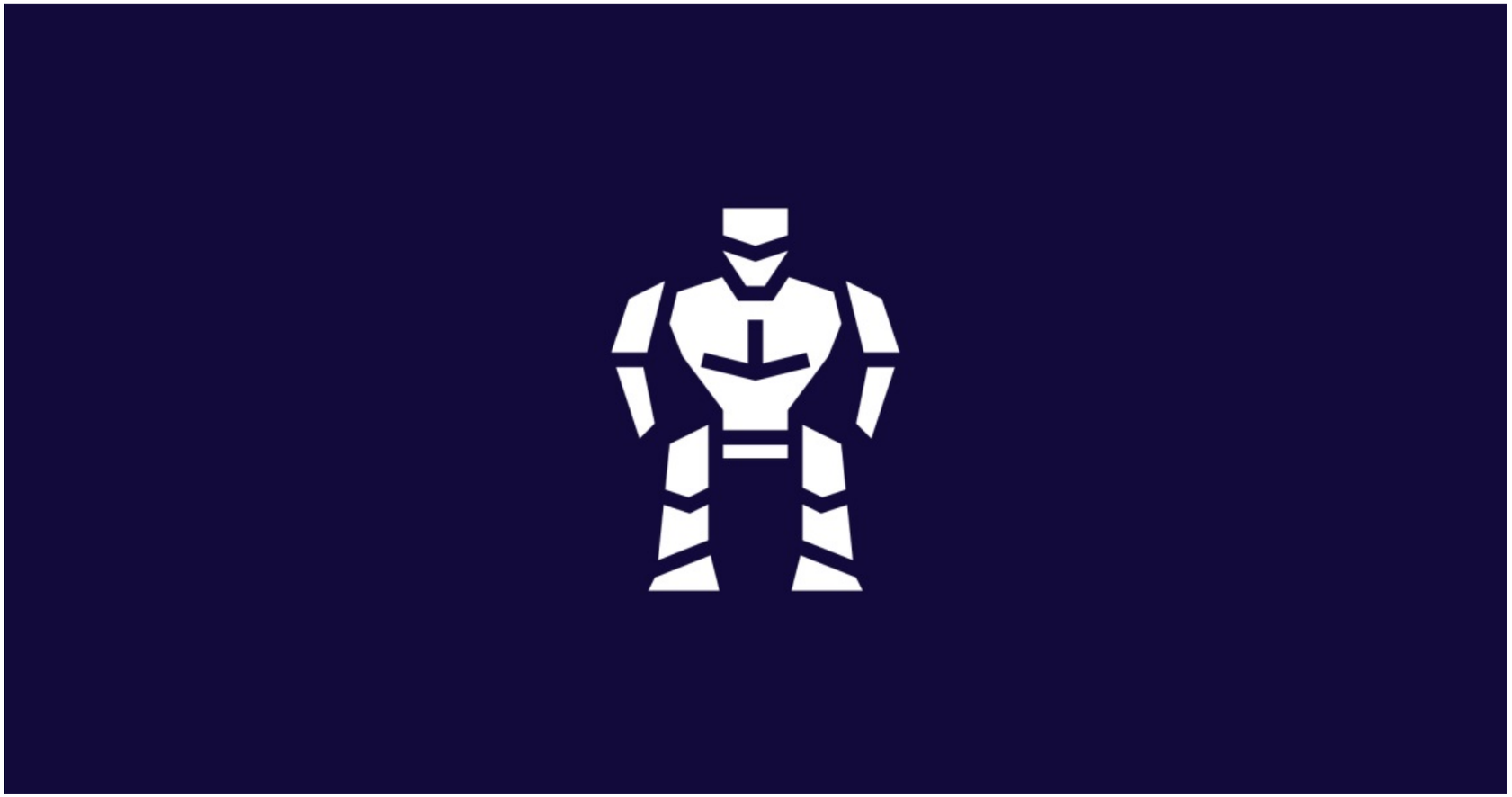

Jak oswoić Bota? – Requester (sheadovas/poradniki/howto/jak-oswoic-bota-requester/)

Lut 03, 2018 / howTo (sheadovas/category/poradniki/howto/)

Powracamy do oswajania botów.

Hej, w tym wpisie chciałbym Wam zaproponować drugie rozwiązanie dotyczące pisania botów – wymagające odrobinę większej finezji przy projektowaniu niż przy poprzednim podejściu (jeżeli nie czytaliście mojego poprzedniego wpisu o botach to zapraszam [tutaj (sheadovas/poradniki/howto/jak-oswoic-bota-clicker/)]).

Jednocześnie nie ukrywam, że pewną inspiracją do napisania tego bota są liczne pytania na *forum pasji-informatyki* dotyczące pewnej bazy danych pytań z pewnej strony w celach szkoleniowych (sytuacja jest nieco podobna do tej z naszej „fabuły”, patrz rozdział: *Praktyka*).

Disclaimer niniejszy wpis służy jedynie w celach edukacyjnych i autor nie ponosi odpowiedzialności za użycie zdobytej wiedzy w „niecnych” (ani żadnych innych) celach. Zdobytą tutaj wiedzę należy używać jedynie na aplikacjach, które są naszą własnością (lub mamy stosowne uprawnienia).

Trochę teorii

Zanim przejdziemy do praktyki to omówmy w dwóch zdaniach czym się charakteryzuje ten typ bota:

Otóż komunikuje się on ze stroną / aplikacją, już nie za pomocą emulacji tego co robi użytkownik (tak jak robił *Clicker*), lecz schodzi „warstwę niżej” i używa do tego zapytań webowych *(ang. requests* – stąd nazwa bota).

Aby móc używać requestów (tak aby rozumiała to aplikacja), to należy najpierw dowiedzieć się jak to robi aplikacja. Tutaj można albo przeczytać dokumentację, albo zapytać autora (jeżeli nim nie jesteśmy), albo jeżeli autor nie pracuje już w naszej pracy, nie zrobił dokumentacji, a źródła aplikacji są długie i zawiłe – to możemy wykonać samodzielną analizę.

Praktyka

Wstęp fabularny (różniący się nieco od inspirowanej aplikacji):

Jesteśmy uczniami jakiejś bliżej nieokreślonej szkoły, uczymy się do egzaminu do którego pytania (z zeszłych lat) udostępnia sama szkoła. Niefortunnie udostępnia je jedynie w formie formularza losującego jedno pytanie.

Co prawda można klikać dużo razy w przycisk „losuj”, lecz wydaje się to mało praktyczne (bo baza jest duża), szczególnie że chcemy mieć możliwość wydrukowania całego zestawu pytań w celu wielokrotnych powtórek na odpowiedzi do tych samych pytań.

Skoro pytania w bazie i tak są dostępne publicznie to postanawiamy napisać bota, który zrzuci wszystkie pytania za nas...

(https://i0.wp.com/szymonsiarkiewicz.pl/wp-content/uploads/2018/02/app-q.png)
Przykładowe pytanie z aplikacji

Rekonesans

UWAGA Ta faza jest do pominięcia w przypadku gdy już wiemy jak działa aplikacja webowa (bo np. mamy dostęp do jej dokumentacji / jesteśmy jej autorem). W naszym przypadku zakładamy, że musimy się sami tego dowiedzieć i właśnie ten proces pokazuję w tym paragrafie.

Szybki rekonesans pozwala nam stwierdzić, że:

- znamy dokładną ilość pytań w bazie (twórcy aplikacji chwalą się tym na stronie głównej quizu),
- pytania są stałego formatu:

```
Format pytania
```

```
Wylosowałeś pytanie z bazy nr: [nr pytania]
[Treść]
[Odp A]
[Odp B]
[Odp C]
[Odp D]
```

- (prawdopodobnie) znamy ID pytania w bazie (wynika to z treści widocznej przy pytaniu).

Aby dowiedzieć się czegoś więcej, to należy gdzieś się „zaczaić” i podejrzeć co się dzieje gdy:

1. losujemy pytanie,
2. wybierzemy odpowiedź.

Losowanie pytania

W naszym przypadku w zupełności wystarczają narzędzia dostarczone przez same przeglądarki, a więc „Narzędzia deweloperskie”, które dostarczają informacji o wysyłanych zapytaniach, w niektórych sytuacjach lepszym / wygodniejszym rozwiązaniem może się okazać lokalne proxy.

(https://i0.wp.com/szymonsiarkiewicz.pl/wp-content/uploads/2018/02/1.png)
Wysłane zapytanie przez przeglądarkę w celu uzyskania mojego bloga

Zobaczmy co się dzieje na kliknięcie przycisku „LOSUJ” w testowanej przez nas aplikacji:

```
Pobieranie pytania
```

```
POST http://target-school.com/src/loadquestion.php
Host: target-school.com
User-Agent: Definitely not a browser :)
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://target-school.com/random
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 70
Cookie: PHPSESSID=[PHP SESSION ID]
Connection: keep-alive

token=[some very long hex token]
```

Na podstawie powyższego requestu jesteśmy w stanie stwierdzić parę rzeczy:

- skrypt losujący znajduje się w *src/loadquestion.php* oraz posiada jeden parametr wejściowy,
- komunikacja odbywa się przez POST'y,
- aplikacja posiada dwie warstwy „zabezpieczeń": PHPSESSID oraz token.

Uruchomienie powyższego requestu parę razy (przepuszczając je dalej) upewnia nas, że token i id sesji PHP są stałe dla jednej sesji, co nas cieszy bo nie musimy się martwić o pobieranie nowego tokenu co nowe losowanie pytania.

Ponowne uruchomienie powyższego żądania, ale bez tokenu pokazuje że autor skryptu chciał się zabezpieczyć przed czymś (na pewno nie przed botem), bo token jest wymagany.

Pobieranie odpowiedzi

Przejdźmy do pobrania odpowiedzi:

Pobieranie odpowiedzi

```
POST http://target-school.com/src/loadanswer.php
Host: target-school.com
User-Agent: Definitely not a browser :)
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://target-school.com/random
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 83
Cookie: PHPSESSID=[PHP SESSION ID]
Connection: keep-alive

token=[some very long hex token]&idp=76&odp=1
```

Widzimy, że:

- skrypt do pobierania odpowiedzi znajduje się w *src/loadanswer.php,*
- wysyłamy do niego swój token sesji PHP,
- przyjmuje 3 argumenty:
  - *token* – tak samo jak wyżej, ważna uwaga: jest on tożsamy z tym który posiadamy także przy pobieraniu pytania (podobnie jak PHPSESSID),
  - *idp* – zgadujemy, że to ID pytania (co sugeruje ten sam numer co występuje w zdaniu: „Wylosowałeś pytanie z bazy nr: [nr pytania]"),
  - *odp* – są to wartości od 1-4 odpowiadające literom odpowiednio A-D.

Sam szkielet odpowiedzi nie różni się zbytnio od tego z ramki zawierającej format pytania, otóż posiada ona gotowy kod HTML, który jest wstrzykiwany w zawartość strony. Jedyną nowością jest string informujący o sukcesie:

Format pytania po otrzymaniu odpowiedzi

```
Twoja odpowiedź na pytanie [nr pytania] jest: ["poprawna" / "niepoprawna, prawidłowa to [litera odpowiedzi]"]:
[Treść]
[Odp A]
[Odp B]
[Odp C]
[Odp D]
```

Wnioski

Podsumujmy powyższe informacje, wiemy że:

1. Skrypty *loadquestion.php* i *loadanswer.php* służą do ładowania pytania i odpowiedzi.
    1. Całkiem możliwe, że do enumeracji całej bazy pytań wystarczy nam jedynie drugi ze skryptów (bo po co je losować, skoro można je pobrać po kolei?).
2. Występuje dość naiwne zabezpieczenie przed botami / osobami (w postaci *tokenu*), które manualnie wchodzą bezpośrednie na powyższe skrypty.
3. Tokeny żyją długo, więc nie musimy ich pobierać co uruchomienie skryptu, ułatwia to całkiem sporo,
4. Komunikacja (ważna dla nas) odbywa się tylko i wyłącznie przy pomocy requestu POST.

Proof of Concept

Zebraliśmy kilka wniosków, czas na weryfikację naszego podejrzenia z punktu 1.1.

Do tego z powodzeniem może nam posłużyć program *curl* wywołany następująco:

Shell
```
curl 'http://target-school.com/src/loadanswer.php' \
  -H 'Host:target-school.com' \
  -H 'User-Agent: [Not a browser]' \
  -H 'Accept: */*' \
  -H 'Accept-Language: en-US,en;q=0.5' --compressed \
  -H 'Referer: http://target-school.com/random' \
  -H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8' \
  -H 'X-Requested-With: XMLHttpRequest' \
  -H 'Cookie: PHPSESSID=[TWÓJ PHPSESSID]' \
  -H 'Connection: keep-alive' \
  --data "token=${TOKEN}&idp=${IDP}&odp=${ODP}"
```

Jako zmienną *idp* proponuję podstawić dowolną wartość różną od ostatnio wylosowanego pytania (np jeżeli ostatnio wylosowaliśmy pytanie nr 3, to spróbujmy pobrać odpowiedź na pytanie nr 7).

Niestety, napotykamy tutaj mały problem:

Odpowiedź z serwera
```
Ta strona jest prawdopodobnie otwarta na wielu kartach twojej przeglądarki.
Zamknij pozostałe i odśwież obecną.
```

Wniosek nasuwa się sam: autorzy porównują id ostatnio wylosowanego pytania z tym co my podajemy jako argument. Jeżeli są różne, to wyskakuje powyższy błąd.

Ta ochrona ma zapobiegać uruchamiania strony na wielu kartach z jaśniej niezrozumiałych powodów.

Powyższy kod co prawda pokazuje, że będzie trudniej, ale nas jeszcze nie eliminuje bo wciąż możemy wyciągnąć bazę chociaż nieco większym nakładem sił.

Oswajamy bota

Do tej pory przedstawiłem jak można poznać sposób działania aplikacji, zobaczyliśmy też że nie nie pobierzemy wszystkich pytań po kolei, musimy dostosować się do tego jak działają powyższe skrypty.

Algorytm

Ogólny algorytm może wyglądać następująco:

1. (opcjonalnie) Wejdź na stronę w celu pobrania PHPSESSID i tokena (można też zrobić to ręcznie, po czym wkleić te wartości).
2. Niech **M** – hash-mapa postaci, **M**[**idp**] – informacje o pytaniu (treść, odpowiedzi, poprawna odpowiedź) o wskazanym ID pytania.
3. Niech **|Q|** – ilość wszystkich pytań w bazie, **|M|** – ilość pobranych pytań do mapy
4. Uruchom skrypt *loadquestion*:
    1. Pobierz ID pobranego pytania, jeżeli istnieje już w mapie to przejdź do kroku (3),
    2. w p.p. stwórz nowy wpis do **M** i dodaj do niego: treść pytania, odpowiedzi.
5. Uruchom skrypt *loadanswer* podając ID pytania oraz odpowiedź ‚1' jako poprawną:
    1. Jeżeli odpowiedź zawiera ciąg znaków „jest: poprawna", to oznacz w **M**[**idp**] odpowiedź ‚A' jako poprawną,
    2. w p.p. pobierz ciąg znaków po stringu „poprawna to " i oznacz ją jako poprawną.
6. Jeżeli **|M|** < **|Q|**, to wróć do kroku (2).
7. Wyświetl pobrane pytania.

Całość może wydawać się skomplikowana, jednakże bot jest prosty – ba! nawet dużo prostszy od *Clikera*, wymaga jedynie nieco większej wiedzy o działaniu aplikacji.

Implementacja

Poniższy kod implementuje nasz algorytm przy użyciu biblioteki *requests* (do wysyłania zapytań) oraz *re* (do wyrażeń regularnych). Zachęcam do samodzielnej analizy całości, zaraz omówię krótko wszystkie części.

dump-db.py — Python

```python
import requests
import re

# | REPLACE ME |
# v            v
token = "RkxBR3tOb09uZUlzV2F0Y2hpbmd9Cg=="
phpsessid = "SW4gY2FzZSBvZiBlbWVyZ2VuY3kgY2FsbDogMDExOCA5OTkgODgxIDk5OSAxMTkgNzI1MyEK"
QUESTIONS_COUNT = 1000
# ^            ^
# | REPLACE ME |

questions = {}
data = {'token' : token}
headers = {'Host' : 'target-school.com',
           'User-Agent' : 'WaterCat 1.2.3.4',
           'X-Requested-With' : 'XMLHttpRequest',
           'Referer' : 'http://target-school.com/random',
           'Cookie' : 'PHPSESSID={}'.format(PHPSESSID)}


def get_unique_question(max_tries=10):
    """Get unique question & add entry to global map,
    returns id of unique question.

    max_tries       - max amount of retries for get unique random question number"""

    idp = None
    while max_tries > 0 and idp is None:
        r = requests.post('http://target-school.com/src/loadquestion.php', data=data, headers=headers)

        regex = r"Wylosowałeś pytanie z bazy nr: (\d*)\n(.*)\n(.*)\n(.*)\n(.*)\n(.*)"
        match = re.match(regex, r.text)
        idp, q, anw_a, ans_b, ans_c, ans_d = match.groups()
```

Parametrów wejściowe

```python
# | REPLACE ME |
# v            v
token = "RkxBR3tOb09uZUlzV2F0Y2hpbmd9Cg=="
phpsessid = "SW4gY2FzZSBvZiBlbWVyZ2VuY3kgY2FsbDogMDExOCA5OTkgODgxIDk5OSAxMTkgNzI1MyEK"
QUESTIONS_COUNT = 1000
# ^            ^
# | REPLACE ME |

questions = {}
data = {'token' : token}
headers = {'Host' : 'target-school.com',
           'User-Agent' : 'WaterCat 1.2.3.4',
           'X-Requested-With' : 'XMLHttpRequest',
           'Referer' : 'http://target-school.com/random',
           'Cookie' : 'PHPSESSID={}'.format(PHPSESSID)}
```

Podświetlone linie zawierają dane które przekazujemy na wejściu do zapytania, tzn. pola które musimy uzupełnić aby wyglądało jak zapytanie od aplikacji webowej, a nie bota. Chyba nie muszę mówić, że pola wewnątrz *REPLACE ME* są do zmiany oraz przydałoby się podmienić nagłówek

`User-Agent`

na coś bardziej „legalnego” niż zmyślona przeglądarka? ;)

Pobieranie unikalnego pytania

```python
def get_unique_question(max_tries=10):
    """Get unique question & add entry to global map,
    returns id of unique question.

    max_tries       - max amount of retries for get unique random question number"""

    idp = None
    while max_tries > 0 and idp is None:
        r = requests.post('http://target-school.com/src/loadquestion.php', data=data, headers=headers)

        regex = r"Wylosowałeś pytanie z bazy nr: (\d*)\n(.*)\n(.*)\n(.*)\n(.*)\n(.*)"
        match = re.match(regex, r.text)
        idp, q, anw_a, ans_b, ans_c, ans_d = match.groups()

        if idp in questions:
            questions[idp] = "({}) Question: {}\n{}\n{}\n{}\n{}\n".format(idp, q, ans_a, ans_b, ans_c, ans_d)
        else:
            max_tries -= 1

    if idp is None:
        print "Max retries reached! Cancelling! :("
        return None

    return idp
```

Następna jest funkcja, której zadaniem jest pobranie unikalnego pytania, tzn takiego którego jeszcze nie ma w naszej mapie / słowniku. Z racji, że zdajemy się na losowość, to została dostarczona razem z dodatkowym parametrem

`max_tries`

który służy do przerwania działania bota w razie gdyby szczęście przestało nam sprzyjać i nie moglibyśmy trafić na pytanie, któ©ego jeszcze nie mamy.

Podświetlone linie pokazują odpowiednio: wysłanie zapytania do serwera w celu uzyskania pytania, wyrażenie regularne do odnalezienia pytania i wyciągnięcia z niego odpowiednich danych oraz zapisanie go do słownika.

Pobieranie odpowiedzi

```python
def get_answer(idp):
    """Get answer for question based on question id

    idp             - question idp"""

    # prepare data
    d = data
    d['idp'] = idp
    d['odp'] = 1 # A

    # request answer
    r = requests.post("http://target-school.com/src/loadanswer.php", data=d, headers=headers)

    if " poprawna" in r.text:
        questions[idp] += "Correct: A\n"

    else:
        regex = r".*prawidłowa to (\W)"
        match = re.match(regex, r.text)

        if match is None:
            print "Error, cannot find correct answer for {} in string: {}\n".format(idp, r.text)
        else:
            questions[idp] += "Correct: {}\n".format(match.group(0))
```

W tej funkcji pobieramy odpowiedź do pytania. Warto zwrócić uwagę, że domyślnie wysyłamy odpowiedź ‚A' jako wybraną, następnie na podstawie odpowiedzi decydujemy jaka jest poprawna.

W przypadku gdyby wygasł token, to nasze wyrażenie regularne nie zadziała i po prostu zwróci błąd.

Główna pętla

```python
def main():
    while(len(data) < QUESTIONS_COUNT):
        idp = get_unique_question()
        if idp is None:
            break
        get_answer(idp)

    # dump data
    for key in questions:
        print questions[key]
```

Zadaniem głównej pętli jest wyciągnięcie wszystkich błędów lub w przypadku błędu zatrzymania bota. Na koniec drukuje na standardowe wyjście wszystkie znalezione pytania.

Podsumowanie

W tym wpisie zobaczyliśmy jak można napisać bota, który potrafi komunikować się z aplikacją przy użyciu zapytań. Niewątpliwie wymaga nieco większego nakładu pracy na etapie przygotowawczym, jednakże w wielu sytuacjach może okazać się znacznie wydajniejszy niż *Clicker*.

Jako zadanie domowe polecam udoskonalić powyższą implementację o działanie na wielu wątkach (zobaczycie że jest niewiele do modyfikacji) oraz automatyczne pobieranie tokenów.

Jeżeli macie jakieś tematy, które chcelibyście zobaczyć, to oczywiście piszcie śmiało w komentarzach, na koniec tradycyjnie zapraszam do śledzenia bloga przez social-media.

C0de On!