Instrukcja laboratoryjna – generowanie i analiza ruchu sieciowego

1. Wprowadzenie

Niniejsza instrukcja dotyczy ćwiczenia generowania i analizy ruchu sieciowego zorganizowanego w szereg zadań. Każde z nich składa się z demonstracji funkcjonalności określonego narzędzia i następującego po niej punktowanego zadania.
Przed przystąpieniem do wykonywania zadań należy skonfigurować maszynę wirtualną tak, aby posiadała dwa interfejsy sieciowe. Pierwszy z interfejsów należy skonfigurować w trybie bridge z kartą Ethernet systemu hypervisora. Drugą – jako sieć Host-only. Następnie należy wyłączyć adres IP w systemie hypervisora na karcie sieciowej eth0 komendą:

```
nmcli c down Wired\ connection\ 1
ip l s eth0 up
```

W ten sposób karta będzie działała, ale bez adresu IP. Na interfejsie Host-only na hypervisorze należy skonfigurować statycznie adresy IP z puli 192.168.11.0/24 zarówno dla hypervisora, jak i maszyny wirtualnej. Dodatkowo – należy skonfigurować bramę domyślną w hypervisorze tak, aby wskazywała adres IP maszyny wirtualnej.

2. Opis narzędzi

W ćwiczeniu zaprezentowano zbiór rozwiązań pozwalających generować i analizować ruch sieciowy. Rozwiązania różnią się dostępnością i dostarczanymi funkcjonalnościami, co pozwala dobierać je zależnie od ograniczeń (np. terminala tekstowego) i potrzeb (np. inspekcji ruchu szyfrowanego).

2.1. IPtables

Jednym z najprostszych sposobów zweryfikowania występowania określonego ruchu IP w systemie Linux jest utworzenie reguły IPtables. Polecenie to daje dostęp do funkcjonalności netFilter kernela systemu Linux odpowiedzialnego za filtrowanie ruchu sieciowego. Pozwala ono w szczególności zliczać datagramy/segmenty, które przystawały do poszczególnych reguł. Przykładowe statystyki przedstawiono poniżej (liczba datagramów wytłuszczona):

```
# iptables -nvL
Chain INPUT (policy ACCEPT 830 packets, 38698 bytes)
 pkts bytes target     prot opt in     out     source               destination
  259 43035 ACCEPT     udp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           udp spt:53
```

Aby na bieżąco obserwować zmianę w liczbie pakietów można posłużyć się poleceniem watch – w przykładzie odświeżanie co 1 sekundę i wyróżnianie zmian względem poprzednich wartości:

```
# watch -d -n 1 iptables -nvL
```

Drugim sposobem monitorowania ruchu sieciowego w oparciu o IPtables jest możliwość logowania pakietów do dziennika zdarzeń (syslog). Zamiast celu ACCEPT należy podać LOG oraz ewentualnie prefiks, po którym łatwiej wyszukać informację w dzienniku zdarzeń. Przykładowa komenda (dla ruchu DNS) może wyglądać następująco:

```
iptables -A INPUT -i eth0 -p udp --sport 53 -j LOG --log-prefix "NF_IN "
```

a tak zawartość odpowiadającej regule linii zalogowanego ruchu w dzienniku /var/log/syslog:

```
NF_IN  IN=eth0  OUT=  MAC=08:00:27:b3:b4:96:00:15:5d:10:10:0b:08:00  SRC=10.1.0.1  DST=10.1.1.190  LEN=129
   TOS=0x00 PREC=0x00 TTL=64 ID=64915 PROTO=UDP SPT=53 DPT=60915 LEN=109
```

Inną funkcjonalnością wykorzystywaną podczas zajęć jest przekierowanie określonego ruchu do usługi działającej lokalnie na maszynie. Wykorzystywana jest funkcjonalność REDIRECT łańcucha PREROUTING w tablicy nat:

```
iptables -t nat -A PREROUTING -s 10.1.0.2 -d 8.8.8.8 -p udp --dport 53 -j REDIRECT --to-ports 10053
```

W przykładzie ruch ze stacji 10.1.0.2 wysyłany pod adres 8.8.8.8 na port 53 zostanie przekierowany na port 10053 na lokalnej maszynie.

2.2. tcpdump i filtry BPF

Program tcpdump stanowi podstawowe narzędzie analizowania ruchu sieciowego w konsoli tekstowej systemów Unix/Linux. Jakkolwiek konsola nakłada duże ograniczenia na ilość wyświetlanych informacji, program tcpdump pozostaje standardem w diagnozie problemów z funkcjonowaniem sieci i usług sieciowych. Ze względu na ilość informacji, które mieszczą się na ekranie podczas analizy istotne jest odfiltrowanie możliwie wielu niepotrzebnych informacji. Można to osiągnąć stosując filtry BPF (ang. Berkeley Packet Filter) służące do selekcji ramek, które zostaną wyświetlone. Uniwersalność tych filtrów, osiągnięta między innymi dzięki wydajności z jaką mogą pracować, powoduje że można je spotkać w większości narzędzi analizy ruchu sieciowego. Mimo ich uniwersalności oferowany jest przeciętnie duży zakres funkcjonalny – dużo bardziej elastyczne są filtry wyświetlania programów Wireshark/Tshark, ale pracują zdecydowanie wolniej.
Najprostszymi operatorami BPF są dyrektywy *host* i *net*:

```
tcpdump -ni eth0 host 10.1.0.1
tcpdump -ni eth0 net 192.168.1.0/24
```

Modyfikatorami mogą być operatory *src* i *dst*, oznaczające odpowiednio pole nadawcy lub odbiorcy w pakiecie:

```
tcpdump -ni eth0 src host 10.1.0.1
```

Nieco bardziej zaawansowaną formą filtrowania jest wybór określonego adresu MAC:

```
tcpdump -ni eth0 ether host 00:de:ad:be:ef:01
```

Można odfiltrować także na podstawie numerów portów tcp lub udp:

```
tcpdump -ni eth0 port 53
```

a także niektórych protokołów warstw niższych:

```
tcpdump -ni eth0 stp
tcpdump -ni eth0 ip6
tcpdump -ni eth0 arp
tcpdump -ni eth0 proto 47 ; ruch protokołu GRE - lista w /etc/protocols
```

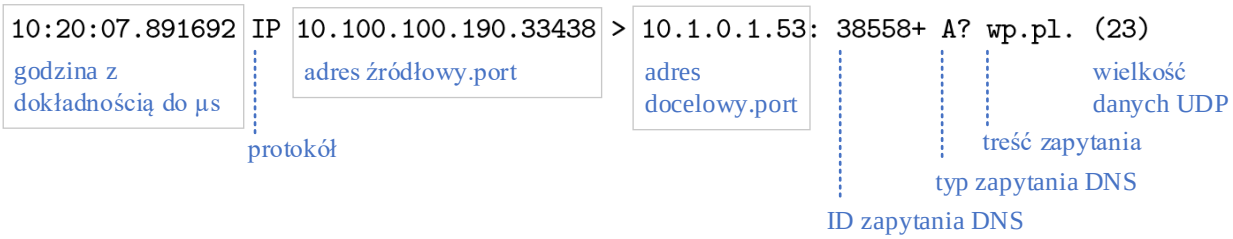
Filtry mogą być łączone operatorami *and*, *or* i *not*, a także nawiasami w dość naturalnej notacji. Jeżeli wyrażenie robi się bardziej skomplikowane, warto użyć pojedynczego cudzysłowu, żeby uniknąć interpretacji wyrażenia przez powłokę (bash). Pierwszy przykład analizuje ruch pochodzący od konkretnej stacji, ale nie IP ani ARP:

```
tcpdump -ni eth0 ether host 00:de:ad:be:ef:01 and not arp and not ip
```

Drugi przykład złożonego filtra to analiza ruchu do dwóch serwerów DNS, opcja -A powoduje wyświetlenie drukowalnych znaków ASCII w datagramach, natomiast -X – szesnastkowo:

```
tcpdump -ni eth0 -A 'port 53 and (host 153.19.48.1 or host 10.1.0.1)'
tcpdump -ni eth0 -X 'port 53 and (host 153.19.48.1 or host 10.1.0.1)'
```

Przydatnymi argumentami programu tcpdump są opcje -v (może być podawana kilkukrotnie) oraz -e, które odpowiadają odpowiednio za zwiększenie szczegółowości prezentowania informacji (verbose) i wyświetlanie nagłówka warstwy II (najczęściej Ethernet, stąd -e).
Pozostaje jeszcze omówienie przykładowego wyniku łapania ruchu:



Rys. 1. Przykładowy wynik łapania ramek programem tcpdump – zapytanie DNS

2.3. Wireshark

Program Wireshark także pozwala posłużyć się filtrami BPF podczas łapania ramek, oferuje jednak więcej możliwości pod względem ich późniejszej selekcji. Przeznaczone są do tego tzw. filtry wyświetlania. Filtry te mają inną składnię, ale oferują znacznie więcej możliwości. Podstawowe odpowiedniki filtrów BPF są następujące:

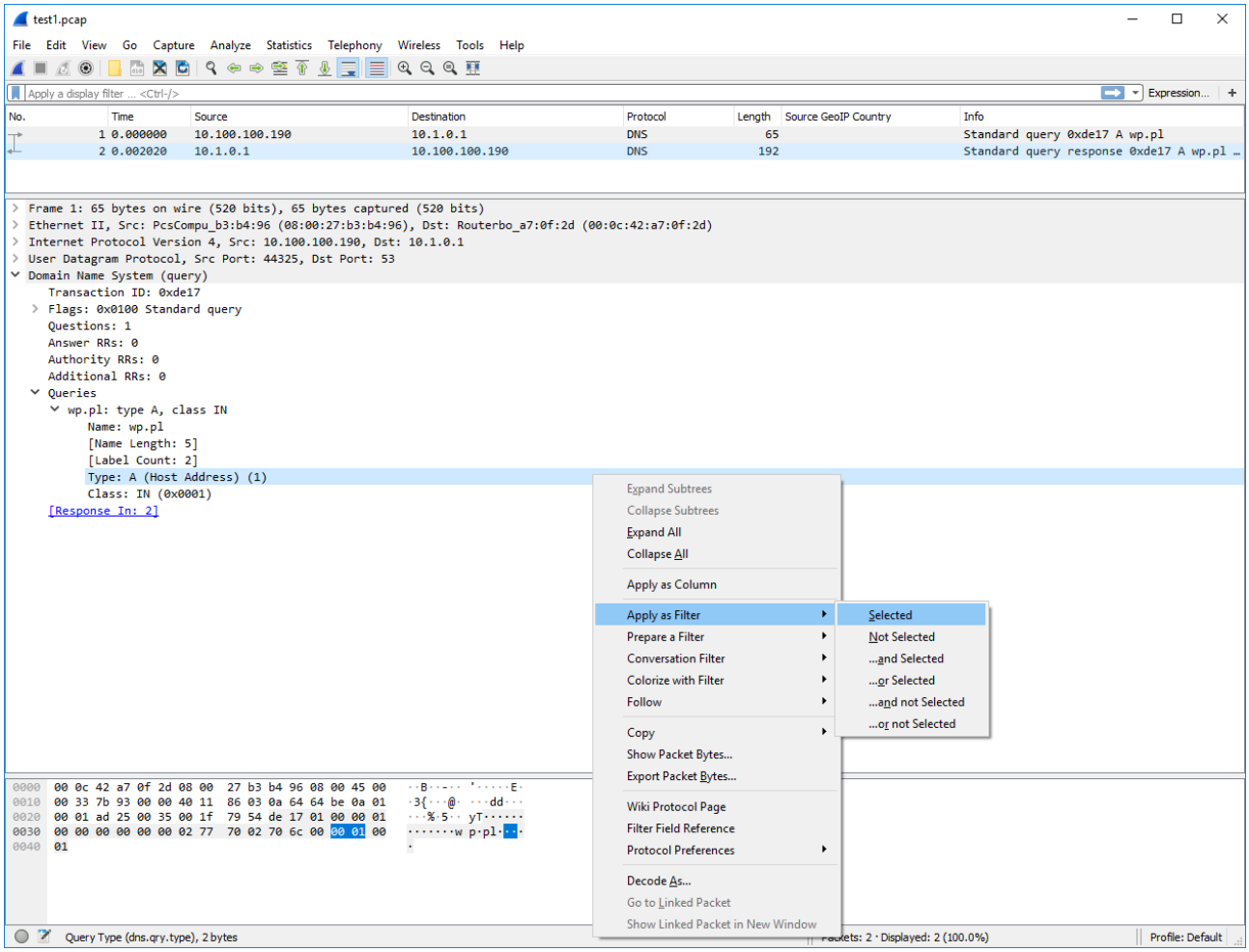
Filtr BPF	**Filtr wyświetlania Wireshark/Tshark**
host 10.1.0.1	ip.addr == 10.1.0.1
src host 10.1.0.1	ip.src == 10.1.0.1
port 53	udp.port == 53 || tcp.port == 53
ether host 00:de:ad:be:ef:01	eth.addr == 00:de:ad:be:ef:01

Filtry wyświetlania pozwalają także na stosowanie operatorów miękkiego przystawania, m. in. operatora *contains*. Zatem filtr:

```
eth.addr contains 00:0c:42
```

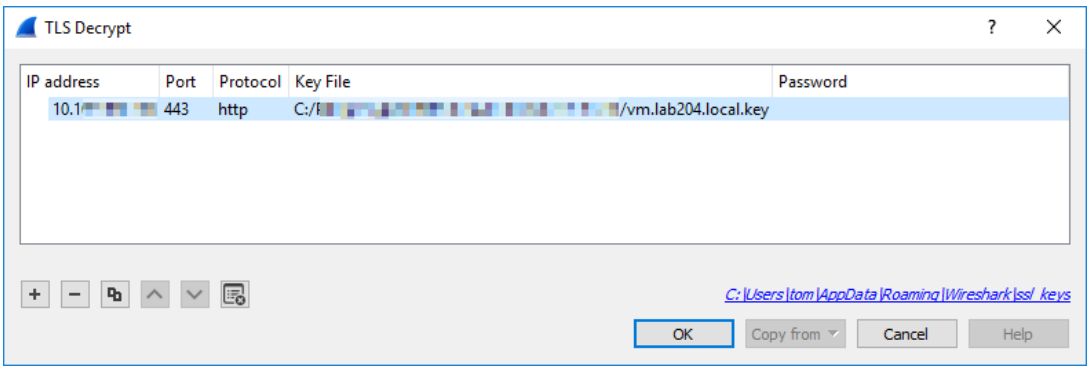
potencjalnie będzie odpowiadał za wszystkie adresy MAC danego producenta (pierwsze trzy bajty adresu MAC to pole OUI identyfikujące producenta interfejsu sieciowego, tutaj identyfikowanego podanym ciągiem szesnastkowym). W tym przypadku do filtra dostaną się także adresy MAC, które na innych niż pierwsze polach mają wskazane wartości.
Jeżeli natomiast wymagane jest użycie filtra, którego składnia nie jest znana, bardzo łatwo można ją ustalić posługując się interfejsem graficznym. Na Rys. 2 przedstawiono zrzut ekranu programu Wireshark, gdzie ponownie analizowany jest ruch DNS. Przykład ilustruje wybór typu zapytania DNS – rekord A (pytanie o adres IP dla podanej nazwy domenowej) jako filtra wyświetlania. Należy w tym celu w środkowym oknie dekodowania ramki (tutaj – segmentu UDP) rozwinąć interesujące gałęzie (Domain Name System query -> Queries -> wp.pl -> Type: A, a następnie kliknąć prawym przyciskiem myszy i z menu kontekstowego wybrać Apply as Filter -> Selected. W ten sposób przykład prowadzi do filtra wyświetlania:

```
dns.qry.type == 1
```



Rys. 2. Tworzenie nietypowego filtra wyświetlania dla typu A zapytania DNS w programie Wireshark

Inną atrakcyjną funkcjonalnością programu Wireshark jest możliwość odszyfrowania próbki ruchu z użyciem odpowiedniego klucza prywatnego. Klucz konfiguruje się wchodząc do ustawień programu (Edit -> Preferences) i wybierając Protocols -> TLS -> RSA keys ->Edit. Można tam wczytać klucze prywatne przeznaczone dla serwera działającego na zadanym porcie (tutaj 443) przenoszącym wskazany protokół (http) pod określonym adresem IP:



Rys. 3. Okno konfiguracji klucza do odszyfrowania ruchu sieciowego określonego serwera

Warto zauważyć, że program Wireshark pozostając pasywny podczas inspekcji ruchu TLS nie ma możliwości odszyfrowania danych, o ile został wykorzystany protokół Diffie-Hellman w dowolnej odmianie (np. z wykorzystaniem krzywych eliptycznych).

2.4. Moduł recent iptables

2.5. Narzędzie hping

Program hping (najczęściej w wersji 3 przyjmuje nazwę hping3) służy do generowania różnorodnego ruchu IP, najczęściej na potrzeby skanowania usług sieciowych.

2.6. Narzędzie IPset

Wydajność filtrowania ruchu IP przez klasyczne rozwiązania netFilter dostępne w jądrze systemu Linux nie dostarczają żadnej optymalizacji wydajnościowej przeszukiwanych reguł. Zakładając, że konfiguracja firewalla zakłada relatywnie dużą liczbę adresów IP lub podsieci, opracowano rozwiązanie IPset, które adresuje problem wydajności ich przeszukiwania. Funkcjonowanie komendy ipset jest podobne do iptables, z tym, że obiektami są zbiory (set) oraz zawarte w nich pozycje (adresy, podsieci). Argument -L powoduje wyświetlenie listy zbiorów oraz ich zawartości (dodając -n można wyświetlić tylko zbiory):

```
ipset -L
ipset -L -n
```

Dodanie nowego zbioru wymaga określenia, czy jego elementami będą pojedyncze adresy IP (wariant pierwszy komendy), czy podsieci zadane w postaci ADRES/DŁ_MASKI (wariant drugi):

```
ipset -N example1 hash:ip
ipset -N example2 hash:net
```

Narzędzie oferuje więcej opcji związanych z optymalizacją procesu przeszukiwania zbiorów, jednak na potrzeby zajęć wystarczą polecenia odpowiednio dodawania i usuwania elementów ze zbioru:

```
ipset -A example2 37.7.0.0/16
ipset -D example2 37.7.0.0/16
```

Usunięcie całego zbioru można osiągnąć komendą:

```
ipset -X example2
```

2.7. Program Tshark

Filtry wyświetlania z programu Wireshark znajdują także zastosowanie w programie tshark, również podczas pracy z ruchem sieciowym na żywo. Tshark domyślnie wyświetla informacje zbliżone do tcpdump, ale pozwala dołożyć pewne inne dodatkowe wyświetlane informacje. Przykładowa komenda przedstawiona poniżej łapie ruch wyłącznie na portach 53 i 5353, a następnie wybiera następujące pola datagramów do wyświetlenia:

- czas od rozpoczęcia łapania (-e frame.time_relative),
- adresy IP (-e ip.addr),
- protokół (-e col.Protocol),
- typ zapytania DNS (-e dns.qry.type) oraz
- wynik dekodowania (-e col.Info).

```
tshark -i ppp0 -f 'port 53 or port 5353' -T fields -E header=y -e frame.time_relative -e ip.addr
   -e col.Protocol -e dns.qry.type -e col.Info
```

A to przykładowy wynik jej działania:

```
frame.time_relative     ip.addr col.Protocol    dns.qry.type    col.Info
0.000000000     10.1.1.2,10.1.0.1    DNS     0x00000001      Standard query 0xf408  A wp.pl
0.006984172     10.1.0.1,10.1.1.2    DNS     0x00000001      Standard query response 0xf408  A 212.77.98.9
```

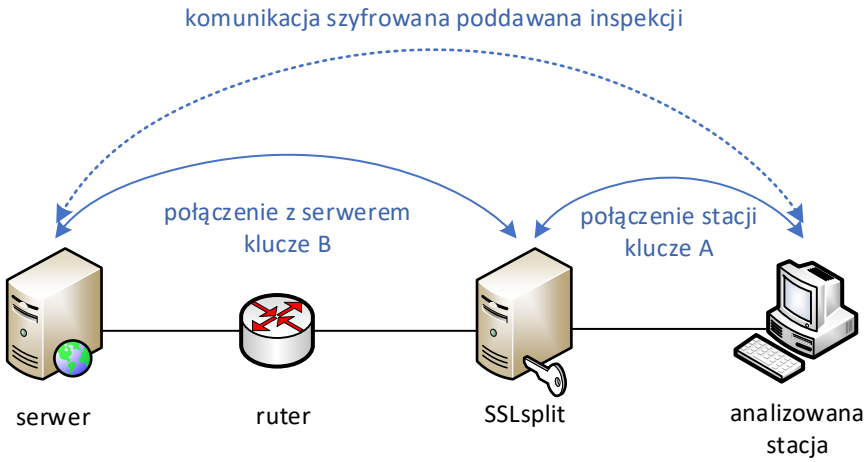
2.8. Narzędzie sslsplit

SSLsplit jest programem, który pozwala wykonać aktywną inspekcję ruchu SSL/TLS. Rozwiązanie to różni się od funkcjonalności odszyfrowywania próbki ruchu w programie Wireshark dwoma aspektami:

a) pozwala wykonać inspekcję ruchu zabezpieczonego także z użyciem algorytmu Diffie-Hellman,
b) jest wykonywane w modelu proxy – pośrednika w procesie komunikacji.

Ograniczeniem programu jest brak możliwości eksportu przechwytywanego ruchu do formatu .pcap. Zamiast tego uzyskiwany jest surowy ciąg danych ze strumienia poddawanego inspekcji.

Inspekcja SSL/TLS wykonywana przez program SSLsplit zakłada generowanie certyfikatu każdego serwera, z którym nawiązuje połączenie maszyna poddana analizie. Certyfikat ten domyślnie posiada skopiowane wszystkie pola oryginalnego certyfikatu serwera, ale jest podpisywany przez urząd certyfikacji konfigurowany w programie SSLsplit. Dlatego aby inspekcja nie uniemożliwiała funkcjonowania aplikacji należy certyfikat urzędu doinstalować na maszynie poddanej analizie. Poglądowo ilustruje to Rys. 4.



Rys. 4. Inspekcja ruchu szyfrowanego z użyciem narzędzia SSLsplit

Podczas inspekcji wykorzystywane są dwa różne zestawy kluczy – komplet A, za który odpowiedzialny jest program SSLsplit poprzez użycie określonego, przygotowanego wcześniej certyfikatu urzędu certyfikacji (CA), a także komplet B, za który odpowiada strona serwera. Najczęściej scenariusz ten, blisko spokrewniony z atakiem MitM na sesję SSL/TLS, ma zastosowanie w środowisku korporacyjnym, gdzie dodatkowo maszyna odszyfrowująca ruch sieciowy posiada mechanizmy selekcji, dla których adresów ją pominąć. W ten sposób pracodawca z jednej strony może poddawać analizie ruch szyfrowany, w szczególności potencjalnie przenoszący złośliwe oprogramowanie, z drugiej – zapewnić prywatność połączeniom, których ujawnienie mogłoby naruszać dobro pracowników. Między innymi do takiej klasy należą połączenia z portalami bankowości internetowej, portalami religijnymi, czy portalami związanymi z medycyną i farmacją, w szczególności apteki internetowe. Selekcja, które kategorie powinny pozostawać bez inspekcji leży w gestii ustawodawstwa danego kraju, natomiast utrzymywanie listy takich portali – w gestii dostawcy rozwiązania inspekcji ruchu szyfrowanego.

Wywołanie polecenia sslsplit wymaga podania następujących argumentów:

- `-D` – uruchomienie w trybie interaktywnym
- `-Z` – wyłączenie kompresji dla wszystkich połączeń
- `-O` – (wielka litera o) zablokowanie wykorzystania protokołu OCSP
- `-l conns.log` – logowanie połączeń do pliku conns.log

- `-S logdir` – logowanie danych z połączeń do osobnych plików w katalogu logdir; katalog musi istnieć wcześniej
- `-k ca.key` – użycie klucza prywatnego CA z pliku ca.key; format Base64
- `-c ca.pem` – użycie certyfikatu CA z pliku ca.pem; format Base64
- `http 0.0.0.0 10080` – otwarcie proxy dla ruchu HTTP na porcie 10080 na dowolnym lokalnym adresie IP
- `https 0.0.0.0 10443` – otwarcie proxy dla ruchu HTTPS na porcie 10443 na dowolnym lokalnym adresie IP

Całość komendy przyjmuje postać:

```
sslsplit -D -Z -O -l conns.log -S logdir -k ca.key -c ca.pem http 0.0.0.0 10080 https 0.0.0.0 10443
```

Następnie należy dodać regułę IPTables, która spowoduje przekierowanie ruchu, który ma zostać poddany inspekcji do programu sslsplit. W oryginalnej dokumentacji podane są reguły bez klasyfikatorów ruchu poza numerem portu. Konfiguracja ta jest błędna, ponieważ inspekcji poddanie zarówno ruch z zewnętrznych maszyn, jak i ponownie ten, który wychodzi z serwera proxy, na którym uruchomiony jest sslsplit. Przykład działającej reguły przedstawiono poniżej:

```
iptables -t nat -A PREROUTING -p tcp --dport 443 -s 10.1.1.2 -d 8.8.8.8 -j REDIRECT --to-ports 10443
```

Widoczne są klasyfikatory adresu źródłowego (-s 10.1.1.2) oraz docelowego (-d 8.8.8.8). Dzięki nim ruch oryginalny, pochodzący od stacji 10.1.1.2 będzie różnił się od ruchu wychodzącego z serwera proxy z programem sslsplit, który działa pod adresem np. 10.2.2.190.

Przykładowa prawidłowo poddana inspekcji sesja z portalem allegro.pl może wyglądać następująco:

```
Connecting to [185.31.25.105]:443
===> Original server certificate:
Subject DN: /C=PL/L=Poznan/O=Allegro.pl sp. z o.o./OU=IT/CN=ssl.allegro.pl
Common Names: ssl.allegro.pl/ssl.allegro.pl/www.ssl.allegro.pl
Fingerprint: 9C:00:3C:16:9B:E3:4A:41:B5:4225:88:DC:55:E6:F0:D3:82:A1:11
Certificate cache: MISS
===> Forged server certificate:
Subject DN: /C=PL/L=Poznan/O=Allegro.pl sp. z o.o./OU=IT/CN=ssl.allegro.pl
Common Names: ssl.allegro.pl/ssl.allegro.pl/www.ssl.allegro.pl
Fingerprint: D3:3C:5B:14:9A:B7:7A:2C:DE:582F:3C:96:94:7F:80:AB:E9:DD:22
SSL connected to [185.31.25.105]:443 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256
Received privsep req type 01 sz 85 on srvsock 10
SSL session cache: MISS
SSL connected from [172.16.1.11]:62982 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256
Error from bufferevent: 0:- 336151576:1048:tlsv1 alert unknown ca:20:SSL routines:148:ssl3_read_bytes
SSL disconnected to [185.31.25.105]:443
SSL disconnected from [172.16.1.11]:62982
SSL_free() in state 00000003 = 0003 = SSLOK  (SSL negotiation finished successfully) [accept socket]
SSL_free() in state 00000003 = 0003 = SSLOK  (SSL negotiation finished successfully) [connect socket]
SNI peek: [n/a] [complete]
Attempt reuse dst SSL session
```

Widoczne jest wykorzystanie protokołu Diffie-Hellman (ciphersuite: ECDHE-RSA-AES128-GCM-SHA256), którego nie można poddać inspekcji metodami pasywnymi, np. z użyciem programu Wireshark.