share/
GeoIP/GeoLiteCity.dat'
```

Następnie otwieramy bibliotekę:

```
gi = GeoIP. open (geoip_lib, GeoIP.
GEOIP_STANDARD)
```

Drugi argument powyżej wskazuje na sposób przechowywania bazy. Oprócz standardowego mamy też możliwość przechowywania bazy w pamięci podręcznej (`GEOIP_MEMORY_CACHE`), przy czym każdy kraj zajmuje około 1 MB, jak również w pamięci dzielonej (`GEOIP_MEMORY_SHARED`). Wybór zależy od zastosowań – powinniśmy go określić podczas testów na danych, na których będziemy operować.

Samo zapytanie wykonujemy następująco:

```
gi. record_by_name ("home. pl")
```

Rezultat możemy obejrzeć na Listingu 3. Jak widzimy, możemy uzyskać całkiem sporo danych, co oczywiście nie oznacza, że zawsze są one dokładne. Mimo to bardzo często dokładność wyników zwracanych przez zapytania GeoIP może nas zdziwić.

Na koniec

Python oferuje znacznie więcej narzędzi związanych z obróbką adresów IP i nazwami domenowymi; przedstawiliśmy tu jedynie dwa przykłady zastosowań. Mamy nadzieję, że zainspirowaliśmy do wykorzystania przedstawionych w artykule mechanizmów we własnych skryptach. ■

INFO

[1] Strona domowa projektu ADNS:
<http://www.gnu.org/software/adns/>

[2] Darmowe bazy geolokacyjne krajów i miast: <http://www.maxmind.com/app/geolitecountry>, <http://www.maxmind.com/app/geolitecity>