الجمهورية العربية السورية

المعهد العالي للعلوم التطبيقية والتكنولوجيا

قسم الاتصالات

العام الدراسي 2025/2024

مشروع تخرج

أعد لنيل درجة الإجازة في هندسة الاتصالات

تصميم وتنفيذ معمارية ثقة-صفرية باستخدام أدوات مفتوحة المصدر

Implementing a Zero-Trust Architecture using open-source tools

تقديم

رسول رفعت بشر

إشراف

د. خلدون خرزم

ما. مفيد دوزكنجي

2025

إهداء

إلى من وددناه ان يكون معنا ولكن أبى إلا أن يرحل وقلوبنا معه الى جوار ربه...

المرحوم خضر محمد خليل

إلى أعز الناس، اللذين كانا ولا يزالان سندي ودعمي الأكبر في كل خطوة أخطوها.

والديّ العزيزين

إلى الذين كانوا دائماً بجانبي، يشاركونني أفراحي وأحلامي.

إخوتي الأحباء

إلى الذين كانوا عوناً لي في كل خطوة على هذا الطريق.

أصدقائي الأعزاء

الحكمة

"The way to get started is to quit talking and begin doing"

Walt Disney

كلمة شكر

أتوجه بامتناني إلى المشرفين **د. خلدون خرزم** و **ما. مفيد دوزكنجي** لمتابعة العمل المنجز خطوة تلو الأخرى ولتوجيهاتهم وملاحظاتهم القيمة طوال فترة إعداد هذا المشروع.

خلاصة

مع ازدياد التهديدات السيبرانية وتعقّد البنى الشبكية الحديثة، أصبحت الحلول التقليدية المعتمدة على حماية المحيط الشبكي غير كافية لضمان أمان الموارد الداخلية. لهذا السبب، يبرز مفهوم الثقة الصفرية (Zero Trust) كنهج أمني حديث يقوم على التحقق المستمر وعدم منح أي ثقة افتراضية للمستخدمين أو الأجهزة.

في هذا المشروع، تم تصميم وتنفيذ نموذج أمني يعتمد على معمارية الثقة الصفرية لحماية الموارد الداخلية من الوصول غير المصرح به. يركّز النموذج على مصادقة المستخدمين والتحقق المستمر من الهوية قبل منح أي صلاحية وصول، مع تطبيق سياسات دقيقة تعتمد على الأدوار واستخدام قناة اتصال آمنة بين المستخدمين والموارد المستهدفة.

تم استخدام أدوات مفتوحة المصدر لبناء البيئة الافتراضية وتنفيذ المكونات الأساسية، بما في ذلك إدارة الهوية، بوابة الوصول، ونفق الاتصال الآمن. يقدّم هذا المشروع نموذجاً عملياً يمكن تطويره مستقبلاً ليشكّل أساساً لبنية تحتية شبكية أكثر أماناً ومرونة في البيئات التعليمية أو الإنتاجية.

الكلمات المفتاحية:

الثقة الصفرية، إدارة الهوية والوصول، المصادقة متعددة العوامل، الوصول المعتمد على الأدوار، بوابة الوصول، الأنفاق الآمنة، WireGuard، Keycloak، الأمن الشبكي.

Abstract

With the rise of cyber threats and the increasing complexity of modern network infrastructures, traditional perimeter-based security approaches are no longer sufficient to ensure the protection of internal resources. As a response, the Zero Trust paradigm has emerged, emphasizing continuous verification and the elimination of implicit trust for all users and devices.

This project presents the design and implementation of a security model based on Zero Trust principles to protect internal resources from unauthorized access. The model focuses on user authentication, continuous identity verification, role-based access control, and secure communication channels between users and protected resources.

Open-source tools were utilized to create a virtualized environment and implement the core components of the system, including identity management, an access gateway, and encrypted network tunnels. This project provides a practical foundation that can be further extended to build more secure and flexible network infrastructures in both educational and production settings.

Keywords:

Zero Trust, Identity and Access Management, Multi-Factor Authentication, Role-Based Access Control, Access Gateway, Secure Tunnels, WireGuard, Keycloak, Network Security.

المحتويات

Abstract 4
المحتويات 5
قائمة الاشكال 9
قائمة الجداول 11
قائمة بأهم الاختصارات 12
الفصل الأول 13
1. تعريف بالمشروع 13
1.1 الهدف من المشروع 13
2.1 خطوات العمل 13
3.1 المتطلبات الوظيفة 13
4.1 المتطلبات الغير وظيفية 14
الفصل الثاني 15
2. الدراسة المرجعية 15
1.2 مقدمة 15
2.2 الشبكات الخاصة الافتراضية VPN 15
1.2.2 أوجه القصور الأمنية 16
3.2 مبادئ الثقة-الصفرية 17
4.2 المكونات المنطقية لمعمارية الثقة-الصفرية 18
5.2 الوصول الشبكي القائم على الثقة الصفرية (Zero Trust Network Access -ZTNA) 19
1.5.2 المفاهيم الأساسية لنموذج ZTNA 19
2.5.2 المكونات الأساسية لمعمارية ZTNA 22
3.5.2 معماريات وتنفيذ نموذج ZTNA 26
6.2 الخاتمة 28
الفصل الثالث 29
3. تصميم النموذج الأمني المقترح 29
1.3 مقدمة 29
2.3 مكونات النموذج المقترح 29
1.2.3 جهاز المستخدم(User Device) 29

2.2.3 نظام المصادقة(Authentication System) 29
3.2.3 بوابة التحكم بالوصول(Access Gateway) 29
4.2.3 محرّك السياسات(Policy Engine) 29
5.2.3 مزود النفق الآمن(Secure Tunnel Provider) 30
6.2.3 المورد المستهدف مع الجدار الناري (Target Resource with Firewall)..... 30
3.3 تسلسل عمل النظام المقترح (Workflow Description)..... 30
1.3.3 مصادقة هوية المستخدم 30
2.3.3 إصدار الرمز وبدء الجلسة 30
3.3.3 تقييم طلب الوصول 30
4.3.3 تقييم السياسات 30
5.3.3 إنشاء جلسة اتصال آمنة 31
6.3.3 توجيه الموارد بناءً على الدور..... 31
7.3.3 إنفاذ الوصول على مستوى المورد..... 31
8.3.3 مراقبة الجلسة والمصادقة المستمرة 31
4.3 أهداف النموذج التصميمية والأمنية 32
1.4.3 الأهداف التصميمية 32
2.4.3 الأهداف الأمنية 33
5.3 اعتبارات التصميم وتخفيف التهديدات..... 33
6.3 الخاتمة..... 34
الفصل الرابع 35
4. بيئة العمل والأدوات المستخدمة..... 35
1.4 المقدمة..... 35
2.4 Keycloak- إدارة الهوية والتحكم بالوصول 35
1.2.4 نظرة عامة 35
2.2.4 دور Keycloak في نموذج ZTNA 35
3.2.4 مبررات اختيار Keycloak 36
3.4 WireGuard- بروتوكول VPN حديث..... 36
1.3.4 نظرة عامة 36
2.3.4 دور WireGuard في نموذج ZTNA..... 36
3.3.4 مبررات اختيار WireGuard..... 36
4.4 Flask web application framework- بوابة الوصول 37

1.4.4 نظرة عامة 37
2.4.4 دور Flask في النموذج ZTNA..... 37
3.4.4 مبررات استخدام Flask..... 38
5.4 VMware- بيئة العمل الافتراضية 38
1.5.4 نظرة عامة 38
2.5.4 دور VMware في نموذج ZTNA 38
3.5.4 مبررات اختيار VMware..... 38
6.4 Google Authenticator – مزود المصادقة متعددة العوامل MFA 39
1.6.4 نظرة عامة 39
2.6.4 دور Google Authenticator في نموذج ZTNA..... 39
3.6.4 مبررات اختيارGoogle Authenticator 39
الفصل الخامس 41
5. تنفيذ واختبار النموذج الأمني 41
1.5 المقدمة..... 41
2.5 إعداد البيئة..... 41
1.2.5 الجهاز الفيزيائي..... 41
2.2.5 الآلة الافتراضية التي تعمل عليها أداة Keycloak..... 41
3.2.5 الآلة الافتراضية التي تعمل عليها الأداتين Flask وWireGuard 41
4.2.5 الآلة الافتراضية التي تمثل الموارد..... 42
3.5 معمارية الشبكة 42
4.5 تكوين النظام..... 43
1.4.5 تكوين الأداة Keycloak 43
2.4.5 تكوين الأداة – WireGuard..... 52
3.4.5 تكوين تطبيق Flask – (Access Gateway Application)..... 53
4.4.5 تطبيق السياسات على مستوى المورد (Enforcement Point / Resource Firewall)..... 65
5.5 سيناريوهات اختبار النظام 66
1.5.5 تسجيل الدخول بصفة مسؤول النظام -Admin 66
2.5.5 تسجيل الدخول بصفة مطوّر-Developer 70
3.5.5 محاولة تجاوز Flask باستخدام WireGuard فقط 72
4.5.5 ملخص حالات الاختبار المنفذة 73
6.5 النتائج والتحليلات 73

الفصل السادس 75
6. الخاتمة والآفاق المستقبلية 75
1.6 الخاتمة 75
2.6 المهارات المكتسبة 75
3.6 الآفاق المستقبلية 75
7. المراجع 77

قائمة الاشكال

الشكل 1 مخطط صندوقي يوضح خطوات العمل 13
الشكل 2 آلية عمل الشبكات الخاصة الافتراضية (VPN).[1] 16
الشكل 3 وصول صفري الثقة.[2] 18
الشكل 4 المكونات البنيوية لمعمارية الثقة الصفرية.[2] 18
الشكل 5 مخطط تدفق عملية طلب الوصول في معمارية الثقة-الصفرية 19
الشكل 6 مخطط التوضيحي لآلية عمل ZTNA.[1] 22
الشكل 7 النموذج الأمني المقترح 32
الشكل 8 طوبولوجي بيئة الاختبار 42
الشكل 9 تشغيل الأداة Keycloak 43
الشكل 10 تسجيل الدخول لمسؤول الأداة Keycloak 43
الشكل 11 واجهة Keycloak بعد تسجيل الدخول من قبل مسؤول الأداة 44
الشكل12 إنشاء نطاق باسم Company 45
الشكل 13 إنشاء مستخدم في Keycloak 45
الشكل 14تعيين كلمة المرور للمستخدم admin1 46
الشكل 15إضافة دور المسؤول 47
الشكل 16إضافة دور المطور 47
الشكل 17إعطاء المستخدم admin1 دور admin 48
الشكل 18إضافة دور الضيف 48
الشكل 19إعدادات العميل VPN-Access-Client 49
الشكل 20إعدادات العميل VPN-Access-Client (Client Secret) 49
الشكل 21ضبط خصائص الرموز Tokens 50
الشكل 22تفعيل استخدام Refresh Tokens 51
الشكل 23الأقسام الرئيسية لـ JWT 51
الشكل 24ضبط إعدادات كلمة المرور لمرة واحدة TOTP 52
الشكل 25توليد زوج مفاتيح التشفير وفك التشفير لمخدم wireguard 52
الشكل 26ملف إعداد المخدم wg0.conf 53
الشكل 27تفعيل واجهة الشبكة الافتراضية wg0 53
الشكل 28تهيئة المتغيرات الأساسية لتكامل عمل النظام 54
الشكل 29التابع المسؤول عن إعادة التوجيه إلى keycloak 55
الشكل 30 استقبال رمز التفويض والية الحصول على الرموز المميزة 56
الشكل 31استخلاص الهوية وتخزين الجلسة 56
الشكل 32آلية التحقق من وجود الجلسة والرموز 57
الشكل 33آلية التحقق من مهلة الجلسة وقطع الاتصال فور انتهائها 57
الشكل 34التحقق من صلاحية الرموز 58
الشكل 35تقييم صلاحية الرمز Access Token 58
الشكل 36آلية طلب الحصول على الرمز المحدث Refresh Token 59
الشكل 37توليد ملف إعدادات الVPN 59

الشكل 38محتوى ملف الإعدادات conf. 60
الشكل 39توليد مفاتيح التشفير الخاصة والعامة للقرين..... 60
الشكل 40آلية تخصيص عناوين IP للمستخدمين 61
الشكل 41تخزين بيانات المستخدم في ملف peer.json..... 61
الشكل 42 تفعيل الPeer وإضافته على واجهة WireGuard..... 61
الشكل 43 إدارة دورة حياة القرين وتعطيله بعد انتهاء الجلسة..... 62
الشكل 44 التابع المسؤول عن تعطيل القرين..... 62
الشكل 45 آلية تسجيل الخروج 63
الشكل 46 مخطط عملية إدارة الجلسة والاتصال في بوابة الوصول 65
الشكل47 قواعد الجدار الناري المطبقة على الموارد..... 66
الشكل 48 طلب تسجيل الدخول بصفة مسوؤل 66
الشكل 49 طلب ادخال كلمة المرور لمرة واحدة 67
الشكل50 صفحة الويب الترحيبية بعد نجاح المصادقة للمستخدم admin1 67
الشكل51 ملف الاعدادات الخاص بالمستخدم admin1 68
الشكل 52 تفعيل ملف اعدادات WireGuard الخاص بالمستخدم admin1 على جهاز المستخدم 68
الشكل 53 محتويات ملف peer.json بعد تسجيل الدخول كمسؤول 69
الشكل 54 نجاح اختبار الوصول للموارد على المنفذ رقم 22 كمسؤول 69
الشكل 55 عرض مدة صلاحية ملف الاعدادات في صفحة الويب 69
الشكل 56 محتويات ملف peer.json بعد انتهاء صلاحية ملف الاعدادات..... 70
الشكل 57 عرض رسالة انتهاء صلاحية ملف الاعدادات وامكانية تحميل ملف اعدادات جديد..... 70
الشكل 58فشل اختبار الوصول للموارد على المنفذ رقم 22 كمسؤول..... 70
الشكل 59صفحة الويب الترحيبية بعد تسجيل الدخول كمطوّر 71
الشكل 60تفعيل ملف اعدادات WireGuard الخاص بالمستخدم dev1 على جهاز المستخدم..... 71
الشكل 61فشل اختبار الوصول للموارد على المنفذ رقم 22 كمطوّر 71
الشكل 62 نجاح اختبار الوصول للموارد على المنفذ رقم 80 كمطوّر..... 72

قائمة الجداول

جدول 1 مقارنة شاملة بين VPN و ZTNA.[1] 17
جدول 2 وظائف مزود الهوية IdP 23
جدول 3 أنواع السياسات الشائعة في ZTNA 24
جدول 4 أمثلة على سيناريوهات تنجيز نقطة التنفيذ 26
جدول 5 أنواع التهديدات وسبل التخفيف منها 34
جدول 6 مقارنة بين WireGuard و OpenVPN 37
جدول 7 متطلبات التصميم التي يحققها Flask 38
جدول 8 مقارنة بين VirtualBox و VMware 39
جدول 9 توصيف الآلة الفيزيائية المنصب عليها الآلات الافتراضية 41
جدول 10 توصيف الآلة التي تعمل عليها الأداة Keycloak 41
جدول 11 توصيف الآلة الافتراضية التي تعمل عليها الأداتين Flaskو WireGuard 42
جدول 12 توصيف الآلة الافتراضية التي تمثل الموارد 42
جدول13 أهم التوابع والمسارات ووظائفها لبوابة الوصول 64

قائمة بأهم الاختصارات

ZTNA	Zero Trust Network Access	الوصول الشبكي صفري الثقة
ZTA	Zero Trust Architecture	معمارية ثقة صفرية
IdP	Identity Provider	مزود الهوية
RBAC	Role-Based Access Control	التحكم بالوصول المعتمد على الأدوار
ABAC	Attribute-Based Access Control	التحكم بالوصول المعتمد على الواصفات
PBAC	Policy-Based Access Control	التحكم بالوصول المعتمد على السياسات
IAM	Identity and Access Management	إدارة الهوية والوصول
MFA	Multi-Factor Authentication	مصادقة متعددة العوامل
TOTP	Time-based One-Time Password	كلمة مرور لمرة واحدة المعتمدة على الزمن
VPN	Virtual Private Network	الشبكة الخاصة الافتراضية
PEP	Policy Enforcement Point	نقطة تنفيذ السياسة
PDP	Policy Decision Point	نقطة إقرار السياسة
ACL	Access Control List	قائمة التحكم بالوصول
JWT	JSON Web Token	-

الفصل الأول

1. تعريف بالمشروع

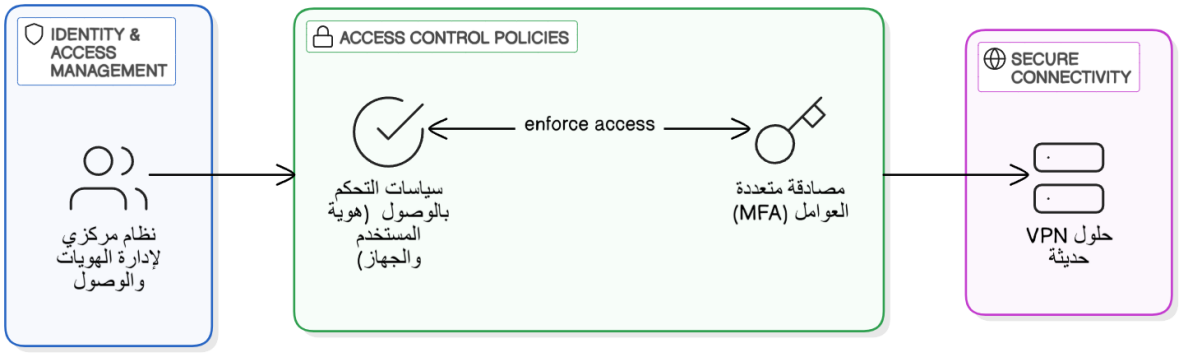
نتعرف في هذا الفصل على الهدف من المشروع مع توضيح خطوات العمل بمخطط صندوقي مفصل لها

1.1 الهدف من المشروع

يهدف المشروع إلى تصميم وتنفيذ معمارية أمنية تعتمد على مبدأ "الثقة-الصفرية (Zero Trust)" لمؤسسة افتراضية صغيرة باستخدام أدوات مفتوحة المصدر، واختبار المعمارية ضد الهجمات الشائعة.

2.1 خطوات العمل

يتضمن المشروع بناء بيئة افتراضية محلية لتطبيق معمارية الثقة الصفرية، تثبيت وتكوين نظام إدارة الهويات، تفعيل المصادقة متعددة العوامل، تصميم حل VPN وربطه بنظام إدارة الهويات، وأخيرا اختبار سياسات الوصول والاتصال الامن.
الشكل (1) يوضح خطوات العمل:



الشكل 1 مخطط صندوقي يوضح خطوات العمل

3.1 المتطلبات الوظيفة

- إعداد بيئة افتراضية لإنشاء مختبر محلي يحاكي بيئة مؤسسة بسيطة.
- تصميم وتنفيذ معمارية الثقة الصفرية (ZTNA) لحماية موارد المؤسسة.
- تطبيق المصادقة متعددة العوامل (MFA) لضمان تحقق قوي من هوية المستخدمين.
- تطبيق سياسة الوصول المعتمدة على الأدوار(RBAC).
- تأمين الاتصالات باستخدام بروتوكولات VPN.
- اختبار المعمارية في مواجهة وصولات غير مشروعة وهجمات أمنية محتملة.

4.1 المتطلبات الغير وظيفية

- ❖ الوثوقية (Reliability)
  - يجب أن يستمر النظام في العمل دون انقطاع حتى في حال حدوث محاولات وصول غير شرعية.
- ❖ قابلية التوسع (Scalability)
  - يجب أن يسمح النظام بإضافة مستخدمين أو أجهزة جديدة بسهولة دون إعادة هيكلة المعمارية.
- ❖ قابلية الإدارة (Manageability)
  - توفر واجهة لإدارة السياسات والمستخدمين والمراقبة الأمنية بسهولة.
- ❖ الأداء (Performance)
  - يجب أن يكون وقت الاستجابة لطلبات الوصول مقبولاً.
  - لا يجب أن تؤثر آليات التحقق والمراقبة سلباً على سرعة الوصول للخدمات.

الفصل الثاني

2. الدراسة المرجعية

سيتم في هذا الفصل تناول أدبيات الدراسة والمفاهيم النظرية

1.2 مقدمة

لقد تطورت البنى التحتية للمؤسسات والشركات بشكل أصبحت أكثر تعقيداً، حيث يمكن أن تعتمد المؤسسة الواحدة على عدة شبكات داخلية، مكاتب تابعة للمؤسسة مزودة ببنى تحتية محلية عن بعد، موظفين يعملون عن بعد أو متنقلين، أو خدمات سحابية متعددة. أدى هذا التعقيد إلى تجاوز حدود أنظمة الحماية التقليدية القائمة على مفهوم المحيط الشبكي Perimeter-Based Security التي تستند في عملها على استخدام الجدران النارية، الشبكات الخاصة الافتراضية VPNs والمناطق منزوعة السلاح DMZs لتأمين المحيط الخارجي للشبكة ضد التهديدات، حيث لم يعد للمؤسسة حدود واضحة وسهلة التحديد. أثبتت هذه الطرق التقليدية عدم كفاءتها[2]، إذ أن اختراق الحدود الخارجية للشبكة يسمح للمهاجمين بالانتقال غير المصرح به داخل الشبكة دون عوائق Lateral Movement وبالتالي لم تعد حدود الأمان التقليدية كافية لحماية البنى التحتية المعقدة والموزعة في عصرنا الحديث.

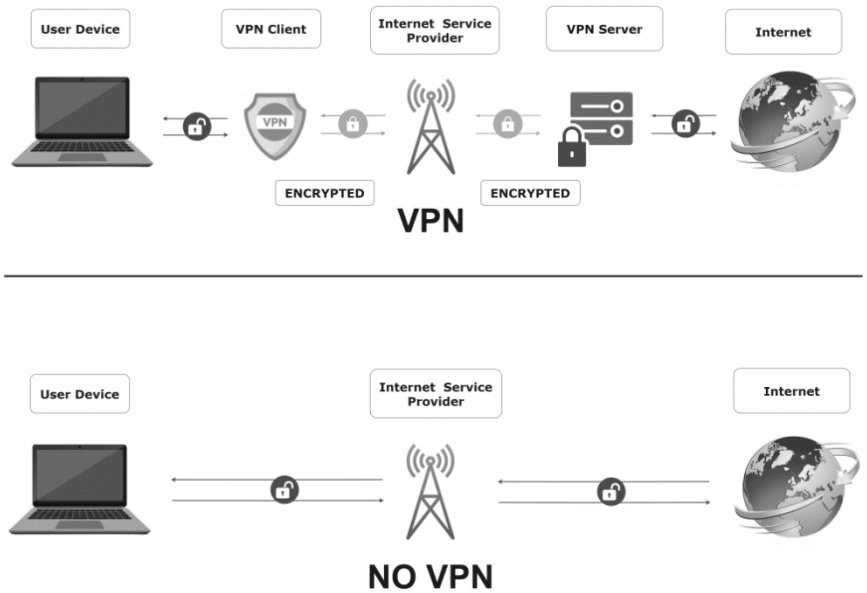
أدى هذا التعقيد في البنى التحتية للمؤسسات الى ظهور نموذج جديد في الأمن السيبراني يعرف باسم الثقة-الصفرية Zero-Trust. يركز نهج الثقة الصفرية بشكل أساسي على حماية البيانات والخدمات، لكنه يمكن -ويجب -أن يمتد ليشمل جميع أصول المؤسسة (مثل الأجهزة، مكونات البنية التحتية، التطبيقات، المكونات الافتراضية والسحابية) والمستخدمين (مثل المستخدم النهائي، التطبيقات والإجرائيات التي تطلب الوصول للموارد). تفترض نماذج الأمان القائمة على نهج الثقة-الصفرية أن المهاجم قد يكون موجود بالفعل داخل البيئة الداخلية، وأن البيئة التي تملكها المؤسسة ليست موثوقة أكثر من أي بيئة خارجية. في هذا النموذج الجديد، لا تفترض المؤسسة أي ثقة ضمنية، بل تقوم بشكل مستمر بتحليل وتقييم المخاطر التي تهدد أصولها ووظائفها التشغيلية وتنفيذ إجراءات حماية للتقليل من هذه المخاطر. عادة ما تتضمن إجراءات الحماية في إطار الثقة-الصفرية تقييد الوصول للموارد (مثل البيانات، وموارد الحوسبة، والتطبيقات/الخدمات) بحيث يقتصر فقط على المستخدمين أو التطبيقات أو الإجرائيات التي تحتاج إليها، والمصادقة والتخويل المستمرين لهوية وصحة كل طلب وصول قبل منحه. وبذلك يصبح الأمان قائما من خلال التحقق الدائم بدلا من افتراض الثقة بشكل مسبق.

2.2 الشبكات الخاصة الافتراضية VPN

الشبكة الخاصة الافتراضية تقنية تقوم بإنشاء اتصال آمن ومشفر عبر شبكة أقل أماناً، مثل شبكة الإنترنت، حيث تتيح للمستخدمين إرسال واستقبال البيانات عبر الشبكات العامة أو المشتركة كما لو أن أجهزتهم الحاسوبية متصلة مباشرةً بالشبكة الخاصة. تستخدم الشبكات الخاصة الافتراضية (VPNs) بشكل شائع من قبل الشركات للسماح للموظفين بالوصول الآمن إلى الشبكة الداخلية للمؤسسة عن بعد، حيث تمكّن الموظفين من الوصول إلى الملفات، والتطبيقات، والموارد الأخرى كما لو كانوا يعملون من داخل مكاتبهم. توفر شبكات VPN اتصالاً آمناً يحمي البيانات من المهاجمين، خاصةً عند استخدام شبكات الـ Wi-Fi العامة. وتضمن تقنيات التشفير أنه حتى لو تم اعتراض البيانات، فلن يكون بالإمكان قراءتها؛ ومن خلال إخفاء عنوان بروتوكول الإنترنت (IP)، تساهم شبكات VPN في حماية الخصوصية على الإنترنت. فهي تمنع المواقع الإلكترونية، والمعلنين، وحتى مزودي خدمة الإنترنت (ISP) من تتبع أنشطة المستخدمين على الشبكة. تتيح شبكات VPN تجاوز القيود الجغرافية والوصول إلى المحتوى المحجوب في بعض المناطق، مثل خدمات البث والمواقع الإلكترونية والخدمات الرقمية التي تتوفر فقط في دول معينة[1].

يوضح الشكل (2) آلية عمل الشبكات الخاصة الافتراضية (VPN)، حيث يبيّن أنه عند اتصال شخص ما بشبكة VPN، يتم تشفير حركة البيانات عبر الإنترنت. هذا يعني أن البيانات المرسلة والمستقبلة تتحول إلى رموز آمنة يصعب على الأطراف غير المخولة فك تشفيرها، مما يضمن حماية المعلومات الحساسة مثل كلمات المرور والبيانات الشخصية من التنصت. يتم تمرير حركة الإنترنت عبر خادم VPN. يقوم هذا الخادم بدور الوسيط بين جهاز المستخدم والإنترنت. فعند محاولة الوصول

إلى موقع إلكتروني أو خدمة عبر الإنترنت، يتم أولاً إرسال الطلب إلى خادم VPN، والذي يقوم بعد ذلك بإعادة توجيهه إلى الوجهة المطلوبة؛ وتعاد الاستجابة من الموقع عبر خادم VPN إلى جهاز المستخدم. ومن الجدير بالذكر أنه من خلال تمرير حركة المرور عبر خادم VPN، يتم إخفاء عنوان IP الحقيقي للمستخدم، ليظهر وكأنه يتصفح الإنترنت من موقع الخادم نفسه، مما يساعد على حماية الهوية والموقع الجغرافي ويوفر طبقة إضافية من الخصوصية.



الشكل 2 آلية عمل الشبكات الخاصة الافتراضية (VPN).[1]

1.2.2 أوجه القصور الأمنية

إن تقنيات VPN التقليدية تمنح المستخدمين، بمجرد المصادقة، وصولاً واسع النطاق إلى الموارد الداخلية، مما يزيد من خطر حركة المهاجم داخل الشبكة. علاوةً على ذلك، تعاني الشبكات الخاصة الافتراضية التقليدية من مشكلات الأداء وعدم التزامن وعدم قابلية التوسع مع زيادة أعداد المستخدمين والاتصالات، مما يؤكد الحاجة إلى نماذج أمنية أكثر تطوراً مثل الوصول الشبكي القائم على الثقة الصفرية (ZTNA). لتوضيح الفروقات الجوهرية بين VPN التقليدي وZTNA، يعرض الجدول (1) المقارنة التالية بين خصائص كل منهما:

البند	ZTNA	VPN
ميزات الأمان	يوفر إعدادات تحكم بالوصول دقيقة مع تقسيم دقيق وثقة تكيفية، مما يقلل من الحركة الجانبية داخل الشبكة	ينشئ نفقاً مشفراً لنقل البيانات بين جهاز المستخدم وشبكة الشركة، لكنه قد يكون عرضة للهجمات عند سوء التهيئة أو استخدام تشفير قديم
نموذج الثقة	يعتمد على نموذج "الثقة الصفرية"، حيث يتم التحقق المستمر من الهوية وصلاحيات الوصول	يتم تأسيس الثقة مرة واحدة عند الاتصال، وبعدها يحصل المستخدم على وصول كامل لكل الموارد

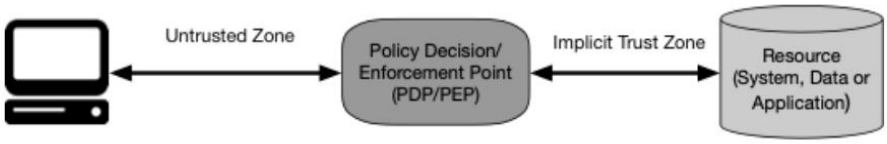
نموذج أمان الوصول	يتم الوصول فقط إلى تطبيقات أو بيانات محددة حسب سياسات دقيقة، معتمداً على هوية المستخدم وحالة الجهاز وحساسية التطبيق	بعد المصادقة، يحصل المستخدمون على وصول واسع، مما يزيد الخطر في حال اختراق بيانات الاعتماد
الأداء	يتم توجيه الحركة مباشرة إلى التطبيق أو الخدمة، مما يقلل التأخير ويتجنب نقاط الاختناق، مع تحسين تجربة المستخدم عبر الخروج المحلي	قد يحدث تأخير بسبب توجيه كل حركة المرور عبر خادم مركزي، مع تدهور الأداء مع المسافة والحمولة العالية
المصادقة	يفرض مصادقة قوية تشمل المصادقة متعددة العوامل، والتحقق من هوية الجهاز، والعوامل السياقية مثل الموقع والوقت	يستخدم طرقاً أساسية مثل اسم المستخدم وكلمة المرور؛ المصادقة متعددة العوامل اختيارية
تعقيد النشر	أكثر تعقيداً، يتطلب تكامل مع مزود الهوية، تعريف سياسات دقيقة، وضمان التوافق مع التطبيقات والبنية التحتية	تثبيت برامج VPN بسيط نسبياً، يتطلب تهيئة خوادم العميل على الأجهزة
القابلية للتوسع	مصمم لدعم التوسع بسهولة، دون الاعتماد على نقطة مركزية، ويدعم التكامل السلس مع الخدمات السحابية	صعب التوسع، لأن كل حركة المرور تمر عبر خوادم VPN مما يتطلب بنية تحتية كبيرة مع زيادة عدد المستخدمين

جدول 1 مقارنة شاملة بين VPN و ZTNA.[1]

3.2 مبادئ الثقة-الصفرية

الثقة-الصفرية ليست ذات معمارية محددة بل هي مجموعة من المبادئ التوجيهية لسير العمل، تصميم النظام والعمليات التشغيلية، والتي يمكن استخدامها لتعزيز الوضع الأمني لأي مستوى من التصنيف أو الحساسية. الانتقال الى نموذج الثقة-الصفرية Zero Trust Architecture -ZTA يتعلق بكيفية تقييم المؤسسة للمخاطر في تحقيق أهدافها، ولا يمكن تحقيقها بمجرد استبدال التقنيات الحالية بشكل كامل. ومع ذلك، فإن العديد من المؤسسات لديها بالفعل بعض عناصر ZTA في بنيتها التحتية الحالية[2].

يركز هذا النموذج الأمني على حماية الموارد ويستند على مبدأ أساسي هو "لا تثق أبدا"، "تحقق دائما". تعتبر معمارية الثقة-الصفرية نهجا كاملا حيث تشمل الهوية (الأفراد والكيانات غير البشرية كالتطبيقات، بيانات الاعتماد، إدارة الوصول، العمليات التشغيلية، بيئات الاستضافة، البنية التحتية للاتصالات) . يجب أن يتركز الاهتمام الأولي على تقييد الوصول إلى الموارد فقط للأشخاص الذين يحتاجون إليها، ومنحهم الحد الأدنى من الصلاحيات اللازمة لأداء المهام المطلوبة. للتقليل من حالات عدم الثقة (التي لا يمكن التخلص منها بشكل نهائي) يتم التركيز على المصادقة Authentication والتخويل Authorizationوتقليص مناطق الثقة الضمنية مع الحفاظ على توفر الخدمة Availability وتقليل التأخير الزمني في اليات المصادقة. تصمم قواعد الوصول بأكبر قدر ممكن من التخصيص لفرض مبدأ الصلاحيات الأدنى Least Privileges لتنفيذ الإجراء المطلوب. يوضح الشكل (3) نموذج مجرد للوصول بحيث يتم طلب وصول الى مورد مؤسسي، يتم منح الوصول عبر مكونين أساسيين: نقطة قرار السياسة Policy Decision Point أو PDP اختصارا، ونقطة تنفيذ السياسة Policy Enforcement Point أو PEP اختصارا. حيث يجب على النظام التأكد من صحة هوية صاحب الطلب وشرعية الطلب. يقوم PDP/PEP باتخاذ القرار المناسب للسماح بالوصول إلى المورد. وهذا يعني أن الثقة الصفرية تنطبق على مجالين أساسيين:



الشكل 3 وصول صفري الثقة.[2]

أولا: المصادقة

- ما مستوى الثقة في هوية الكيان لهذا الطلب الفريد؟
- هل الجهاز المستخدم للطلب يتمتع بوضع أمني مناسب؟

ثانيا: التخويل

- هل يسمح بالوصول إلى المورد نظراً لمستوى الثقة في هوية الكيان؟
- هل هناك عوامل أخرى يجب مراعاتها وتغير مستوى الثقة؟ (مثل الوقت، موقع الكيان، حالته الأمنية)

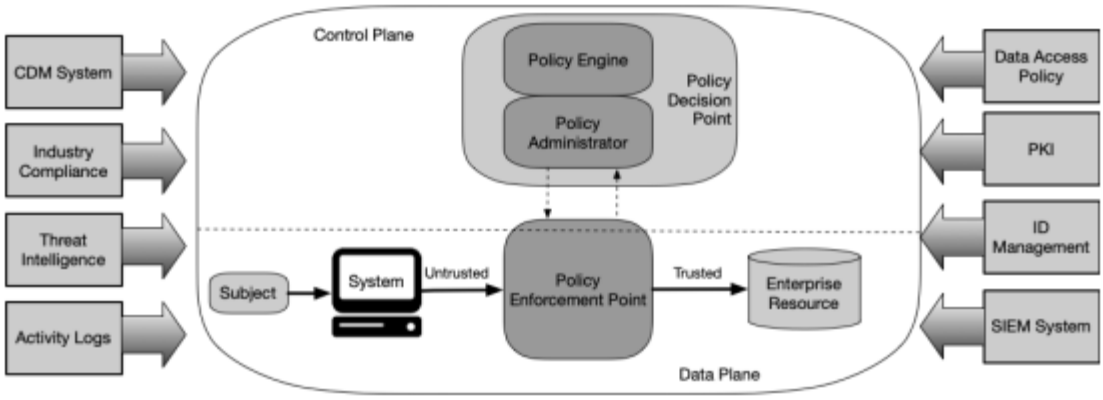
على المستوى الشامل، تحتاج المؤسسات إلى:

- تطوير سياسات ديناميكية قائمة على المخاطر للوصول إلى الموارد
- إعداد نظام لضمان التنفيذ الصحيح والمتسق لهذه السياسات لكل طلب وصول فردي

هذا يعني أنه لا ينبغي للمؤسسة الاعتماد على الثقة الضمنية، حيث إذا استوفى الكيان مستوى مصادقة أساسي (مثل تسجيل الدخول إلى أصل معين)، لا يفترض أن جميع طلبات الموارد اللاحقة صالحة بنفس الدرجة. بل يجب تقييم كل طلب على حدة[2].

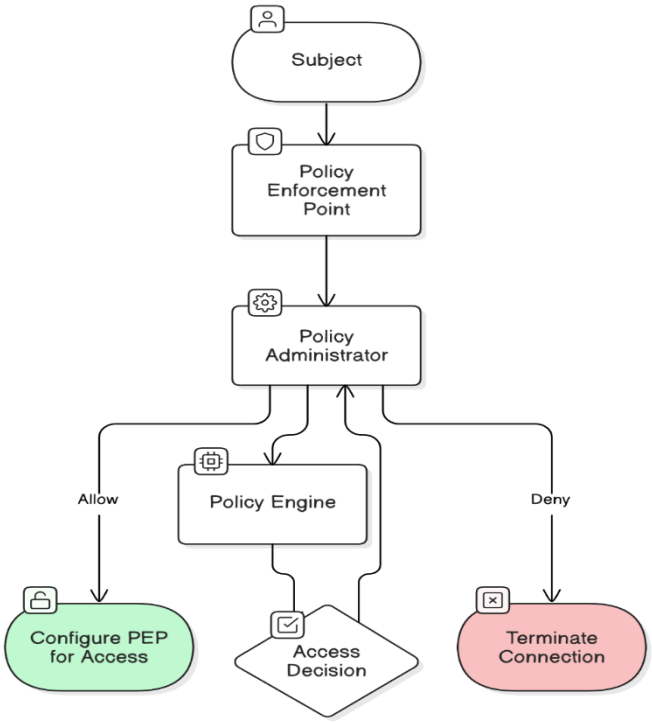
4.2 المكونات المنطقية لمعمارية الثقة-الصفرية

يوجد العديد من المكونات المنطقية التي تتكون منها معمارية الثقة-الصفرية في المؤسسات التي قد تعمل كخدمات محلية او عبر السحابة. يوضح النموذج الإطار المفاهيمي في الشكل (4) العلاقات الأساسية بين هذه المكونات واليات تفاعلها مع ملاحظة أن هذا النموذج مثالي يظهر المكونات المنطقية وعلاقاتها الوظيفية. من الشكل (4) نلاحظ انه تقسم نقطة قرار السياسة PDP الى مكونين رئيسيين هما محرك السياسة Policy Engine او اختصارا PE ومسؤول السياسة Policy Administrator او اختصارا PA.



الشكل 4 المكونات البنيوية لمعمارية الثقة الصفرية.[2]

- محرك السياسة (PE): هذا المكون مسؤول عن القرار النهائي بمنح أو رفض أو الغاء الوصول الى مورد معين لأي كيان. يعتمد في أخذ القرار على سياسة المؤسسة التي تأخذ المدخلات من مصادر خارجية مثل أنظمة التكوين المستمر Continues Diagnostics and Mitigation أو اختصارا CDM Systems وغيرها وتكون هي دخل لخوارزمية تقييم الثقة. يتم إقران الـ PE مع مكون مسؤول السياسة PA بحيث يتم اتخاذ القرار ويسجل (موافقة/رفض/الغاء) من قبل محرك السياسة وبعدها يتم تنفيذ القرار من قبل مسؤول السياسة.
- مسؤول السياسة (PA): يكون مسؤول عن إنشاء/إيقاف مسار الاتصال بين الكيان والمورد عن طريق أوامر تتعلق بـ PEPs وتوليد رموز مميزة tokens او بيانات اعتماد جلسة العمل الخاصة بالمصادقة والتخويل للعميل. وهي مرتبطة ارتباط وثيق بنقطة تنفيذ السياسة وتعتمد على قرارها بالسماح بنهاية المطاف بقبول الجلسة او رفضها. في حال الموافقة على الجلسة يقوم PA بتهيئة نقطة تنفيذ السياسة لبدء الاتصال. في حال الرفض او الغاء تصريح سابق يقوم PA بأمر نقطة تنفيذ السياسة بقطع الاتصال. قد يتم في بعض التطبيقات دمج المكونين PA/PE تحت اسم PDP.
- نقطة تنفيذ السياسة (PEP): تؤدي ثلاثة وظائف رئيسية هي تمكين الاتصال بين الكيان والمورد، مراقبة هذه الاتصالات، انهاؤها عند الحاجة. يوضح الشكل (5) مخطط تدفق عملية طلب الوصول في معمارية الثقة-الصفرية[2].



الشكل 5 مخطط تدفق عملية طلب الوصول في معمارية الثقة-الصفرية

5.2 الوصول الشبكي القائم على الثقة الصفرية (Zero Trust Network Access -ZTNA)

1.5.2 المفاهيم الأساسية لنموذج ZTNA

يعتمد نموذج ZTNA على مجموعة من المبادئ الأساسية التي تشكّل أساس منهجه في تأمين الشبكات. وقد استمدت هذه المبادئ من أفضل الممارسات الأمنية المتعارف عليها، لضمان أن الوصول إلى الموارد يتم التحكم فيه بشكل صارم ويراقب

باستمرار. ومن خلال الالتزام بهذه المبادئ، يوفر ZTNA إطاراً أمنياً قوياً يعالج التحديات الأمنية الحديثة، مثل التهديدات الداخلية، والتهديدات المستمرة المتقدمة(APT) ، وتعقيد الشبكات الموزعة[3] .

1.1.5.2 مبدأ الصلاحيات الأدنى Least Privilege

يعد مبدأ الصلاحيات الأدنى جوهرياً في نموذج ZTNA، إذ يضمن أن يحصل المستخدمون على الحد الأدنى فقط من الصلاحيات اللازمة لأداء مهامهم. هذا النهج يحد من الأضرار المحتملة في حال اختراق أحد الحسابات، حيث تكون الصلاحيات محصورة بدور المستخدم، مما يقلل من مساحة الهجوم. وفي النماذج الأمنية التقليدية، غالباً ما يكون لدى المستخدمين صلاحيات أوسع مما يلزم، وهو ما يعد ثغرة شائعة. من خلال تطبيق مبدأ الصلاحيات الأدنى، يساهم ZTNA في احتواء أي نشاط غير مصرح به ضمن نطاق ضيق، ما يعزز مرونة الشبكة. يطبق هذا المبدأ عملياً باستخدام التحكم في الوصول المعتمد على الأدوار Role-Based Access Control أو اختصارا (RBAC)، أو الواصفات Attribute-Based Access Control أو اختصارا (ABAC)، حيث تحدد الصلاحيات بناءً على وظائف العمل أو خصائص أخرى ذات صلة. ويسهم هذا النوع من الإدارة الدقيقة في تقليل مخاطر تسرب البيانات أو إساءة استخدامها داخلياً.

1.1.1.5.2 التحكم في الوصول المعتمد على الأدوار (RBAC)

هو أسلوب لإدارة وصول المستخدمين إلى الأنظمة أو الشبكات أو الموارد، استناداً إلى الدور الذي يشغله كل مستخدم ضمن فريق العمل أو المنظمة بشكل عام. وبدلاً من تكوين صلاحيات وصول مخصصة لكل مستخدم على حدة، يسمح نظام RBAC لمسؤولي تقنية المعلومات بتحديد مستوى الوصول المطلوب لجميع المستخدمين الذين يؤدون وظيفة معينة، ومن ثم تعيين دور محدد لهم يحتوي على مجموعة الصلاحيات المناسبة. يتيح ذلك لفرق تقنية المعلومات إضافة أو تعديل أو إزالة الصلاحيات لجميع المستخدمين في مجموعة معينة دفعة واحدة، أو تغيير مستوى وصول مستخدم فردي بسهولة عبر تعيين دور جديد له أو إزالة الدور الحالي.

في جوهره، يتبع كل نظام تحكم بالوصول قائم على الدور المبادئ الأساسية التالية:

- يعيّن لكل مستخدم دور أو أكثر.
- تحدد الصلاحيات على مستوى الأدوار.
- يحصل المستخدمون على الصلاحيات من خلال كونهم أعضاء فعّالين ضمن دور معين.

ويعد أحد أكثر أنواع التحكم بالوصول شيوعاً، حيث يوفر نهجاً أكثر دقة لإدارة الهوية والوصول Identity and Access Management أو اختصارا (IAM)، مقارنة بقوائم التحكم في الوصول Access-Control Lists أو اختصارا (ACLs)، لكنه يبقى أبسط وأسهل في التطبيق من نظام التحكم بناءً على (ABAC) وعلى الرغم من أن هناك أساليب أخرى لإدارة الهوية والوصول، فإن نظام RBAC يعتبر خياراً ملائماً لمعظم المؤسسات التي تبحث عن حل حكومي سهل الإدارة وقابل للتوسّع. في كثير من الحالات، تنشئ نماذج RBAC هيكلاً هرمياً للأدوار يعكس البنية التنظيمية للمؤسسة، وقد يشمل أدواراً للمسؤولين والمستخدمين النهائيين والزوار، وأي مجموعات متخصصة بين تلك الفئات. بعض الهياكل الهرمية قد تكون قائمة على الوراثة، حيث يتم منح الأدوار ذات المستوى الأعلى بشكل تلقائي الأدوار التي تقع دونها مع كافة الصلاحيات المرتبطة بها. في حالات أخرى، قد تكون البنية الهرمية اعتباطية، بحيث لا يرث المستخدم ذو الدور الأعلى بالضرورة صلاحيات الأدوار الأدنى.

يعتمد RBAC على مبدأ الصلاحيات الأدنى Principle of Least Privilege أو اختصارا (PoLP)، وهو أحد المبادئ الأساسية في نموذج الأمان القائم على الثقة الصفرية (Zero Trust) [5] .

2.1.5.2 المصادقة المستمرة Continuous Authentication

تختلف المصادقة المستمرة عن أساليب المصادقة التقليدية القائمة على التحقق لمرة واحدة في بداية الجلسة. إذ يعتمد ZTNA على عمليات تحقق متكيّفة ومستمرة، تقوم بالتحقق من هوية المستخدم ووضع أمان الجهاز طوال فترة الجلسة.

يتيح هذا النهج الديناميكي تعديل صلاحيات الوصول في الزمن الحقيقي، استناداً إلى عوامل مثل موقع المستخدم، وصحة الجهاز، وأنماط استخدام الشبكة. وتعد هذه المصادقة المستمرة أمراً حيوياً في مواجهة التهديدات الأمنية الحديثة، خصوصاً أن الاعتماد على كلمات مرور ثابتة لم يعد كافياً. وتستخدم تقنيات مثل المصادقة متعددة العوامل (MFA)، التحقق الحيوي، والتحليلات السلوكية لمراقبة الهوية والسلوك باستمرار؛ وفي حال اكتشاف نشاط مريب، يمكن لـ ZTNA إلغاء الوصول فوراً أو طلب خطوات تحقق إضافية، مما يقلل من خطر الوصول غير المصرح به ويخفف من التهديدات الداخلية.

3.1.5.2 التقسيم الدقيق (Microsegmentation)

يعتبر التقسيم الدقيق أحد المفاهيم المحورية في ZTNA، حيث يتم تقسيم الشبكة إلى مناطق صغيرة ومعزولة، تطبق عليها ضوابط أمنية مستقلة. يسهم هذا النهج في الحد من حركة المهاجم داخل الشبكة (Lateral Movement)، بحيث يتم احتواء الاختراق -إن حصل -ضمن نقطة واحدة فقط. ومن خلال فرض حدود صارمة بين أجزاء الشبكة، يقلل ZTNA من احتمال انتشار الهجمات، خصوصاً في البيئات الحساسة مثل القطاعات الصحية والمالية. يطبق ZTNA ضوابط وصول دقيقة على مستوى حجم العمل (Workload Level)، لضمان وصول المستخدمين والأجهزة فقط إلى الموارد التي يحتاجونها. ومن خلال عزل كل جزء والتحكم الصارم في التفاعل بين الأجزاء المختلفة، يتم تقليل مساحة الهجوم، ما يعزز حماية الأنظمة الحيوية من التهديدات الجانبية.

4.1.5.2 التحكم في الوصول القائم على السياسات (Policy-Based Access Control -PBAC)

يشكل التحكم في الوصول القائم على السياسات (PBAC) محوراً رئيسياً في نموذج ZTNA، حيث تتم إدارة عمليات الوصول إلى الموارد عبر سياسات أمنية محددة مسبقاً. تقوم هذه السياسات بتحديد من يسمح له بالوصول إلى موارد معينة، وتحت أي شروط، ومن خلال أي أجهزة. وعلى عكس النماذج التقليدية التي تعتمد على أذونات ثابتة، يتسم PBAC في ZTNA بالمرونة والديناميكية، حيث يتكيف مع البيانات الفعلية المتعلقة بالمستخدمين، والأجهزة، وبيئة الشبكة، لاتخاذ قرارات وصول فورية.

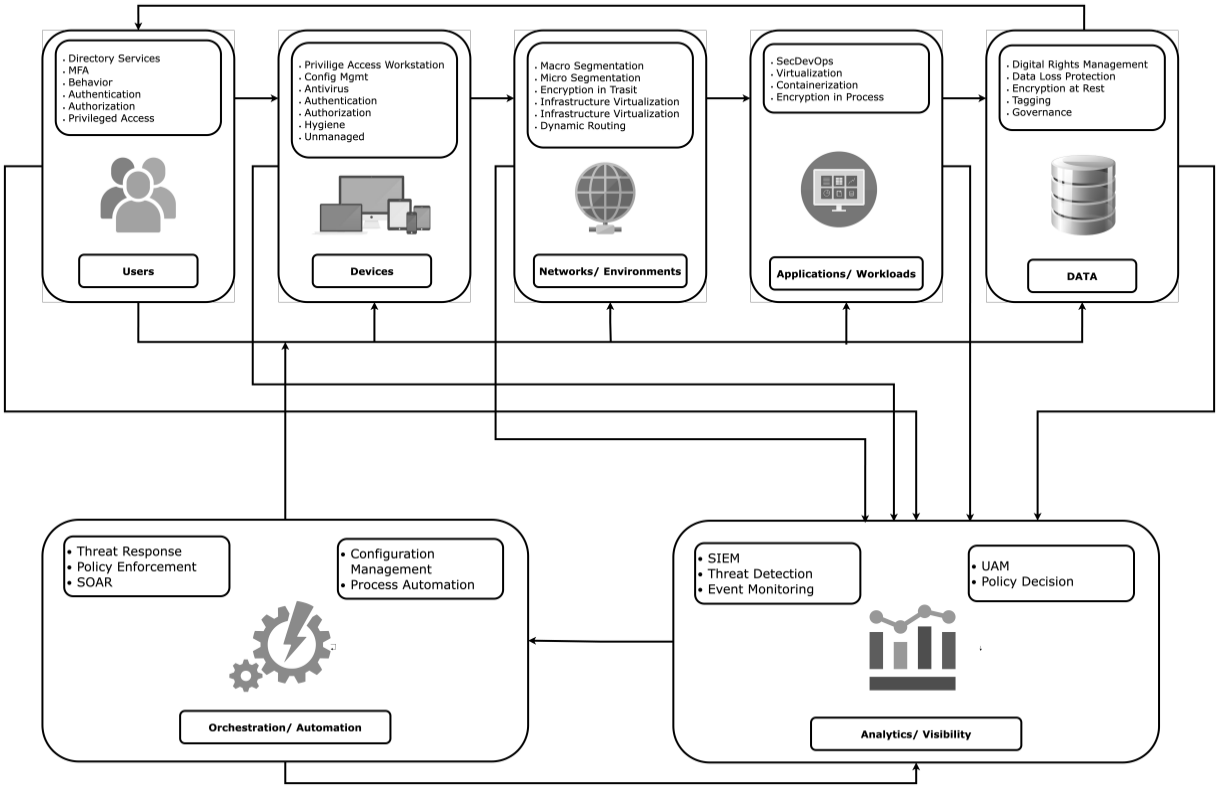
تتميز سياسات ZTNA بالقدرة على التخصيص العالي والدقة المتناهية، مما يسمح للمؤسسات بوضع قواعد محددة لكل مستخدم، أو مجموعة مستخدمين، أو أجهزة معينة. ويمكن أن تستند قرارات التحكم بالوصول إلى عوامل متعددة مثل توقيت الطلب، والموقع الجغرافي، وحالة أمان الجهاز. ومن خلال تحديث السياسات بشكل مستمر عبر الشبكة، يضمن ZTNA أن تبقى أذونات الوصول متوافقة مع متطلبات الأمان الآنية، مما يسمح بالتعامل الفعال مع الطبيعة الديناميكية للبيئات الشبكية الحديثة حيث يصل المستخدمون إلى الموارد من مواقع وأجهزة متنوعة.

5.1.5.2 التنفيذ الديناميكي للسياسات وتحديثها (Dynamic Policy Enforcement and Updates)

إلى جانب التحكم القائم على السياسات، يؤكد نموذج ZTNA على أهمية التنفيذ الديناميكي للسياسات، بحيث لا يتم تطبيق السياسات الأمنية فقط عند المصادقة الأولية، بل يتم تقييمها وتعديلها بشكل مستمر بناءً على الظروف الآنية. تتيح هذه القدرة التأكد من أن صلاحيات المستخدمين تظل مناسبة مع تطور السياق الأمني، مثل تغيّر حالة الجهاز، أو محاولة المستخدم القيام بأنشطة غير معتادة.

وتعد عملية تحديث السياسات بشكل ديناميكي ضرورية بشكل خاص لمواجهة التهديدات المستمرة المتقدمة Advanced Persistent Threats أو اختصارا (APTs)، حيث يسعى المهاجمون للحصول على وصول طويل الأمد إلى الشبكة وزيادة امتيازاتهم تدريجياً. ومن خلال التعديل المستمر لضوابط الوصول، يمنع ZTNA المهاجمين من استغلال الأذونات الثابتة، مما يقلل بشكل كبير من قدرتهم على تصعيد الامتيازات أو التنقل داخل الشبكة[3].

يوضح الشكل (6) التالي المفهوم العام لطريقة عمل نموذج الوصول الشبكي القائم على الثقة الصفرية (ZTNA)، حيث يتم مصادقة كل طلب بشكل مستقل بناءً على هوية المستخدم والجهاز، مع فرض سياسات وصول دقيقة بناءً على السياق



الشكل 6 مخطط توضيحي لآلية عمل ZTNA.[1]

الأمني.

2.5.2 المكونات الأساسية لمعمارية ZTNA

يعمل نظام الوصول إلى شبكة الثقة الصفرية (ZTNA) من خلال بنية متعددة الطبقات تفصل بين مسؤوليات الهوية والسياسة والتنفيذ حيث لا تحدد من خلال تقنية واحدة أو منتج معين من قبل جهة مصنّعة، بل تعتمد على مجموعة من المكونات المتكاملة التي تطبّق ضوابط وصول صارمة تستند إلى الهوية والسياق والسياسات. تعمل هذه المكونات على التحقق بشكل صريح، وتطبيق الحد الأدنى من الصلاحيات، وتقييم الثقة باستمرار قبل منح الوصول إلى أي مورد. يعرض هذا القسم المكونات الأساسية المشتركة بين معظم نماذج ZTNA:

- **مزود الهوية (Identity Provider - IdP)** مسؤول عن التحقق من هوية المستخدمين أو الكيانات.
- **محرك السياسات (Policy Engine)** يقيّم الطلبات بناءً على السياسات المعتمدة والسياق الأمني.
- **بوابة الوصول (Access Gateway)** تعمل كوسيط يتحكم في الوصول إلى الموارد المحمية.

- **نقطة التنفيذ (Enforcement Point)** تطبّق القرار المتخذ من قبل محرك السياسات وتفرضه على مستوى الشبكة أو التطبيق.

تنفّذ هذه المكونات مجتمعةً مبدأ "الثقة، ولكن تحقق" الذي يشكّل جوهر أطر العمل الخاصة بـ Zero Trust، مثل ما هو موضح في [2].

1.2.5.2 مزود الهوية (IdP)

يعد مزود الهوية (IdP) أحد المكونات المركزية في بنية ZTNA، إذ يتولى مسؤولية مصادقة المستخدمين والأجهزة، وإصدار رموز الهوية(Identity Tokens) ، وإدارة سمات المستخدم مثل الأدوار، والانتماءات للمجموعات، والصلاحيات. في إطار ZTNA، تعتبر المصادقة الخطوة الأولى والأكثر أهمية في عملية بناء الثقة. حيث تتلخص وظيفة هذا المكون في ثلاث نقاط هي أن:

- يمثّل نقطة الثقة الأولى التي تبنى عليها جميع قرارات الوصول.
- يضمن أن كل كيان—سواء أكان بشرياً أو آلياً—قد تم تمييزه وتوثيقه بشكل فريد قبل البدء في تقييم السياسات.
- يقوم بإصدار رموز الوصول (Access Tokens) وفق معيارOAuth2 ، ورموز الهوية **(ID Tokens)** باستخدام OpenID Connect (OIDC)، والتي تستخدم لاحقاً من قبل مكونات أخرى مثل محرك السياسات وبوابة الوصول.

الوظائف التي يتعامل معها IdP:

الوظيفة	الغرض
مصادقة المستخدم	التحقق من الهوية باستخدام كلمات المرور، أو المصادقة متعددة العوامل (MFA)، أو البيانات الحيوية(Biometrics)
التكامل مع مصادر خارجية	التكامل مع مصادر هوية خارجية مثل LDAP أوSAML
تسجيل الدخول لمرة واحدة (SSO)	تمكين المستخدم من تسجيل الدخول مرة واحدة للوصول إلى خدمات متعددة بأمان
إصدار الرموز(OIDC/OAuth2)	إصدار رموز آمنة تستخدم في التواصل مع واجهات برمجة التطبيقات (APIs)والجلسات
إثراء الواصفات (Attribute Enrichment)	إضافة بيانات وصفية إلى هوية المستخدم مثل القسم الوظيفي أو الدور المؤسسي

جدول 2 وظائف مزود الهوية IdP

2.2.5.2 محرك السياسات (Policy Engine)

يعد محرك السياسات المعالج المركزي في بنية الوصول إلى الشبكة صفري الثقة (ZTNA). يتولى مسؤولية تقييم ما إذا كان يجب السماح بالوصول إلى مورد معيّن، وذلك استناداً إلى الهوية، والسياق، ومجموعة من قواعد التحكم في الوصول المعرّفة مسبقاً. وعلى عكس أنظمة الأمان التقليدية التي تتخذ قرارات الوصول مرة واحدة فقط عند دخول المستخدم إلى النظام، فإن محرك السياسات في ZTNA يقيّم السياسات بشكل مستمر أو على الأقل عند كل طلب جديد أو تجديد للجلسة، مما يضمن حماية ديناميكية وقابلة للتكيف مع تغيّر الظروف.

الوظائف الأساسية لمحرك السياسات:

- تقييم السياسات(Policy Evaluation) : تفسير سياسات التحكم في الوصول التي تحدد من يمكنه الوصول إلى ماذا، وتحت أي شروط.
- اتخاذ قرارات مبنية على السياق (Context-Aware Decision Making):
  تحليل سياقات متعددة مثل:
  - الدور الوظيفي أو الانتماء إلى مجموعة وفق(RBAC)
  - الوقت أو الموقع الجغرافي
  - حالة التوافق الأمني للجهاز (Device Posture)
  - سلوك الجلسة الجارية ضمن المصادقة المستمرة(Continuous Authentication)
- التحقق من الرموز (Token Validation):
  التأكد من صحة وسريان الرموز المصدّرة من مزود الهوية (مثل رموز الوصول ورموز الهوية).
- ضوابط مبنية على مستوى الخطر (Risk-Based Controls):
  دعم قرارات وصول تكيفية مثل طلب المصادقة متعددة العوامل (MFA) أو منع الوصول عند ارتفاع مستوى الخطر.

أنواع السياسات الشائعة في ZTNA:

نوع السياسة	الوصف
التحكم في الوصول القائم على الدور (RBAC)	منح الوصول استناداً إلى الدور المعيّن للمستخدم.
التحكم في الوصول القائم على الواصفات(ABAC)	أخذ واصفات المستخدم أو الجهاز أو البيئة بعين الاعتبار.
التحكم في الوصول القائم على السياسات(PBAC)	دمج الأدوار والواصفات والسياق في سياسات مرنة ومعقدة.
تطبيق مبدأ الحد الأدنى من الامتيازات (Least Privilege)	ضمان منح المستخدمين الحد الأدنى فقط من الوصول اللازم لأداء مهامهم.

جدول 3 أنواع السياسات الشائعة في ZTNA

3.2.5.2 بوابة الوصول (Access Gateway)

تعد بوابة الوصول نقطة التنفيذ لقرارات السياسات وممر الدخول الوحيد الذي تمر من خلاله حركة مرور المستخدمين نحو الموارد المحمية. توضع هذه البوابة استراتيجياً بين المستخدمين (أو أجهزتهم) والخدمات الداخلية، لضمان عدم وصول أي حركة مرور إلى الأصول الحساسة دون تحقق مسبق من الهوية وتفويض مناسب.

الوظائف الأساسية لبوابة الوصول:

- **اعتراض حركة المرور (Traffic Interception)**
  التقاط طلبات الاتصال الواردة من المستخدمين أو الأجهزة التي تم التحقق من هويتها.
- **نقطة تنفيذ السياسات (Policy Enforcement Point -PEP)**
  تطبيق قرارات السماح أو الرفض الصادرة عن محرك السياسات.
- **توسيط القنوات الآمنة (Secure Channel Mediation)**
  إنشاء وصيانة أنفاق اتصال آمنة (مثل VPN أو mTLS) بين المستخدمين والموارد الداخلية.
- **إدارة الجلسات (Session Management)**
  تتبّع الجلسات النشطة وتطبيق إبطال الجلسة أو فرض إعادة المصادقة عند الحاجة، بما يدعم مفهوم المصادقة المستمرة.(Continuous Authentication)
- **التسجيل والمراقبة (Logging and Monitoring)**
  توفير الرؤية حول محاولات الوصول والقرارات المتخذة وسلوك المستخدم، لأغراض التدقيق واكتشاف السلوكيات غير الطبيعية.

موقع تنفيذ بوابة الوصول في معمارية ZTNA:

- يتم وضعها بعد عملية المصادقة (من قبل مزود الهوية -IdP) وقبل منح الوصول إلى الموارد.
- غالباً ما تدمج مع تقنيات مثل شبكات VPN، أو الخوادم العكسية(Reverse Proxies) ، أو بوابات ZTNA المخصصة.
- تعمل كنقطة دخول موحدة لجميع طلبات الوصول، مما يقلل من مساحة الهجوم (Attack Surface).

الفوائد الأمنية التي تقدمها بوابة الوصول:

- إلغاء إمكانية الوصول المباشر إلى الموارد، مما يحد من احتمالية الاختراق المباشر.
- فرض المصادقة في كل مرة قبل توجيه الطلبات إلى الموارد.
- دعم التحكم الدقيق في الوصول، مثل تطبيق سياسات مختلفة تبعاً لدور المستخدم أو نوع الجهاز.
- تسهيل تقسيم الشبكة إلى مقاطع صغيرة (Microsegmentation) من خلال توجيه حركة المرور فقط إلى الخدمات المحددة بناءً على السياسات.

4.2.5.2 نقطة التنفيذ (Enforcement Point)

يعرف أيضاً باسم نقطة تنفيذ الموارد(Resource Enforcement Point -REP) ، وهي الحاجز الأخير في بنية ZTNA، والمسؤولة عن تطبيق قرارات الوصول الصادرة عن النظام بشكل صارم. لا يتم منح الوصول إلى المورد المستهدف أو الخدمة إلا بعد إتمام جميع عمليات المصادقة والتفويض السابقة.

الوظائف الأساسية لنقطة التنفيذ

- **تطبيق الحكم النهائي لمحرك السياسات (Policy Engine)**
  تقوم بمنح أو رفض الوصول استناداً إلى قرارات السياسات التي تم تقييمها مسبقاً (وقد تكون مخزنة مؤقتاً لتحسين الأداء).
- **الحفاظ على قنوات الاتصال الآمن (Secure Communication Channels)**
  لا تستقبل سوى الطلبات القادمة عبر المسار الآمن الذي تم إنشاؤه من قبل بوابة الوصول (مثل نفق VPN أو اتصال TLS متبادل – mTLS).
- **لتحقق من الهوية على جانب المورد (Resource-Side Validation)**
  التحقق من هويات الأقران أو رموز الجلسة (إن وجدت) لمنع التزوير أو تجاوز آلية التحقق.

- **الاندماج مع التطبيقات أو الخدمات**
يمكن تنفيذها كجزء من التطبيق ذاته، أو كخادم عكسي وسيط(Reverse Proxy) ، أو مدمجة ضمن بنية البنية التحتية.

سيناريوهات التنفيذ:

نوع التقنية	سلوك التنفيذ
خوادم VPN	تقبل فقط مفاتيح الأقران المصادق عليها وتقيّد الوصول بناءً على عناوين IP المخصصة.
الخوادم العكسية(Reverse Proxies)	تفرض التحقق من بادئة الجلسة/الرمز (Token Headers) أو هويات TLS المتبادلة.
بوابات واجهات برمجة التطبيقات (API Gateways)	تتحقق من صلاحية رموز الوصول OAuth2 قبل توجيه الطلبات.
جدران الحماية على مستوى المضيف-Host) Based Firewalls)	تطبق قواعد جدار حماية ديناميكية بناءً على المستخدم أو الجلسة.

جدول 4 أمثلة على سيناريوهات تنجيز نقطة التنفيذ

الميزات الأمنية:

- **ضمان عدم الوصول غير المصرح به حتى في حال وجود أخطاء في الطبقات السابقة**
تعمل نقطة التنفيذ كخط الدفاع الأخير، مما يضمن أن المورد لا يمكن الوصول إليه إلا إذا تم السماح بذلك صراحةً، حتى في حال وجود خلل في تكوين بوابة الوصول أو محرك السياسات.
- **فرض العزل على مستوى الشبكة والتجزئة الدقيقة (Microsegmentation)**
يتم ربط كل مستخدم بعنوان IP أو نطاق فرعي (Subnet) محدد، مما يمنع الاتصال العشوائي ويحد من مساحة التعرض للهجمات.
- **منع الحركة الجانبية داخل الشبكة (Lateral Movement)**
يسمح للمستخدم فقط بالوصول إلى الخدمات المصرح بها خلال الجلسة أو حسب الدور(Role) ، مما يمنع المهاجم من التنقل بين الأنظمة أو الموارد الداخلية في حال حصوله على صلاحيات غير مشروعة[4] .

3.5.2 معماريات وتنفيذ نموذج ZTNA

تنقسم معماريات نموذج الوصول الشبكي القائم على انعدام الثقة (ZTNA) عادةً إلى فئتين رئيسيتين:

- ZTNA المعتمد على الوكيل (Agent-Based ZTNA)
- ZTNA غير المعتمد على الوكيل (Agentless ZTNA)

حيث يتم تصميم كل منهما لتلبية احتياجات أمنية وسياقية تشغيلية مختلفة. ولمعالجة متطلبات الشبكات الخاصة وبيئات النشر المختلفة، يمكن أن تتضمن هذه النماذج تطبيقات متخصصة، مثل:

- المحيط المعرف بالبرمجيات(Software-Defined Perimeter -SDP)
- ونموذج ZTNA القائم على الحوسبة السحابية.(Cloud-Based ZTNA)

تعتمد كلا المعماريتين على اتخاذ قرارات وصول ديناميكية قائمة على تقييم المخاطر، بهدف الحفاظ على أمان متواصل من خلال تعديل الأذونات بشكل لحظي بناءً على تغير تقييمات المخاطر مع مرور الوقت.

1.3.5.2 نموذج ZTNA المعتمد على الوكيل (Agent-Based ZTNA)

يتطلب نموذج ZTNA المعتمد على الوكيل تثبيت برنامج وكيل (Agent) على كل جهاز يحتاج إلى الوصول إلى الشبكة. يقوم هذا الوكيل بتطبيق السياسات الأمنية محلياً، ويراقب باستمرار صحة الجهاز وامتثاله للمعايير الأمنية، مما يتيح إدارة أمنية دقيقة تعتمد على خصائص محددة لكل جهاز، مثل:

- إصدار نظام التشغيل
- مستوى تحديث التصحيحات الأمنية
- إعدادات الجهاز

ورغم أن هذا النهج يوفر مستوى عالياً من التحكم والسيطرة، إلا أن إدارة الوكلاء عبر عدد كبير من الأجهزة قد تشكل تحدياً، خاصة في بيئات العمل التي تعتمد سياسة "استخدام الأجهزة الشخصية Bring Your Own Device أو اختصارا BYOD. ومع ذلك، يعد نموذج ZTNA المعتمد على الوكيل مناسباً جداً للبيئات المدارة مركزياً، حيث تدار الأجهزة بشكل موحد، مما يتيح تطبيق سياسات وصول دقيقة وضبط الامتثال الأمني بشكل ديناميكي وفقاً للحالة الفعلية للجهاز.

2.3.5.2 نموذج ZTNA غير المعتمد على الوكيل (Agentless ZTNA)

يعتمد نموذج ZTNA غير المعتمد على الوكيل على مكونات البنية التحتية الشبكية، مثل:

- بوابات الويب الآمنة(Secure Web Gateways)
- الوكلاء(Proxies)
- وجدران الحماية المستندة إلى الهوية(Identity-Aware Firewalls)

لتطبيق السياسات الأمنية دون الحاجة إلى تثبيت برامج وكيلة على الأجهزة النهائية. يعد هذا النموذج فعالاً بشكل خاص في البيئات التي تحتوي على أنواع متباينة من الأجهزة، مثل الأجهزة الطرفية منخفضة القدرة، حيث قد يكون تثبيت الوكلاء غير عملي أو غير ممكن. ورغم أن ZTNA غير المعتمد على الوكيل يفتقر إلى الدقة العالية في التعامل مع خصائص الجهاز الفردية مقارنة بالنموذج المعتمد على الوكيل، إلا أنه يوفر:

- سهولة أكبر في النشر
- وقابلية توسع أعلى

لذلك، يعتبر هذا النهج مثالياً للمؤسسات التي تعطي أولوية للتوافق الواسع والمرونة التشغيلية. ومع ذلك، فإن القدرة على المراقبة المستمرة لصحة الأجهزة تكون محدودة مقارنة بالنماذج التي تعتمد على وكيل مثبت محلياً.

3.3.5.2 تطبيقات متخصصة Specialized Implementations

1.3.3.5.2 المحيط المعرف بالبرمجيات (Software-Defined Perimeter -SDP)

يعزز نموذج SDP الأمان عبر إخفاء الموارد الشبكية عن المستخدمين غير المصرح لهم، ومنح حق الوصول فقط للكيانات المصادق عليها. يستخدم SDP مستوى التحكم Control Plane للمصادقة على المستخدمين والأجهزة، مما يقلل من حركة المهاجم داخل الشبكة (Lateral Movement) عبر تقسيم الوصول إلى الموارد. يتكوّن نموذج SDP من ثلاثة مكونات رئيسية:

- العميل (Initiating Client)
- نقطة قرار السياسة (PDP)
- نقطة تنفيذ السياسة (PEP)

يقوم العميل بالتواصل مع نقطة قرار السياسة (PDP) للتحقق من الهوية والتوافق، وبعدها تتولى نقطة تنفيذ السياسة (PEP)تنفيذ قرار السماح أو المنع بالوصول. يعد هذا الإطار الأمني القائم على الهوية والسياق قابلاً للتكيف بشكل كبير مع بيئات الحوسبة السحابية الهجينة والبيئات المحلية.

2.3.3.5.2 نموذج ZTNA القائم على الحوسبة السحابية(Cloud-Based ZTNA)

تم تصميم حلول ZTNA القائمة على الحوسبة السحابية لمعالجة التحديات الأمنية الفريدة المرتبطة بالبيئات السحابية، مثل:

- تعددية الإيجارات (Multi-Tenancy)
- والتخصيص الديناميكي للموارد

وعلى عكس النماذج التقليدية للأمان المحلي، يعتمد ZTNA السحابي على فرض ضوابط وصول متعددة الطبقات، تمتد من نقاط دخول الشبكة والأجهزة إلى طبقة التطبيقات، مما يضمن تغطية شاملة للأمان ضمن بنى الحوسبة السحابية.

يقوم هذا النموذج بتقييم الوصول بشكل مستمر بناءً على:

- هوية المستخدم
- حالة الجهاز
- أنماط السلوك

ويعد هذا النهج الديناميكي ضرورياً في البيئات السحابية حيث يتم الوصول إلى الموارد من مواقع متعددة، وهو مثالي للبنى التحتية السحابية المختلطة والمتعددة، مما يضمن اتساق السياسات الأمنية عبر مختلف مقدمي الخدمات السحابية. من الجدير بالذكر أن مكونات SDP قد تختلف بناءً على حالة الاستخدام، فبعض التطبيقات قد تضيف عناصر إضافية لدعم التعديلات الفورية على الوصول، خاصة في البيئات التي تتسم بتغيرات متكررة في الأجهزة[3].

6.2 الخاتمة

بعد تقديم فهم واضح للمفاهيم الأساسية والمكوّنات الجوهرية للوصول الشبكي صفري الثقة(ZTNA) ، ننتقل الآن إلى المرحلة العملية لتطبيق هذه المبادئ. لقد وضع هذا الفصل الأساس النظري من خلال تناول إدارة الهوية، وتطبيق السياسات، وآليات الوصول الآمن، والتي تعدّ جميعها عناصر حاسمة في بناء بنية ZTNA فعّالة.

يرتكز الفصل التالي على هذا الأساس النظري ليعرض تصميم النموذج المقترح. حيث يشرح كيفية بناء المعمارية التقنية، بالإضافة إلى كيفية فرض التحكم بالوصول وتطبيق تدابير الأمان بما يتوافق مع إرشادات نموذج الثقة الصفرية.

الفصل الثالث

3. تصميم النموذج الأمني المقترح

سنقدم في هذا الفصل تصميم النموذج الأمني والية تفاعل مكوناته

1.3 مقدمة

بعد تحليل مبادئ المعماريات عديمة الثقة (ZTA) واستعراض التطبيقات النظرية والأطر المعتمدة في هذا السياق، بات من الضروري الانتقال من المستوى المفاهيمي إلى المستوى التطبيقي، وذلك عبر وضع تصور شامل لحل متكامل ينفذ المبادئ الأمنية التي تم تناولها. يهدف هذا التصميم إلى تقديم بنية مرنة وآمنة لإدارة الوصول إلى الموارد، بالاعتماد على الهوية، سياق الاستخدام، ومستوى الثقة اللحظي.

يركز التصميم المقترح على تقليل سطح الهجوم من خلال عزل الموارد الداخلية عن الشبكة العامة، وذلك من خلال المصادقة المعتمدة على الهوية، وتطبيق الحد الأدنى من الصلاحيات عبر منح المستخدمين أقل قدر ممكن من الصلاحيات اللازمة لأداء مهامهم، والتحقق المستمر من صحة الجلسة لتوفير وصول آمن وديناميكي إلى موارد الشبكة المحمية. يتم تنفيذ هذا الحل عبر تقسيمه إلى مكونات منطقية رئيسية، تنسق فيما بينها لضمان تجربة وصول آمنة، دون الكشف المباشر عن بنية الشبكة أو مواردها الحساسة. ويتضمن التصميم مسارات متكاملة للمصادقة، اتخاذ القرار، تطبيق السياسات، وإنشاء نفق اتصال آمن مع الموارد المعزولة.

2.3 مكونات النموذج المقترح

1.2.3 جهاز المستخدم(User Device)

يمثّل نقطة البداية في سلسلة الطلب، حيث يباشر المستخدم عملية المصادقة من جهازه (حاسوب، هاتف ذكي...) باستخدام بيانات اعتماد متعددة العوامل (MFA)، في بيئة لا تفترض فيها الثقة المسبقة. يتمثل دوره في تقديم طلبات الوصول إلى النظام، وهو مسؤول عن تقديم بيانات الهوية، والتفاعل مع مكونات النظام الأخرى ضمن نفق مشفر. يهدف إلى تمكين التفاعل الآمن والمراقب مع الموارد الداخلية، ويحقق بذلك نقطة انطلاق موثوقة تخضع كل جلسة تحقق لسياقها وخصائصها.

2.2.3 نظام المصادقة(Authentication System)

يتحقق من هوية المستخدم استناداً إلى بيانات الاعتماد المقدمة. عند نجاح العملية، يصدر رمز تعريف (Token) يستخدم لاحقاً لتقديم طلب الوصول. في حال فشل المصادقة، يتم إعلام المستخدم بذلك دون كشف أي تفاصيل داخلية عن البنية الخلفية للنظام. يتمثل دوره في التأكد من أن المستخدم هو كيان معروف ومعتمد، وهدفه هو منع الوصول غير المشروع من المصدر. يحقق بذلك أول مستوى من الحماية عبر فلترة مبكرة لأي محاولة غير مشروعة.

3.2.3 بوابة التحكم بالوصول(Access Gateway)

تستلم رمز الهوية من المستخدم وتبدأ عملية التحقق. تعد هذه البوابة هي نقطة التنفيذ الفعلي للسياسات، حيث تمنع الاتصال المباشر مع الموارد، وتعزلها خلف آليات تحقق متعددة. دورها يتمثل في الوساطة بين المستخدم والموارد، وتنفيذ قرارات محرّك السياسات. تهدف إلى منع الوصول غير المرخّص وحماية البنية الداخلية من التعرض المباشر. وتحقق عزلاً منطقياً فعالاً يعزز أمن الشبكة بالكامل.

4.2.3 محرّك السياسات(Policy Engine)

تحلل هذه الوحدة هوية المستخدم وسياق الطلب (الموقع، الوقت، نوع الجهاز...) لتقييم التوافق مع السياسات الأمنية المعتمدة. وتعيد قرار الوصول مرفقاً بمجموعة القواعد التي يجب تطبيقها على الجلسة. مهمته الأساسية اتخاذ قرارات

الوصول الديناميكية بناءً على عوامل متعددة. يهدف إلى تنفيذ مبدأ الامتياز الأقل من خلال تخصيص الصلاحيات بدقة. ويحقق مستوى عالٍ من التكيّف مع التغيرات اللحظية في التهديدات أو سلوك المستخدم.

5.2.3 مزود النفق الآمن(Secure Tunnel Provider)

القناة المشفرة بين جهاز المستخدم والبوابة، تضمن سرية وسلامة البيانات المنقولة، وتمنع التنصت أو التلاعب. يدار النفق وفقاً للسياسات المفروضة من محرّك السياسات، مع مراعاة الحدود الزمنية أو الجغرافية أو نوع الخدمة. تتمثل وظيفته في توفير مسار آمن لنقل البيانات. يهدف إلى الحفاظ على خصوصية الاتصال، ويحقق بذلك متطلباً جوهرياً في بيئات غير موثوقة.

6.2.3 المورد المستهدف مع الجدار الناري (Target Resource with Firewall)

يمثل المورد المستهدف الخدمة أو الخادم الذي يهدف المستخدمون إلى الوصول إليه عبر شبكة VPN. في هذا النموذج، لا يصادق المورد نفسه على الجلسات أو الرموز مباشرةً. بل يفرض المصادقة والتفويض مسبقاً عبر بوابة الوصول ومزوّد الهوية. ويقيّد الوصول بشكل أكبر باستخدام قواعد جدار الحماية المهيأة على الجهاز الافتراضي للمورد، مما يسمح فقط بحركة المرور من عناوين IP أو الأدوار المعتمدة. يساعد هذا المكوّن على تقليل مساحة الهجوم من خلال منع الوصول المباشر غير المصرّح به وضمان إنشاء جميع الاتصالات عبر نفق الآمن، بما يتماشى مع مبادئ ZTNA للتجزئة والتحكم الدقيق في الوصول. دوره يتمثل في تقديم الخدمة المطلوبة دون الانكشاف على التهديدات المباشرة. يهدف إلى توفير الموارد بأمان، ويحقق بذلك استمرار العمل دون المساس بسرّية أو تكامل البيانات.

3.3 تسلسل عمل النظام المقترح (Workflow Description)

1.3.3 مصادقة هوية المستخدم

يبدأ المستخدم عملية الوصول إلى النظام المحمي من خلال واجهة ويب، حيث يعاد توجيهه إلى نظام مصادقة خارجي يقوم بالتحقق من هويته باستخدام بيانات الاعتماد (اسم المستخدم وكلمة المرور) مع إمكانية استخدام المصادقة متعددة العوامل (MFA)لتعزيز الأمان.

2.3.3 إصدار الرمز وبدء الجلسة

عند اكتمال المصادقة بنجاح، يقوم نظام المصادقة بإصدار رمز الوصول (Access Token) موقّع يحتوي على خصائص الهوية مثل اسم المستخدم والدور (مسؤول، مطور، ضيف). يعاد هذا الرمز إلى المستخدم ليستخدم في مراحل الوصول اللاحقة. يقوم أيضا نظام المصادقة بإصدار رمز التحديث (Refresh token) يتم التحكم بزمن انتهاء صلاحية كل من الرمزين من خلال نظام المصادقة.

3.3.3 تقييم طلب الوصول

يقدّم المستخدم الرمز إلى بوابة الوصول(Access Gateway) ، والتي تعمل كوسيط بين المستخدم والموارد المحمية. تقوم هذه البوابة بالتحقق من صحة الرمز واستخلاص معلومات الهوية والجلسة الضرورية.

4.3.3 تقييم السياسات

تقوم بوابة الوصول بإرسال طلب إلى محرك السياسات (Policy Engine) لتحديد ما إذا كان ينبغي منح الوصول، وذلك بالاعتماد على مجموعة من العوامل، منها:

- دور المستخدم.

- حساسية المورد المطلوب.
- قواعد التحكم في الوصول المحددة مسبقاً.

5.3.3 إنشاء جلسة اتصال آمنة

إذا وافق محرك السياسات على الطلب، تقوم بوابة الوصول بإنشاء قناة اتصال مشفّرة وآمنة (نفق آمن) بين المستخدم والموارد المستهدفة.

6.3.3 توجيه الموارد بناءً على الدور

يمنح المستخدم نطاق وصول محدد وفقاً لدوره. يستخدم هذا النطاق من قبل آليات الإنفاذ لضبط رؤية الموارد المتاحة للمستخدم.

7.3.3 إنفاذ الوصول على مستوى المورد

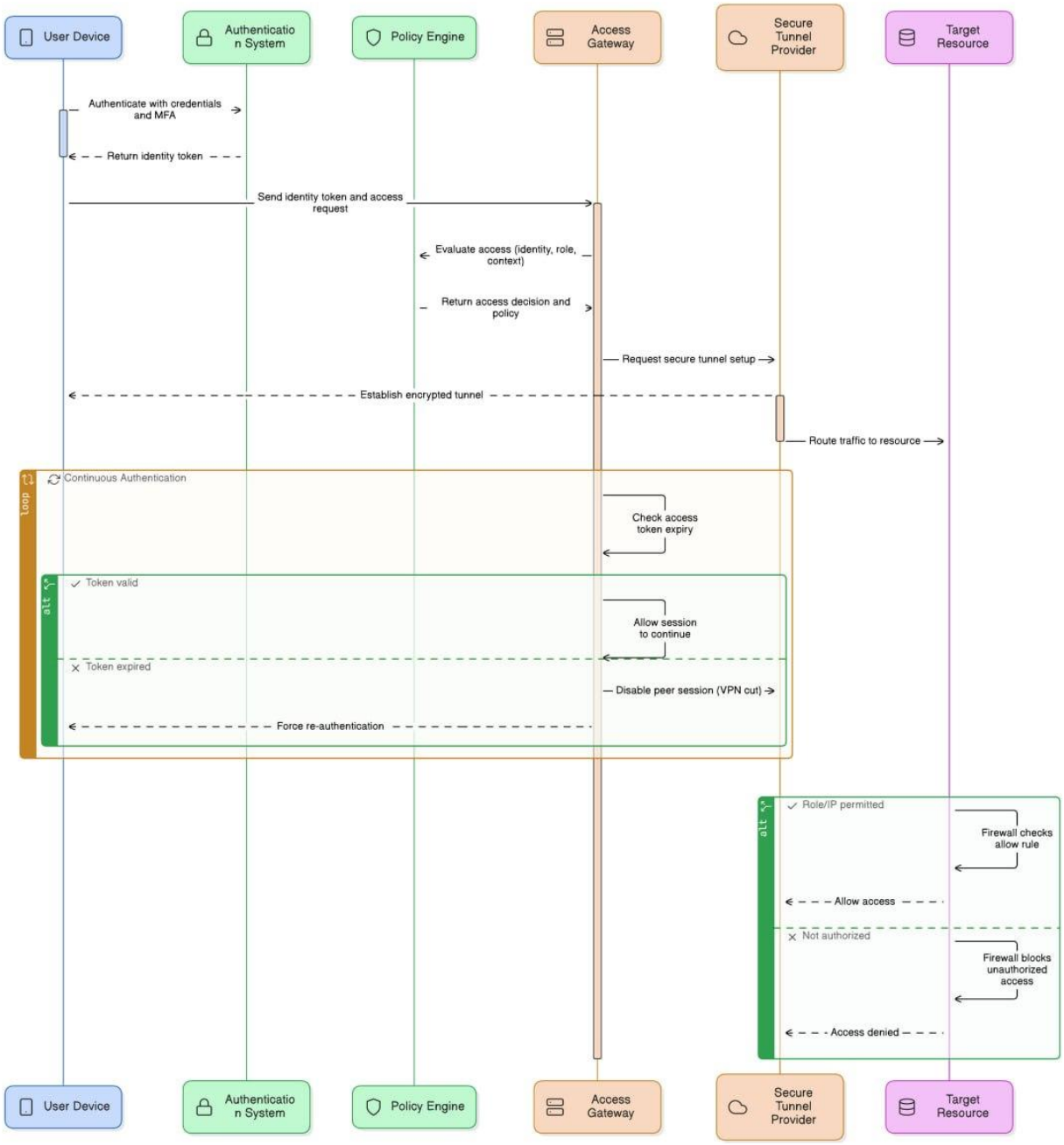
يحمى المورد المستهدف بجدار حماية محلي أو نقطة إنفاذ(Enforcement Point) ، والتي تتحقق من حركة المرور الواردة بناءً على:

- جداول تعيين/IP الدور المعرفة مسبقاً.
- المنافذ المسموح بها لكل فئة مستخدم.

8.3.3 مراقبة الجلسة والمصادقة المستمرة

طوال مدة الجلسة، يفرض النظام صلاحية الجلسة باستخدام آليات انتهاء وتجديد الرموز (Tokens). في حال انتهاء صلاحية الرموز أو إلغائها، يتم إبطال الوصول وإيقاف النفق الآمن تلقائياً.

يعرض الشكل (7) المخطط العام لسير عمل النظام المقترح، موضحاً تسلسل الخطوات من لحظة طلب الوصول وحتى التفاعل الآمن مع الموارد الداخلية.



الشكل 7 النموذج الأمني المقترح

4.3 أهداف النموذج التصميمية والأمنية

يعتمد تصميم هذا النموذج على الأهداف والمبادئ التالية المتوافقة مع مفهوم ZTNA:

1.4.3 الأهداف التصميمية

- **تخفيض مستوى الثقة**
لا يتم الوثوق بأي جهاز أو مستخدم بشكل ضمني. يجب أن تتم المصادقة على كل جلسة والوصول مصرح به بشكل صريح.
- **التحكم الدقيق في الوصول**
يسمح للمستخدمين بالوصول فقط إلى الموارد المحددة وفقاً لدورهم أو السياسة المطبقة عليهم.
- **فصل الهوية عن الشبكة**
يعتمد الوصول على هوية المستخدم والسياق الذي تم التحقق منه، وليس على موقعه في الشبكة.
- **فصل المهام**
تتولى مكونات مختلفة مهام المصادقة واتخاذ قرارات السياسات وتطبيق حركة المرور بشكل مستقل.

2.4.3 الأهداف الأمنية

- **مصادقة قوية**
يجب على المستخدمين تقديم بيانات اعتماد يمكن التحقق منها قبل منحهم الوصول.
- **التحكم الديناميكي في الجلسات**
تنتهي صلاحية الجلسات تلقائياً ويتم التحقق من صحتها باستمرار.
- **تطبيق الأدوار**
تحدد حقوق الوصول بشكل صارم بناءً على أدوار المستخدمين.
- **نقل آمن**
يجب أن تتم جميع الاتصالات عبر قنوات مشفرة وآمنة.
- **تطبيق مبدأ أقل امتياز**
يتم منح المستخدمين الحد الأدنى من الوصول اللازم لأداء مهامهم.
- **مقاومة للحركة الجانبية**
تمنع الجدران النارية والتجزئة الدقيقة الوصول إلى ما هو خارج نطاق التصريح.

5.3 اعتبارات التصميم وتخفيف التهديدات

يأخذ النموذج بعين الاعتبار عدداً من المخاطر المحتملة ويطبق استراتيجيات تخفيف مناسبة:

التهديد / الخطر	استراتيجية التخفيف
سرقة بيانات الاعتماد	فرض المصادقة متعددة العوامل (MFA) والرموز قصيرة العمر يقللان من الضرر طويل الأجل.
هجمات إعادة استخدام الرموز	تواقيع الرموز وتواريخ انتهاء الصلاحية تضمن أن الرموز تستخدم لمرة واحدة ومحددة زمنياً.
الأجهزة المخترقة	الوصول المبني على الدور مع التجزئة الدقيقة يقلل من نطاق الضرر.

التهديد / الخطر	استراتيجية التخفيف
الحركة الجانبية في الشبكة	يتم تقييد الوصول إلى الموارد بعناوين IP ثابتة وتطبيق ذلك عبر جدران حماية خاصة بالموارد.
اختطاف الجلسات	تتم مراقبة الجلسات وإيقافها عند انتهاء صلاحية الرموز أو عند عدم صلاحيتها.
سوء تكوين سياسات الوصول	يضمن تطبيق السياسات من خلال نقطة مركزية تقييماً متسقاً للقواعد.

جدول 5 أنواع التهديدات وسبل التخفيف منها

6.3 الخاتمة

من خلال هذا النموذج، لا نقوم فقط بالتخفيف من نقاط الضعف التقليدية في الشبكات الخاصة الافتراضية — (VPN) مثل الوصول المفرط وعدم وجود رؤية على مستوى الجلسة — بل نقدّم أيضاً آليات تحكم حديثة تتماشى مع مبادئ الثقة الصفرية. يتميز التصميم بطبيعته النمطية (Modular) وقابليته للتوسّع، مما يسمح باستخدام تقنيات مختلفة في التنفيذ مع الحفاظ على نفس الأهداف المعمارية.

يمهّد هذا الفصل الطريق لمرحلة التنفيذ، حيث سيتم تحويل النموذج النظري إلى واقع عملي باستخدام أدوات مفتوحة المصدر. وستظهر الفصول التالية كيفية تحقيق كل مكون على حدة، وكيف يضمن النظام وصولاً آمناً ومدفوعاً بالسياسات وموجهاً بحسب هوية المستخدم إلى الأصول المحمية.

الفصل الرابع

4. بيئة العمل والأدوات المستخدمة

سيتم في هذا الفصل عرض بيئة العمل والأدوات المستخدمة

1.4 المقدمة

عند تنفيذ نموذج الوصول إلى الشبكة بناءً على مبدأ الثقة المعدومة(ZTNA) ، كان اختيار الأدوات المناسبة عاملاً حاسماً لضمان الأمان وقابلية التوسع والامتثال لمبادئ هذا النموذج. يعرض هذا الفصل الأدوات والمنصات التي تم استخدامها في المشروع، حيث تم اختيار كل أداة لتؤدي دوراً محدداً داخل المعمارية—بدءاً من إدارة الهوية ووصولاً إلى إنشاء الأنفاق الآمنة وتطبيق سياسات التحكم في الوصول.

جميع الأدوات المختارة مفتوحة المصدر، مما يتماشى مع هدف المشروع المتمثل في تقديم بيئة "ثقة معدومة" عملية وفعالة من حيث التكلفة وقابلة للتنفيذ واقعياً. يناقش هذا الفصل دور كل أداة على حدة، يلي ذلك تبرير واضح لاختيارها ضمن سياق تنفيذ هذا النموذج الأمني.

2.4 Keycloak- إدارة الهوية والتحكم بالوصول

1.2.4 نظرة عامة

Keycloak هي أداة مفتوح المصدر لإدارة الهوية والتحكم في الوصول، تم تطويره من قبلRed Hat . يوفر دعماً جاهزاً لبروتوكولات المصادقة الحديثة مثل OAuth 2.0 و OpenID Connect (OIDC)وSAML 2.0، مما يتيح للتطبيقات والخدمات تفويض عملية المصادقة إلى مزود هوية مركزي(IdP) .

في سياق معمارية الثقة الصفرية(ZTA) ، يعتبر مزود الهوية (IdP) مكوناً أساسياً—فهو يحدد حدود الثقة، ويصدر رموزاً موقعة تمثل الهويات المصادق عليها، ويطبق ضمان الهوية قبل منح الوصول إلى أي مورد. يدعم Keycloak التحكم في الوصول بناءً على الدور(RBAC) ، والمصادقة متعددة العوامل(MFA) ، وإدارة الجلسات المعتمدة على الرموز، واتحاد الهوية (Identity Federation)، مما يجعله مناسباً جداً لتنفيذ نموذج ZTNA .

يستخدم Keycloak رموزاً من نوع **JWT (JSON Web Token)** لتمثيل Access Tokenو ID Token، وذلك ضمن إطار بروتوكولOpenID Connect . وتوقع هذه الرموز رقمياً باستخدام خوارزميات مثل RSA256 لضمان النزاهة والتحقق من صحتها. اعتماد JWT يجعل هذه الرموز قابلة للتحقق ذاتياً (self-contained) وقابلة للنقل بين الأنظمة بطريقة آمنة وفعالة.

2.2.4 دور Keycloak في نموذج ZTNA

في المعمارية المقترحة لدينا:

- يقوم Keycloak بمصادقة المستخدمين باستخدام بيانات الاعتماد و MFA (عبرGoogle Authenticator/ (TOTP (Time-based One-time Password)).
- بعد المصادقة الناجحة، يصدر رموز الهوية (ID tokens) ورموز الوصول (Access tokens) بما يتوافق مع معاييرOIDC.
- تستخدم هذه الرموز لاحقاً من قبل بوابة الوصول (Access Gateway) للتحقق من الهوية، واستخراج الأدوار، وتطبيق سياسات التحكم في الوصول.

- يعمل Keycloak أيضاً كسلطة RBAC، حيث يعيّن لكل مستخدم دوراً (مثلadmin ، developer، guest)، ويتم ربط هذا الدور مباشرةً بمستوى الوصول عبر الـ VPN والقيود المفروضة على الموارد الداخلية.

من خلال استخدامKeycloak ، يمكننا من مركزة المصادقة، وتطبيق المصادقة المستمرة باستخدام منطق انتهاء وتجديد الرموز، وفصل طبقة المصادقة عن منطق التطبيق—وهي مبادئ أساسية في نموذج الثقة الصفرية.

3.2.4 مبررات اختيار Keycloak

- بروتوكولات قائمة على المعايير :يدعم OAuth 2.0 وOpenID Connect، اللازمة للمصادقة المفوضة الآمنة وإصدار الرموز. حيث أن البروتوكول OpenID Connect يعتمد على JSON Web Tokens (JWT) بشكل رئيسي
- إدارة الأدوار :يتيح تعريف وتعيين RBAC بسهولة دون الحاجة إلى إضافات خارجية.
- دعم MFA مدمج :يوفر دعماً للمصادقة متعددة العوامل باستخدام كلمات مرور لمرة واحدة تعتمد على الوقت (TOTP)بدون خدمات خارجية.
- مفتوح المصدر ومدعوم بنشاط :مدعوم من Red Hat ويحظى بمجتمع كبير ودعم على مستوى المؤسسات.
- واجهة إدارة سهلة :تتيح إدارة المستخدمين والمجالات والعملاء بكفاءة دون الحاجة إلى برمجة متقدمة.
- مرونة التكامل :يمكن دمجه بسهولة مع(Python (Flask ، وعملاء VPN، وبوابات الوصول باستخدام بروتوكولات التعامل مع الرموز القياسية.

وبخلاف حلول IAM التجارية الثقيلة، وفر Keycloak المرونة والتحكم ودعم البروتوكولات المطلوبة لتطبيق ZTNA في بيئة تعليمية وتجريبية [6] .

3.4 WireGuard- بروتوكول VPN حديث

1.3.4 نظرة عامة

WireGuardهو بروتوكول شبكة افتراضية خاصة (VPN) حديث ومفتوح المصدر، تم تطويره مع التركيز على البساطة، الأداء العالي، والمبادئ التشفيرية القوية. أنشأه جيسون دوننفيلد (Jason A. Donenfeld) وتم دمجه لاحقاً في نواة نظام Linux، مع دعم لأنظمة تشغيل متعددة مثل Windows وmacOS وAndroid وiOS.

يعمل WireGuard على الطبقة الثالثة من نموذج الشبكات (طبقة الشبكة) ويعتمد في تنجيزه على رماز صغير نسبيا (~4000 سطر)، ما يجعله أسهل في التدقيق والصيانة مقارنة بحلول VPN التقليدية. ويعتمد على خوارزميات تشفير متطورة وحديثة، ويهدف إلى استبدال بروتوكولات قديمة مثل IPsec وOpenVPN في العديد من الاستخدامات [7].

2.3.4 دور WireGuard في نموذج ZTNA

في النموذج المقترح، يستخدم WireGuard لـ:

- إنشاء نفق اتصال آمن بين جهاز المستخدم وشبكة الموارد.
- تطبيق التحكم في الوصول من خلال ربط كل قرين (Peer) بعنوان IP معين وصلاحيات قائمة على الدور.
- تفعيل الإلغاء الديناميكي للجلسات: يمكن تعطيل أو إزالة الأقران برمجياً حسب حالة الرمز(Token) أو الجلسة.
- الحد من الانكشاف: لا يتم كشف الموارد أبداً للمستخدمين غير المصادق عليهم، وتمرر كل الحزم عبر قناة آمنة.

3.3.4 مبررات اختيار WireGuard

في سياق تنفيذ نموذج الوصول إلى الشبكة وفق الثقة الصفرية (ZTNA)، يعد النفق الآمن للتواصل نقطة إنفاذ مركزية. بعد تقييم عدد من حلول VPN [8]، اخترنا WireGuard بدلاً من OpenVPN للأسباب التالية:

الميزة	OpenVPN	WireGuard
حجم الرماز	100,000~سطر	4000~سطر
التصميم التشفيري	تقليدي(TLS مع RSA و AESوغيرها)	حديث (Curve25519, ChaCha20, BLAKE2s)
الأداء	متوسط (يعمل في مساحة المستخدم)	عالي جداً (متكامل في نواة Linux)
تعقيد الإعداد	معقد (شهادات، تبادل مفاتيح، إعدادات متعددة)	بسيط (ملفات إعداد ثابتة)
قابلية التدقيق والأمان	صعب التدقيق، احتمالية أكبر للثغرات	سهل التدقيق، سطح هجوم صغير
دعم التنقل (Roaming)	يتطلب إعدادات إضافية	مدمج وسلس
عبورNAT	مدعوم لكن يتطلب تعديلات	مدمج باستخدامUDP
التكامل مع ZTNA	أكثر تعقيداً ويتطلب شهادات	سهل الدمج مع منطق التحكم في الوصول

جدول 6 مقارنة بين WireGuard و OpenVPN

4.4 Flask web application framework- بوابة الوصول

1.4.4 نظرة عامة

في بنية الوصول الشبكي بدون ثقة(ZTNA) ، تعمل بوابة الوصول (Access Gateway) كوسيط بين المستخدمين والموارد المحمية. وتتمثل مسؤولياتها في التحقق من رموز الهوية(Tokens) ، تطبيق سياسات الوصول، والتحكم في الاتصالات الآمنة بناءً على صلاحية الجلسة والتفويض.

في هذا المشروع، قمنا ببناء بوابة الوصول باستخدام **Flask**، وهو إطار عمل ويب خفيف الوزن ومعياري بلغة Python. وجاء اختيارنا لـ Flask نتيجة الحاجة إلى حل قابل للتخصيص، منخفض التعقيد، ويتيح دمج منطق المصادقة، التحكم بالجلسات، والتفاعل المباشر مع أدوات الشبكة على مستوى النظام مثل WireGuard.

2.4.4 دور Flask في النموذج ZTNA

يمثل Flask البنية الأساسية لتطبيق مكون بوابة الوصول في نموذج ZTNA الخاص بنا، حيث يقوم بما يلي:

- **التحقق من الرموز** :تحليل والتحقق من رموز Jason Web Tokens أو JWT اختصارا الصادرة عن Keycloak.
- **منطق التحكم بالوصول** :تطبيق السياسات بناءً على صلاحية الجلسة والرمز.
- **إدارة دورة حياة الأقران** :**(Peers)** إنشاء وحذف أقران WireGuard حسب حالة الجلسة.
- **توليد الإعدادات بأمان** :إنشاء ملفات إعداد WireGuard .conf بشكل ديناميكي لكل مستخدم.
- **تطبيق تسجيل الخروج** :تعطيل الأقران تلقائياً عند انتهاء صلاحية الجلسة أو الرمز المحدث (Refresh Token).

3.4.4 مبررات استخدام Flask

تم اختيار Flask للأسباب التالية [9]:

متطلبات التصميم	ما يقدّمهFlask
منطق المسارات المخصص	سهولة تعريف وإدارة المسارات مثل: /login, /callback, /download-config, /logout.
التعامل مع رموز الهوية	تكامل سهل مع Keycloak عبر بروتوكول OpenID Connect باستخدام مكتبات Python مثلPyJWT و requests.
تطبيق التحكم بالوصول الديناميكي	إمكانية برمجة منطق لتفعيل/تعطيل أقران WireGuard والتحقق من صلاحية الجلسة باستخدام middleware أو decorators.
التكامل مع نظام Linux	استخدام subprocess.run() لتنفيذ أوامر WireGuard وجدران الحماية من الخادم الخلفي بشكل آمن.
إدارة الجلسات والرموز	استخدمنا جلسات Flask لتنفيذ مصادقة مستمرة، وتتبع انتهاء صلاحية الرموز، وفرض إعادة المصادقة.
سرعة التطوير والمرونة	Flaskمثالي للمشاريع التي تتطلب تصميماً واضحاً، تطويراً سريعاً، وقابلية للتجريب، خصوصاً في السياقات البحثية أو الأكاديمية.

جدول 7 متطلبات التصميم التي يحققها Flask

5.4 VMware- بيئة العمل الافتراضية

1.5.4 نظرة عامة

VMware Workstation هو برنامج محاكاة افتراضية من النوع (Type 2 Hypervisor) ، يتيح للمستخدمين تشغيل أنظمة تشغيل متعددة في وقت واحد على جهاز مادي واحد. يستخدم على نطاق واسع في بيئات التطوير، والاختبار، والتعليم، نظراً لاستقراره ودعمه لأنظمة تشغيل ضيف متنوعة، وميزاته الغنية.

2.5.4 دور VMware في نموذج ZTNA

تم استخدام VMware لاستضافة المكونات التالية في بيئات افتراضية معزولة ولكن مترابطة:

- مزود الهوية Keycloak (IdP)
- بوابة الوصول والنفق الآمن Flask (Access Gateway / Policy Enforcement) and WireGuard Server (VPN tunnel provider)
- الموارد Resources

بحيث يمكن إدارة كل أداة (كجهاز افتراضي) بسهولة وبشكل معزول مما يدعم الاختبار الآمن.

3.5.4 مبررات اختيار VMware

رغم أن VirtualBox يعد خياراً مفتوح المصدر وشائعاً، إلا أنه يفتقر إلى بعض الميزات المتقدمة الموجودة في VMware[10]:

الميزة	VirtualBox	VMware Workstation
تخصيص الشبكات	أساسي (تحكم محدود)	Bridged ، شبكات افتراضية) متقدم NAT)
تحسين الأداء	مناسب للاستخدام العام	دعم أفضل لضبط المعالج
إدارة اللقطات والاستنساخ	وظيفية ولكن أقل كفاءة في الواجهة	واجهة سهلة ومستقرة
التكامل مع أدوات المؤسسات	محدود أو غير موجود	دعم قوي (vSphere, vCenter)

جدول 8 مقارنة بين VirtualBox و VMware

6.4 Google Authenticator – مزود المصادقة متعددة العوامل MFA

1.6.4 نظرة عامة

هو تطبيق جوال تم تطويره بواسطة Google ويستخدم لتوليد كلمات مرور لمرة واحدة تعتمد على الوقت (TOTP) كجزء من المصادقة متعددة العوامل (MFA). يعزز أمان تسجيل الدخول التقليدي من خلال طلب عامل تحقق ثانٍ بالإضافة إلى اسم المستخدم وكلمة المرور — عادةً ما يكون رمزاً مكوّناً من 6 أرقام يتغير كل 30 ثانية.

يتوافق هذا التطبيق مع أي خدمة تدعم خوارزمية TOTP كما هو محدد في المعيارRFC 6238 ، وقد حاز على انتشار واسع بفضل بساطته، وعمله دون اتصال بالإنترنت، وتوافقه الواسع مع مزودي الهوية مفتوحي المصدر مثل **Keycloak**.

2.6.4 دور Google Authenticator في نموذج ZTNA

في سياق المشروع، يعمل**Google Authenticator** كمزود للمصادقة متعددة العوامل (MFA) أثناء تسجيل دخول المستخدم. فعند محاولة المستخدم تسجيل الدخول عبر **Keycloak**، وبعد إدخال بيانات الاعتماد الصحيحة، يطلب منه إدخال رمز مكوّن من 6 أرقام من تطبيق Google Authenticator الخاص به. وفقط بعد التحقق بنجاح من هذا الرمز، يصدر Keycloak رموز الوصول (Access Token) والتحديث (Refresh Token) التي تتيح تحميل إعداداتVPN .

وبالتالي، يلعب هذا التطبيق دوراً أساسياً في ضمان هوية المستخدم ويعزز حدود الثقة المطلوبة في نموذج الوصول الشبكي المعتمد على الثقة الصفرية.

3.6.4 مبررات اختيار Google Authenticator

وقد تم اختيار Google Authenticator للأسباب التالية:

- **التوافق مع معيار TOTP**: يتوافق تماماً مع بروتوكول TOTP المستخدم في Keycloak وأنظمة إدارة الهوية الأخرى.
- **العمل دون اتصال**: لا يتطلب اتصالاً بالإنترنت، مما يجعله مناسباً للبيئات الآمنة أو المعزولة عن الشبكة.
- **البساطة**: سهل الإعداد وسهل الاستخدام من قبل المستخدمين دون الحاجة إلى أجهزة خاصة أو حسابات معقدة.

- **متعدد المنصات** :متوفر على أجهزة Android وiOS.
- **تكامل مفتوح** :يمكن دمجه بسهولة مع إعدادات MFA المدمجة في Keycloak دون الحاجة إلى واجهات برمجة تطبيقات أو تراخيص خارجية [11].

الفصل الخامس

5. تنفيذ واختبار النموذج الأمني

سيتم في هذا الفصل تفصيل خطوات العمل واختبار النموذج المقترح

1.5 المقدمة

يعرض هذا الفصل التنفيذ العملي لنموذج معمارية الوصول الشبكي القائم على الثقة الصفرية المقترح باستخدام أدوات مفتوحة المصدر. بعد وضع الأسس النظرية والمعمارية في الفصول السابقة، يقوم هذا القسم بترجمة تلك المفاهيم إلى نموذج أولي واقعي وفعّال.

الهدف هو إثبات إمكانية تحقيق المبادئ الأساسية لبنية الثقة الصفرية —مثل التحقق الصارم من الهوية، والمصادقة المستمرة، والوصول حسب مبدأ الصلاحيات الأدنى، وعزل الموارد —باستخدام الأدوات **Keycloak** و **WireGuard** و **Flask** و**VMware**.

يبدأ الفصل بوصف بيئة الاختبار وتكوين المكونات، متبوعاً بشرح تفصيلي لعملية التكامل، وآليات تطبيق السياسات، والمنطق المتبع في إدارة الأقران(peers) . وفي النهاية، يتم تنفيذ عدد من حالات الاختبار للتحقق من سلوك النظام تحت سيناريوهات مختلفة، بما يضمن توافقه مع نموذج أمان الثقة الصفرية وتحقيقه للأهداف الأمنية المنشودة.

2.5 إعداد البيئة

تم بناء بيئة اختبار افتراضية كمختبر محلي باستخدام VMware Workstation. تحاكي هذه البيئة شبكة خاصة مؤمنة يتم فيها تطبيق المصادقة والسياسات والتحكم بالوصول استنادا الى مبادئ الثقة الصفرية. تم استخدام تجهيزة فيزيائية من نوع DELL مع مجموعة من الآلات الافتراضية لتنفيذ النموذج. نجد أهم المعلومات التفصيلية في الجداول التالية:

1.2.5 الجهاز الفيزيائي

Motherboard	DELL 0PGC5N
Processor	AMD Ryzen 5 5600H (12 CPUs)~3.3GHz
Memory	16 GB
Storage	1 TB
Operating System	Windows 11 Home 64-bit
Network Adapter	NAT (VMnet8) with IP 192.168.39.1/24

جدول 9 توصيف الآلة الفيزيائية المنصب عليها الآلات الافتراضية

2.2.5 الآلة الافتراضية التي تعمل عليها أداة Keycloak

Processor	2 Dual Core Virtual Processors
Memory	4 GB
Storage	20 GB
Operating System	Ubuntu Server 24.04
Network Adapter	NAT (VMnet8) with IP 192.168.39.128/24

جدول 10 توصيف الآلة التي تعمل عليها الأداة Keycloak

3.2.5 الآلة الافتراضية التي تعمل عليها الأداتين Flask وWireGuard

Processor	2 Dual Core Virtual Processors
Memory	4 GB
Storage	20 GB
Operating System	Ubuntu Server 24.04
Network Adapter	NAT (VMnet8) with IP 192.168.39.129/24
Network Adapter 2	Host-Only (VMnet1) with IP 192.168.139.129/24

جدول 11 توصيف الآلة الافتراضية التي تعمل عليها الأداتين Flaskو WireGuard

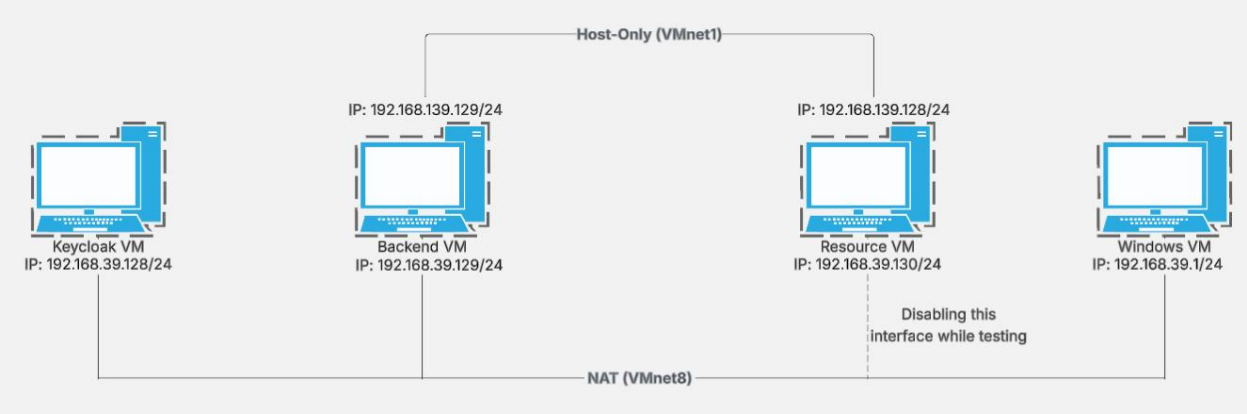
4.2.5 الآلة الافتراضية التي تمثل الموارد

Processor	2 Dual Core Virtual Processors
Memory	4 GB
Storage	20 GB
Operating System	Ubuntu Server 24.04
Network Adapter	NAT (VMnet8) with IP 192.168.39.130/24
Network Adapter 2	Host-Only (VMnet1) with IP 192.168.139.128/24

جدول 12 توصيف الآلة الافتراضية التي تمثل الموارد

3.5 معمارية الشبكة

بعد إعداد مجموعة الآلات الافتراضية التي تم عرضها أعلاه قمنا بتنصيب كل أداة على الآلة الموافقة. يوضح الشكل (8) طوبولوجي بيئة الاختبار حيث تضم البيئة أربع مكونات رئيسية.



الشكل 8 طوبولوجي بيئة الاختبار

تكون التجهيزة الفيزيائية التي تعمل بنظام Windows بحد ذاتها هي الآلة التي يقوم المستخدم من خلالها بالدخول للشبكة.

تم ربط كل Backend VM وKeycloak VM وWindows machine وResource VM بشبكة واحدة VMnet8 بحيث تكون محوّلات الشبكة لكل من الآلات ضمن هذه الشبكة من نوع NAT بحيث يمكننا ذلك من الوصول لشبكة الانترنت

وتحميل الأدوات من المواقع المخصصة لها واضافة كل ما يلزم من إضافات لتعمل بشكل صحيح. وأيضا يمكننا ذلك من سهولة استخدام هذه الآلات من خلال استخدام برنامج MobaXterm (المنصب على التجهيزة الفيزيائية) بحيث يمكننا من انشاء جلسات عن بعد من خلال استخدام البروتوكول SSH وذلك لما يفيد من زيادة التفاعل مع واجهة الأوامر السطرية CLI. تكون الشبكة الخاصة VMnet1 التي تربط كل الآلتين Backend وResources من نوع Host-Only مما يحقق عزل آلة الموارد عن الآلات الأخرى بحيث لا يمكن سوى لآلة الـ Backend الوصول لها (وذلك كما هو مبين بالشكل (8) بعد إيقاف واجهة الشبكة للآلة Resource VM).

4.5 تكوين النظام

سيتم شرح الإعدادات بالتفصيل التي تم تطبيقها لتنفيذ مبادئ الثقة الصفرية عبر جميع المكونات، والتي تشمل إدارة الهوية والوصول، تطبيق السياسات، إنشاء نفق اتصال مشفر، وحماية الموارد.

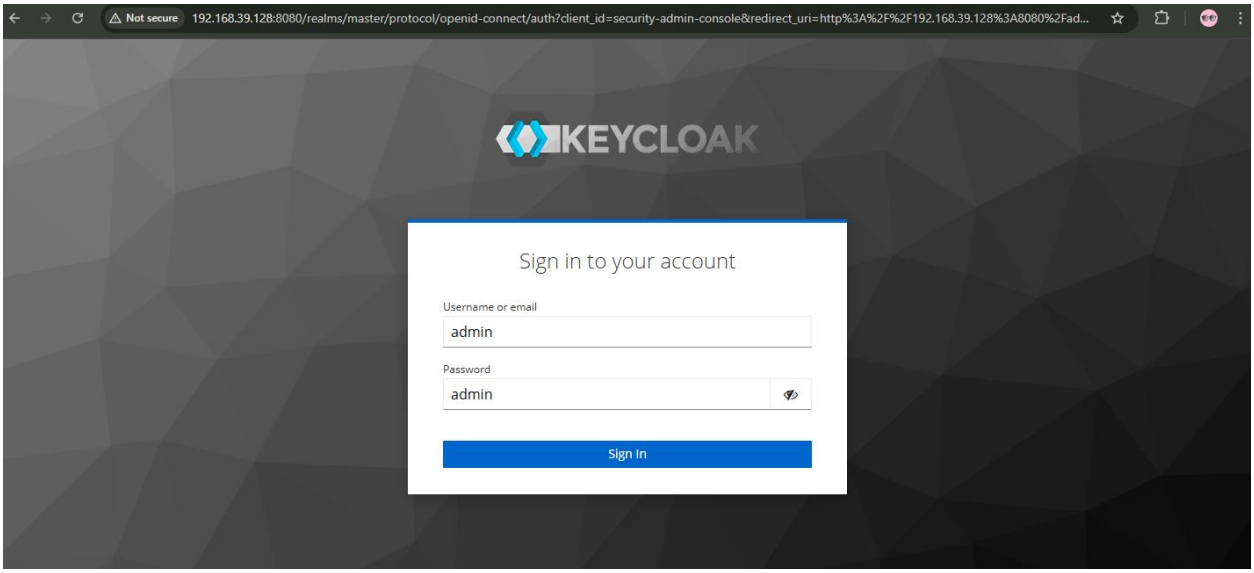
1.4.5 تكوين الأداة Keycloak

يمثل Keycloak نظام إدارة الهوية والوصول المركزي، بحيث يكون مسؤولا عن المصادقة، انشاء الرموز وإعطاء الأدوار لكل من المستخدمين. فيما يلي خطوات الإعداد الرئيسية. بعد تنصيب الأداة على الآلة الافتراضية الموافقة تم تشغيل هذه الأداة عبر تنفيذ ما يلي:

```
keyck@keycloak:~$ cd keycloak-24.0.0/bin/
keyck@keycloak:~/keycloak-24.0.0/bin$ sudo KEYCLOAK_ADMIN=admin KEYCLOAK_ADMIN_PASSWORD=admin ./kc.sh start-dev --http-port 8080
```

الشكل 9 تشغيل الأداة Keycloak

بحيث تقوم هذه التعليمة بتشغيل الأداة على الآلة الافتراضية ذات العنوان 192.168.39.128 على المنفذ 8080. بعدها باستخدام المتصفح (Chrome) يمكن استخدام الأداة. يعرض الشكل (10) واجهة الأداة بحيث تطلب اسم المستخدم وكلمة المرور لمسؤول الأداة كما هو مبين بالشكل (9).

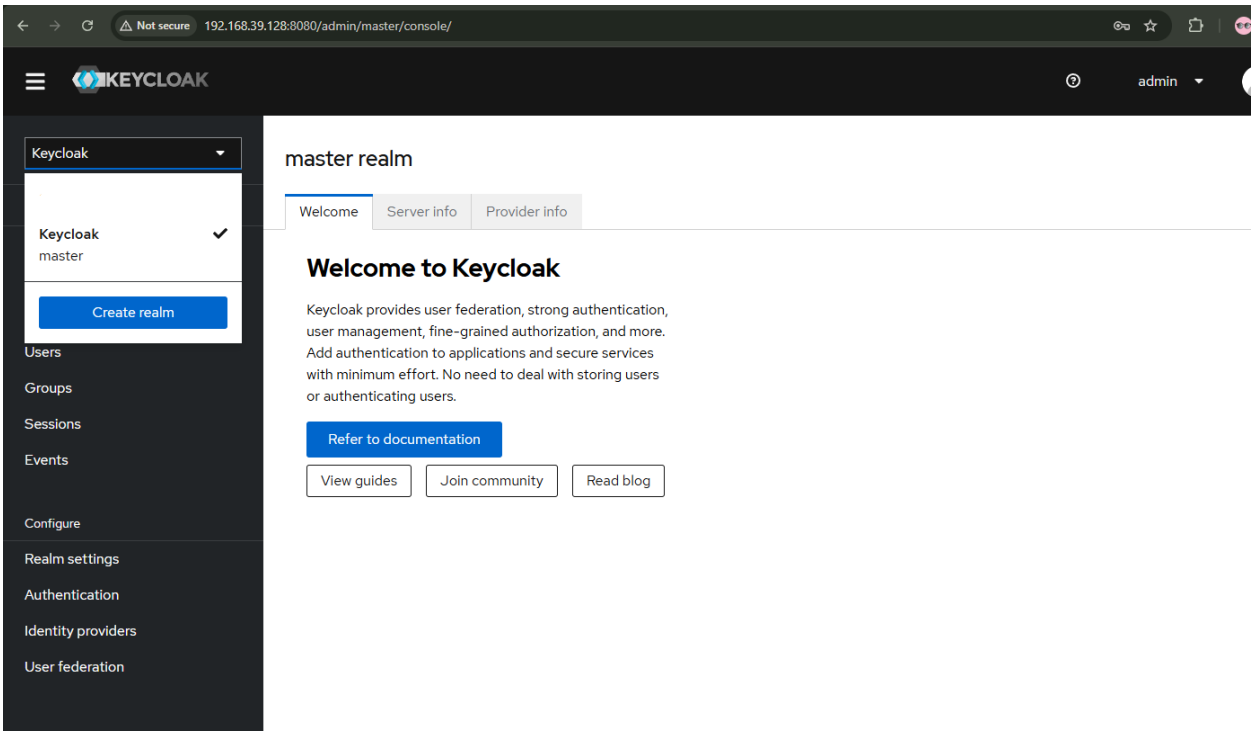


الشكل 10 تسجيل الدخول لمسؤول الأداة Keycloak

1.1.4.5 تكوين النطاق - Realm

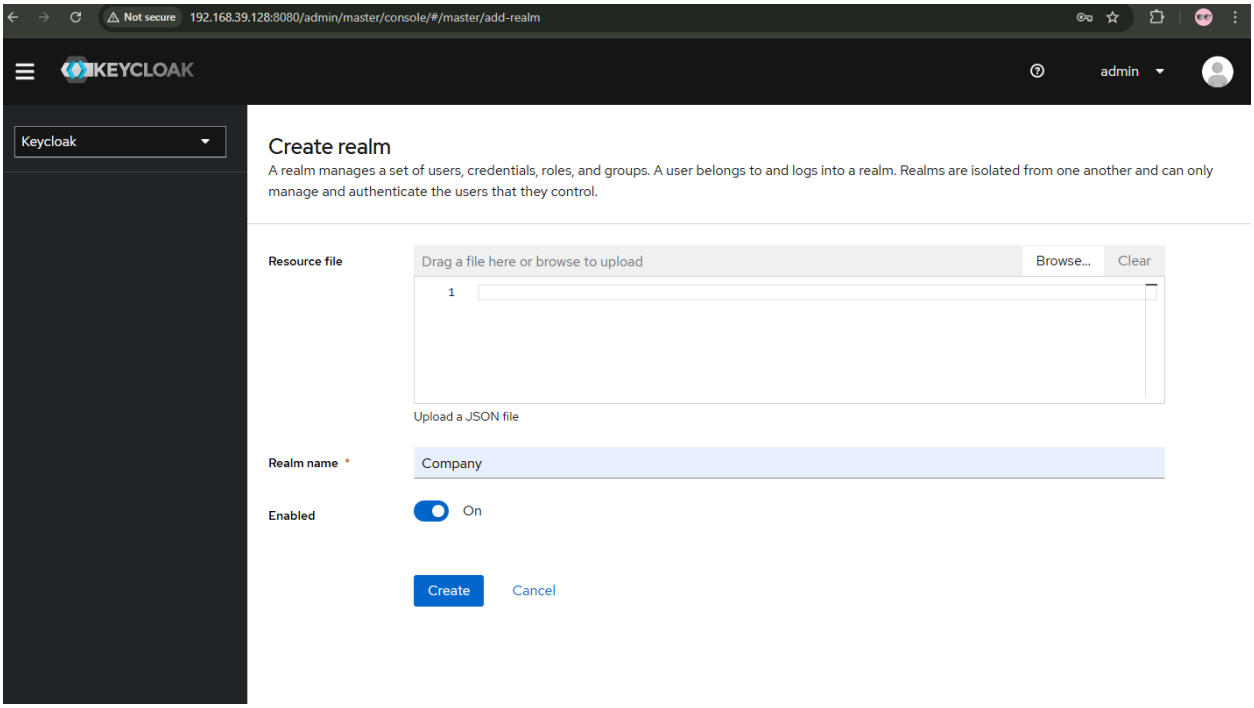
توفر الأداة العديد من الخصائص التي يمكن استخدماها حسب متطلبات النظام المطلوب تصميمه، قمنا من خلال هذه الأداة بإنشاء نطاق اسمه Company يمثل بحيث تكون جميع الإعدادات اللاحقة ضمن نفس النطاق (حيث أن Keycloak

يقدم ميزة التحكم وإنشاء أكثر من نطاق بحيث يكون كل نطاق معزول عن الاخر بما فيه من مستخدمين وقواعد وصول وزبائن وغيرها). يوضح الشكل (11) واجهة الأداة بعد تسجيل الدخول ومن ثم كيفية إنشاء نطاق.



الشكل 11 واجهة Keycloak بعد تسجيل الدخول من قبل مسؤول الأداة

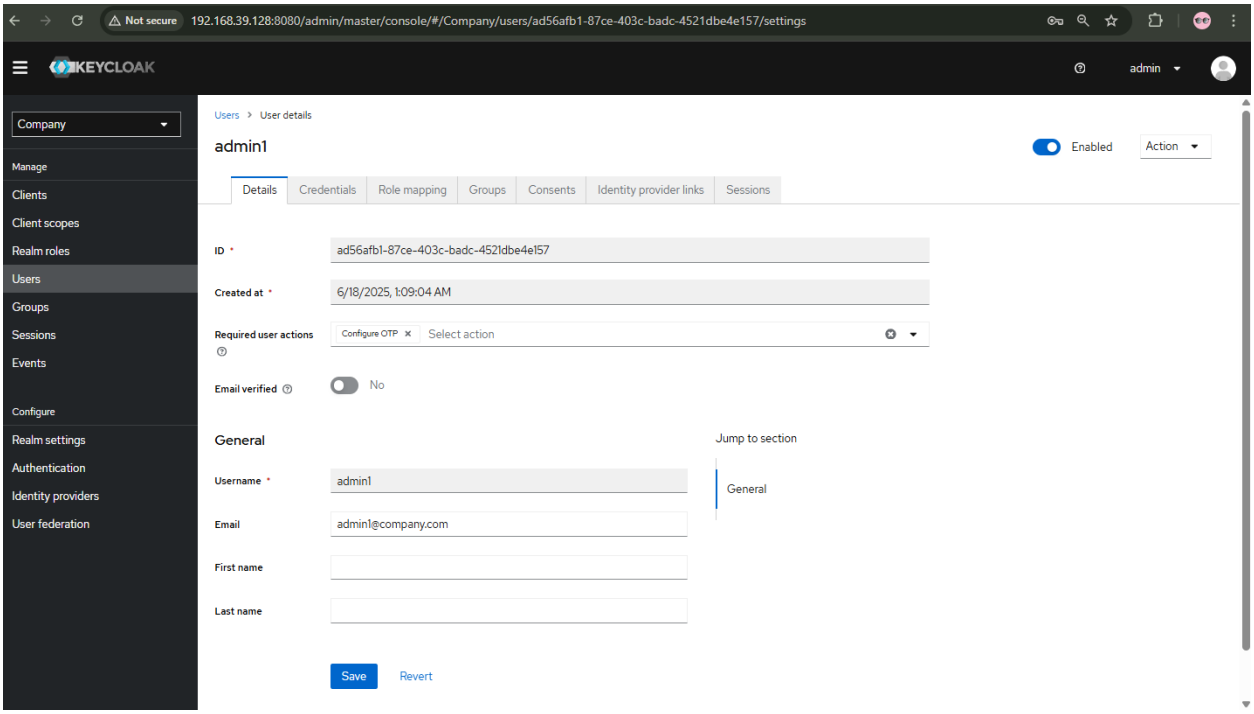
يوضح الشكل (12) كيفية إنشاء نطاق اسمه Company بحيث سيضم هذا النطاق جميع المستخدمين وتعيين الأدوار لكل منهم ومن ثم السياسات مثل كيفية تسجيل الدخول وإعداد المصادقة متعددة العوامل والزبائن (بحالتنا هو تطبيق بوابة الوصول) وتعيين جميع الإعدادات التي سنقوم بتطبيقها ضمن هذا النطاق بما يشمل إعدادات الرموز التي سنتطرق لها بالتفصيل لاحقا.



الشكل12 إنشاء نطاق باسم Company

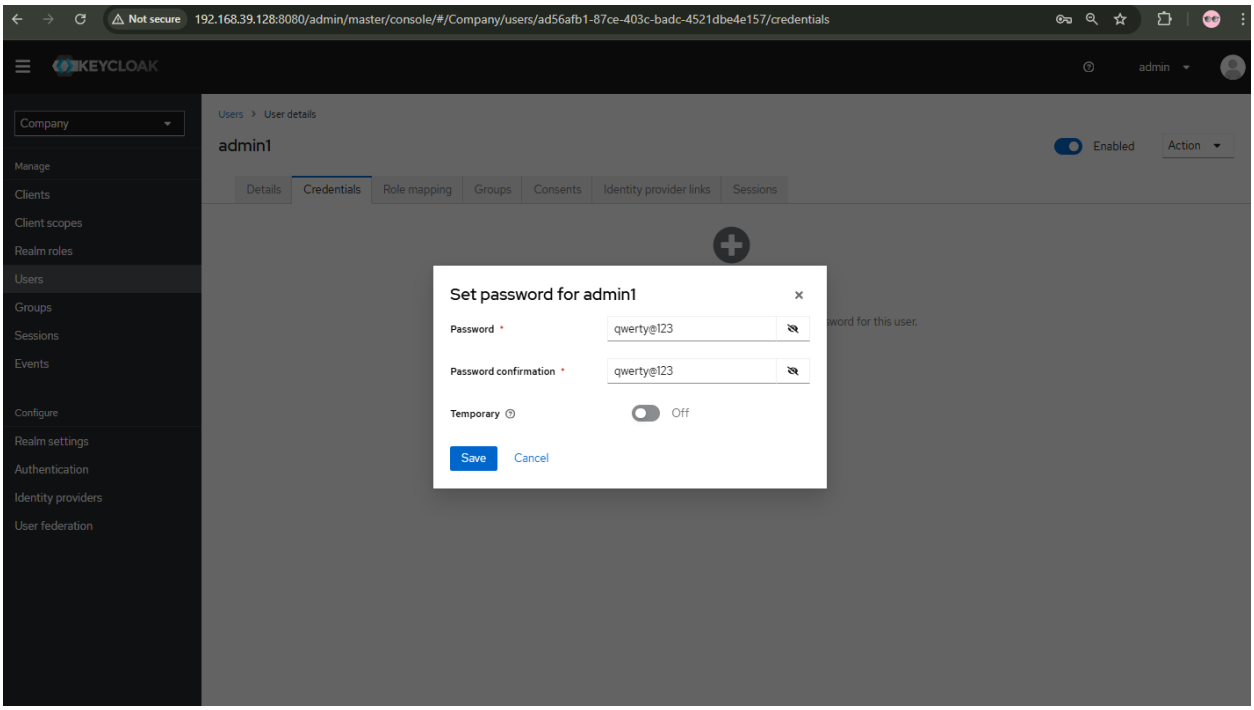
2.1.4.5 تكوين المستخدمين – Users

بعد إنشاء النطاق Company سنقوم بإنشاء المستخدمين الذين ينتمون لهذا النطاق. يوضح الشكل (13) كيفية انشاء المستخدمين. بحيث تم تفعيل ميزة كلمة المرور لمرة واحدة One-Time Password أو اختصارا OTP مما يحقق المصادقة



الشكل 13 إنشاء مستخدم في Keycloak

متعددة العوامل وتم إعطاء اسم المستخدم admin1 وعنوان بريد الكتروني admin1@company.com ويتم بعد ذلك توليد ID تلقائي خاص بهذا المستخدم كما هو موضح بالشكل (13) بالإضافة لإعدادات أخرى. بالانتقال للخانة Credentials يتم من خلالها تعيين كلمة المرور الخاصة بهذا المستخدم. يوضح الشكل (14) كيفية وضع كلمة مرور للمستخدم admin1. ثم نقوم بنفس الخطوات لإضافة جميع المستخدمين (في مشروعنا تم إضافة مسؤلي نظام (admins) و أربعة مطورين

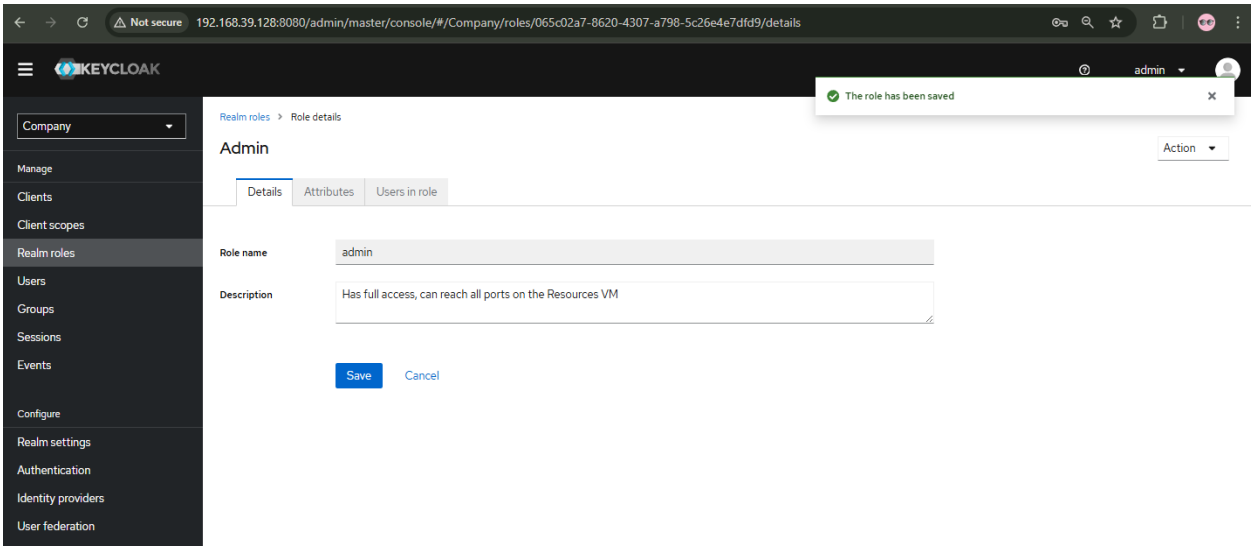


الشكل 14تعيين كلمة المرور للمستخدم admin1

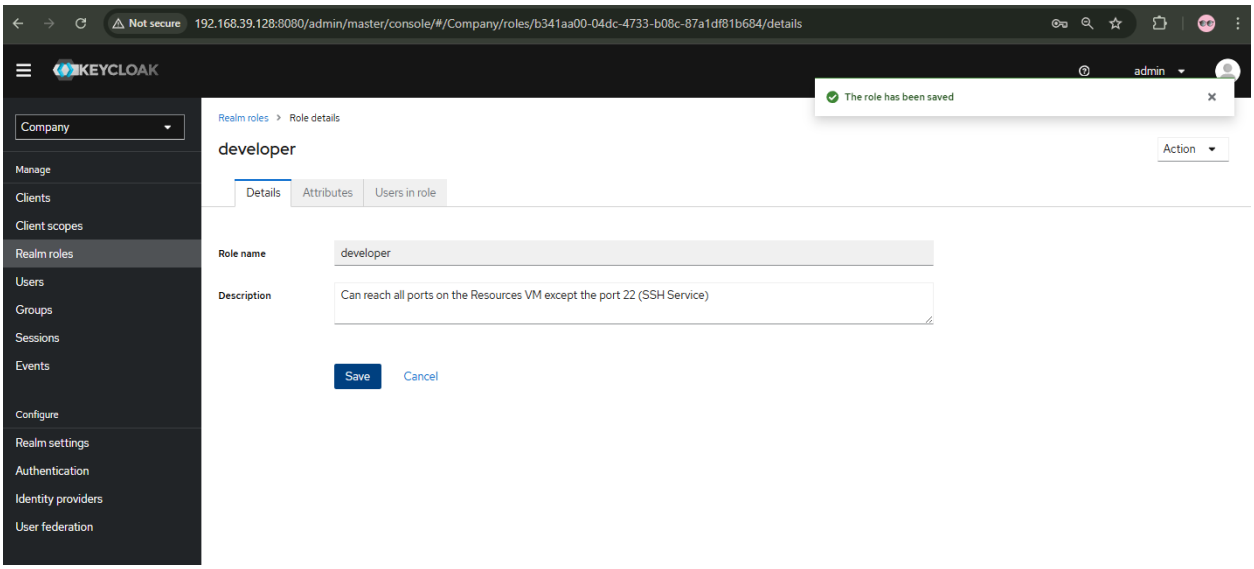
(Developers)، وثمانية ضيوف (Guests).

3.1.4.5 تكوين الأدوار – Roles

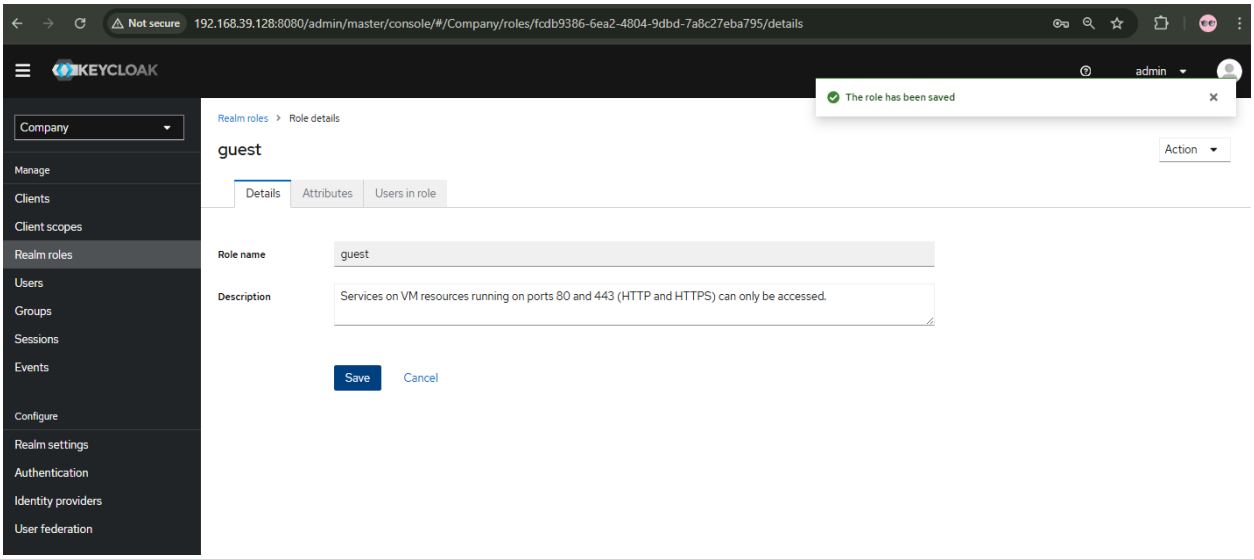
من خلال الخيار Realm Roles يمكن ضبط الأدوار في من النطاق Company فيما يلي تم تعيين ثلاثة أدوار هم المسؤول (admin) والمطور (developer) و الضيف (guest). يوضح الشكل (15) و(16) و(17) أضافة كل منهم مع إعطاء وصف لكل منهم كما هو مبين:



الشكل 15إضافة دور المسؤول

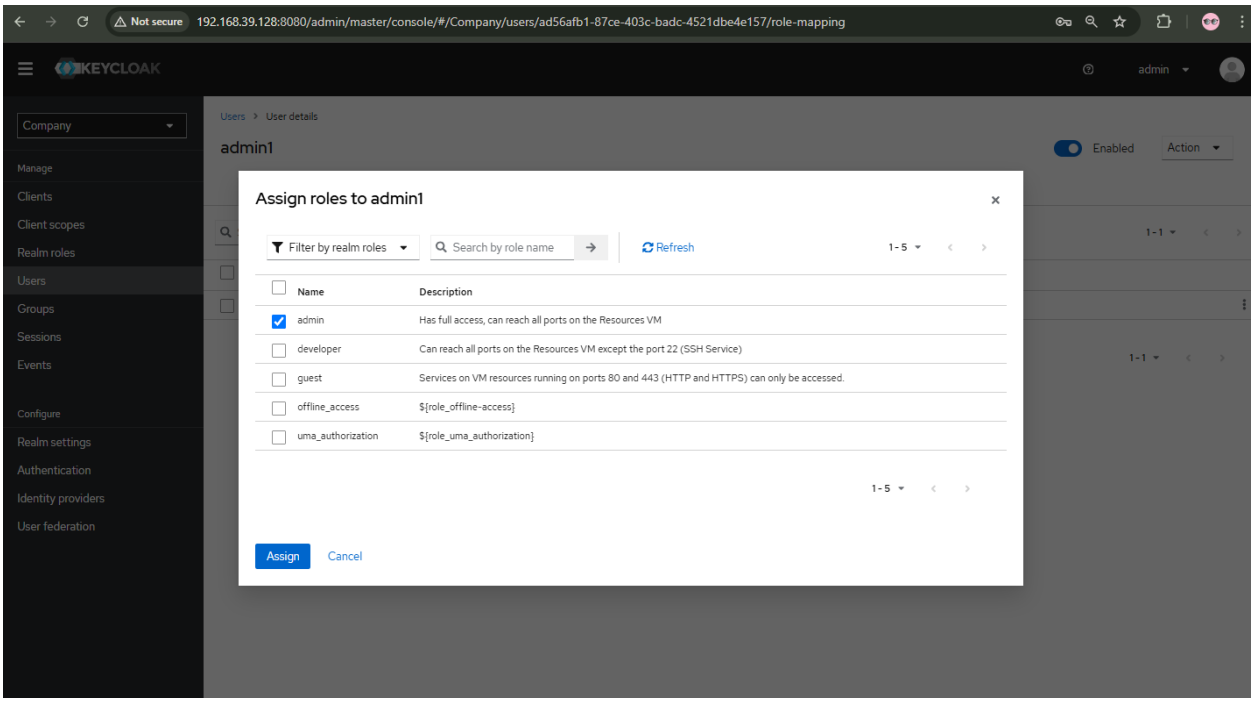


الشكل 16إضافة دور المطور



الشكل 18إضافة دور الضيف

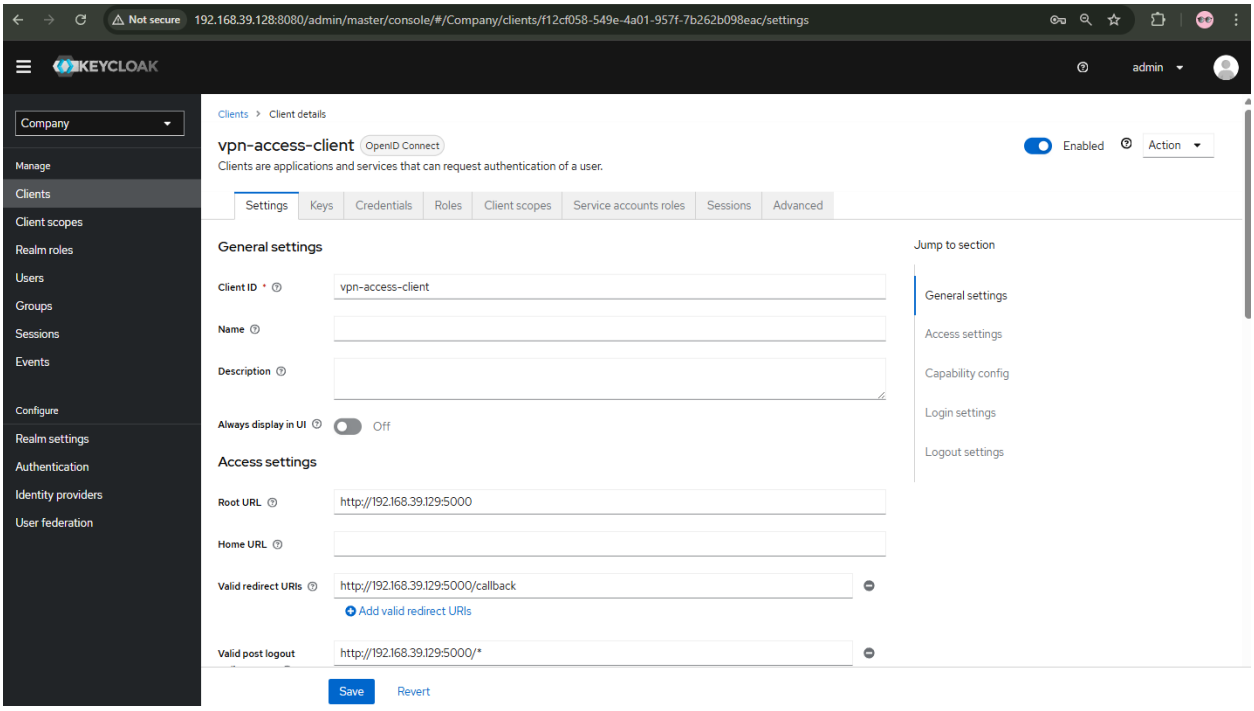
ثم نقوم بإجراء مقابلة كل مستخدم مع الدور المناسب له كما هو موضح بالشكل (18) حيث تم هنا إعطاء دور المسؤول (admin) للمستخدم (admin1). ثم نقوم بنفس العملية من أجل جميع المستخدمين.



الشكل 17إعطاء المستخدم admin1 دور admin

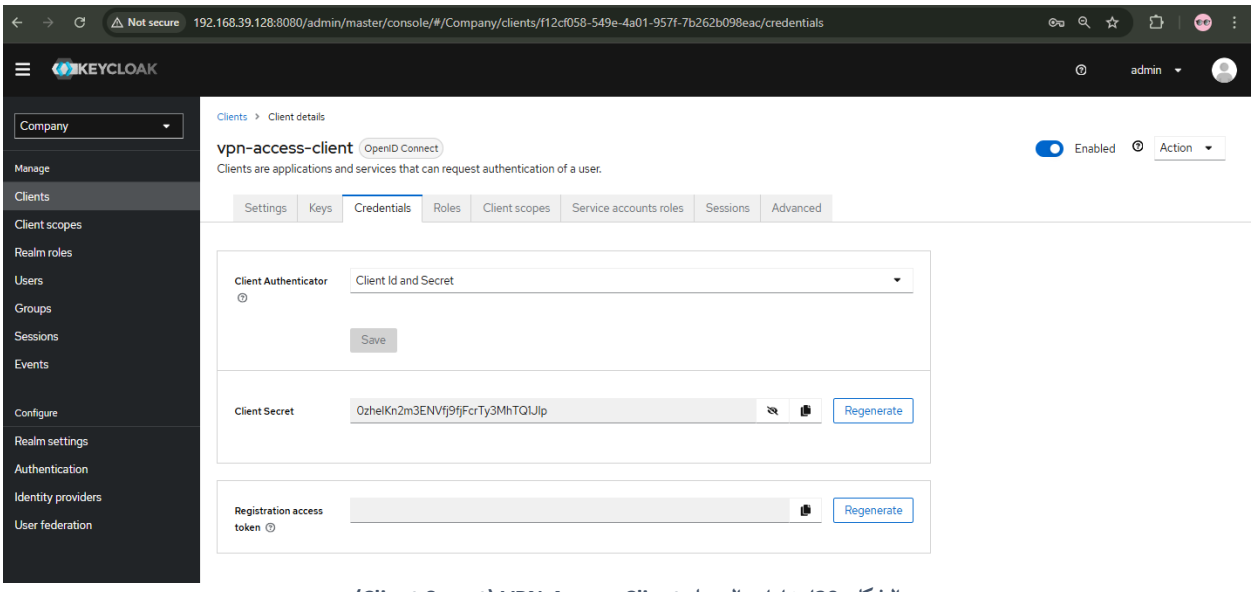
4.1.4.5 تكوين العملاء -Clients

يقصد هنا بالعملاء هم الخدمات و التطبيقات التي يمكنها طلب مصادقة المستخدم وفي سياق مشروعنا يشار الى بوابة الوصول الذي يقوم بتزويد المستخدم بنفق آمن بعد تحقق شروط المصادقة وتطبيق سياسات الوصول. يوضح الشكل (19) كيفية ضبط إعدادات هذا العميل.



الشكل 19إعدادات العميل VPN-Access-Client

حيث يتم إعطاء ID لهذا عميل ومن ثم نقوم بضبط إعدادات الوصول. يكون Root URL الممثل بالعنوان http://192.168.39.129:5000 حيث يمثل هذا العنوان التطبيق الذي سيتم إعادة توجيه المستخدم إليه بعد تسجيل الدخول بنجاح. أما العنوان http://192.168.39.129:5000/callback فيعد هذا الإعداد مهماً جداً من ناحية الأمان، حيث يمنع الهجمات التي تعتمد على إعادة التوجيه إلى مواقع ضارة. عادةً ما يكون المسار callback/هو النقطة التي يستقبل فيها التطبيق رمز التفويض(Authorization Code) ، ثم يستخدمه للحصول على الرموز (Access Token, ID Token). حيث أن Keycloak لن يقبل أي عنوان إعادة توجيه غير مسجل هنا. أما Valid Post Logout الممثل بالعنوان http://192.168.39.129:5000/* يمكّن التطبيق من إعادة توجيه المستخدم الى الصفحة الرئيسية أو الى صفحة تؤكد



الشكل 20إعدادات العميل VPN-Access-Client (Client Secret)

له أنه تم تسجيل الخروج بنجاح. بالانتقال للخيار Credentials كما هو موضح بالشكل (20) نجد القيمة Client Secret التي يتم إنشاؤها عند إنشاء العميل من نوع Confidential وتستخدم كوسيلة مصادقة بين العميل (التطبيق) و Keycloak أثناء تبادل الرموز. حيث يحتاج keycloak الى التأكد من ان التطبيق الذي يطلب الرموز هو تطبيق موثوق حيث يقوم بالتحقق من هوية التطبيق فعندما يرسل التطبيق طلبا للحصول على رموز الوصول (Access Token) باستخدام Authorization Code، فانه يرفق معه Client ID وClient Secret.

بعد تكوين كل من المستخدمين والأدوار وإجراء مقابلة بين كل مستخدم والدور الذي يشغله وتعريف العميل الممثل ببوابة الوصول ننتقل الآن لضبط إعدادات الرموز.

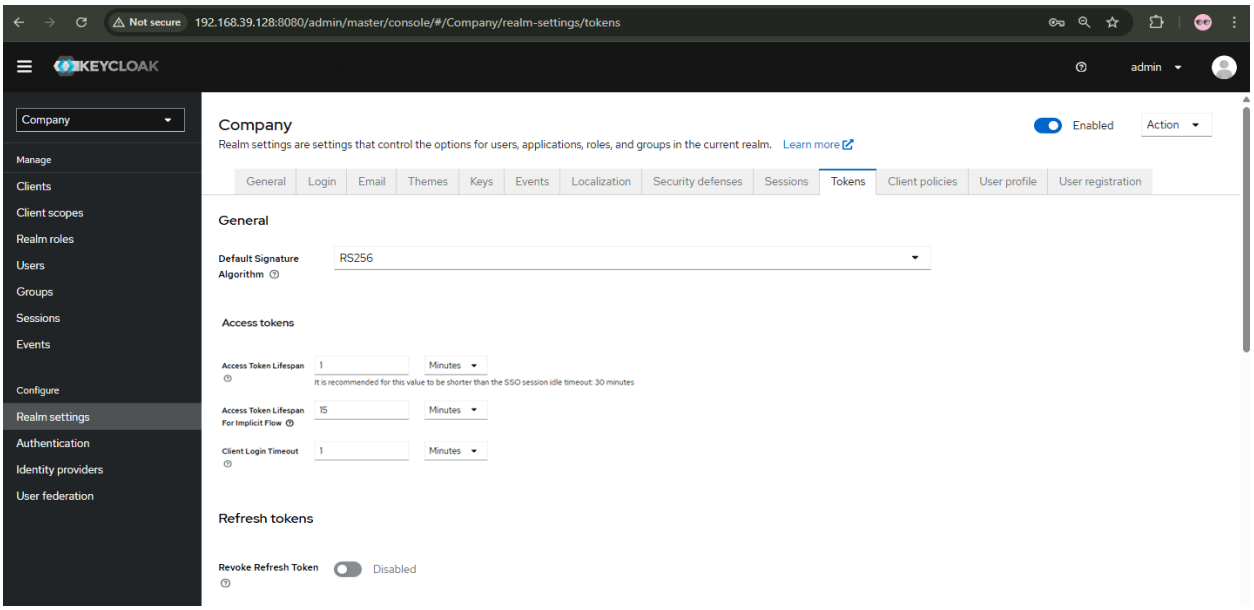
5.1.4.5 تكوين الرموز – Tokens

ضمن إعدادات العميل في keycloak يعد قسم **Tokens** من أهم الأجزاء التي تتحكم في طريقة إنشاء الرموز (Tokens) وإدارتها. تعتمد هذه الرموز على بروتوكول **OpenID Connect**، ويتم استخدامها للتحكم بالوصول إلى الموارد المحمية وتحديد هوية المستخدم.

يقوم Keycloak بإصدار عدة أنواع من الرموز، من أبرزها:

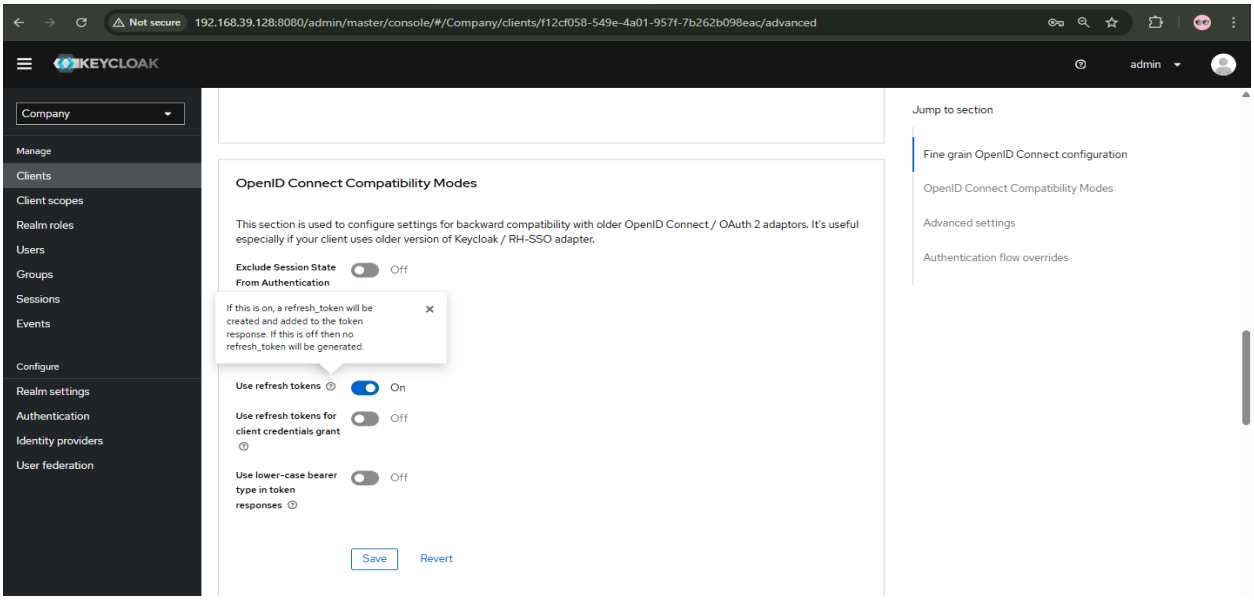
- Access Token: يستخدم للوصول إلى الموارد (مثل واجهات برمجة التطبيقات)، ويحتوي على معلومات مثل هوية المستخدم وصلاحياته.
- ID Token: يحتوي على معلومات عن المستخدم (claims) تستخدم عادة في تطبيقات الويب.
- Refresh Token: يستخدم لتجديد صلاحية الوصول دون الحاجة لإعادة تسجيل الدخول.

يعرض الشكل (21) إدارة الرموز ضمن خيار Realm settings، حيث تم استخدام خوارزمية التوقيع RSA256 وهي خوارزمية تشفير غير متماثل (مفتاح عام/خاص). مما يضمن عدم تعديل أو تزوير الرموز من قبل أطراف خارجية. مدى حياة رموز الوصول (Access Token Lifespan) تشير الى المدة الزمنية التي يكون فيها رمز الوصول صالحا قبل ان تنتهي صلاحيته في مشروعنا تم ضبطها على دقيقة واحدة مما يعني أن التطبيق يحتاج الى استخدام Refresh Tokenلتجديده



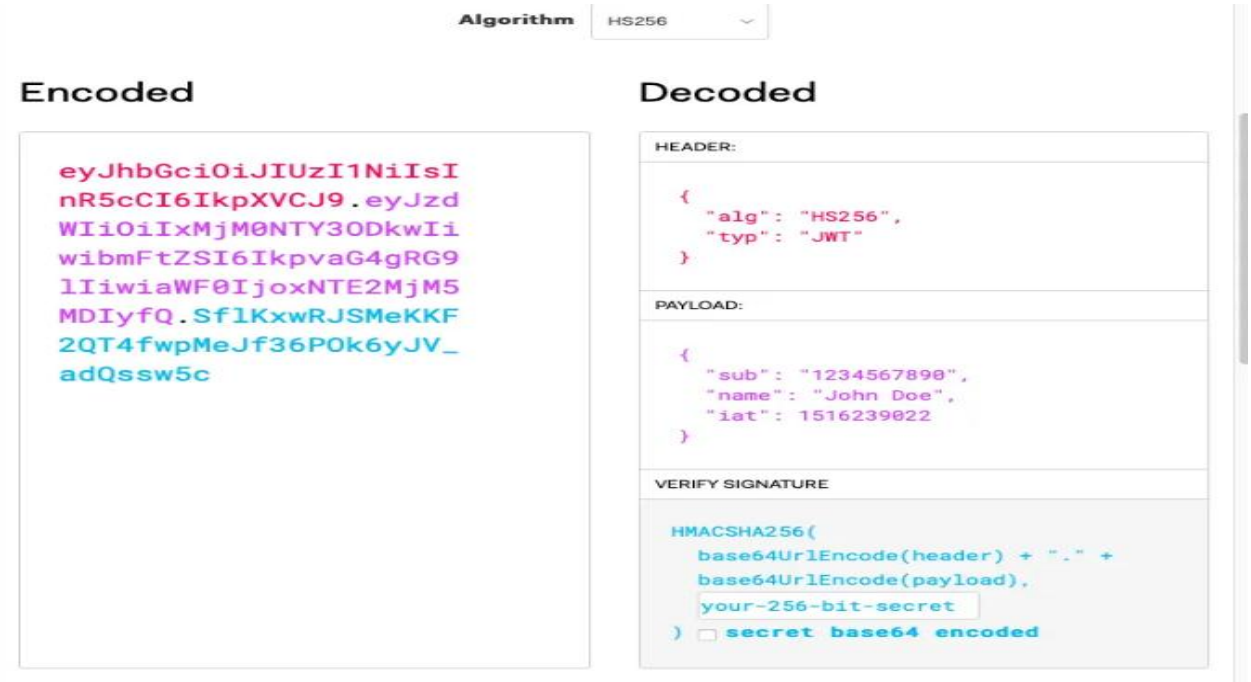
الشكل 21ضبط خصائص الرموز Tokens

بعد هذه المدة. يوضح الشكل (22)كيفية تفعيل استخدام Refresh Tokens وذلك عبر الخيار Clients ومن ثم اختيار العميل VPN-Access-Client.



الشكل 22تفعيل استخدام Refresh Tokens

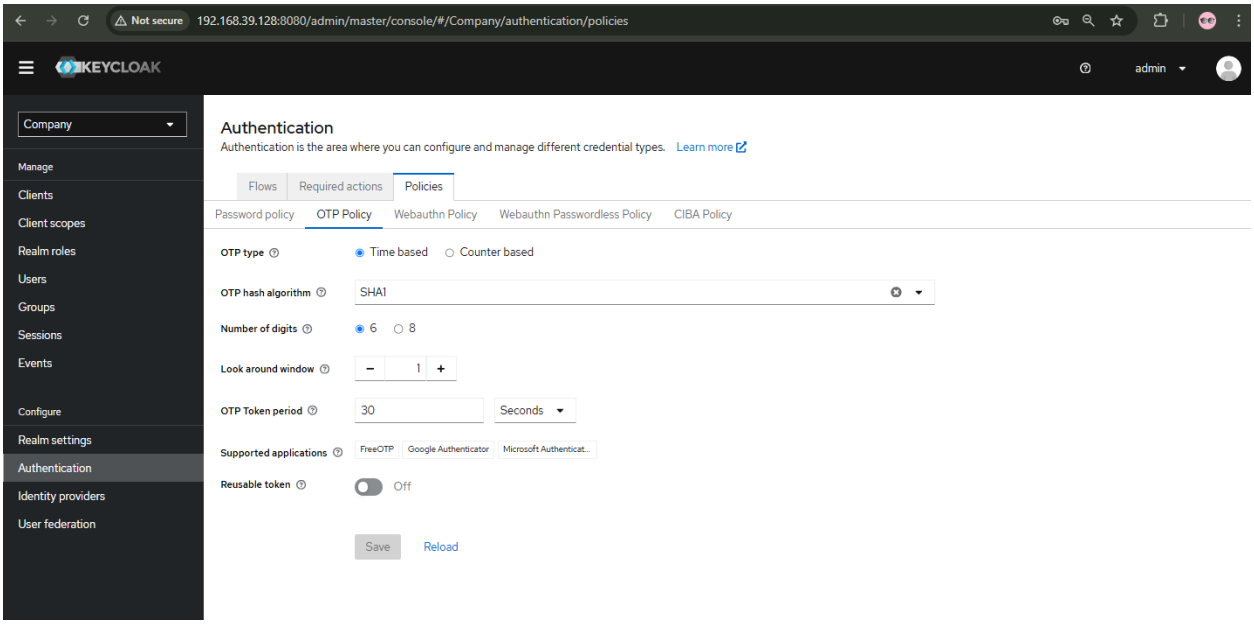
يعرض الشكل (23) الأقسام الرئيسية لـ JWT، بحيث تضم كل من الترويسة (Header) والبيانات (Payload) والتوقيع الرقمي (Signature) [12]:



الشكل 23الأقسام الرئيسية لـ JWT

6.1.4.5 تكوين المصادقة متعددة العوامل – TOTP (Time-based One-Time Password)

من خلال الخيار Authentication يمكن ضبط إعدادات تكوين المصادقة متعددة العوامل من خلال إنشاء كلمات المرور لمرة واحدة المعتمدة على الزمن. حيث يوضح الشكل (24) نوع كلمات المرور لمرة واحدة وخوارزمية التهشير المتبعة في إنشاء كلمة المرور وعدد خانات كلمة المرور والمدة الزمنية لصلاحيتها والتطبيقات التي تدعم استخدام هذه الميزة. حيث تم



الشكل 24ضبط إعدادات كلمة المرور لمرة واحدة TOTP

استخدام خوارزمية تهشير SHA1 بحيث تقوم هذه الخوارزمية بتوليد رمز لا يمكن عكسه (أي لا يمكن من الرمز معرفة المفتاح المشترك السري بين keycloak وجهاز المستخدم) وتحقيق النزاهة حيث لا يمكن تعديل الرمز دون الكشف عن التلاعب و توفير أمان كافي لتقنيات المصادقة الثنائية (2FA). وقمنا باختيار عدد خانات كلمة المرور من ستة خانات ومن ثم وضعنا مدة صلاحية هذه الكلمة 30 ثانية. فعند تسجيل الدخول يقوم المستخدم بإدخال اسم المستخدم او البريد الالكتروني وكلمة المرور وبعدها يقوم بمسح QR Code من خلال التطبيق Google Authenticator على جوال المستخدم بحيث يقوم بتوليد كلمة المرور لمرة واحدة وذلك اثناء عملية تسجيل الدخول لأول مرة وفي عمليات تسجيل الدخول اللاحقة تظهر واجهة ادخال كلمة المرور لمرة واحدة مباشرة بعد تسجيل اسم المستخدم وكلمة المرور.

2.4.5 تكوين الأداة – WireGuard

يمثل WireGuard مزود النفق المشفر الذي يسمح للمستخدمين بالوصول الى المورد المطلوب بعد عملية المصادقة والتخويل بناء على دور المستخدم. تم تنصيب الأداة على الآلة الافتراضية الموافقة ذات العنوان 192.168.39.129 وعلى التجهيزة الفيزيائية التي ستقوم لاحقا بدور العميل الذي يقوم بطلب الخدمة. جميع الخطوات التالية تم تنفيذها على الآلة الافتراضية Backend VM.

1.2.4.5 تكوين مخدم الـ VPN Server- VPN

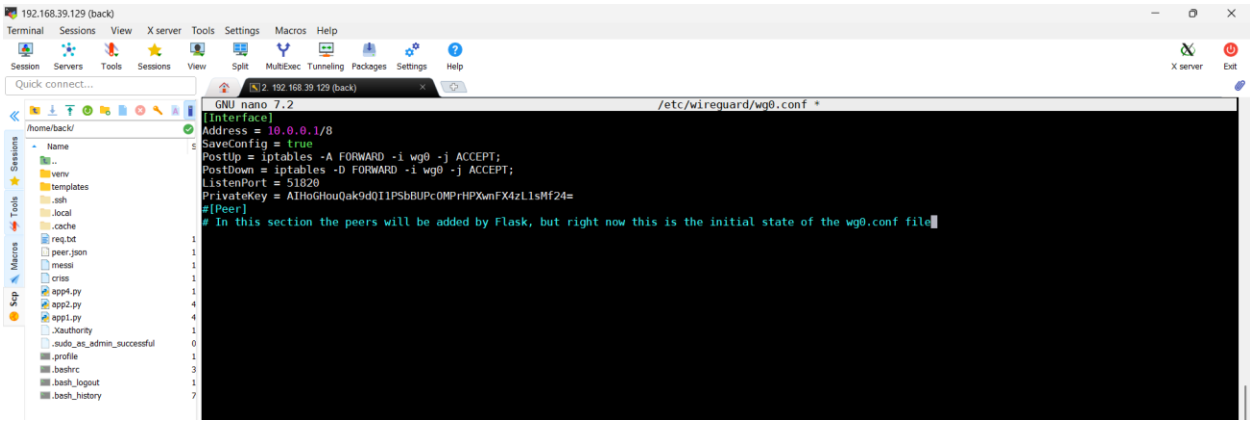
تم إعداد المخدم ليكون نقطة الدخول المركزية للشبكة حيث تم اجراء الخطوات التالية:

توليد زوج المفاتيح (عام/خاص) للمخدم عبر تنفيذ التعليمة الموضحة بالشكل (25).

```
back@back:~$ wg genkey | tee server_private.key | wg pubkey > server_public.key
```

الشكل 25توليد زوج مفاتيح التشفير وفك التشفير لمخدم wireguard

ومن ثم يتم إنشاء ملف إعداد المخدم في المسار التالي /etc/wireguard/wg0.conf حيث يحوي هذا الملف على المعلومات التالية الموضحة بالشكل (26).



الشكل 26ملف إعداد المخدم wg0.conf

[Interface]: يشير إلى إعدادات واجهة الشبكة الافتراضية (الداخلية) الخاصة بـ wireguard.

PrivateKey: هو المفتاح الخاص بالخادم الذي تم توليده في الخطوة السابقة.

Address: هو عنوان IP الداخلي الذي سيتم منحه للخادم داخل شبكة VPN هنا (10.0.0.1/24).

ListenPort: هو المنفذ الذي سيستمع عليه الخادم للاتصالات القادمة من العملاء(UDP 51820).

PostUp وPostDown: هاتين التعليمتين تمثلان قاعدتي جدار ناري تقوم بالسماح للبيانات بالمرور عبر الواجهة wg0.

أما في قسم الـ Peer فيتم لاحقا إضافة العملاء في هذا القسم (سيتم شرح ذلك في قسم تكوين الأداة Flask).

بعدها يتم تفعيل واجهة الشبكة الافتراضية wg0 بناء على إعدادات الملف wg0.conf عبر التعليمة الموضحة بالشكل (27). وبعد تشغيل العميل لملف الإعدادات username.conf (الذي سنقوم بشرح الية توليده واضافته في قسم تكوين

```
back@back:~$ sudo wg-quick up wg0
[sudo] password for back:
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.1/8 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] iptables -A FORWARD -i wg0 -j ACCEPT;
back@back:~$
```

الشكل 27تفعيل واجهة الشبكة الافتراضية wg0

الأداة Flask) يأخذ العميل عنوان ضمن الشبكة 10.0.0.0/8 وبالتالي يتم انشاء نفق مشفر بين العميل ومخدم الـ VPN. سيتم أيضا شرح كيفية تعيين فضاء العناوين لكل دور في التطبيق Flask.

3.4.5 تكوين تطبيق Flask – (Access Gateway Application)

يعد تطبيق **Flask** أحد المكونات الأساسية في تنفيذ نموذج Zero Trust ، حيث يعمل كـ **بوابة وصول** مسؤولة عن التحقق من هوية المستخدمين، فرض السياسات الأمنية، وإدارة الاتصال الآمن عبر VPN بناءً على صلاحيات كل مستخدم.

يتمركز دور هذا التطبيق بين نظام التحقق من الهوية (Keycloak) ونظام النفق الآمن(WireGuard) ، مما يتيح تحقيق مبدأ "عدم الثقة الضمنية" من خلال التأكد المستمر من الجلسة وقيود الوصول. تم تشغيل التطبيق Flask على الآلة الإفتراضية Backend VM ذات العنوان 192.168.39.129 على المنفذ 5000. يعرض الشكل (28) المتغيرات الأساسية التي تم تعريفها في بداية تطبيق الـ Flask، حيث تشمل إعدادات Keycloak وWireGuard ومسارات البروتوكول OAuth2.

```
# --- Keycloak Config ---
KEYCLOAK_URL = "http://192.168.39.128:8080"
REALM = "Company"
CLIENT_ID = "vpn-access-client"
CLIENT_SECRET = "0zheIKn2m3ENVfj9fjFcrTy3MhTQ1JIp"
REDIRECT_URI = "http://192.168.39.129:5000/callback"

# --- OAuth2 Endpoints ---
AUTH_URL = f"{KEYCLOAK_URL}/realms/{REALM}/protocol/openid-connect/auth"
TOKEN_URL = f"{KEYCLOAK_URL}/realms/{REALM}/protocol/openid-connect/token"
JWKS_URL = f"{KEYCLOAK_URL}/realms/{REALM}/protocol/openid-connect/certs"

# --- WireGuard Config ---
WG_SERVER_PUBLIC_KEY = "TJmL+jaJX546+h2F1TOgid4EpRcbqoG5xgokLJS2By8="
WG_ENDPOINT = "192.168.39.129:51820"
WG_ALLOWED_IPS = "192.168.139.128/32"
WG_INTERFACE = "wg0"
PEER_DB = "peer.json"
```

الشكل 28تهيئة المتغيرات الأساسية لتكامل عمل النظام

حيث تم تعريف مجموعة من المتغيرات الثابتة التي تستخدم لربط نظام Flask بمكونات البنية المعمارية الأخرى، وتحديداً مزود الهوية **Keycloak** ونظام النفق المشفر **WireGuard.** يمثّل هذا القسم نقطة الانطلاق التي تربط منطق التطبيق الداخلي بالمصادر الخارجية، ويحتوي على عناصر تكوين مهمة تستخدم في كل من المصادقة، إصدار الرموز، وإعدادات الاتصال.

إعدادات Keycloak :

- **KEYCLOAK_URL:** عنوان مخدم Keycloak .
- **REALM**: اسم النطاق الذي يستخدم لفصل سياسات وهوية المؤسسة.
- **CLIENT_ID / CLIENT_SECRET:** بيانات اعتماد العميل المسجل ضمن Keycloak، والتي تستخدم خلال عملية تبادل الرموز.
- **REDIRECT_URI:** عنوان نقطة النهاية في Flask التي تستقبل رمز التفويض (authorization_code) بعد نجاح المصادقة.

نقاط النهاية لبروتوكول OAuth2:

- **AUTH_URL**: عنوان صفحة تسجيل الدخول ضمن OpenID Connect .
- **TOKEN_URL**: نقطة تبادل authorization_code للحصول على رموز access وrefresh.
- **JWKS_URL**: نقطة تحميل المفاتيح العامة (JWKS) المستخدمة للتحقق من توقيع الرموز المصدّقة.

إعدادات WireGuard :

- **WG_SERVER_PUBLIC_KEY:** المفتاح العام الخاص بمخدم WireGuard، ويدمج داخل إعدادات العميل.
- **WG_ENDPOINT:** عنوان خادم VPN الذي سيتصل به العميل.
- **WG_ALLOWED_IPS:** النطاق المسموح بالوصول إليه من خلال النفق، ويمثل الموارد المحمية Resource VM ذات العنوان 192.168.139.128.
- **WG_INTERFACE:** اسم واجهة WireGuard التي يتم تفعيل الأقران عليها (wg0).
- **PEER_DB**: اسم الملف المستخدم لتخزين معلومات النظائر (Peers) مثل حالة التمكين، IP، توقيت الانتهاء.

1.3.4.5 تسجيل الدخول باستخدام OAuth2 وتكامل المصادقة مع Keycloak

تعد المصادقة (Authentication) الخطوة الأولى والأكثر حساسية في نموذج الثقة الصفرية(ZTNA). في هذا السياق، يعتمد النظام على بروتوكول **OAuth2** لتنفيذ تدفق التفويض (Authorization Code Flow) بطريقة آمنة، ومركزية، ومستقلة عن التطبيق الرئيسي.

عند قيام المستخدم بطلب الوصول إلى النظام (في المتصفح عند طلب العنوان http://192.168.39.129:5000)، يتم أولاً إعادة توجيهه إلى صفحة تسجيل الدخول الخاصة بـ **Keycloak**، والذي يعمل كمزود هوية (Identity Provider - IdP). يتضمن هذا التفاعل أربع مراحل أساسية:

1. إعادة التوجيه إلى Keycloak :

- يقوم تطبيق Flask بتنفيذ توجيه (Redirect) إلى رابط التفويض الخاص بـ Keycloak. يتم إرسال الرموز الأساسية مثل client_id ، redirect_uri، و scope إلى خادم المصادقة. يعرض الشكل (29) التابع المسؤول عن إعادة التوجيه الى keycloak.

```
@app.route("/login")
def login():
    return redirect(
        f"{AUTH_URL}?client_id={CLIENT_ID}&response_type=code&scope=openid&redirect_uri={REDIRECT_URI}"
    )
```

الشكل 29التابع المسؤول عن إعادة التوجيه إلى keycloak

2. استقبال رمز التفويض(Authorization Code) :

- بعد إدخال المستخدم لبيانات الاعتماد بنجاح، يعيد Keycloak توجيه المستخدم إلى المسار /callback داخل تطبيق Flask .
- يحتوي هذا التوجيه على رمز مؤقت يعرف بـ authorization_code.

3. تبادل الرمز للحصول على الرموز المميزة(Tokens) :

- يقوم تطبيق Flask بإرسال طلب خلفي (Backend Request) إلى Keycloak لاستبدال authorization_code بـ:
  - **Access Token**: يستخدم لتأكيد هوية المستخدم خلال الجلسة.

```
@app.route("/callback")
def callback():
    code = request.args.get("code")
    if not code:
        return "Error: no code in callback", 400

    data = {
        "grant_type": "authorization_code",
        "code": code,
        "redirect_uri": REDIRECT_URI,
        "client_id": CLIENT_ID,
        "client_secret": CLIENT_SECRET,
    }

    token_response = requests.post(TOKEN_URL, data=data)
    token_json = token_response.json()
    access_token = token_json.get("access_token")
    refresh_token = token_json.get("refresh_token")

    if not access_token:
        return "Login failed", 401
```

الشكل 30 استقبال رمز التفويض والية الحصول على الرموز المميزة

- **Refresh Token**: يستخدم لتجديد Access Token عند قرب انتهاء صلاحيته دون الحاجة لإعادة تسجيل الدخول. يعرض الشكل (30) جزء من التابع الذي يقوم بعملية استقبال رمز التفويض والية تبادل الرمز للحصول على الرموز المميزة.

4. استخلاص الهوية وتخزين الجلسة:

- يتم تحليل محتوى Access Token باستخدام مكتبة (PyJWT) لاستخراج المعلومات المتعلقة بهوية المستخدم (مثل preferred_username وroles).
- تخزن هذه البيانات ضمن الكائن session الخاص بـ Flask ، والذي يستخدم طوال فترة التفاعل مع التطبيق. يعرض الشكل (31) تكملة للتابع ()callback الذي يقوم باستخلاص الهوية وتخزين الجلسة.

```
    session["access_token"] = access_token
    session["refresh_token"] = refresh_token

    # Parse token for username and roles
    jwks_client = PyJWKClient(JWKS_URL)
    signing_key = jwks_client.get_signing_key_from_jwt(access_token).key
    decoded = jwt.decode(access_token, signing_key, algorithms=["RS256"], options={"verify_aud": False})
    username = decoded.get("preferred_username")
    roles = decoded.get("realm_access", {}).get("roles", [])

    session["username"] = username
    session["roles"] = roles
```

الشكل 31استخلاص الهوية وتخزين الجلسة

2.3.4.5 التحقق المستمر من صلاحية الجلسة

في سياق نموذج ZTNA لا يعد التحقق من الهوية حدثاً يحدث لمرة واحدة عند تسجيل الدخول، بل هو **عملية مستمرة** ترافق كل تفاعل مع النظام. لضمان هذا السلوك، يعتمد تطبيق Flask على Decorator يدعى @token_required يتم ربطه بأي مسار (Route) يتطلب التحقق من الهوية. يقوم @token_required بتنفيذ سلسلة من التحققات التي تضمن بقاء الجلسة فعالة وآمنة وتضمن المراحل التالية:

1. التحقق من وجود الجلسة والرموز:

- إذا لم يكن access_token موجوداً في الجلسة، يعتبر المستخدم غير مصادق عليه.

- تتم إعادة توجيهه مباشرة إلى login/لإعادة المصادقة. يعرض الشكل (32) آلية التحقق من وجود الجلسة والرموز

```
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = session.get("access_token")
        refresh_token = session.get("refresh_token")

        if not token:
            print("[Session] No access token found.")
            return redirect("/login")
```

الشكل 32آلية التحقق من وجود الجلسة والرموز

2. التحقق من مهلة الجلسة(Hard Timeout) :

- يتم تخزين توقيت بداية الجلسة (session_start) في الجلسة.
- إذا تجاوزت الجلسة مدة 3 دقائق (180 ثانية)، يتم قطع الاتصال عبر تعطيل العميل (Peer) (سيتم شرح كيفية قطع الاتصال في المراحل التالية لهذه المرحلة) وإلغاء الجلسة (يتم التحكم بهذه المدة حسب متطلبات النظام قمنا بوضعها 3 دقائق لسهولة الاختبار). وعندها يجبر المستخدم على تسجيل الخروج عبر إعادة توجيهه الى المسار logout/ وتعديل المعاملات في الملف peer.json (سيتم أيضا شرح هذا الملف وفائدته في عملية المصادقة المستمرة). يعرض الشكل (33) آلية التحقق من مهلة الجلسة وقطع الاتصال فور انتهاء المهلة المحددة.

```
        # ? Hard session limit (e.g., 3 minutes)
        session_start = session.get("session_start", 0)
        if time.time() - session_start > 180:  # 3 * 60
            print("[Session] Hard session expired, force logout.")
            # ? Disable peer
            username = session.get("username")
            peers = load_peers()
            if username in peers:
                peer = peers[username]
                if peer.get("enabled"):
                    print(f"[Session Expired] Disabling peer for {username}")
                    disable_peer(peer["public_key"], peer["ip"])
                    peer["enabled"] = False
                    save_peers(peers)

            session.clear()
            return redirect("/logout")
```

الشكل 33آلية التحقق من مهلة الجلسة وقطع الاتصال فور انتهائها

3. التحقق من صلاحية الرموز:

- يتم التحقق من تاريخ انتهاء refresh_token.
- إذا انتهت صلاحيته، تعتبر الجلسة منتهية ويتم تسجيل الخروج تلقائياً. يعرض الشكل (34) التحقق من صلاحية الرموز وقطع الاتصال بحال انتهاء الصلاحية بنفس آلية قطع الاتصال المذكورة أعلاه.

```
136         # ? Refresh token expiry check
137         refresh_exp = session.get("refresh_token_exp", 0)
138         if time.time() > refresh_exp:
139             print("[Token] Refresh token expired, force logout.")
140             # ? Disable peer
141             username = session.get("username")
142             peers = load_peers()
143             if username in peers:
144                 peer = peers[username]
145                 if peer.get("enabled"):
146                     print(f"[Refresh Expired] Disabling peer for {username}")
147                     disable_peer(peer["public_key"], peer["ip"])
148                     peer["enabled"] = False
149                     save_peers(peers)
150             session.clear()
151             return redirect("/logout")
```

الشكل 34التحقق من صلاحية الرموز

4. تجديد رمز الوصول عند الحاجة:

- يتم تقييم ما إذا كان access_token يقترب من الانتهاء باستخدام التابع ()should_refresh_token. يوضح الشكل (35) التابع المسؤول عن تقييم صلاحية الرمز (Access Token) وذلك من خلال المعامل access_token_exp المخزن ضمن session الذي يمثل لحظة انتهاء صلاحية رمز الوصول الذي تم الحصول عليه سابقا من keycloak، إذا لم يكن هناك تاريخ انتهاء مخزن (لأي سبب كان)، يقوم التابع بإرجاع **True**، مما يعني أنه من الأفضل تجديد الرمز كإجراء احترازي لضمان استمرار الأمان ومن ثم يتم حساب الوقت المتبقي حيث يقوم بحساب عدد الثواني المتبقية قبل انتهاء صلاحية الرمز، من خلال طرح الوقت الحالي ()time.time من وقت الانتهاءexp. إذا الوقت المتبقي أقل من 30 ثانية يرجع التابع **True** ما يشير إلى أن النظام يجب أن **يبدأ عملية التجديد** لتجنّب انقطاع الجلسة.

```
100 def should_refresh_token():
101     exp = session.get("access_token_exp")
102     if not exp:
103         return True  # If no expiry stored, better to refresh
104 
105     remaining = exp - int(time.time())
106     return remaining < 30  # Refresh only if access token will expire in next 30 seconds
```

الشكل 35تقييم صلاحية الرمز Access Token

- يتم إرسال طلب تجديد إلى Keycloak باستخدام refresh_token، ويتم تحديث الرموز الجديدة في الجلسة. يعرض الشكل (36) آلية تحديث رمز الوصول عند الحاجة (بنفس طريقة طلب الحصول على رمز الوصول ولكن هنا للحصول على الرمز المحدث Refresh Token).

```
153         # ? Refresh access token if it's about to expire
154         if should_refresh_token():
155             print("[Access Token] About to expire, refreshing...")
156             if not refresh_token:
157                 print("[Token] No refresh token found.")
158                 session.clear()
159                 return redirect("/logout")
160
161             data = {
162                 "grant_type": "refresh_token",
163                 "client_id": CLIENT_ID,
164                 "client_secret": CLIENT_SECRET,
165                 "refresh_token": refresh_token
166             }
167             resp = requests.post(TOKEN_URL, data=data)
168             if resp.status_code != 200:
169                 print("[Token Refresh] Failed to refresh token.")
170                 session.clear()
171                 return redirect("/logout")
```

الشكل 36آلية طلب الحصول على الرمز المحدث Refresh Token

3.3.4.5 توليد وتحميل ملف إعدادات WireGuard – (VPN Configuration Generation)

بمجرد مصادقة المستخدم والتحقق من الجلسة، يصبح بإمكانه تحميل إعداد الاتصال بـ WireGuard، وهو البروتوكول المسؤول عن إنشاء النفق الآمن (Secure Tunnel). تنفّذ هذه الوظيفة عبر المسار download-config/ في تطبيق Flask(يتم ذلك عن طريق صفحة الويب)، والذي يقوم بتنفيذ الآتي:

1. التحقق من صلاحية الجلسة:

- يتم أولاً تمرير الطلب من خلال @token_required، والذي يضمن أن الجلسة نشطة والرموز صالحة.

2. استدعاء بيانات العميل من الملف peer.json:

- يتم البحث عن سجل العميل المرتبط باسم المستخدم داخل الملف peer.json

3. توليد عميل (Peer) جديد إن لزم:

- إذا لم يكن هناك عميل نشط أو انتهت صلاحيته، يتم استدعاء التابع generate_peer() لتوليد مفاتيح جديدة وتعيين عنوان IP جديد بناءً على الدور(Role) .

يعرض الشكل (37) الخطوات السابقة.

```
319 @app.route("/download-config")
320 @token_required
321 def download_config():
322     username = session.get("username")
323     roles = session.get("roles")
324     peers = load_peers()
325
326     if username in peers:
327         peer = peers[username]
328         if peer.get("expires_at", 0) < time.time():
329           print(f"[Download] Peer expired for {username}, regenerating...")
330           peer = generate_peer(username, roles)
331     else:
332         peer = generate_peer(username, roles)
```

الشكل 37توليد ملف إعدادات الـ VPN

4. إنشاء محتوى ملف conf.:

بعد تنفيذ الخطوات السابقة يتم إنشاء محتوى ملف الإعدادات conf. ، حيث يعرض الشكل (38)محتوى هذا الملف:

```
    config_content = f"""
[Interface]
PrivateKey = {peer['private_key']}
Address = {peer['ip']}
DNS = 10.0.0.1

[Peer]
PublicKey = {WG_SERVER_PUBLIC_KEY}
Endpoint = {WG_ENDPOINT}
AllowedIPs = {WG_ALLOWED_IPS}
PersistentKeepalive = 25
"""
```

الشكل 38محتوى ملف الإعدادات conf.

يتم إنشاء ملف إعداد WireGuard بصيغة معيارية، يتضمّن:

- المفتاح الخاص للمستخدم.
- عنوان IP ثابت مخصص له.
- عنوان خادم WireGuard والمفتاح العام له.
- النطاقات المسموح له بالوصول إليها (عنوان الآلة الافتراضية Resource VM).
- إعداد Keepalive للحفاظ على استقرار الاتصال.

5. إرسال الملف للمستخدم:

من خلال واجهة تطبيق الويب يمكن للمستخدم تحميل الملف وتخزينه.

4.3.4.5 إنشاء القرين وإضافته لمخدم WireGuard – (Peer Generation)

تمثل عملية **توليد القرين (Peer Generation)** خطوة محورية في تنفيذ نموذج ZTNA، حيث يتم من خلالها ربط هوية المستخدم الرقمية (كما تم التحقق منها عبر Keycloak) بهوية شبكية خاصة وفريدة داخل نفق WireGuard، تستخدم لتطبيق قواعد التحكم في الوصول والتجزئة، والتتبع الأمني.

يتم تنفيذ هذه الوظيفة تلقائياً عند تسجيل الدخول لأول مرة أو انتهاء صلاحية القرين الحالي أو تحميل إعداد VPN جديد بعد إعادة المصادقة. فيما يلي مراحل التنفيذ:

1. توليد المفاتيح الخاصة والعامة:

يتم استخدام أوامر wg genkey وwg pubkey لتوليد مفاتيح WireGuard بطريقة آمنة ومشفّرة. يعرض الشكل (39) توليد المفاتيح الخاصة والعامة للقرين.

```
def generate_peer(username, roles):
    private_key = subprocess.check_output("wg genkey", shell=True).decode().strip()
    public_key = subprocess.check_output(f"echo {private_key} | wg pubkey", shell=True).decode().strip()
```

الشكل 39توليد مفاتيح التشفير الخاصة والعامة للقرين

2. تحديد عنوان IP حسب الدور:

يتم استدعاء التابع assign_ip()، الذي يربط كل مستخدم بعنوان IP ضمن نطاق 10.10.0.X/32 بناءً على دوره (admin, developer, guest)، ما يمكّن من تطبيق قواعد جدار ناري مختلفة لاحقاً لكل فئة مستخدم. حيث تم تخصيص المجال 10.10.0.16/31 لدور المسؤول (admin) حيث يضم مسؤولين اثنين، والمجال 10.10.0.32/30 لدور المطور

(developer) حيث يضم أربع مطورين، وأخيرا المجال 10.10.0.64/29 لدور الضيف حيث يضم ثمانية ضيوف. يعرض الشكل (40) التابع assign_ip() :

```
def assign_ip(username, roles):
    def hash_index(name, size):
        return sum(ord(c) for c in name) % size

    if "admin" in roles:
        base_ip = 16
        index = hash_index(username, 2)
        return f"10.10.0.{base_ip + index}/32"
    elif "developer" in roles:
        base_ip = 32
        index = hash_index(username, 4)
        return f"10.10.0.{base_ip + index}/32"
    elif "guest" in roles:
        base_ip = 64
        index = hash_index(username, 8)
        return f"10.10.0.{base_ip + index}/32"
    else:
        raise Exception("Unknown role or no valid role assigned.")
```

الشكل 40آلية تخصيص عناوين IP للمستخدمين

3. تخزين بيانات القرين (Peer):

- يتم تخزين معلومات القرين في ملف peer.json، مع تعيين حالة "**enabled= True**" وتحديد مدة صلاحية الجلسة .يبين الشكل (41) كيفية تخزين بيانات المستخدم.

```
    # Set peer expiry time (e.g., 1  X now)
    expiry_minutes = 30
    expires_at = datetime.now(timezone.utc) + timedelta(minutes=expiry_minutes)

    peer = {
        "private_key": private_key,
        "public_key": public_key,
        "ip": client_ip,
        "enabled": True,
        "expires_at": expires_at.timestamp()
    }

    peers = load_peers()
    peers[username] = peer
    save_peers(peers)
    ensure_peer_enabled(public_key, client_ip)
    return peer
```

الشكل 41تخزين بيانات المستخدم في ملف peer.json

4. تفعيل القرين (المستخدم) على واجهة WireGuard

- يتم استدعاء التابع() ensure_peer_enabled حيث يضيف القرين إلى واجهة wg0 باستخدام wg set، بحيث يصبح المستخدم قادراً على الاتصال بعد تحميل ملف الإعداد conf. . يعرض الشكل (42) التعليمة التي يتم تنفيذها لإضافة القرين وتفعليه على واجهة Wireguard.

```
def ensure_peer_enabled(public_key, client_ip):
    subprocess.run([
        "sudo", "wg", "set", WG_INTERFACE,
        "peer", public_key,
        "allowed-ips", client_ip
    ], check=True)
```

الشكل 42 تفعيل الPeer وإضافته على واجهة WireGuard

5.3.4.5 إدارة دورة حياة القرين وتعطيله بعد انتهاء الجلسة

في نموذج ZTNA، لا يسمح بأي اتصال مفتوح إلى أجل غير مسمى، حتى لو تم التحقق من هوية المستخدم في البداية. ولهذا، يجب أن تخضع كل جلسة VPN لرقابة زمنية صارمة، ويتم إنهاؤها تلقائياً عند انتهاء صلاحيتها. هنا يأتي دور التابع ()cleanup_expired_peers، والذي يعتبر **أداة إدارة دورة الحياة(Peer Lifecycle Manager)** . فيما يلي مراحل عمل هذا التابع:

1. تحميل جميع الأقران (Peers)

يتم تحميل محتوى ملف peer.json، الذي يحتوي على كافة بيانات المستخدمين المتصلين، بما في ذلك توقيت انتهاء كل جلسة عن طريق التابع ()load_peers.

2. التحقق من صلاحية كل قرين

إذا كانت حالة القرين enabled = True وكان وقت الانتهاء (expires_at) قد مضى، يتم تنفيذ أمر تعطيله مباشرة عن طريق التابع ()disable_peer. يعرض الشكل (43) آلية التحقق من صلاحية كل قرين فيما يعرض الشكل آلية تعطيل القرين (44).

3. حفظ التحديثات

يتم تحديث ملف peer.json بعد تعديل الحالة

4. تشغيل دوري

يتم تنفيذ التابع دوريا كل دقيقة باستخدام نيسب Thread يعمل بالخلفية.

يعرض الشكل (43) التابع ()cleanup_expired_peers.

```
371 def cleanup_expired_peers():
372     peers = load_peers()
373     now = time.time()
374     changed = False
375     for username, peer in peers.items():
376         if peer.get("enabled") and peer.get("expires_at", 0) < now:
377             print(f"[Cleanup] Disabling expired peer: {username}")
378             disable_peer(peer["public_key"], peer["ip"])
379             peer["enabled"] = False
380             changed = True
381     if changed:
382         save_peers(peers)
```

الشكل 43 إدارة دورة حياة القرين وتعطيله بعد انتهاء الجلسة

```
97 def disable_peer(public_key, client_ip):
98     subprocess.run([
99         "sudo", "wg", "set", WG_INTERFACE, "peer", public_key, "allowed-ips", client_ip, "remove"], check=True)
```

الشكل 44 التابع المسؤول عن تعطيل القرين

6.3.4.5 تسجيل الخروج وتعطيل القرين يدويا

يمثل تسجيل الخروج نقطة إنهاء صريحة في دورة حياة جلسة VPN ضمن بيئة Zero Trust .
رغم أن النظام يطبّق آلية انتهاء الجلسة تلقائياً، إلا أن السماح للمستخدم بتسجيل الخروج يدوياً يعزز من التحكم الذاتي في الجلسة وتعزيز الثقة في النظام. فيما يلي خطوات تنفيذ التابع logout.

1. استرداد اسم المستخدم

يتم تحميل سجل المستخدم من الملف peer.json للتحقق من وجود قرين نشط مرتبط به.

2. تعطيل القرين

إذا كان القرين نشط أي "enabled = True" يتم حذفه من واجهة WireGuard باستخدام التابع disable_peer().

3. مسح الجلسة

يتم حذف جميع البيانات المرتبطة بالمستخدم من session، بما في ذلك الرموز (access_token وrefresh_token) وتوقيت الجلسة.

4. إعادة التوجيه إلى keycloak لتسجيل الخروج المركزي

يتم توجيه المستخدم إلى رابط تسجيل الخروج الرسمي الخاص بKeycloak ، لضمان إنهاء الجلسة من جانب مزود الهوية أيضاً. يعرض الشكل (45) آلية تسجيل الخروج.

```
354 @app.route("/logout")
355 def logout():
356     username = session.get("username")
357     peers = load_peers()
358     if username in peers:
359         peer = peers[username]
360         peer["enabled"] = False
361         disable_peer(peer["public_key"], peer["ip"])
362         save_peers(peers)
363
364     session.clear()
365     logout_url = (
366         f"{KEYCLOAK_URL}/realms/{REALM}/protocol/openid-connect/logout"
367         f"?client_id={CLIENT_ID}"
368         f"&post_logout_redirect_uri=http://192.168.39.129:5000/"
369     )
370     return redirect(logout_url)
```

الشكل 45 آلية تسجيل الخروج

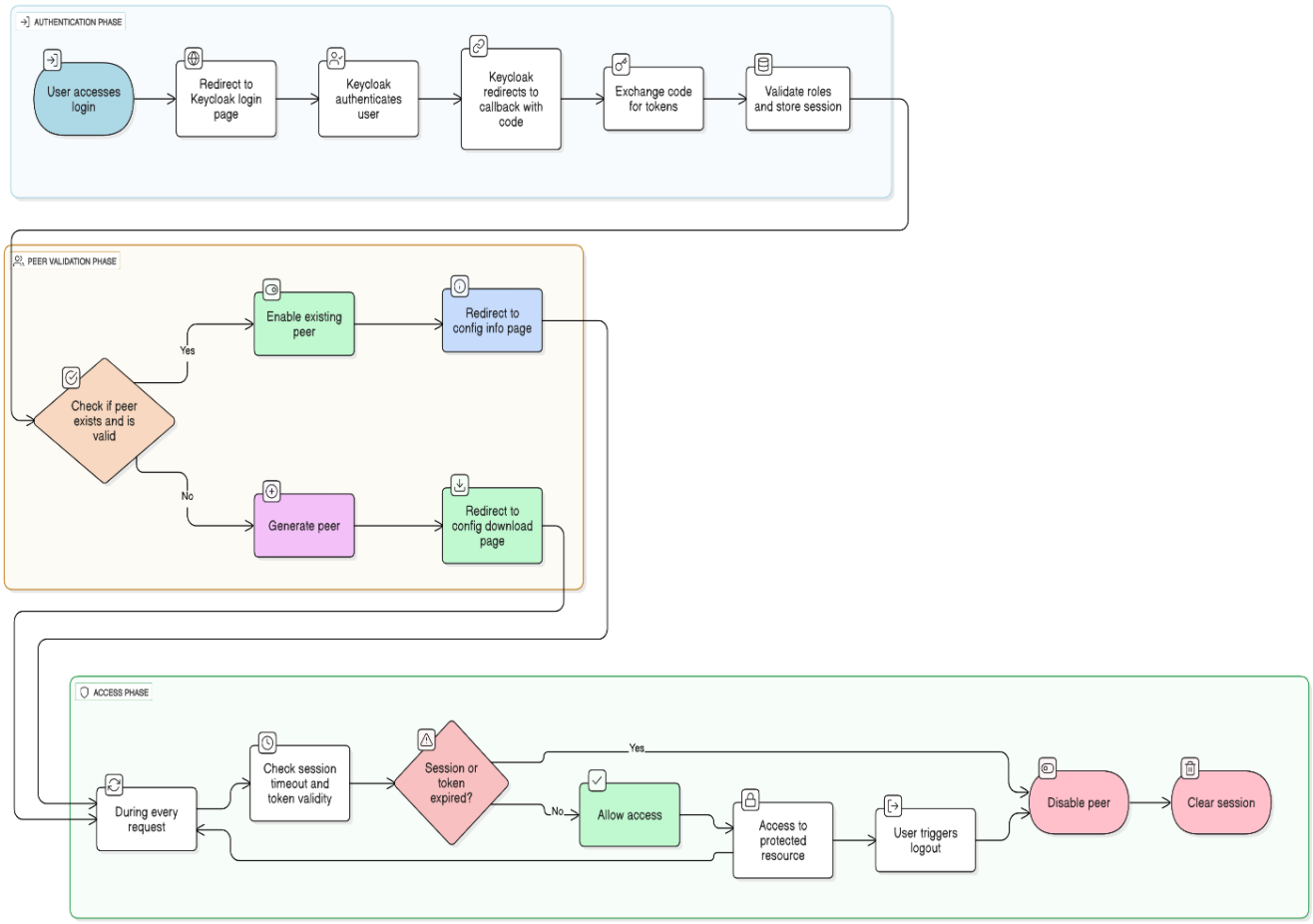
فيما يلي الجدول (13) يوضح التوابع ووظائفها لبوابة الوصول

اسم التابع / المسار	نوعه	الوصف الوظيفي
/login	Route	يعيد توجيه المستخدم إلى صفحة تسجيل الدخول في Keycloak
/callback	Route	يستقبل authorization_code ويستبدله بـ access/refresh tokens، ثم يبدأ الجلسة.
token_required(f)	Decorator	يتحقق من صلاحية الجلسة والرموز، ويفرض مهلة الجلسة، ويجدد الرموز تلقائياً.
/download-config	Route	يولد ملف إعداد WireGuard ويرسله للمستخدم بناءً على دوره وصلاحية الجلسة.

اسم التابع / المسار	نوعه	الوصف الوظيفي
generate_peer(username, roles)	Function	يولّد مفاتيح WireGuard ، يعيّن IP حسب الدور، ويسجّل القرين في peer.json
assign_ip(username, roles)	Function	يحدد عنوان IP المناسب للمستخدم اعتماداً على دوره في النظام.
ensure_peer_enabled(peer)	Function	يضيف القرين إلى واجهة WireGuard ويفعّله.
disable_peer(peer)	Function	يزيل القرين من WireGuard ويمنع وصوله.
/logout	Route	يعطل القرين، ينهي الجلسة، ويعيد توجيه المستخدم إلى Keycloakلتسجيل الخروج.
cleanup_expired_peers()	Function	ينظف الأقران منتهية الصلاحية ويعطلها تلقائياً.
should_refresh_token()	Function	يحدد ما إذا كان access_token بحاجة إلى تحديث قبل انتهاء صلاحيته.

جدول13 أهم التوابع والمسارات ووظائفها لبوابة الوصول

يعرض الشكل (46) مخطط عملية إدارة الجلسة والاتصال في بوابة الوصول باستخدام Flask.



الشكل 46 مخطط عملية إدارة الجلسة والاتصال في بوابة الوصول

4.4.5 تطبيق السياسات على مستوى المورد (Enforcement Point / Resource Firewall)

ضمن نموذج الوصول الشبكي المعتمد على انعدام الثقة(ZTNA) ، لا يكفي منح حق الوصول لمستخدم مصادق عليه عبر بوابة الوصول، بل يجب أيضاً فرض قيود دقيقة على مستوى المورد نفسه لضمان أن كل اتصال يتم التحقق منه وتقييده حسب الدور الوظيفي.

ولهذا الغرض، تم تنفيذ **نقطة تطبيق السياسات (Enforcement Point)** على الآلة الافتراضية Resource VM التي تلعب دور الموارد باستخدام جدار حماية مبني على iptables. تحدد هذه القواعد مسبقاً حسب الفئة الوظيفية للمستخدم (الدور)، بناءً على عنوان IP الذي تم تعيينه للمستخدم داخل نفق WireGuard .

يعرض الشكل (47) الرماز الذي يتم من خلاله تطبيق قواعد الجدار الناري الذي يقوم بعزل الموارد بناء على دور المستخدم.

```
GNU nano 7.2                                        firewall.sh *
#!/bin/bash

# Flush old rules

iptables -F
# --- Admins: Full Access ---

iptables -A INPUT -s 10.10.0.16/31 -j ACCEPT


# --- Developers: Everything except SSH ---

iptables -A INPUT -s 10.10.0.32/30 -p tcp --dport 22 -j DROP    # Block SSH first

iptables -A INPUT -s 10.10.0.32/30 -j ACCEPT                    # Then allow all else


# --- Guests: Only HTTP/HTTPS ---

iptables -A INPUT -s 10.10.0.64/29 -p tcp --dport 80 -j ACCEPT

iptables -A INPUT -s 10.10.0.64/29 -p tcp --dport 443 -j ACCEPT
echo "Firewall rules applied."
```

الشكل47 قواعد الجدار الناري المطبقة على الموارد

حيث يتم تقييد محيط الوصول على أساس عنوان الـ IP حيث تم منح كل مستخدم بناء على دوره مجموعة من السياسات، يعرضها الجدول (14).

الدور	المجال(IP Subnet)	السياسة المطبقة
Admins	10.10.0.16/31	وصول كامل دون قيود(All ports)
Developers	10.10.0.32/30	منع منفذ SSH فقط، والسماح بكل شيء آخر
Guests	10.10.0.64/29	السماح فقط بالوصول إلى خدمات (80) HTTP و HTTPS (443)

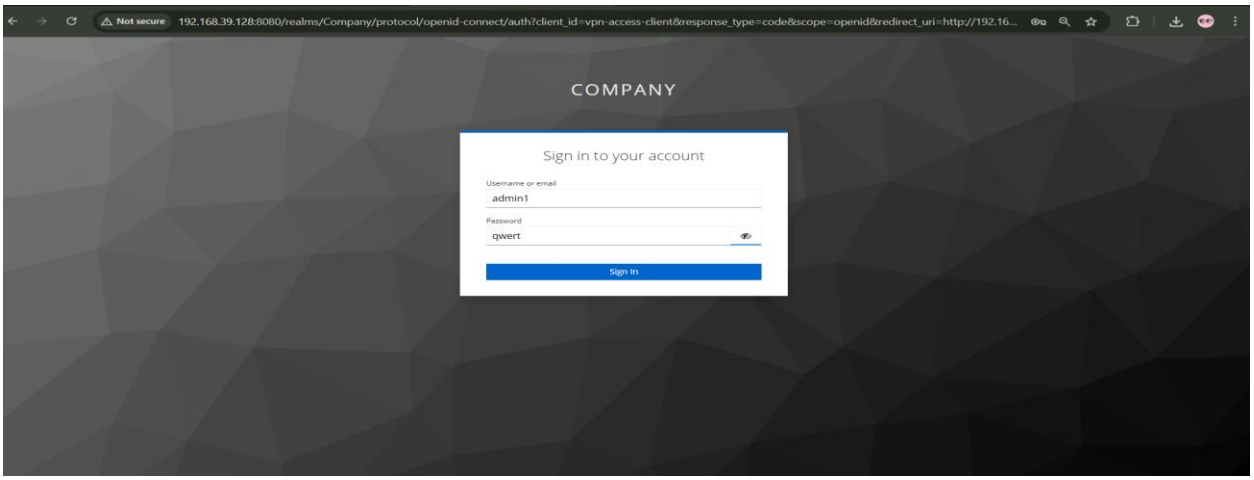
جدول 14السياسات المطبقة بناء على دور المستخدم

5.5 سيناريوهات اختبار النظام

1.5.5 تسجيل الدخول بصفة مسؤول النظام -Admin

بداية تم وضع زمن انتهاء صلاحية كل مما يلي بهدف الاختبار ويمكن لمسؤول النظام ان يقوم بتغييرها حسب المتطلبات:

1- زمن انتهاء رمز الوصول (Access Token) 1 دقيقة



الشكل 48 طلب تسجيل الدخول بصفة مسؤول

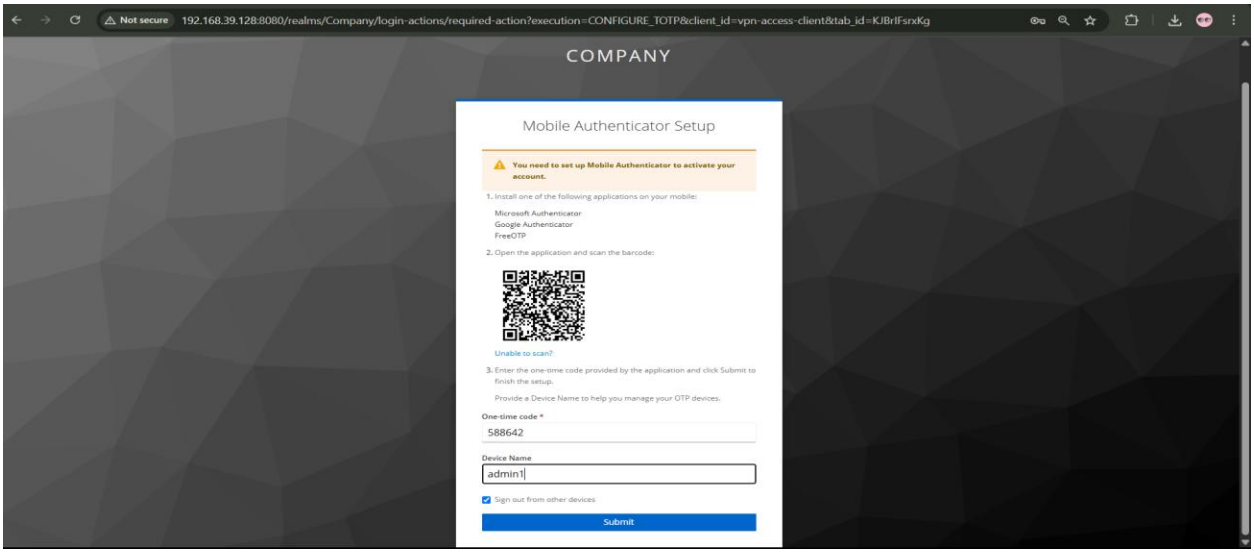
2- زمن انتهاء رمز التجديد(Refresh Token) 2 دقيقة

3- زمن انتهاء الجلسة (Hard limit session) 3 دقيقة

4- زمن صلاحية ملف الاعدادات 30 دقيقة

يتم طلب العنوان التالي http://192.168.39.129:5000 الذي يمثل بوابة الوصول من خلال المتصفح كما هو موضح بالشكل (48) ومن ثم يقوم تطبيق Flask بإعادة توجيه طلب تسجيل الدخول الى الأداة Keycloak على العنوان التالي http://192.168.39.128:8080 لإتمام عملية المصادقة.

يقوم المستخدم بإدخال اسم المستخدم وكلمة المرور كما هو موضح بالشكل (48). بحال نجاح المصادقة يتم بعدها عرض QR code يقوم المستخدم بمسحه باستخدام جواله عن طريق التطبيق Google Authenticator ويقوم بعدها بإدخال



الشكل 49 طلب ادخال كلمة المرور لمرة واحدة

الرمز (كلمة المرور لمرة واحدة) واسم الجهاز كما هو موضح بالشكل (49).

بعد مسح الـ QR code يتم إعطاء رمز مكون من 6 خانات يتجدد كل 30 ثانية. يتم ادخال الرمز من ثم نجاح المصادقة (بحال الرمز كان صحيحا). بعد نجاح المصادقة يتم عرض صفحة الويب الموضحة بالشكل (50) التي يمكن من خلالها تحميل ملف الاعدادات الخاص بـ WireGuard بحال كانت الجلسة صالحة.

الشكل50 صفحة الويب الترحيبية بعد نجاح المصادقة للمستخدم admin1

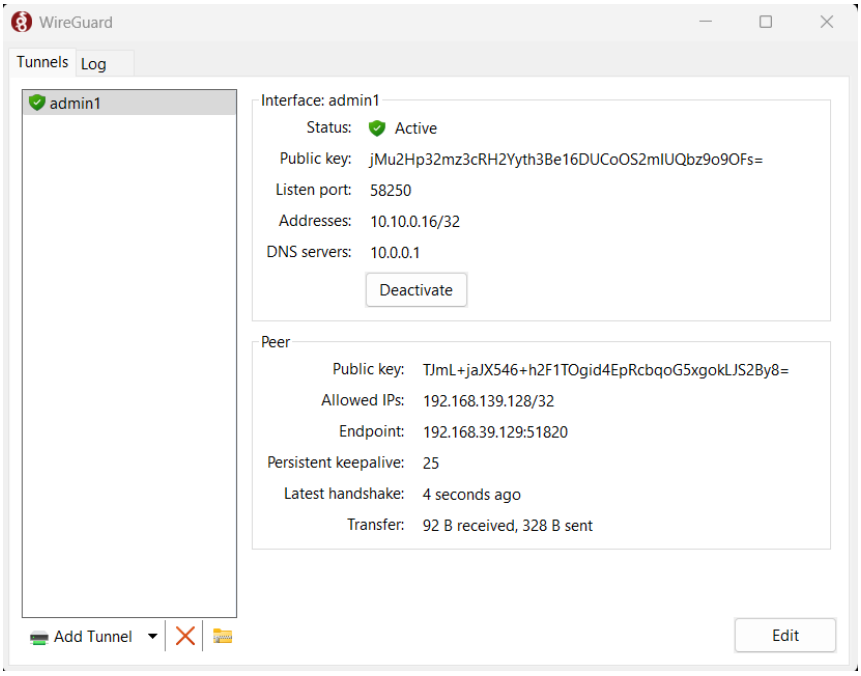
تم التأكد من أن ملف peer.json بالبداية لا يملك معلومات عن المستخدم admin1 في اول مرة تم فيها تسجيل الدخول وبالتالي تم عرض رسالة "No VPN configuration found" ثم يكون بإمكان المستخدم تحميل ملف الاعدادات الخاص بـ WireGuard. يعرض الشكل (51) محتويات ملف الاعدادات الخاص بالمستخدم admin1 (المفتاح الخاص وعنوان IP بناء على دور المستخدم (admin))، ويكون القرين هنا هو مخدم الـ VPN المعرف بالمفتاح العام والعنوان الذي يعمل عليه مع رقم المنفذ 192.168.39.129:51820 والعنوان المسموح طلبه من خلال هذا النفق هو عنوان الالة الافتراضية التي تمثل الموارد ذات العنوان 192.168.139.128.

```
[Interface]
PrivateKey = oANyTsizX/3VDldxYzSA3CkxYORGu11B73mhh6B96WQ=
Address = 10.10.0.16/32
DNS = 10.0.0.1

[Peer]
PublicKey = TJmL+jaJX546+h2F1TOgid4EpRcbqoG5xgokLJS2By8=
Endpoint = 192.168.39.129:51820
AllowedIPs = 192.168.139.128/32
PersistentKeepalive = 25
```

الشكل51 ملف الاعدادات الخاص بالمستخدم admin1

بعد تحميل ملف الاعدادات يتم تفعيل واجهة الشبكة الافتراضية على الالة الفيزيائية التي تعمل بنظام Windows باستخدام WireGuard، كما هو موضح بالشكل (52).



الشكل 52 تفعيل ملف اعدادات WireGuard الخاص بالمستخدم admin1 على جهاز المستخدم

بعد تفعيل واجهة الشبكة الخاصة بـ WireGuard يتم إضافة هذا المستخدم الى ملف peer.json لكي يتم فيما بعد عملية التحقق المستمرة بناء على معلومات الجلسة Continuous Authentication، فيما يلي يعرض الشكل (53) محتويات ملف peer.json بعد دخول المستخدم بنجاح وتفعيل ملف الاعدادات.

```
back@back:~$ cat peer.json
{
    "admin1": {
        "private_key": "oANyTsizX/3VDldxYzSA3CkxY0RGu11B73mhh6B96WQ=",
        "public_key": "jMu2Hp32mz3cRH2Yyth3Be16DUCoOS2mIUQbz9o9OFs=",
        "ip": "10.10.0.16/32",
        "enabled": true,
        "expires_at": 1751963339.532188
    }
```

الشكل 53 محتويات ملف peer.json بعد تسجيل الدخول كمسؤول

يضم الملف peer.json معلومات المستخدم المفاتيح الخاصة والعامة وعنوان المستخدم وحالته وزمن انتهائه بصيغة UNIX Time. بعد ذلك نقوم باختبار إمكانية الوصول للموارد بحيث نقوم بالتحقق من إمكانية الوصول للمنفذ 22 الذي يمثل خدمة SSH، يعرض الشكل (54) إمكانية الوصول للموارد وأن المنفذ رقم 22 فعال ويمكن استخدام الالة الافتراضية التي تمثل الموارد عن بعد.

```
PS C:\Users\DELL> Test-NetConnection -ComputerName 192.168.139.128 -Port 22


ComputerName     : 192.168.139.128
RemoteAddress    : 192.168.139.128
RemotePort       : 22
InterfaceAlias   : admin1
SourceAddress    : 10.10.0.16
TcpTestSucceeded : True
```

الشكل 54 نجاح اختبار الوصول للموارد على المنفذ رقم 22 كمسؤول

بعد نجاح تسجيل الدخول وتحميل ملف الاعدادات يتم عرض توقيت انتهاء صلاحية ملف الاعدادات كما هو موضح بالشكل (55). في هذه الحالة يتم عرض انتهاء صلاحية ملف الاعدادات خلال 24 دقيقة.

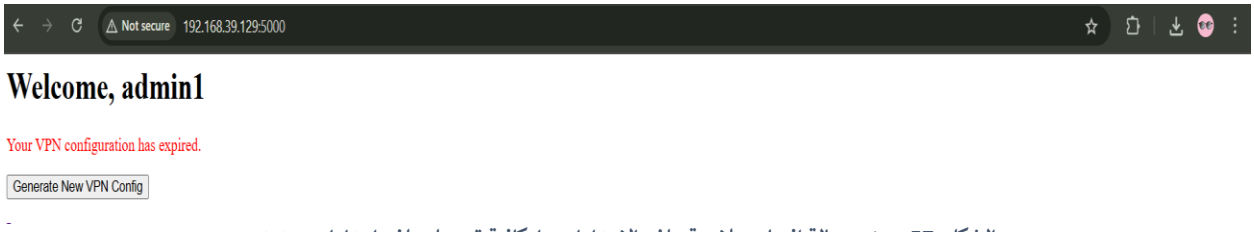
الشكل 55 عرض مدة صلاحية ملف الاعدادات في صفحة الويب

بعد انقضاء مدة زمنية قدرها 1 دقيقة تنتهي صلاحية رمز الوصول (Access Token) ثم يقوم Flask بإرسال طلب تجديد الرموز باستخدام (Refresh Token) وذلك دون الحاجة لتسجيل الدخول مرة أخرى حيث تتم المصادقة في الخلفية. بعد انتهاء مدة صلاحية الجلسة يتم تسجيل الخروج بشكل تلقائي. من ثم يتم إعادة تسجيل الدخول بكلمة المرور واسم المستخدم وكلمة المرور لمرة واحدة ولا حاجة لإعادة تحميل ملف الاعدادات كما هو موضح بالشكل (55). ثم بعد انتهاء

صلاحية ملف الاعدادات تقوم بوابة الوصول بتعديل حالة المستخدم في ملف peer.json من enable= True الى enable= False كما هو موضح بالشكل (56) ويتم عرض في صفحة الويب رسالة تخبر المستخدم بان صلاحية ملف الاعدادات قد انتهت كما هو موضح بالشكل (57).

```
back@back:~$ cat peer.json
{
    "admin1": {
        "private_key": "oANyTsizX/3VDldxYzSA3CkxY0RGu11B73mhh6B96WQ=",
        "public_key": "jMu2Hp32mz3cRH2Yyth3Be16DUCo0S2mIUQbz9o90Fs=",
        "ip": "10.10.0.16/32",
        "enabled": false,
        "expires_at": 1751961281.479366
    }
```

الشكل 56 محتويات ملف peer.json بعد انتهاء صلاحية ملف الاعدادات



الشكل 57 عرض رسالة انتهاء صلاحية ملف الاعدادات وامكانية تحميل ملف اعدادات جديد

للتحقق تم اختبار إمكانية الاتصال والوصول للموارد وطلب خدمة SSH وكانت النتيجة كما هو موضح بالشكل (58) من عدم القدرة على الاتصال بالآلة الافتراضية التي تمثل الموارد.

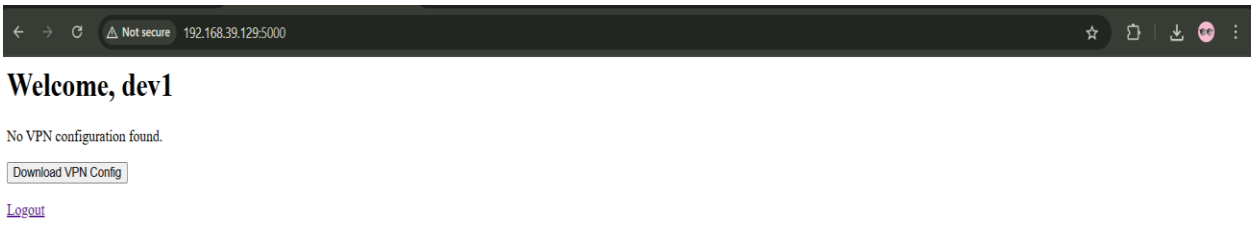
```
PS C:\Users\DELL> Test-NetConnection -ComputerName 192.168.139.128 -Port 22
WARNING: TCP connect to (192.168.139.128 : 22) failed
WARNING: Ping to 192.168.139.128 failed with status: TimedOut


ComputerName            : 192.168.139.128
RemoteAddress           : 192.168.139.128
RemotePort              : 22
InterfaceAlias          : admin1
SourceAddress           : 10.10.0.16
PingSucceeded           : False
PingReplyDetails (RTT)  : 0 ms
TcpTestSucceeded        : False
```

الشكل 58فشل اختبار الوصول للموارد بعد انتهاء صلاحية ملف الإعدادات

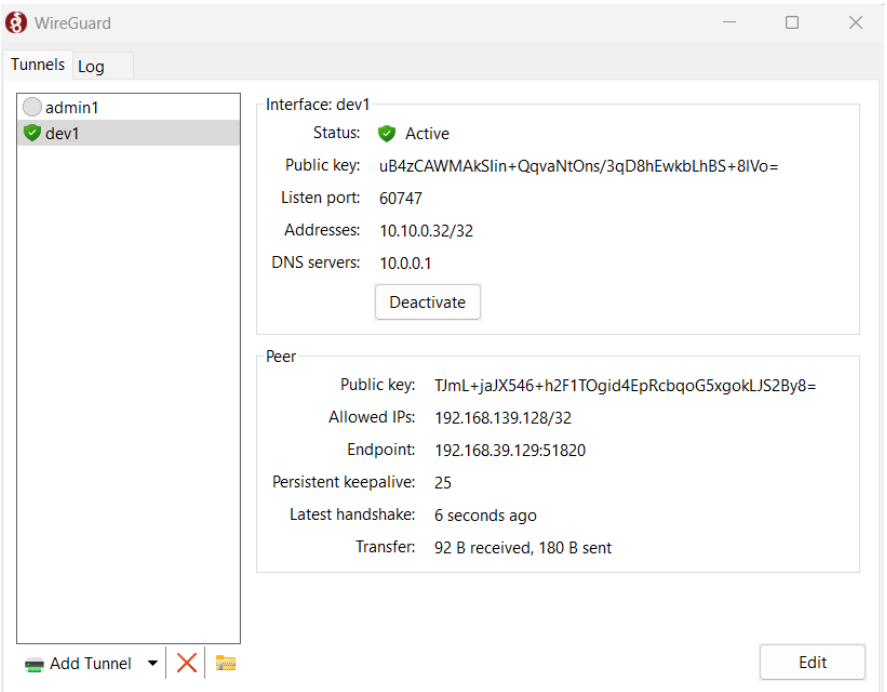
2.5.5 تسجيل الدخول بصفة مطوّر-Developer

بنفس الية تسجيل الدخول التي تم عرضها بالفقرة السابقة (تسجيل الدخول بصفة مسؤول النظام)، بعد نجاح المصادقة يتم عرض رسالة ترحيبية ويمكن تحميل ملف الاعدادات الخاص بالمطوّر كما هو موضح بالشكل (59).



الشكل 59صفحة الويب الترحيبية بعد تسجيل الدخول كمطوّر

يتم تفعيل ملف الاعدادات الخاص بالمستخدم dev1 بعد تحميله على جهاز المستخدم كما هو موضح بالشكل (60).



الشكل 60تفعيل ملف اعدادات WireGuard الخاص بالمستخدم dev1 على جهاز المستخدم

تم اختبار الوصول للموارد بناء على دور المستخدم (Developer) المسموح له بالوصول لكافة الخدمات ما عدا خدمة SSH على المنفذ رقم 22 (هذا فقط بهدف الاختبار وانما بالتطبيقات العملية يتم وضع الشروط حسب متطلبات النظام). يعرض الشكل (61) عدم إمكانية الوصول لخدمة SSH.

```
PS C:\Users\DELL> Test-NetConnection -ComputerName 192.168.139.128 -Port 22
WARNING: TCP connect to (192.168.139.128 : 22) failed


ComputerName            : 192.168.139.128
RemoteAddress           : 192.168.139.128
RemotePort              : 22
InterfaceAlias          : dev1
SourceAddress           : 10.10.0.32
PingSucceeded           : True
PingReplyDetails (RTT)  : 0 ms
TcpTestSucceeded        : False
```

الشكل 61فشل اختبار الوصول للخدمة SSH كمطوّر

بينما يمكن الوصول لخدمة HTTP على المنفذ رقم 80 كما هو موضح بالشكل (62).

```
PS C:\Users\DELL> Test-NetConnection -ComputerName 192.168.139.128 -Port 80


ComputerName     : 192.168.139.128
RemoteAddress    : 192.168.139.128
RemotePort       : 80
InterfaceAlias   : dev1
SourceAddress    : 10.10.0.32
TcpTestSucceeded : True
```

الشكل 62 نجاح اختبار الوصول لخدمة HTTP كمطوّر

3.5.5 محاولة تجاوز Flask باستخدام WireGuard فقط

التحقق مما إذا كان مهاجم (أو مستخدم غير مصرّح له) يمكنه الاتصال بشبكة الـ VPN مباشرة عبر بروتوكول WireGuard، دون المرور بمسار المصادقة الرسمي المعتمد في Flask (Keycloak + Flask Gateway).

شروط الاختبار تتمثل في امتلاك المهاجم بعض المعلومات العامة عن مخدم WireGuard حيث يملك المفتاح العام الخاص بالمخدم و Endpoint (192.168.39.129:51820) والـ AllowedIPs (192.168.139.128/32). تم توليد مفتاح خاص و مفتاح عام خاص به باستخدام (wg genkey / wg pubkey) ومن ثم تم انشاء ملف اعدادات WireGuard ومن ثم تم تفعيل هذا الملف. عندما يرسل المستخدم اول حزمة (Handshake initiation) مشفرة باستخدام المفتاح العام الخاص بالمخدم يلتقى مخدم WireGuard الحزمة ويفك تشفيرها باستخدام مفتاحه الخاص. قبل الرد على المستخدم يتم التحقق من وجود هذا القرين في ملف الاعدادات الخاص بالمخدم wg0.conf ولأن المفتاح العام للمهاجم غير مسجل ولا يوجد AllowedIPs لهذا القرين عندها المخدم لا يرسل أي رد ولن يتم انشاء النفق ولا يتم إدخال أي معلومات في ملف peer.json لان Flask لم يكن طرفا في المصادقة.

لأن WireGuard مصمم بطريقة فقط الأقران المسجلون في ملف الإعدادات (يدويا او برمجيا) يمكنهم تبادل الحزم، حتى لو أرسل طرف ما حزمة صحيحة التشفير، فإن المخدم لا يرد إن لم يكن يعرفه مسبقا وهذا يعتمد على تطابق المفتاح العام.

يوضح الجدول (15) شروط نجاح هذا الاختبار.

الشرط	الغرض
عدم التمكن من الوصول إلى wg0.conf	يمنع تسجيل يدوي أو تعديل يسمح بإضافة المفتاح العام للمهاجم
peer.json فقط يتحكم به Flask	يمنع التلاعب بالمفاتيح وتوقيتات الجلسة
Session Timeout مفعّلة	يعطّل الأقران المنتهية تلقائيا
Endpoint محدود بجدار ناري	يضمن أن الحزم غير المصرح بها تُرفض

جدول 15 شروط نجاح اختبار تجاوز Flask باستخدام WireGuard فقط

4.5.5 ملخص حالات الاختبار المنفذة

تم تنفيذ مجموعة من اختبارات الوصول ضمن البيئة الافتراضية المصممة لتقييم فعالية النموذج الأمني حيث شملت هذه الاختبارات حالات استخدام طبيعية وأخرى هجومية بهدف قياس قدرة النظام على فرض التحقق المستمر من الهوية، ضبط الجلسات بشكل ديناميكي، حماية النفق المشفر من التجاوزات، ومنع الوصول غير المصرح به بعد انتهاء صلاحية الرموز. يوضح الجدول (X) ملخص لحالات الاختبار التي تم اختبارها بنجاح.

رقم الحالة	وصف الاختبار	الهدف
1	تسجيل الدخول بشكل صحيح وتحميل إعداد VPN بنجاح	التأكد من سير عملية المصادقة وإعداد الجلسة بشكل سليم
2	انتهاء الجلسة تلقائيا وتعطيل القرين بعد 3 دقائق	التحقق من تطبيق مهلة الجلسة وتنفيذ الحظر التلقائي
3	محاولة استخدام إعداد VPN منتهي الصلاحية بعد انتهاء الجلسة	التأكد من عدم صلاحية الإعدادات السابقة بعد انتهاء الجلسة
5	تسجيل الخروج يدويا وتعطيل القرين	اختبار إنهاء الجلسة المباشر وإبطال الاتصال بمجرد طلب المستخدم
6	محاولة تجاوز Flask والاتصال مباشرة بـ WireGuard	التحقق من أن الخادم لا يقبل الاتصالات إلا من أقران مصادق عليهم مسبقا
7	هجوم القوة الغاشمة لكشف كلمة المرور (Brute Force Attack)	منع الوصول والحصول على Access Token حتى بحال تم كشف كلمة المرور

6.5 النتائج والتحليلات

أسفر تنفيذ واختبار نموذج الثقة الصفرية المقترح باستخدام أدوات مفتوحة المصدر عن مجموعة من النتائج التي تؤكد فعالية التصميم وقدرته على تحقيق الأهداف الأمنية الموضوعة. فقد أثبتت التجارب العملية أنّ النظام يوفّر بيئة وصول آمنة تعتمد على التحقق المستمر من هوية المستخدم، مع عزل الموارد الداخلية، وتطبيق مبدأ الصلاحيات الأدنى (Least Privilege) على نحو دقيق.

أولاً، أظهرت الاختبارات نجاح النظام في تحقيق مصادقة متعددة العوامل باستخدام Keycloak و Google Authenticator، مما منع أي محاولة وصول حتى في حال تسرب كلمة المرور. كما تمكّن النظام من تطبيق سياسات وصول قائمة على الأدوار (RBAC) مربوطة بعناوين IP مخصصة لكل جلسة، ما سمح بالتحكم الدقيق في الموارد وفق فئة المستخدم.

ثانياً، أثبتت الاختبارات مقاومة النظام لهجمات شائعة مثل هجمات القوة الغاشمة(Brute Force) ، إذ فشلت جميع المحاولات في الوصول إلى الموارد بعد انتهاء صلاحية الرموز أو دون إدخال رمز التحقق الديناميكي(TOTP) . كما أظهر النظام قدرة واضحة على إدارة الجلسات ديناميكياً؛ حيث يتم تعطيل قرين WireGuard تلقائياً عند انتهاء الجلسة، ومنع أي إعادة استخدام للرموز المنتهية.

ثالثاً، أثبتت النتائج أنّ التصميم عزّز الحماية الشبكية من خلال العزل الكامل للموارد الداخلية، ومنع الحركة الجانبية (Lateral Movement) بفضل الجمع بين الجدار الناري المحلي على الخادم المستهدف ونفق WireGuard المشفر الذي لا يقبل سوى الأقران المصرح لهم. وبذلك، قلّل النموذج بشكل كبير سطح الهجوم مقارنة بالشبكات التقليدية المعتمدة على VPN.

الفصل السادس

6. الخاتمة والآفاق المستقبلية

نجد في هذا الفصل شرح مختصر لكل ما قمنا به في فترة العمل في المشروع، إضافة للمهارات والخبرة المكتسبة في العمل مع الآفاق المستقبلية

1.6 الخاتمة

يمثل هذا المشروع تطبيقاً عملياً ناجحاً لمفهوم معمارية الثقة الصفرية (Zero Trust Network Access – ZTNA) باستخدام أدوات مفتوحة المصدر. فقد تمكّنا من تصميم نموذج أمني متكامل يربط بين مصادقة الهوية متعددة العوامل، وتطبيق سياسات وصول دقيقة قائمة على الأدوار، وإنشاء نفق اتصال مشفر باستخدام WireGuard مع عزل الموارد الداخلية عبر جدار ناري محلي.

أثبتت التجارب العملية أنّ النظام قادر على مقاومة محاولات الوصول غير المصرح به، وإدارة الجلسات ديناميكياً، وتطبيق مبدأ أقل امتياز (Least Privilege)، مع الحد من سطح الهجوم وحماية الموارد من الحركة الجانبية (Lateral Movement). وعليه، فإن هذا المشروع يقدم نموذجاً تطبيقياً واقعياً يمكن الاعتماد عليه كأساس لمشاريع أكبر في مجال الأمن الشبكي.

2.6 المهارات المكتسبة

من خلال إنجاز هذا المشروع، تم اكتساب مجموعة واسعة من المهارات التقنية والبحثية، أبرزها:

- **مهارات الشبكات وأمن المعلومات** :إعداد بيئة شبكية افتراضية معزولة باستخدامVMware ، وتطبيق مفاهيم العزل الشبكي والتحكم الدقيق بالوصول.
- **إدارة الهوية والوصول** :**(IAM)** ضبط Keycloak كمزوّد هوية، وتطبيق المصادقة متعددة العوامل (MFA) وإدارة الرموز (Access & Refresh Tokens).
- **برمجة البنى الأمنية** :تطوير بوابة وصول (Access Gateway) باستخدام **Flask** بلغة Python لربط المصادقة بالتحكم في أقران WireGuard وإدارة الجلسات تلقائياً.
- **التكامل مع بروتوكولات VPN الحديثة** :إعداد **WireGuard**كنفق آمن وتطبيق سياسات أمان دقيقة بالاعتماد على عناوين IP المرتبطة بالأدوار.
- **البحث والتحليل الأمني** :اختبار سيناريوهات واقعية للهجمات، مثل هجمات القوة الغاشمة (Brute Force Attacks)، وتحليل استجابة النظام لها.

3.6 الآفاق المستقبلية

يمكن تطوير هذا المشروع في المستقبل بعدة طرق لتعزيز مستوى الأمان وإضافة مزايا أكثر تكاملاً، ومن أبرز هذه الآفاق:

- **توسيع البنية لتشمل بيئات حقيقية أو سحابية**: نقل النموذج إلى بيئة إنتاجية أو منصات Cloud مع الحفاظ على نفس المبادئ الأمنية.
- **دمج أنظمة كشف التسلل (IDS) والتحليل المركزي (SIEM):** لإضافة مراقبة أوسع على مستوى حركة المرور وتحليل محاولات الاختراق بشكل مركزي.

- **تطبيق إدارة الهوية الديناميكية القائمة على السياق(Context-Aware IAM) :** بحيث يتم أخذ عوامل إضافية مثل موقع المستخدم أو حالة الجهاز في قرار الوصول.
- **تحسين تجربة المستخدم**: من خلال أتمتة تنزيل وإدارة ملفات VPN مع واجهة ويب أكثر سلاسة وتوافقاً مع أنظمة متعددة.

7. المراجع

[1] S. M. Zohaib et al, "Zero Trust VPN (ZT-VPN): A Cybersecurity Framework for Modern Enterprises to Enhance IT Security and Privacy in Remote Work Environments". (2024)

[2] S. Rose et al, “NIST Publication 800-207, Zero Trust Architecture”. (2020)

[3] Vasilios Mavroudis, *Zero-Trust Network Access (ZTNA)*, the Alan Turing Institute. (2025)

[4] N. F. Syed et al, “Zero Trust Architecture (ZTA): Comprehensive Survey”. (2022)

[5] Red Hat. (n.d.). Retrieved May 2, 2025 from https://www.redhat.com/en/topics/security/what-is-role-based-access-control

[6] Red Hat. (n.d.). *Keycloak: Open Source Identity and Access Management*. Retrieved May 15, 2025 from https://www.keycloak.org

[7] Donenfeld, J. (2020). *WireGuard: Next Generation Kernel Network Tunnel*. Retrieved May 18, 2025 from https://www.wireguard.com

[8] S. Mackey, I. Mihov, A. Nosenko, F. Vega, and Y. Cheng, “A Performance Comparison of WireGuard and OpenVPN,” in *Proc. ACM Conf. Data and Application Security and Privacy (CODASPY)*, New Orleans, LA, USA, Mar. 2020. [Online]. Available: https://doi.org/10.1145/3374664.3379532

[9] Pallets Projects. (n.d.). *Flask: Python web development, one drop at a time.* Retrieved May 19, 2025 from https://flask.palletsprojects.com/

[10] VMware. (n.d.). *VMware Workstation Pro: Run multiple OS on a single PC.* Retrieved May 23, 2025, from https://www.vmware.com/products/workstation-pro.html

[11] Google. (n.d.). *Google Authenticator.* Retrieved May 30, 2025, from https://support.google.com/accounts/answer/1066447

[12] JWT.io. "JSON Web Tokens - Introduction." Retrieved June 8, 2025. https://jwt.io/