

Al-Saaiydeh Sandra, Michalska Aleksandra,
Pelczar Jacek, Sitarczyk Aleksandra

Bezpieczeństwo Komunikacji

Politechnika Warszawska
Semestr 23Z

Sprawozdanie z projektu 2
Bezpieczna architektura sieci

Spis treści

| | |
|---|----|
| 1. Wstęp | 2 |
| 2. Projekt bezpiecznej architektury | 2 |
| 2.1. Strefy DMZ | 2 |
| 2.2. Firewalle w architekturze | 2 |
| 2.3. Plan konfiguracji dla wszystkich firewalli | 3 |
| 2.4. Plan podziału na VLAN | 5 |
| 2.5. Wymagania konfiguracji portów i usług dla każdego hosta | 5 |
| 2.6. Wymagania konfiguracji poszczególnych usług bezpieczeństwa | 5 |
| 2.7. Diagram bezpiecznej architektury sieci | 6 |
| 2.8. Przepływy między obszarami | 7 |
| 3. Wdrożenie bezpiecznej architektury | 8 |
| 3.1. Diagram wdrożonej architektury sieci | 8 |
| 3.2. Firewalle w architekturze | 9 |
| 3.2.1. Firewall Gateway | 9 |
| 3.2.2. Firewall Central | 10 |
| 3.2.3. Firewall w drugim biurze | 10 |
| 3.3. DNS | 11 |
| 3.4. Reverse Proxy + WAF | 11 |
| 3.5. Serwer WWW | 11 |
| 3.6. Baza danych | 13 |
| 3.7. Site-to-site VPN | 15 |
| 3.8. Pulpit zdalny dla pracownika z jump server | 16 |
| 4. Audyt bezpieczeństwa sieci | 17 |
| 4.1. Część aktywna | 17 |
| 4.2. Audyt względem standardu | 19 |
| 4.3. Prezentacja | 19 |
| 5. Podsumowanie | 19 |
| 6. Materiały dodatkowe | 19 |

1. Wstęp

Celem tego projektu jest zaplanowanie bezpiecznej architektury sieci dla firmy, która przygotowuje się do otwarcia nowego biura. Następnie, w ramach laboratorium 2, będziemy musieli zaimplementować architekturę, której projekt w tym sprawozdaniu opiszemy.

2. Projekt bezpiecznej architektury

W instrukcji zostały wyodrębnione następujące obszary, na które mamy zwrócić uwagę w trakcie projektowania i implementacji:

1. Część lokalna

- a) Segment żółty - segment stykający się z Internetem i dostępem do innych lokalizacji (cloud, drugie biuro),
- b) Segment niebieski - segment z usługami współdzielonymi, w tym dostępnymi z Internetu (w naszym przypadku serwer web),
- c) Segment zielony - segment reprezentujący podstawowe środowisko pracy w biurze,
- d) Segment czerwony - segment o zaokrąglonych wymaganiach dla cyberbezpieczeństwa (w naszym przypadku baza danych).

2. Część zdalna

- a) Drugie biuro firmy
- b) Cloud

W dalszej części sprawozdania będziemy się odwoływać do zdefiniowanego powyżej nazewnictwa, np. segment żółty itd.

2.1. Strefy DMZ

DMZ (Demilitarized Zone) to specjalnie skonfigurowana strefa sieciowa, która znajduje się pomiędzy zewnętrzną (publiczną) i wewnętrzną (prywatną) siecią komputerową. DMZ ma na celu zwiększenie bezpieczeństwa systemów informatycznych, zwłaszcza w kontekście serwerów udostępnianych publicznie, takich jak serwery www. DMZ znajduje zatem sensowne zastosowanie w naszej architekturze sieci.

W skład stref zdemilitaryzowanych wchodzi segment żółty i segment niebieski. Taki podział wynika z faktu, że działanie obu opiera się o interakcję z siecią publiczną.

DMZ zostanie wdrożona dzięki odpowiedniej konfiguracji firewalli.

- **Gateway Firewall** filtruje ruch na wejściu do DMZ,
- **Central Firewall** odgradza koniec DMZ od sieci lokalnej biura.

2.2. Firewall w architekturze

Podjęliśmy decyzję projektową o umiejscowieniu następujących firewalli w naszej architekturze:

1. **Gateway Firewall** - Firewall w tym przypadku pełni rolę ochrony pomiędzy naszą siecią a Internetem, kontrolując ruch sieciowy i zabezpieczając przed potencjalnymi zagrożeniami.
2. **Central Firewall** - Firewall w tym przypadku pełni rolę ochrony środowiska pracy w biurze (segment zielony) i bazę danych (segment czerwony). Przesiewa, co do tych segmentów jest wysyłane i co przez te segmenty jest wysyłane. Umożliwia on ograniczony dostęp do Internetu segmentu zielonego.
3. **Firewall w segmencie cloudowym** - Firewall, który filtruje i kontroluje ruch przychodzący z zewnątrz dla segmentu cloud'owego.
4. **Firewall w drugim biurze** - Firewall, który filtruje i kontroluje w drugim biurze ruch przychodzący z zewnątrz.

2.3. Plan konfiguracji dla wszystkich firewalli

| | | Source | | | | | | | | | |
|-------------|---------------------|----------|------------|-------|-----------------|------------|---------------------|------------|------------------|------|-----------------|
| | | Internet | 2nd Office | Cloud | Remote Employee | WWW Server | Reverse Proxy + WAF | DNS Public | Central Firewall | NIDS | Log Collector 1 |
| Destination | Internet | | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ |
| | 2nd Office | ✓ | | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ |
| | Cloud | ✓ | ✗ | | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ |
| | Remote Employee | ✓ | ✗ | ✗ | | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ |
| | WWW Server | ✗ | ✗ | ✗ | ✗ | | ✓ | ✗ | ✓ | ✗ | ✓ |
| | Reverse Proxy + WAF | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✗ | ✓ |
| | DNS Public | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | | ✓ | ✗ | ✓ |
| | Central Firewall | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✗ |
| | NIDS | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | | ✓ |
| | Log Collector 1 | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | |

- komunikacja zablokowana
 - komunikacja dozwolona
 - nie dotyczy

Rys. 1: Macierz dopuszczalnych i blokowanych połączeń dla Gateway Firewall.

| | | Source | | | | | | | |
|-------------|------------------|------------------|------|---------------|-------------|------------|-------------|-----------------|----------|
| | | Gateway Firewall | SIEM | Scanning Host | DNS Private | EDR Server | EDR Clients | Log Collector 2 | Database |
| Destination | Gateway Firewall | | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| | SIEM | ✗ | | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| | Scanning Host | ✗ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✗ |
| | DNS Private | ✓ | ✗ | ✓ | | ✓ | ✓ | ✓ | ✓ |
| | EDR Server | ✗ | ✗ | ✓ | ✓ | | ✓ | ✓ | ✗ |
| | EDR Clients | ✓ | ✗ | ✓ | ✓ | ✓ | | ✗ | ✓ |
| | Log Collector 2 | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | | ✓ |
| | Database | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | |

- komunikacja zablokowana
 - komunikacja dozwolona
 - nie dotyczy

Rys. 2: Macierz dopuszczalnych i blokowanych połączeń dla Central Firewall.

2.4. Plan podziału na VLAN

Zostaną utworzone następujące sieci VLAN:

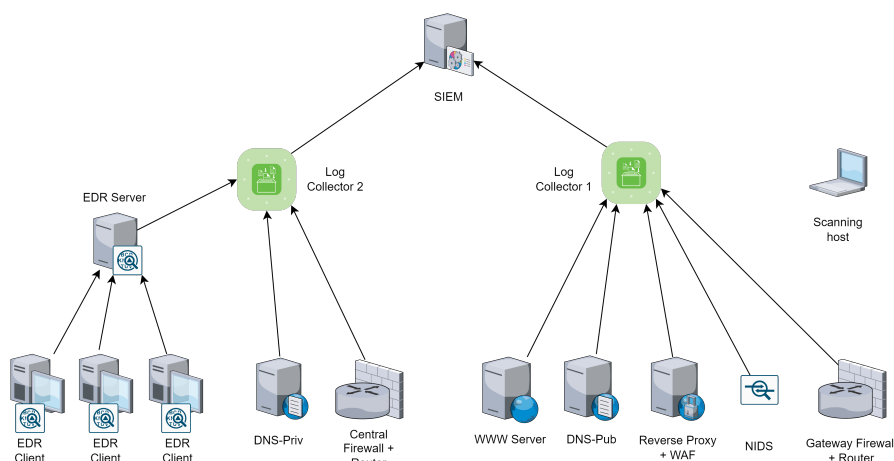
1. VLAN Yellow - dla DMZ 1. Hosty: DNS-Pub, Reverse Proxy + WAF
2. VLAN Blue - dla DMZ 2. Host: WWW Sever
3. VLAN Green - dla strefy Office. Hosty: EDR Clients, EDR Server, DNS-Priv, Scanning Host.
4. VLAN Red - strefa Restricted. Host: Database.
5. VLAN Black - dla infrastruktury bezpieczeństwa. Hosty: EDR Server, Log Collector 2, SIEM, DNS-Priv, Log Collector 1, WWW Server, DNS-Pub, NIDS, Reverse Proxy + WAF.

2.5. Wymagania konfiguracji portów i usług dla każdego hosta

| Host | Wymagania |
|----------------------|---|
| DNS Public | Port do obsługi zapytań DNS z segmentów żółtego i niebieskiego Możliwość wysyłania logów do Log Collector 1 |
| DNS Private | Port do obsługi zapytań DNS z segmentu zielonego Możliwość wysyłania logów do Log Collector 2 |
| Reverse Proxy + WAF | Port do udostępniania proxowanej usługi www Możliwość wysyłania logów do Log Collector 1 |
| WWW Server | Port do komunikacji HTTPS Możliwość wysyłania logów do Log Collector 1 |
| SIEM | Port do zbierania logów Port umożliwiający dostęp do panelu obsługi SIEM |
| EDR Server | Port do odbierania logów i alertów z EDR Clients Możliwość wysyłania logów do Log Collector 2 |
| EDR Clients | Port do wysyłania logów i alertów do EDR Server Port do obsługi usługi zdalnego dostępu Porty umożliwiające korzystanie z zasobów internetu m.in. 443, 80, 53 |
| Scanning Host | Porty umożliwiające poprawne działanie skanera podatności |
| Database | Port do komunikowania poleceń SQL |
| Log Collector 1 & 2 | Port do odbierania logów od wybranych elementów architektury Port do wysyłania logów/alertów do SIEM |
| Remote Employee Host | Port do zdalnego połączenia do wybranych maszyn biurowych w strefie zielonej Porty umożliwiające korzystanie z zasobów internetu m.in. 443, 80, 53 |
| NIDS | Port na który trafia zebrana kopia ruchu Port do wysyłania logów do Log Collector 1 Port do konfiguracji narzędzia do detekcji |

Tabela 1: Wykaz wymagań związanych z portami i usługami

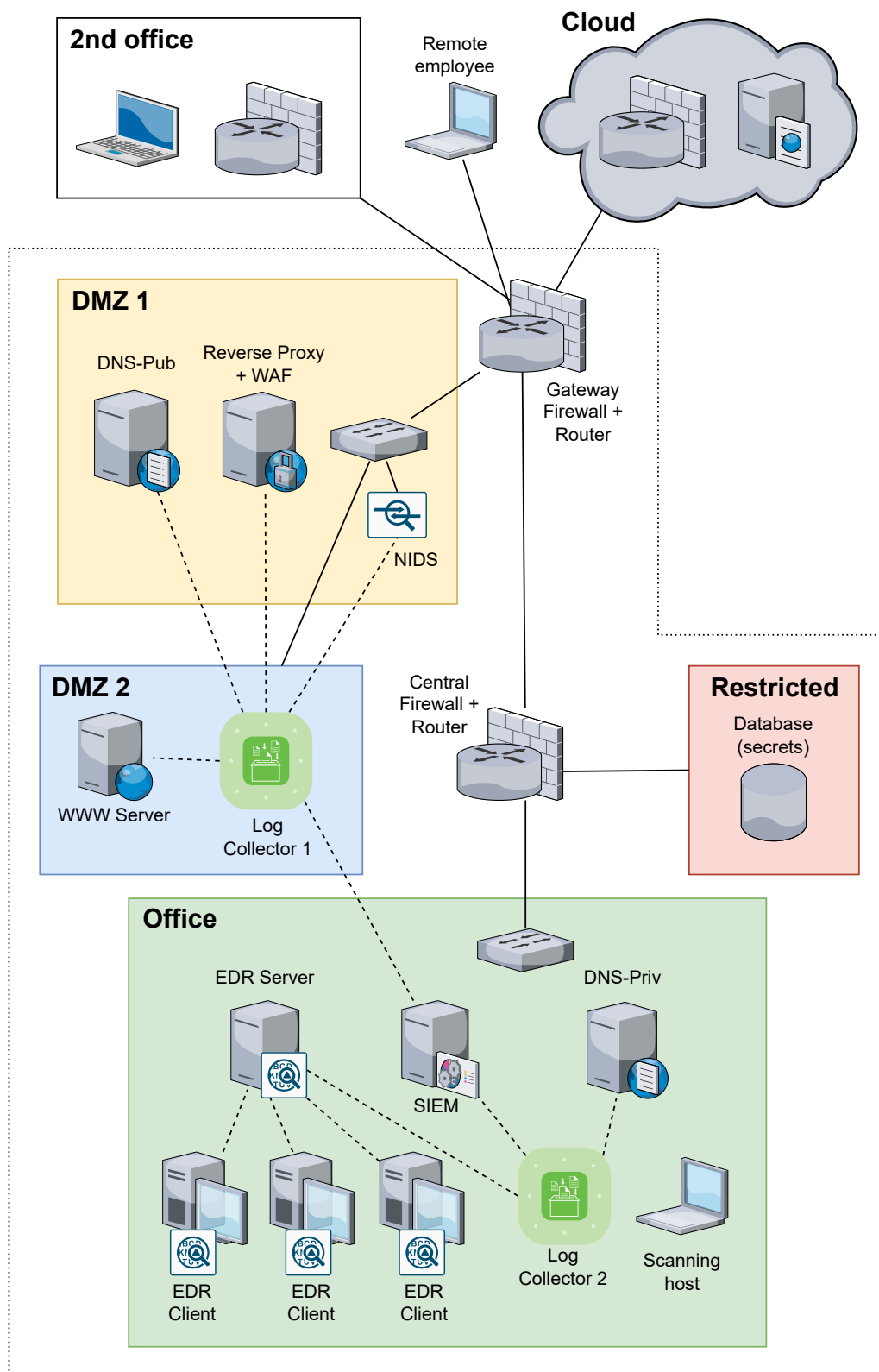
2.6. Wymagania konfiguracji poszczególnych usług bezpieczeństwa



Rys. 3: Diagram konfiguracji usług bezpieczeństwa, w tym przepływ logów.

2.7. Diagram bezpiecznej architektury sieci

Poniżej znajduje się schemat projektowanej przez nas architektury sieci z uwzględnieniem elementów zapewniających bezpieczeństwo.



Rys. 4: Schemat bezpiecznej architektury sieci.

2.8. Przepływy między obszarami

Poniżej znajduje się macierz z przepływu między obszarami. Znakiem ✓ zostały oznaczone pola, które wskazują na obustronną i bezpośrednią (w większości przypadków) komunikację między obszarami. Znakiem X są oznaczone pola wskazujące na brak komunikacji między obszarami. Niektóre z połączeń realizowane są jedynie przy konkretnych warunkach:

- Pracownik Zdalny - Obszar niebieski: komunikacja jest możliwa dopiero po połączeniu z odpowiednim hostem pracownika w strefie zielonej,
- Pracownik Zdalny - Obszar zielony: jest realizowane jedynie przy pomocy VPN,
- Obszar zielony - Obszar czerwony: zakładamy, że grupa użytkowników z dostępem do bazy danych będzie ograniczona tylko do osób z odpowiednimi uprawnieniami.

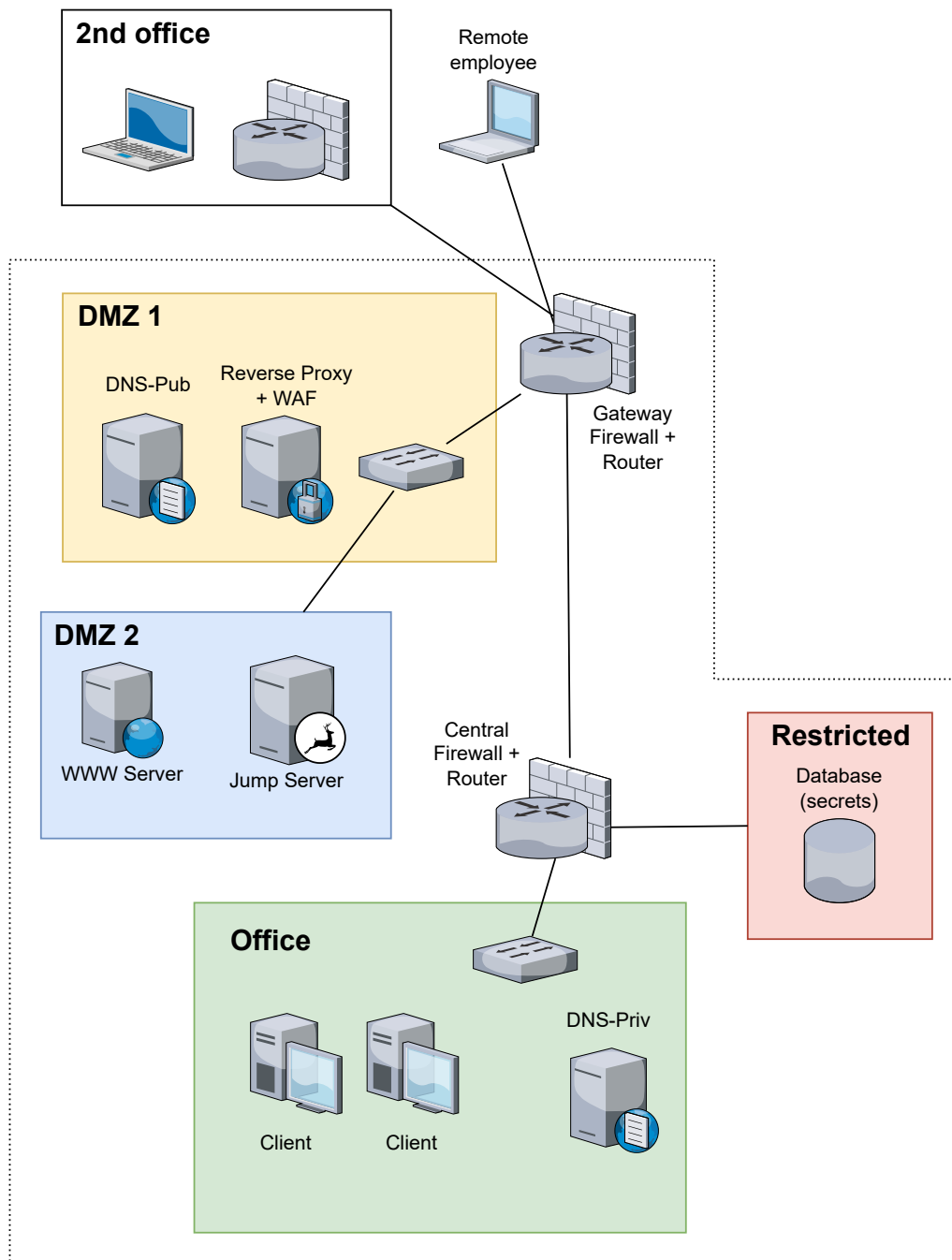
| | | Source | | | | | | |
|-------------|------------------|-------------------------------|-------|--------------|-------|-------------------------------|------------------------------|----------|
| | | Pracownik Zdalny | Cloud | Drugie Biuro | Żółty | Niebieski | Zielony | Czerwony |
| Destination | Pracownik Zdalny | | X | X | ✓ | ✓ (poprzez strefę zieloną) | ✓ (przez VPN) | X |
| | Cloud | X | | X | ✓ | X | X | X |
| | Drugie Biuro | X | X | | ✓ | X | X | X |
| | Żółty | ✓ | ✓ | ✓ | | ✓ | X | X |
| | Niebieski | ✓ (poprzez strefę zieloną) | X | X | ✓ | | ✓ | X |
| | Zielony | ✓ (przez VPN) | X | X | X | ✓ | | ✓ |
| | Czerwony | X | X | X | X | X | ✓ (uprawnieni pracownicy) | |

| | |
|--|---------------------------|
| | - komunikacja zablokowana |
| | - komunikacja dozwolona |
| | - nie dotyczy |

Rys. 5: Macierz przepływu między obszarami.

3. Wdrożenie bezpiecznej architektury

3.1. Diagram wdrożonej architektury sieci



Rys. 6: Schemat wdrożonej architektury sieci.

3.2. Firewall w architekturze

3.2.1. Firewall Gateway

Firewall Gateway został wdrożony zgodnie z założeniami projektowymi. Pełni on rolę jednocześnie servera DHCP, routera, firewalla i switcha. Kontroluje on przepływ pomiędzy vlan yellow, vlan blue i siecią zewnętrzną a siecią lokalną w której znajduje się central firewall. W celu umożliwienia dostępu do zasobów serverów ze stref DMZ zastosowano mechanizm *port forwarding*. Udostępniono stronę WWW na porcie 443, usługę DNS na porcie 53 oraz 2 porty dla ruchu VPN.

| | States | Protocol | Source | Port | Destination | Port | Gateway | Queue | Schedule | Description |
|-------------------------------------|--------------|------------------------------|------------------|------|--------------------|-----------------|---------|-------|----------|-------------------------------------|
| <input type="checkbox"/> | ✓ 0/1.01 MiB | IPv4 UDP | * | * | * | 53 (DNS) | * | none | | NAT wan to dns-pub |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 ICMP <i>echo req</i> | * | * | WAN address | * | * | none | | ping from wan |
| <input type="checkbox"/> | ✓ 0/1.37 MiB | IPv4 * | * | * | LAN subnets | * | * | none | | wan to lan |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 UDP | * | * | 10.10.1.13 | 51820 | * | none | | NAT wireguard |
| <input type="checkbox"/> | ✓ 0/120 KiB | IPv4 TCP | * | * | 10.10.3.9 | 443 (HTTPS) | * | none | | NAT wan to www(waf) |
| <input type="checkbox"/> | ✓ 0/10 KiB | IPv4 UDP | * | * | 10.10.2.9 | 8999 | * | none | | NAT wg jumpserver |
| | States | Protocol | Source | Port | Destination | Port | Gateway | Queue | Schedule | Description |
| <input checked="" type="checkbox"/> | ✓ 0/4.65 MiB | * | * | * | LAN Address | 443 80 22 | * | * | | Anti-Lockout Rule |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 UDP | 10.10.5.0/24 | * | 10.10.3.10 | 53 (DNS) | * | none | | green vlan to dns-pub |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 * | 10.10.5.14 | * | 10.10.2.9 | * | * | none | | lan to jumpserver |
| <input type="checkbox"/> | ✓ 0/100 B | IPv4 TCP | LAN subnets | * | 10.10.3.9 | 443 (HTTPS) | * | none | | lan to www(waf) |
| <input type="checkbox"/> | ✗ 0/6.04 MiB | IPv4 * | * | * | VLANYELLOW subnets | * | * | none | | restrict lan to vlan yellow |
| <input type="checkbox"/> | ✗ 0/0 B | IPv4 * | * | * | VLANBLUE subnets | * | * | none | | restrict lan to vlan blue |
| <input type="checkbox"/> | ✓ 1/6.40 MiB | IPv4 * | LAN subnets | * | * | * | * | none | | lan to wan |
| | States | Protocol | Source | Port | Destination | Port | Gateway | Queue | Schedule | Description |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 TCP | 10.10.2.11 | * | 10.10.3.9 | * | * | none | | www to waf |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 UDP | 10.10.2.9 | 8999 | * | * | * | none | | wireguard jump server |
| <input type="checkbox"/> | ✓ 0/4.47 MiB | IPv4 * | 10.50.0.2 | * | 10.10.5.14 | * | * | none | | jumpserver to remote desktop server |
| <input type="checkbox"/> | ✓ 0/0 B | IPv4 UDP | VLANBLUE subnets | * | 10.10.3.10 | 53 (DNS) | * | none | | vlanblue to dnspublic |
| <input type="checkbox"/> | ✓ 0/830 KiB | IPv4 UDP | 10.10.3.10 | * | * | * | * | none | | dns-pub to wan dns |
| <input type="checkbox"/> | ✓ 0/4 KiB | IPv4 TCP | 10.10.3.9 | * | * | 443 (HTTPS) | * | none | | waf to wan https |
| <input type="checkbox"/> | ✓ 0/73 KiB | IPv4 TCP | 10.10.3.9 | * | 10.10.2.11 | 5000 | * | none | | waf to www |

Rys. 7: Reguły na Firewall Gateway

3.2.2. Firewall Central

Firewall Central stanowi barierę bezpieczeństwa w ruchu sieciowym między vlan green, vlan red oraz resztą świata. Założenia projektowe zakładały wykorzystanie między innymi oprogramowania typu SIEM, hosta skanującego, serwera EDR oraz kolektora logów. Są to elementy, których nie wdrożyliśmy w części laboratoryjnej, dlatego też macierz dopuszczalnych i blokowanych połączeń dla Firewall Central po modyfikacji prezentuje się następująco.

| | | Source | | | | |
|-------------|------------------|------------------|-------------|----------------|-----------------|----------|
| | | Gateway Firewall | DNS Private | Office1 Worker | Inni Pracownicy | Database |
| Destination | Gateway Firewall | | ✓ | ✓ | ✓ | ✗ |
| | DNS Private | ✓ | | ✓ | ✓ | ✓ |
| | Office1 Worker | ✓ | ✓ | | ✓ | ✓ |
| | Inni Pracownicy | ✓ | ✓ | ✓ | | ✗ |
| | Database | ✗ | ✓ | ✓ | ✗ | |

- komunikacja zablokowana
 - komunikacja dozwolona
 - nie dotyczy

Rys. 8: Macierz dopuszczalnych i blokowanych połączeń dla Central Firewall.

3.2.3. Firewall w drugim biurze

Ten firewall został skonfigurowany dla dwóch sieci: VLANOFFICE2 i WAN.

W przypadku sieci VLANOFFICE2 w trakcie tworzenia reguł skupiamy się na akceptowaniu ruchu z i do strefy żółtej. Inna komunikacja ma zostać zablokowana. Jest to zgodne z macierzą przepływu między obszarami.

Reguły dla VLANOFFICE2 prezentują się następująco:

- Pozwolenie na ruch z hosta pracownika (10.10.3.9) na reverse proxy, które jest w strefie żółtej (10.10.3.9),
- Pozwolenie na ruch z reverse proxy, które jest w strefie żółtej (10.10.3.9) na hosta pracownika (10.20.1.10),
- Pozwolenie na ruch z hosta pracownika (10.10.3.9) na DNS public, które jest w strefie żółtej (10.10.3.10),
- Pozwolenie na ruch z DNS public, które jest w strefie żółtej (10.10.3.10) na hosta pracownika (10.20.1.10).

W przypadku sieci WAN w trakcie tworzenia reguł skupiamy się na zapewnieniu bezpiecznego dostępu do Internetu.

Reguły dla WAN prezentują się następująco:

- Pozwolenie na ruch z hosta pracownika (10.20.1.10) na porty WorkerPorts,
- Pozwolenie na ruch z WorkerPorts na host pracownika (10.20.1.10).

WorkerPort to stworzony przez nas alias z listą portów, które są standardowe i na których świadczone są podstawowe usługi potrzebne do używania Internetu. Lista tych portów wygląda następująco: 80, 443, 21, 25, 110, 143, 22, 53, 67, 68, 123, 20, 79, 70, 220, 995, 514.

3.3. DNS

Zgodnie z projektem wdrożono dwa serwery DNS: publiczny w strefie żółtej oraz prywatny w strefie zielonej. Zastosowano do tego system BIND9 (Berkeley Internet Name Domain).

W zakresie prywatnym serwery DNS tłumaczą adresy domeny wewnętrznej na prywatne adresy IP. Żądania dotyczące domen publicznych są przekazywane do serwerów DNS autorytarnych, za które przyjęliśmy serwery DNS Politechniki (IP 10.255.255.10). Dla zapewnienia separacji sieci wewnętrznej od strefy DMZ serwery są zarządzane niezależnie, a transfer informacji o zonach jest zablokowany.

Konfiguracja serwera DNS złożona jest m.in. z plików:

- `/etc/bind/named.conf` - główny plik deklarujący gdzie znajdują się pliki konfiguracyjne
- `/etc/bind/named.conf.options` - globalna konfiguracja serwera, forwarding
- `/etc/bind/zones/db.*` - pliki z deklaracjami zon

3.4. Reverse Proxy + WAF

Wdrożono serwer pełniący funkcję Reverse Proxy z funkcjonalnością WAF wykorzystując oprogramowanie nginx.

Instalacji nginx dokonano z kodu źródłowego poprzez pobranie narzędzia ze strony: <https://nginx.org/en/download.html> w wersji *nginx-1.22.0*. Konfiguracja przedstawia się następująco:

```
./configure --sbin-path=/usr/bin/nginx --conf-path=/etc/nginx/nginx.conf
↳ --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log
↳ --with-pcre --pid-path=/var/run/nginx.pid --add-dynamic-module=../../naxsi/naxsi_src
↳ --with-http_ssl_module
```

Wartym odnotowania jest tag `--add-dynamic-module=../../naxsi/naxsi_src`, który wskazuje na potrzebę wprowadzenia do procesu kompilacji modułu naxsi.

Naxsi jest modulem zewnętrznym dla nginx chroniącym przed atakami tj. SQL Injection czy XSS. Pozyskano i zainstalowano z <https://github.com/wargio/naxsi>.

Po kompilacji i instalacji narzędzi przystąpiono do konfiguracji serwera aby pełnił rolę *reverse proxy*.

Proxowana jest witryna pod adresem lokalnym `https://10.10.2.11:5000`. Do zapewnienia bezpiecznego połączenia https wykorzystano wygenerowane, samodzielnie podpisane klucze "snakeoil" - zalecane tylko do stosowania w fazie development lub debug, ale na potrzeby tego zadania wystarczające do przedstawienia PoC.

3.5. Serwer WWW

Serwer www został zrealizowany zgodnie z założeniami projektowymi. Przy pomocy frameworka Flask udało nam się napisać aplikację webową `app.py` w języku Python. W języku HTML zaś udało nam się napisać szablon do renderowania widoku aplikacji internetowej, `index.html`.

Aplikacja webowa `app.py` znajduje się w folderze `/var/www/`. Jeśli zaś chodzi o `index.html`, to znajduje się w folderze `/var/www/templates`.

W Flask, katalog `/templates` jest domyślnym katalogiem, w którym framework szuka plików szablonów HTML. Oznacza to, że pliki HTML używane jako szablony powinny znajdować się w tym katalogu, aby Flask mógł je znaleźć i wykorzystać podczas renderowania stron. Podczas korzystania z funkcji `render_template` w Flask, framework automatycznie szuka plików w folderze `/templates` bez konieczności podawania pełnej ścieżki.

Strona internetowa jest prosta i zawiera informacje kontaktowe naszej firmy.
Kod strony w języku HTML prezentuje się następująco:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Strona Firmy</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }

    header {
      background-color: #B2A1AB;
      color: #3B3538;
      padding: 15px;
      text-align: center;
      position: relative;
    }

    section {
      padding: 20px;
      color: #363033;
    }

    footer {
      background-color: #B2A1AB;
      color: #363033;
      text-align: center;
      padding: 10px;
      position: fixed;
      bottom: 0;
      width: 100%;
    }
  </style>
</head>
<body>
  <header>
    <h1>Strona firmy</h1>
  </header>

  <section>
    <p>Witaj na stronie naszej firmy.</p>
    <p><strong>Kontakt:</strong></p>
    <p>Tel.: +48 111 222 333</p>
    <p>E-mail: naszafirm@gmail.com</p>
  </section>

  <footer>
    © 1998-2024 Nasza Firma,
    Pl. Firmy 1, 00-661 Warszawa
  </footer>
</body>
</html>
```

Kod aplikacji napisanej w języku Python prezentuje się następująco:

```
from flask import Flask, render_template

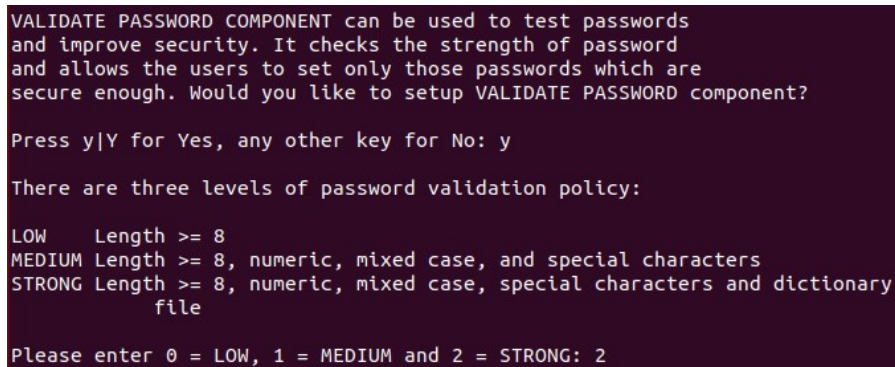
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True, ssl_context='adhoc', host='10.10.2.11')
```

3.6. Baza danych

Zgodnie z projektem sieci do zbudowania bazy danych wykorzystaliśmy narzędzie MySQL. Zaczęliśmy od bezpiecznej instalacji serwera MySQL na maszynie odpowiadającej za bazę danych za pomocą komendy *sudo mysql_secure_installation*. W przeciwieństwie do szybkiej instalacji umożliwia ona między innymi wybór dodatkowych zabezpieczeń serwera. Między innymi daje możliwość dodania sprawdzania siły haseł, z których korzystają użytkownicy przy logowaniu do bazy danych.



```
VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW      Length >= 8
MEDIUM  Length >= 8, numeric, mixed case, and special characters
STRONG  Length >= 8, numeric, mixed case, special characters and dictionary
        file

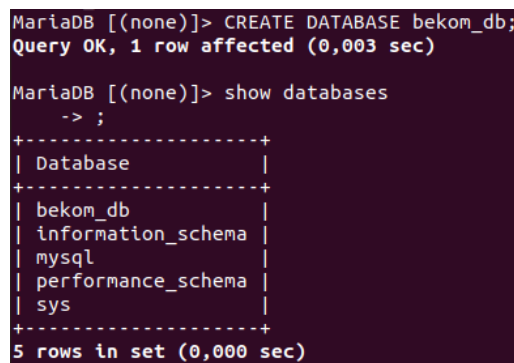
Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 2
```

Rys. 9: Instalacja serwera MySQL.

Podczas instalacji serwera MySQL tworzone są domyślne przykładowe konta użytkowników. W procesie instalacji również pojawia się pytanie o to, czy usunąć tych użytkowników. Zgodziliśmy się na taki zabieg, ponieważ domyślni użytkownicy posiadają słabe hasła oraz mogą mieć zbyt duże uprawnienia co staje się potencjalną luką bezpieczeństwa systemu.

Pierwszym krokiem po instalacji było uruchomienie serwera MySQL przy pomocy komendy *sudo systemctl enable --now mysql*. Następnie zalogowaliśmy się do serwera na konto root poprzez komendę *sudo mysql -u root -p*.

Działania na serwerze zaczęliśmy od utworzenia nowej bazy danych - *bekom_db*.



```
MariaDB [(none)]> CREATE DATABASE bekom_db;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> show databases
-> ;
+-----+
| Database |
+-----+
| bekom_db |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.000 sec)
```

Rys. 10: Utworzenie bazy danych.

Następnie dodaliśmy do `bekom_db` encję przechowującą informacje o klientach firmy - *Clients*. Każdy rekord w tablicy ma 4 atrybuty:

- `Id_client` - identyfikator klienta, jest on jednocześnie kluczem głównym tablicy; przyjmuje wartości typu `INTEGER`, czyli liczby całkowite. Wartość `NOT NULL` oznacza, że ten atrybut nie może być pusty.
- `Name` - imię klienta; przyjmuje wartości typu `VARCHAR(50)`, czyli łańcuchy znaków o maksymalnej długości 50 znaków. Wartość `NOT NULL` oznacza, że ten atrybut nie może być pusty.
- `Surname` - nazwisko klienta; przyjmuje wartości typu `VARCHAR(50)`, czyli łańcuchy znaków o maksymalnej długości 50 znaków. Wartość `NOT NULL` oznacza, że ten atrybut nie może być pusty.
- `Zodiac_sign` - znak zodiaku klienta; przyjmuje wartości typu `VARCHAR(30)`, czyli łańcuchy znaków o maksymalnej długości 30 znaków. W tym przypadku nie ma wartości `NOT NULL`, co oznacza, że ten atrybut może pozostać pusty.

```
MariaDB [(none)]> USE bekom_db
Database changed
MariaDB [bekom_db]> CREATE TABLE Clients(Id_client INT NOT NULL, Name VARCHAR(50) NOT NULL, Surname VARCHAR(50) NOT NULL, Zodiac_sign VARCHAR(30), PRIMARY KEY(Id_client));
Query OK, 0 rows affected (0,020 sec)

MariaDB [bekom_db]> DESCRIBE Clients;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Id_client | int(11) | NO | PRI | NULL | |
| Name | varchar(50) | NO | | NULL | |
| Surname | varchar(50) | NO | | NULL | |
| Zodiac_sign | varchar(30) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0,001 sec)
```

Rys. 11: Utworzenie encji w bazie danych.

Następnym elementem było zapełnienie tablicy *Clients* rekordami.

```
MariaDB [bekom_db]> INSERT INTO Clients(Id_client, Name, Surname, Zodiac_sign) VALUES(1, 'Andrzej', 'Kwiatkowski', 'Rak');
Query OK, 1 row affected (0,004 sec)

MariaDB [bekom_db]> INSERT INTO Clients(Id_client, Name, Surname, Zodiac_sign) VALUES(2, 'Kamil', 'Ślimak', 'Panna');
Query OK, 1 row affected (0,003 sec)

MariaDB [bekom_db]> INSERT INTO Clients(Id_client, Name, Surname, Zodiac_sign) VALUES(3, 'Jan', 'Kowalski', 'Ryby');
Query OK, 1 row affected (0,003 sec)

MariaDB [bekom_db]> INSERT INTO Clients(Id_client, Name, Surname, Zodiac_sign) VALUES(4, 'Tomasz', 'Nowak', 'Koziorożec');
Query OK, 1 row affected (0,004 sec)

MariaDB [bekom_db]> SELECT * FROM Clients;
+-----+-----+-----+-----+
| Id_client | Name | Surname | Zodiac_sign |
+-----+-----+-----+-----+
| 1 | Andrzej | Kwiatkowski | Rak |
| 2 | Kamil | Ślimak | Panna |
| 3 | Jan | Kowalski | Ryby |
| 4 | Tomasz | Nowak | Koziorożec |
+-----+-----+-----+-----+
4 rows in set (0,000 sec)
```

Rys. 12: Dodanie rekordów do encji 'Clients'.

Po utworzeniu bazy danych potrzebowaliśmy jeszcze konta użytkownika `bekom_db`, który będzie się mógł zdalnie logować z maszyny serwera WWW do serwera MySQL, skąd będzie pobierał dane o klientach firmy. W tym celu utworzyliśmy użytkownika 'user', który może zalogować się do serwera bazy danych z urządzenia o adresie IP 10.10.5.14, czyli adresie, pod którym jest widoczna maszyna Worker 1 w naszej sieci. Jest to jedyny host, który może korzystać z naszej bazy danych.

```
MariaDB [bekom_db]> CREATE USER 'user'@'10.10.5.14' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0,004 sec)
```

Rys. 13: Dodanie użytkownika bazy danych `bekom_db`.

Na koniec sprawdziliśmy poprawność konfiguracji użytkownika bazy danych poprzez próbę zdalnego połączenia się z serwerem MySQL z maszyny Worker 1 za pomocą komendy `mysql -u user -p -h 10.10.4.9`. Adres 10.10.4.9 to IP, pod którym jest widoczna maszyna bazy danych w naszej sieci. Jak możemy zauważyć próba przebiegła pomyślnie, a użytkownik jest w stanie odczytywać dane z tablic zawartych w `bekom_db`.

```
user@worker1:~$ mysql -u user -p -h 10.10.4.9
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.4-MariaDB-1~deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE bekom_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [bekom_db]> SELECT * FROM Clients;
+-----+-----+-----+-----+
| Id_client | Name   | Surname | Zodiac_sign |
+-----+-----+-----+-----+
| 1         | Andrzej | Kwiatkowski | Rak         |
| 2         | Kamil  | Ślimak    | Panna       |
| 3         | Jan    | Kowalski   | Ryby        |
| 4         | Tomasz | Nowak     | Koziorożec  |
+-----+-----+-----+-----+
4 rows in set (0.001 sec)
```

Rys. 14: Zdalne logowanie do 'bekom_db' z Worker 1.

Możemy również się zalogować do bazy danych korzystając z naszej domeny `db.amogus.sus`.

```
user@worker1:~$ mysql -u user -p -h db.amogus.sus
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.11.4-MariaDB-1~deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

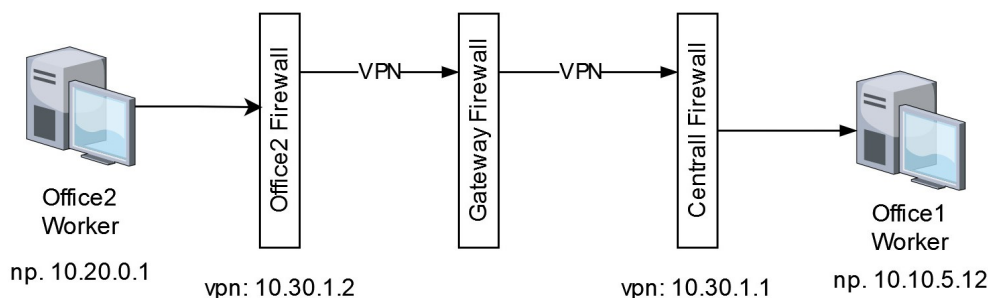
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> exit
Bye
user@worker1:~$
```

Rys. 15: Zdalne logowanie do 'bekom_db' z Worker 1 za pomocą domeny.

3.7. Site-to-site VPN

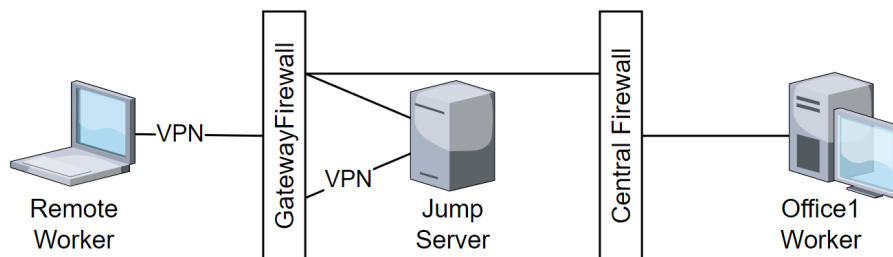
Usługę site-to-site VPN zrealizowano przy pomocy oprogramowania *Wireguard*, które zainstalowano i skonfigurowano na *Central Firewall* oraz *Firewall w drugim biurze*. W ten sposób utworzono połączenie pomiędzy siecią w drugim biurze a siecią w biurze głównym.



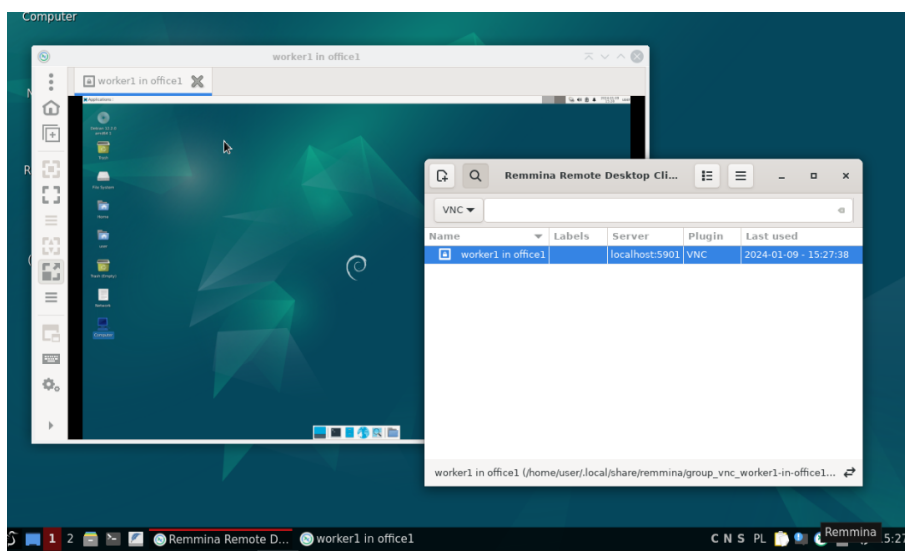
Rys. 16: Architektura site-to-site VPN

3.8. Pulpit zdalny dla pracownika z jump server

Pulpit zdalny zrealizowany został przy pomocy protokołu VNC. W Vlan Blue postawiony został Jump Server do którego można połączyć się z sieci WAN poprzez tunel VPN Wireguard. Z Jump Server można połączyć się do zdalnego pulpitu na hostach w Vlan Green. Zrealizowany został za pomocą oprogramowania TigerVNC. Żeby połączyć się do zdalnego pulpitu trzeba najpierw nawiązać połączenie z Jump Server następnie ustawić tunelowanie portu 5901 po SSH do hosta na którym żądamy zdalnego pulpitu, a następnie nawiązać połączenie VNC za pomocą agenta VNC np. Remmina.



Rys. 17: Schemat połączenia zdalnego pracownika z hostem w Office 1.



Rys. 18: Widok zdalnego pulpitu dla pracownika z sieci zewnętrznej.

4. Audyt bezpieczeństwa sieci

Jako zespół odpowiedzialny za cyberbezpieczeństwo, otrzymaliśmy zadanie oceny środowiska testowego nowej, bezpiecznej architektury sieciowej (zaimplementowanej w ramach laboratorium 2). Naszym celem jest przeprowadzenie testów i badań konfiguracji sieci w celu rzetelnej oceny, czy infrastruktura jest odpowiednio zabezpieczona zgodnie z wymaganiami dotyczącymi cyberbezpieczeństwa.

4.1. Część aktywna

Wykonane zostały aktywne testy bezpieczeństwa badanej sieci. Obejmowały one skanowanie przy użyciu narzędzia NMAP. Dla każdej sieci vlan istniejącej w ramach głównej części sieci (vlan blue, yellow, green, red) wykonano skanowanie w poszukiwaniu hostów dostępnych z widoku tego vlan-a. Takie działanie przybliża sposób w jaki złośliwy aktor może prowadzić rekonesans sieci w celu np. wykonania lateral movement. Uzyskane wyniki pozwalają stwierdzić, że architektura sieci w zakresie routingu i switchingu jest bezpieczna, ponieważ hosty mogą połączyć się między sobą tylko w zakresie dostępu do usług wynikających z projektu sieci. Skan miał także sprawdzić, czy dostępne usługi są przestarzałe, co dawałoby cię szansy na możliwą exploitację, na szczęście wszystkie usługi są aktualne do najnowszych wersji.

Potencjalnym słabym punktem może być działanie serwerów SSH na wielu hostach na których jest to niepotrzebne, np. serwer dns, server waf itd. Może to tworzyć punkt wejścia dla złośliwego aktora. Proponowanym rozwiązaniem jest stworzenie Jump Server i skonfigurowanie hostów w taki sposób, aby akceptowały połączenia SSH tylko od niego, a dla wszystkich innych hostów serwery SSH byłyby niewidoczne. Użytkownik łączył by się bezpiecznym kanałem na Jump Server a następnie nawiązywałby połączenie do wybranego hosta z usługą serwera SSH.

```
user@worker1:~$ sudo nmap -sV 10.10.5.0/24 10.10.4.0/24 10.10.3.0/24 10.10.2.0/24 --top-ports 1000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-14 20:17 CET
Nmap scan report for 10.10.5.1
Host is up (0.00050s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.4 (protocol 2.0)
80/tcp    open  http     nginx
443/tcp   open  ssl/http nginx
MAC Address: BC:24:11:7E:47:BA (Unknown)

Nmap scan report for dns-priv.amogus.sus (10.10.5.11)
Host is up (0.00016s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
53/tcp    open  domain   ISC BIND 9.18.19-1~deb12u1 (Debian Linux)
MAC Address: BC:24:11:C7:F6:25 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for worker2.amogus.sus (10.10.5.16)
Host is up (0.00019s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u1 (protocol 2.0)
MAC Address: BC:24:11:36:D1:62 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.10.5.14
Host is up (0.0000060s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.10.3.9
Host is up (0.0012s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
443/tcp   open  ssl/http nginx
```

Rys. 19: Wyniki skanów w sieci vlan green.

```

user@dns-pub:~$ sudo nmap -sV 10.10.5.0/24 10.10.3.0/24 10.10.2.0/24 10.10.4.0/24 --top-ports 1000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-14 13:25 CST
Nmap scan report for 10.10.3.1
Host is up (0.00082s latency).
All 1000 scanned ports on 10.10.3.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: BC:24:11:2B:E7:95 (Unknown)

Nmap scan report for waf.amogus.sus (10.10.3.9)
Host is up (0.00016s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
80/tcp    open  http     nginx 1.22.0
443/tcp   open  ssl/http nginx
MAC Address: BC:24:11:06:0C:07 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for dns-pub.amogus.sus (10.10.3.10)
Host is up (0.000050s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
53/tcp    open  domain   ISC BIND 9.18.19-1~deb12u1 (Debian Linux)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Rys. 20: Wyniki skanów w sieci vlan yellow.

```

user@www:~$ sudo nmap -sV 10.10.5.0/24 10.10.4.0/24 10.10.2.0/24 10.10.3.0/24 --top-ports 1000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-14 20:32 CET
Nmap scan report for 10.10.2.1
Host is up (0.0020s latency).
All 1000 scanned ports on 10.10.2.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: BC:24:11:2B:E7:95 (Unknown)

Nmap scan report for 10.10.2.9
Host is up (0.00019s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
MAC Address: BC:24:11:05:53:D9 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 10.10.2.11
Host is up (0.000050s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u1 (protocol 2.0)
5000/tcp  open  ssl/upnp?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for waf.amogus.sus (10.10.3.9)
Host is up (0.00048s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
80/tcp    open  http     nginx 1.22.0
443/tcp   open  ssl/http nginx
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1024 IP addresses (4 hosts up) scanned in 567.86 seconds

```

Rys. 21: Wyniki skanów w sieci vlan blue.

```

user@db:~$ sudo nmap -sV 10.10.5.0/24 10.10.3.0/24 10.10.2.0/24 10.10.5.0/24 --top-ports 1000
Starting Nmap 7.93 ( https://nmap.org ) at 2024-01-14 13:51 CST
Nmap done: 1024 IP addresses (0 hosts up) scanned in 822.49 seconds
user@db:~$ _

```

Rys. 22: Wyniki skanów w sieci vlan red.

4.2. Audyt względem standardu

W ramach audytu względem kontroli pochodzących z NIST Cybersecurity Framework wykorzystaliśmy dokument <https://www.nist.gov/document/2018-04-16frameworkv11core1xlsx>. W kolumnie E dodaliśmy naszą ocenę (rating) środowiska zaimplementowanego w ramach laboratorium 2. Do oceny kontroli użyliśmy następujących oznaczeń:

- 1: zrealizowane,
- 0,5: częściowo zrealizowane,
- 0: niezrealizowane,
- N/D: kontrole niepodlegające problemowi (nie dotyczy).

Podsumowanie audytu względem standardu NIST:

- Uzyskane punkty: 17
- Ile kontroli dotyczyło: 48
- Ile kontroli nie dotyczyło: 60
- W sumie kontroli: 108

Plik PDF zawierający wyniki audytu znajduje się na kanale Teams zespołu SPAM. Dodatkowo, można go znaleźć pod linkiem dostępnym na końcu tego dokumentu w sekcji Materiały dodatkowe [1].

4.3. Prezentacja

Prezentacja zawierająca wyniki audytu znajduje się na kanale Teams zespołu SPAM. Dodatkowo, można ją znaleźć pod linkiem dostępnym na końcu tego dokumentu w sekcji Materiały dodatkowe [2].

5. Podsumowanie

Celem tego projektu było zaplanowanie architektury sieci, którą następnie zaimplementowaliśmy w ramach laboratorium 2. Wcielamy się w rolę członków zespołu cyberbezpieczeństwa firmy, która przygotowuje się do otwarcia nowego biura. Naszym zadaniem było zaprojektowanie, jak infrastruktura IT w nowym biurze zostanie zabezpieczona.

Według naszego projektu i implementacji, w ramach laboratorium, w sieci nowego biura znajdują się 4 obszary odpowiadające za poszczególne części sieci. Do tych obszarów należą:

- Obszar Żółty (DMZ): jest pośrednikiem między siecią a Internetem i innymi lokalizacjami
- Obszar Niebieski (DMZ): to źródło usługi WWW
- Obszar Czerwony: zawiera bazę danych, które wymagają szczególnej ochrony
- Obszar Zielony: obejmuje sieć biurową m.in. hosty pracowników

Ponadto zadbaliśmy o uwzględnienie bezpieczeństwa w sieci. W tym celu zaplanowaliśmy wdrożenie takich środków jak: kolektor logów, SIEM, NIDS, WAF oraz firewall. W ramach laboratorium z kolei wdrożyliśmy: serwer WWW, bazę danych, site-to-site VPN, Reverse proxy + WAF, firewall, DNS i zdalny pulpit z jump serverem.

W ramach laboratorium 3 przeprowadziliśmy audyt bezpieczeństwa stworzonej przez nas sieci. Mieliśmy okazję zobaczyć, na ile elementów składa się taki audyt i jak wiele należy wziąć pod uwagę w trakcie projektu i implementacji architektury sieci.

Do tworzenia diagramów wykorzystaliśmy oprogramowanie draw.io, które w przystępny i przejrzysty sposób pozwoliło nam na graficzne przedstawienie zaprojektowanej przez nas architektury sieci. Z kolei do stworzenia macierzy dopuszczalnych i blokowanych połączeń wykorzystaliśmy Excel.

6. Materiały dodatkowe

1. Audyt względem kontroli pochodzących z NIST Cybersecurity Framework
2. Prezentacja z wynikami audytu