Numbex

*System wymiany danych o numeracji w architekturze peer-to-peer*

# Opis pakietu

Numbex to system wymiany danych o numeracji działający w oparciu o architekturę P2P. Założeniami projektu była możliwie duża dostępność zgromadzonych danych bez względu na działanie zewnętrznych serwerów (np. ENUM) lub dostępu do internetu, a także odporność na awarie pojedynczych węzłów systemu.

# Wymagania

* POSIX-owy system operacyjny (testowane tylko na Linuksie)
* Python 2.5
* SQLite3
* Python Web Services 2.1 alpha1

(http://pywebsvcs.sourceforge.net/)

* Python M2Crypto 0.18.2

(http://chandlerproject.org/bin/view/Projects/MeTooCrypto)

# Instalacja zależności

Na systemie Debian Lenny (ostatnia wersja stabilna Debiana) wymagane jest zainstalowanie pakietów:

* python-zsi
* python-m2crypto
* git-core

Można tego dokonać wykonując jako root polecenie

aptitude install python-zsi python-m2crypto git-core

W celu zainstalowania tych bibliotek na innych dystrybucjach lub systemach operacyjnych należy skonsultowac się z ich dokumentacją. Można również zainstalować powyższe biblioteki ze źródeł.

# Instalacja numbex

W pakiecie znajdują się katalogi src/ i docs/. W pierwszym jest znaleźć można pliki źródłowe (programy i biblioteki Pythona), z których składa się program, a także testy jednostkowe (w katalogu src/tests/).

Dostępne programy:

* tracker.py - tracker do służący do koordynacji wymiany danych w rozproszonej infrastrukturze.
* numbex-daemon.py - program podłączający się do trackera i wymieniający dane o innych demonach funkcjonujących w internecie lub sieciach prywatnych. Odpowiada również za wymianę danych o numeracji w architekturze p2p.
* numbexctl.py - program umożliwiający kontrolowanie i wpływanie na działanie numbex-daemon.py.
* numbex\_client.py - klient SOAP służący do aktualizacji wpisów w lokalnej bazie numerów.
* numbex\_server.py - serwer SOAP realizujący powyższe funkcje.
* numbex\_udp\_server.py - serwer UDP spełniający jedną, bardzo prostę rolę - odpowiadać możliwie szybko na zapytania "czy mogę połączyć się z danym numerem poprzez SIP i jeśli tak, to przez jakie SIP proxy".
* numbex\_udp\_client.py - prosty, przykładowy klient realizujący takie zapytanie.
* testrunner.py - uruchamia testy jednostkowe.

# Konfiguracja i uruchamianie

Przed uruchomieniem systemu należy go skonfigurować. Przykładowy plik konfiguracyjny znajduje się w etc/numbex\_default.conf. Warto zwrócić uwagę w szczególności na pole "prefix" w sekcji [DEFAULT] - oznacza katalog, który w standardowej konfiguracji jest miejscem przechowywania gitowego repozytorium numerów i pliku sqlite3 (var/repo i var/db/db.sqlite3).

Aby nasz węzeł dobrze współpracował ze swoimi peerami, należy także skonfigurować adres trackera i adres rozgłoszeniowy, pod którym peery będą mogły znaleźć udostępnione repozytorium. Są to odpowiednio pola "trackers" w sekcji [PEER] oraz "repo\_url" w sekcji [GIT]. Trackerów można podać więcej niż jeden, oddzielanych białymi znakami.

# Uruchamianie

Wszystkie programy powinny być albo uruchamiane z katalogu src/, albo mieć ustawioną zmienną środowiskową PYTHONPATH tak, aby wskazywała na ten katalog. (Dotyczy to także uruchamianych pojedynczo testów jednostkowych z katalogu tests/.)

## Uruchamianie demona p2p::

numbex-daemon.py -C path/to/config/file

Informacje pojawiające się po uruchomieniu powinny być zbliżone do przkładu poniżej:

## Uruchamianie Kontrolera Demona:

#### Uruchamianie

numbexctl.py – klient XML-RPC

Informacje pojawiające się po uruchomieniu powinny być zbliżone do przkładu poniżej:

#### Eksport danych z bazy do p2p

numbexctl.py localhost p2p-export

Dobrym pomysłem jest wyłączenie systemu p2p i updatera przed dokonaniem tej operacji:

numbexctl.py localhost p2p-stop

numbexctl.py localhost updater-stop

Należy zwrócić uwagę, iż eksport nie jest szczególnie szybką operacją. Średnio obsługiwane jest 1000 rec/min.

#### Import danych do bazy z p2p

numbexctl.py localhost p2p-import

Import uruchamiany jest automatycznie, o ile działa updater (updater-start). W przypadku wystąpienia jakiegokolwiek błędu importu następuje natychmiastowe zatrzymanie updatera i ustawienie flagi błędu.

## Uruchamianie trackera::

tracker.py [-t <timeout>] [-p <port>] [-b <host do przybindowania>]

## Uruchamianie serwera UDP::

numbex\_udp\_server.py [-p <port>] [-b <host do przybindowania>] <plik sqlite3>

## Uruchamianie serwera SOAP::

numbex\_server.py [-p <port>] [-d <plik sqlite3>]

# Aktualizowanie danych w lokalnej bazie

Aktualizowanie danych jest realizowane przez program numbex\_client.py. Obsługuje on kilka operacji:

* pullall - pobieranie całej bazy
* pull <data> - pobieranie rekordów zmienionych od danej daty
* \* pullsign - pobranie rekordów nie zawierających podpisu. Przydatne po wysłaniu aktualizacji, jeśli w wyniku automatycznego wydzielania przedziałów powstały nowe wpisy.
* send <plik csv> - wysyłanie rekordów \_z podpisami\_.
* sendsign <plik csv> <klucz prywatny .pem> - wysyłanie rekordów \_bez podpisów\_. Jeśli rekordy miały podpis cyfrowy, jest on ignorowany, a zamiast niego wstawiany jest nowy.
* addpubkey <właściciel> <klucz publiczny .pem> - przypisuje klucz publiczny do właściciela. Tworzy właściciela, jeśli nie istniał.
* rmpubkey <id klucza> - usuwa klucz o podanym id
* listpubkeys <właściciel> - wyświetla klucze publiczne należące do danego właściciela.

Od pliku csv bez podpisów oczekuje się następującego formatu:

początek zakresu,koniec zakresu,adres sip,właściciel,data modyfikacji

Z podpisami:

początek zakresu,koniec zakresu,adres sip,właściciel,data modyfikacji,podpis

Podpis ma postać dwóch ciągów bajtów zakodowanych w base64, oddzielonych

spacją.

numbex\_client.py obsługuje ponadto opcję -o <plik wyjściowy> (jeśli nie jest podana, program wypisuje pozyskane dane na standardowe wyjście), a także opcję -u <url serwera soap> (defaultowo localhost:8000).