

Skrypty automatyzujące zadania FTP

# BEZ TRZYMANKI

Jeśli często wykonujemy takie same, rutynowe czynności z użyciem protokołu FTP, warto je zautomatyzować.

DAVID TANSLEY



**F**ile Transfer Protocol (FTP) jest popularnym standardem transmitowania plików w sieciach TCP/ IP. Podstawowy protokół FTP ma już wiele lat i, choć nie jest najbezpieczniejszy, wróży mu się dalsze długie lata ze względu na wydajność i popularność. W tym artykule prezentujemy techniki automatyzacji transmisji FTP. Nie poruszamy tematyki serwerów FTP takich jak wu-ftp czy vs-ftp; zamiast tego koncentrujemy się na stronie klienckiej. Pokazujemy, jak za pomocą skryptu połączyć się z serwerem FTP i pobrać pliki.

Część artykułu poświęcimy opisowi bezpieczeństwa podczas logowania. Musimy jednak pamiętać, że siła zabezpieczeń zależy od tego, jakie zasady przyjęto w danej firmie czy organizacji. Na przykład możliwe jest przesyłanie plików protokołem FTP przez tunel VPN. Istnieją także bezpieczne wersje protokołu FTP, zapewniające szyfrowanie. Te szczegóły pozostawiamy do rozstrzygnięcia Czytelnikowi, a dziś koncentrujemy się na automatyzacji transferu plików.

## Plik .netrc

Bez względu na to, czy korzystamy z protokołu FTP w sposób wsadowy czy interaktywny, warto skorzystać z mechanizmu zapewniającego automatyczne logowanie do zdalnego serwera. W katalogu *\$HOME* użytkownika istnieje specjalny plik *.netrc* automatyzujący logowanie do usług FTP. Prawa dostępu do pliku *.netrc* powinny być ustawione tak, aby tylko właściciel mógł odczytywać i zapisywać w nim dane (*chmod 600*). Plik może zawierać wiele wpisów odpowiadających poszczególnym serwerom zdalnym. Jeżeli plik *.netrc* istnieje, próba nawiązania połączenia z jednym z wymienionych tam komputerów spowoduje, że zostanie użyta nazwa użytkownika i hasło wskazane w odpowiedniej pozycji pliku *.netrc*.

Istnieją dwa formaty pliku *.netrc*. Autor niniejszego artykułu woli format następujący:

```
machine <zdalny_host> \\  
login <nazwa_uzytkownika> \\  
password <haslo> \\  
password <haslo_konta>
```

```
password <haslo> \\  
password <haslo_konta>
```

Każdemu hostowi zdalnemu odpowiada jeden taki wpis. „Hasło” to ciąg znaków wymagany do połączenia z hostem zdalnym. Hasło konta jest potrzebne tylko wtedy, gdy zdalny host wymaga specjalnej metody uwierzytelniania. Zazwyczaj wystarczy podanie pierwszych trzech pozycji.

## Wydruk 1: plik .netrc

```
machine ftp.emea.ibm.com login  
anonymous password david.tan-  
sley@btinternet.com  
machine uk011x6001 login dxtans  
password 10opy  
machine uk041x6003 login dxtans  
password mas123
```

## Wydruk 2: ftp 1

```
#!/bin/bash
# ftp1
ftp -i -v <<mayday
open uk011x6001
ascii
lcd /tmp
cd /etc
get hosts
quit
mayday
```

Na Wydruku 1 przedstawiono plik *.netrc* z trzema wpisami. Pierwszy wpis to publiczny serwer FTP. Przez nazwę użytkownika *anonymous* zazwyczaj uzyskuje się dostęp do folderu publicznego, na który można wysłać pliki lub je z niego pobierać. Za dobry zwyczaj przy stosowaniu nazwy użytkownika *anonymous* uważa się podawanie własnego adresu e-mail jako hasła; nie jest to jednak obowiązkowe. Pozostałe wpisy na Wydruku 1 dotyczą hostów znajdujących się w sieci wewnętrznej.

Aby uzyskać połączenie z hostem uk011x6001 wystarczy wpisać:

```
$ ftp uk011x6001
```

Jak wspomnieliśmy, podczas uruchamiania program FTP poszukuje pliku *.netrc*. W tym przypadku, ponieważ łączę się z hostem uk011x6001, klient poszukuje w pliku *.netrc* nazwy hosta uk011x6001; następnie pobiera stamtąd nazwę i hasło i używa ich do zalogowania się na serwer.

## Pobieranie pliku

Napiszmy teraz prosty skrypt FTP, który łączy się ze zdalnym hostem i pobiera stam-

## Wydruk 3: Wynik działania skryptu ftp 1

```
$ ftp1
Connected to uk011x6001 (168.14.2.4).
220 uk011x6001 FTP server (Version 4.1 Wed Mar 26 16:45:44 CST 2003) ready.
331 Password required for dxtans.
230-Last unsuccessful login: Tue Jan 18 12:18:34 GMT 2005 on /dev/pts/0
230-Last login: Tue Mar 29 18:40:33 BST 2005 on ftp from ::ffff:168.14.2.9
230 User dxtans logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
200 Type set to A; form set to N.
Local directory now /tmp
250 CWD command successful.
local: hosts remote: hosts
227 Entering Passive Mode (162,14,2,4,209,161)
150 Opening data connection for hosts (2370 bytes).
226 Transfer complete.
2443 bytes received in 0.00135 secs (1.8e+03 Kbytes/sec)
221 Goodbye.
```

ąd plik */etc/hosts* (Wydruk 2). Zwróćmy uwagę, że program FTP uruchamiamy z wyłączonym trybem interaktywnym, a włączonym trybem dostarczania dodatkowych informacji. Stosujemy metodę *w tym miejscu dokument* (here document) implementowaną przez zastosowanie symbolu *<<*. Tłumacząc na polski: wszystko, co znajduje się między pierwszym a drugim wystąpieniem słowa *mayday*, jest interpretowane jako dane odczytywane ze standardowego wejścia. Najpierw otwieramy połączenie z hostem uk011x6001, zlecamy programowi FTP tryb transferu ASCII i zmie-

niamy katalog lokalny na */tmp*, a zdalny na */etc*. Następnie pobieramy plik *hosts*, w wyniku czego zapisywany jest on w katalogu lokalnym */tmp*. Na koniec opuszczamy program *FTP*. Drugie wystąpienie słowa *mayday* powoduje zakończenie danych dostarczanych ze standardowego wejścia; ponieważ dalej nie ma już żadnych linijek, skrypt kończy działanie.

Po uruchomieniu kodu z Wydruku 2 na ekranie wyświetlane są dane pokazane na Wydruku 3. Zauważmy, że domyślnie FTP próbuje wykonać transfer w trybie binarnym. Zmieniamy to poleceniem *ascii*. Zwróćmy

## Często stosowane opcje programu FTP

Zanim zagłębimy się w temat automatyzacji transferów FTP, przyjrzyjmy się najczęściej stosowanym opcjom programu FTP:

-p : określa, że połączenie ma być realizowane w trybie pasywnym. Ta opcja jest często wykorzystywana po stronie klienta. Metoda pasywna umożliwia transfer użytkownikom znajdującym się za zaporami, ponieważ w takiej konfiguracji to klient informuje serwer, który port ma zostać użyty do transmisji. Obecnie program domyślnie łączy się w trybie pasywnym; w przypadku starszych wersji opcję tę trzeba podać jawnie.

-i : wyłącza interaktywne zapytania podczas negocjowania transferów FTP. Jeśli z góry znamy pliki, które chcemy umieścić na serwerze lub z niego pobrać, na pewno warto użyć tej opcji.

-n : blokuje próbę automatycznego zalogowania przy wstępnym połączeniu. Jeśli automatyczne logowanie jest włączone, FTP poszukuje informacji wymaganych do zalogowania w pliku *.netrc*. Jeśli w pliku tym nie ma pasującego pliku, program prosi o podanie nazwy i hasła.

-v : powoduje, że program dostarcza więcej informacji o wykonywanych czynnościach.

## Wydruk 4: ftp2

```
#!/bin/bash
# ftp2
list="hosts hosts.allow hosts.deny"
for files in $list
do
  ftp -i -v <<mayday
  open uk011x6001
  ascii
  lcd /tmp
  cd /etc
  get $files
  quit
  mayday
done
```

także uwagę, że program FTP pokazuje, iż faktycznie dane są transmitowane w trybie pasywnym.

## Lista plików

Metody „w tym miejscu dokument” można także użyć do przesłania plików wymienionych na liście; jednak oznacza to konieczność nawiązania i zakończenia połączenia dla każdego transferu. Technikę tę demonstrujemy na Wydruku 4. Nie można tutaj zastosować w kodzie sterującym programem FTP pętli *or*, ponieważ na tym etapie nie wykonujemy operacji na lokalnej powłoce *bash*, lecz na zdalnym hoście FTP.

## Sztuczki z plikiem .netrc

W niektórych przypadkach przed rozpoczęciem sesji FTP musimy sprawdzić, czy plik *.netrc* istnieje. Na przykład możemy zażyć sobie, aby skrypt wykonywał uwierzytelnienie w pewien sposób, gdy plik *.netrc* istnieje, ale w inny, gdy tego pliku nie ma. O ile tylko to możliwe, dobrze jest korzystać z pliku *.netrc*. Jeśli nie trzeba, nie należy stosować uwierzytelniania interaktywnego; a już na pewno nie wolno zapisywać hasła bezpośrednio w skrypcie sterującym transmisją FTP. Na Wydruku 5 pokazano przykładowy kod sprawdzający, czy plik *.netrc* istnieje i jest do odczytania przez skrypt; jeśli te warunki są spełnione, skrypt wyświetla ten plik na wyjściu standardowym.

Jeśli chcemy umożliwić użytkownikowi wybranie jednego z wpisów *.netrc*, musimy przedstawić mu opcje dostępne do wybo-

## Wydruk 5: ftp4

```
#!/bin/bash
# ftp4
netrc_file=$HOME/.netrc
if [ -r "$netrc_file" ]
then
  cat $netrc_file | awk '{print $2,$4}'
else
  echo "brak pliku $netrc_file "
fi
```

ru. Można to zrobić przekazując plik *.netrc* poleceniu *cat* z opcją *-n* – każda linijka wyświetlona na standardowym wyjściu jest wtedy opatrzona numerem. Na Wydruku 6 pokazano, jak zrealizować takie zadanie. Najpierw sprawdzamy, czy plik *.netrc* daje się odczytać (a więc także czy istnieje). Określamy, ile w pliku jest rekordów:

```
max_recs=`cat \\  
$netrc_file | awk \\  
'END{print NR}'`
```

Dlaczego nie zastosowaliśmy polecenia *wc -l*? Przecież byłoby prościej. Niestety, w wyniku zastosowania tego programu na ekran wyrzucane są dane sformatowane przez wypełnianie spacjami, a to nie wygląda najlepiej. Polecenie *cat* powoduje wyświetlenie wszystkich wpisów na wyjściu standardowym; na ekranie widać jednak tylko pola hosta i nazwy użytkownika. Na Wydruku 7 pokazano wynik działania tego kodu. Teraz prosimy użytkownika o wprowadzenie liczby odpowiadającej żądanemu rekordowi; posługujemy się przy tym zmienną *\$max\_recs*, która zawiera liczbę wszystkich rekordów. Na podstawie wybranej liczby funkcja *NR* języka *awk* pobiera wymagane pola z odpowiedniego rekordu. Wynik jest wyświetlany na wyjściu standardowym.

Na Wydruku 6 zaprezentowano jeden ze sposobów zaimplementowania interfejsu z menu. Nie zastosowano tutaj kontroli błędów z prawdziwego zdarzenia (zaledwie sprawdzenie, czy wartość mieści się w odpowiednim przedziale liczb), ponieważ chodzi jedynie o zademonstrowanie, że przygotowanie menu na bazie skryptu w *bashu* nie zajmuje wiele czasu.

## Wydruk 6: ftp5

```
#!/bin/bash
# ftp5
netrc_file=$HOME/.netrc
if [ -r "$netrc_file" ]
then
  max_recs=`cat $netrc_file | awk 'END{print NR}'`
  cat -n $netrc_file | awk '{print $1,": serwer [",$3,"], użytkownik [",$5,"]}'

  echo -n " Wybierz rekord [ 1 .. $max_recs ] : "
  read ans
  if [ $ans -ge 1 ] && [ $ans -le $max_recs ]
  then
    host=`cat $netrc_file | awk "NR==$ans"|awk '{print $2}'`
    user=`cat $netrc_file | awk "NR==$ans"|awk '{print $4}'`
    password=`cat $netrc_file | awk "NR==$ans"|awk '{print $6}'`
  else
    echo "nieprawidłowa wartość, wybierz z przedziału [ 1 .. $max_recs ]"
    exit 1
  fi
  # $ans w odpowiednim przedziale
  echo -e "wybrano:\nhost [$host]\nużytkownik [$user]\nhasło [$password]"
else
  echo " Nie można odczytać pliku $netrc_file"
fi
# plik netrc istnieje
```

## Kontrola błędów

Pod koniec działania skryptu FTP powinien sprawdzić, czy nie wystąpiły błędy. Protokół FTP udostępnia całkiem spory zestaw kodów błędów. W Tabeli 1 pokazano często spotykane kody wyjścia programu, które można interpretować jako błędy lub ostrzeżenia.

Jednym ze sposobów sprawdzenia, czy nie wystąpiły błędy, jest wykonanie polecenia `egrep` na końcu skryptu, tak jak to zademonstrowano na Wydruku 9. Przede wszystkim dane zwracane przez program FTP musimy przekierować do pliku:

```
ftp -i -v >> $log 2>&1 <<mayday
```

FTP będzie zapisywał wszystkie dane wyjściowe, w tym informacje o błędach sesji FTP, w pliku dziennika określonym w zmiennej `$log`. Po zamknięciu sesji FTP po prostu przeszukujemy ten plik poleceniem `egrep` pod kątem wystąpienia odpowiednich

kodów lub słów; poszczególne wzorce rozdzielamy znakiem `|`. Zawsze, gdy w skrypcie używa się polecenia `egrep`, dobrze jest przekierować dane wyjściowe z dopasowania wzorców do `/dev/null`; dzięki temu na standardowym wyjściu nie pojawiają się zbędne komunikaty:

```
if egrep \\  
"202|421|426" \\  
$log > /dev/null 2>&1
```

Jeśli dla dowolnego wyrażenia `egrep` zwróci wartość „prawda”, skrypt kończy działanie zwracając kod 1, a odpowiedni komunikat przekazywany jest na wyjście standardowe i do pliku dziennika; w przeciwnym razie kończymy działanie zwracając kod 0. Jeśli skrypty są uruchamiane wsadowo, a nie interaktywnie z wiersza poleceń, żadne dane nie powinny być wysyłane na wyjście standardowe – wszystkie komunikaty należy kierować do pliku dzien-

nika. Fragment odpowiedzialny za zapisywanie kodu wyjścia w przypadku błędu uzupełniamy wtedy o linijkę:

```
echo "Błędy" >> $log
```

Podobnie można zmodyfikować kod odpowiedzialny za bezbłędne zakończenie programu:

```
echo "OK" >>$log
```

## Podsumowanie

Transfer plików między hostami można zautomatyzować i w ten sposób ułatwić sobie pracę. Dla różnych wymagań można przygotować kilka różnych skryptów FTP. Zawsze jednak dane wymagane na potrzeby logowania należy umieszczać w pliku `.netrc`, a nie w samym skrypcie. ■

### Wydruk 7: Wynik działania skryptu ftp5

```
$ ftp5  
1 : serwer [ ftp.emea.ibm.com ],  
użytkownik [ anonymous ]  
2 : serwer [ uk011x6001 ],  
użytkownik [ dxtans ]  
3 : serwer [uk041x6003 ],  
użytkownik [ dxtans]
```

```
Wybierz rekord [ 1 .. 3 ] :2  
wybrano:  
host [uk011x6001]  
użytkownik [dxtans]  
hasło [10opy]
```

### Tabela 1: Kody zwracane przez program FTP

202	Nieobsługiwane polecenie
421	Usługa niedostępna
426	Przerwany transfer
450	Plik niedostępny
500	Błąd składniowy
501	Błąd składniowy w liście argumentów
503	Użytkownik niezalogowany
550	Plik niedostępny
553	Nieprawidłowa nazwa pliku
666	Plik lub katalog nie istnieje
777	Nieznany host
999	Nieprawidłowe polecenie

### Wydruk 9: ftp7

```
#!/bin/bash  
# ftp7  
log=ftp.log  
>$log  
  
list="hosts telnet.conf"  
host="uk011x6001"  
  
echo "Nazwa skryptu [ `basename $0` ]" >>$log  
for files in $list  
do  
ftp -i -v >> $log 2>&1 <<mayday  
open $host  
ascii  
lcd /tmp  
cd /etc  
get $files  
quit  
mayday  
done  
if egrep "202|421|426|450|500|501|503|550|553|666|777|999" \  
$log > /dev/null 2>&1  
then  
echo "Błędy" | tee -a $log  
exit 1  
else  
echo "OK" | tee -a $log  
exit 0  
fi
```