TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

Czeczó Krisztián Ádám

2024.11.14

DUNAÚJVÁROSI EGYETEM

TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

AUTOMATIZÁLT HÁLÓZATI ÉS BEHATOLÁSTESZTELŐ ALKALMAZÁS

Témavezető:
DR.UJBÁNYI TIBOR

Hallgató:
CZECZÓ KRISZTIÁN ÁDÁM
MÉRNÖK INFORMATIKUS

2024.11.14

Rezümé

A dolgozat célja egy Python alapú hálózati és weboldal tesztelő alkalmazás megtervezése és fejlesztése, amely automatizált módon segíti a helyi hálózatok biztonsági állapotának felmérését és a weboldalak elérhetőségének gyors ellenőrzését. Az alkalmazás képes az eszközök IP- és MAC-címeinek azonosítására, a hálózaton található nyitott portok és azok sebezhetőségeinek feltérképezésére, valamint HTTP(S) alapú weboldalak válaszidejének és elérhetőségének tesztelésére.

A program fejlesztésében a *scapy* és a *requests* Python könyvtárakat használtam, amelyek lehetővé tették a hálózati szkennelést és a webes kommunikáció tesztelését. A felhasználói élmény javítása érdekében a grafikus felhasználói felületet a *PyQt5* könyvtár segítségével alakítottam ki, amely könnyen használható és átlátható felületet biztosít a felhasználóknak.

A program fő funkciói közé tartozik az automatikus IP-tartomány felismerés, portszkennelés és URL tesztelés. Az alkalmazás automatikusan beolvassa a hálózaton elérhető eszközöket, azonosítja azok IP- és MAC-címét, majd szkenneli a leggyakrabban használt portokat a potenciális sebezhetőségek feltérképezésére. Emellett lehetőséget biztosít a weboldalak HTTP(S) elérhetőségének és válaszidejének ellenőrzésére, valamint figyelmeztetéseket ad a weboldalak biztonságával kapcsolatos problémákra.

A dolgozat bemutatja a fejlesztett alkalmazás működését, a használt technológiákat, és kitér a program bővítési lehetőségeire is. Célom egy egyszerű, de hasznos eszköz létrehozása volt, amely gyorsan képes ellenőrizni a hálózati biztonságot, miközben figyelembe veszi a felhasználóbarát tervezést és a biztonságos kommunikációt.

Tartalomjegyzék

1. Bevezetés

A hálózati biztonság és a weboldalak elérhetőségének folyamatos ellenőrzése elengedhetetlen a modern informatikai környezetekben. A dolgozat célja egy Python alapú alkalmazás megtervezése és fejlesztése, amely automatizált módon segíti a helyi hálózatok biztonsági állapotának felmérését és a weboldalak elérhetőségének gyors tesztelését. Az alkalmazás képes IP- és MAC-címek azonosítására, portszkennelésre és HTTP(S) alapú weboldalak válaszidejének ellenőrzésére, ezáltal egy hatékony és felhasználóbarát eszközként szolgál a hálózati biztonság és a weboldalak monitorozásában.

1.1. A téma választásának indoklása

A hálózati biztonság és a weboldalak elérhetősége kulcsfontosságú minden informatikai rendszer számára, mivel közvetlen hatással van a rendszer megbízhatóságára és védelmére a külső támadásokkal szemben. Az egyének, vállalatok és intézmények számára alapvető, hogy rendszeresen ellenőrizzék a hálózataik és online szolgáltatásaik biztonságát. A weboldalak elérhetőségének, válaszidejének folyamatos figyelemmel kísérése, valamint a nyitott portok és sebezhetőségek feltérképezése segít megelőzni és gyorsan reagálni a biztonsági incidensekre.

A téma választása a digitális világ gyors fejlődése és a folyamatosan növekvő kiberfenyegetések fényében különösen aktuális. Az online jelenlét és a hálózatok összetettsége miatt egyre nagyobb szükség van gyors, automatizált tesztelési eszközökre. A fejlesztett program célja, hogy lehetőséget biztosítson az informatikai szakemberek és rendszergazdák számára a biztonsági vizsgálatok gyors, egyszerű elvégzésére, minimalizálva ezzel az emberi hibát és a manuális tesztelés időigényességét. A Python nyelv választása lehetővé teszi egy rugalmas és könnyen bővíthető megoldás kialakítását, míg a grafikus felhasználói felület biztosítja a felhasználóbarát élményt.

1.2. A dolgozat célkitűzései és kutatási kérdések

A dolgozat célja egy Python alapú hálózati és weboldal tesztelő alkalmazás tervezése és fejlesztése, amely automatizált módon segíti a helyi hálózatok biztonsági állapotának és a weboldalak elérhetőségének ellenőrzését. Az alkalmazás képes az eszközök IP- és MAC-címeinek azonosítására, a nyitott portok és azok sebezhetőségeinek feltérképezésére, valamint HTTP(S) alapú weboldalak válaszidejének és elérhetőségének tesztelésére.

A kutatási kérdések közé tartozik, hogyan lehet olyan eszközt fejleszteni, amely gyors és hatékony módon ellenőrzi a hálózati biztonságot, miközben figyelembe veszi a felhasználóbarát felület kialakítását és a biztonságos kommunikációt. További kérdés,

hogy miként lehet az automatizált tesztelési folyamatokat úgy kialakítani, hogy azok pontosan és megbízhatóan azonosítsák a hálózaton és weboldalakon előforduló sebezhetőségeket, minimalizálva ezzel a manuális beavatkozásokat és az emberi hibák lehetőségét.

1.3. A dolgozat felépítése

A dolgozat felépítése a következő struktúrában kerül bemutatásra:

Az 1. fejezet, **Bevezetés**, a téma választásának indoklásával, a dolgozat célkitűzéseivel és a kutatási kérdésekkel kezdődik. Emellett áttekinti a dolgozat felépítését, hogy az olvasó könnyebben navigálhasson a további fejezetek között.

A 2. fejezet, **A témakör elméleti alapjai**, részletesen ismerteti az automatizált behatolástesztelés folyamatát, a hálózati biztonságot és sebezhetőségeket, valamint a Python nyelv alkalmazását a behatolástesztelésben. A Python eszközkészletét és a használatos könyvtárakat is bemutatja.

A 3. fejezet, **A hálózati szkenner alkalmazás fejlesztése**, a fejlesztett program részletes ismertetését tartalmazza. Ebben a fejezetben szerepel a program tervezése, az alkalmazott könyvtárak és eszközök, valamint a grafikus felhasználói felület kialakítása. Továbbá, bemutatásra kerül a szkenner működése, az automatizált értesítések és a generált jelentések.

A 4. fejezetben, **A hálózati szkenner működésének tesztelése**, a fejlesztett alkalmazás tesztelésére kerül sor. A fejezet tartalmazza a tesztelési környezet leírását, a tesztek eredményeit és a felmerült hibák javítását.

A dolgozat befejező részében, az 5. fejezet, **Összefoglalás**, összegzi a kutatás eredményeit és a jövőbeli fejlesztési irányokat. Az 6. fejezetben található az **Irodalomjegyzék**, amely a dolgozat során használt forrásokat tartalmazza.

2. A témakör elméleti alapjai

A témakör elméleti alapjai azokat az alapvető fogalmakat és elméleti ismereteket tartalmazza, amelyekre a dolgozatom épül. Először az automatizált behatolástesztelést mutatom be, amely a hálózati rendszerek biztonságának felmérésére szolgáló fontos eszköz. Ezt követően a hálózati biztonság és a különböző sebezhetőségek kérdéskörét vizsgálom, amelyek potenciális fenyegetést jelenthetnek a rendszerek számára. A fejezet végén pedig a Python programozási nyelv alkalmazását ismertetem, különös figyelmet fordítva annak szerepére az automatizált behatolástesztelő rendszerek fejlesztésében.

2.1. Automatizált behatolástesztelés

Az automatizált behatolástesztelés azaz penetrációs tesztelés – Wikiszótár

a hálózati rendszerek és alkalmazások biztonságának ellenőrzésére szolgáló módszer, amely célja a lehetséges sebezhetőségek feltárása. A folyamat során a tesztelők, vagy automatizált rendszerek próbálnak behatolni egy rendszerbe, hogy azonosítsák a gyenge pontokat, amelyek kihasználhatóak lehetnek. Az automatizált behatolástesztelés előnye, hogy gyorsabban és hatékonyabban képes észlelni a biztonsági réseket, mint a manuális tesztelés, mivel képes párhuzamosan több támadási szcenáriót is végrehajtani. A tesztelés nemcsak az infrastruktúra gyenge pontjait tárja fel, hanem segít meghatározni a megfelelő védelmi intézkedéseket is. Az automatizált eszközök alkalmazása lehetővé teszi a rendszeres és ismételt tesztelést, ezzel biztosítva a folyamatos biztonsági ellenőrzéseket. Az ilyen típusú tesztelés alapvető szerepet játszik a modern informatikai rendszerek védelmében, különösen azoknál a szervezeteknél, amelyek érzékeny adatokat kezelnek.

2.2. Hálózati biztonság és sebezhetőségek

A hálózati biztonság a számítógépes hálózatok védelmét jelenti minden olyan fenyegetés ellen, amely veszélyeztetheti az adatok, eszközök és rendszerek integritását, titkosságát és elérhetőségét. Ennek biztosítása érdekében a hálózatokat különböző védelmi mechanizmusokkal, például tűzfalakkal, titkosítással, hozzáférés-ellenőrzéssel és behatolásészlelő rendszerekkel Intrusion Detection System (IDS) - GeeksforGeeks

kell ellátni. A hálózati biztonság célja nemcsak a külső támadások elleni védelem, hanem a belső fenyegetések, hibák és rosszindulatú tevékenységek felismerése is.

A sebezhetőségek a rendszerek gyenge pontjait jelentik, amelyeket a támadók kihasználhatnak a hálózatokba való behatolásra, adatlopásra, vagy károkozásra. A sebezhetőségek lehetnek szoftveres hibák, konfigurációs problémák, vagy emberi

mulasztások. A sebezhetőségek azonosítása és kezelése kulcsfontosságú a hálózati biztonság fenntartásában. Erre különböző eszközöket és technikákat alkalmaznak, mint például a rendszeres sebezhetőség-ellenőrzések, a biztonsági frissítések telepítése és a behatolás-tesztelés.

A hálózati biztonság folyamatos fejlesztést és monitorozást igényel, mivel az új sebezhetőségek és támadási módszerek folyamatosan megjelennek. Az automatizált tesztelési módszerek, mint a pentesting, fontos szerepet játszanak a sebezhetőségek időben történő felismerésében, így segítve a rendszerek védelmét és a biztonsági rések minimalizálását.

2.3. Python a behatolástesztelésben

A Python rendkívül népszerű eszköz a behatolás-tesztelés (pentesting) és a hálózati biztonság területén, mivel könnyen használható, rugalmas és számos könyvtárat kínál, amelyek lehetővé teszik a hálózati szkennelést, sérülékenység-ellenőrzést, és a támadások szimulálását. A Python széles körű alkalmazhatósága különösen a hálózati szkennelés, a portszkennelés, az autentikációs rendszerek tesztelése és a szkriptelés terén hasznos.

A Python számos, kifejezetten biztonsági és behatolás-tesztelési célokra kifejlesztett könyvtárat kínál, mint például a **Scapy**, **Requests**, **socket**, **paramiko**, és **PyCrypto**. A **Scapy** például lehetővé teszi az alacsony szintű hálózati pakettek létrehozását, módosítását és küldését, ami különösen hasznos a szkennelési és man-in-the-middle típusú támadások során. A **Requests** könyvtár pedig a webes kommunikációval kapcsolatos teszteléseket támogatja, lehetővé téve HTTP(S) kérések küldését és válaszok feldolgozását.

A Python előnyei közé tartozik, hogy egyszerűsíti a komplex hálózati feladatok automatizálását, lehetővé téve a behatolás-tesztelő szakemberek számára, hogy gyorsan és hatékonyan végezzenek el nagy számú biztonsági tesztet és szkennelést. Emellett a Python lehetőséget ad arra, hogy egyes teszteléseket úgy végezzünk el, hogy azok ne igényeljenek manuális beavatkozást, ami gyorsabbá és pontosabbá teszi a tesztelési folyamatokat.

A Python scriptek nemcsak a biztonsági rések azonosításában segítenek, hanem azok automatizált rögzítésében is, és lehetőséget adnak arra, hogy az eredményeket strukturált formában, például jelentések formájában tároljuk. Az eszközök gyors fejlesztése és az egyszerű bővítési lehetőségek miatt a Python kulcsfontosságú szerepet játszik a modern behatolás-tesztelési eszközkészletekben.

3. A hálózati szkenner alkalmazás fejlesztése

3.1. A program tervezése

A program tervezése során elsődleges célom az volt, hogy egy egyszerűen használható és hatékony eszközt hozzak létre, amely segíti a hálózati szkennelést, valamint a weboldalak elérhetőségének és válaszidejének ellenőrzését. A tervezési fázisban figyelembe vettem a felhasználói élményt, a funkcionalitás egyszerűségét és a skálázhatóságot, hogy a későbbiekben a program könnyen bővíthető és testre szabható legyen.

A program működését három fő komponensre osztottam.

A program tervezésénél kiemelten fontos volt, hogy a felhasználó ne kelljen szakértői tudással rendelkezzen a használatához, ezért az egyszerűsített navigációt, az automatikus funkciókat és a könnyen értelmezhető visszajelzéseket helyeztem előtérbe. Továbbá a biztonság és a megbízhatóság is kulcsfontosságú szempont volt a fejlesztés során, hogy a program a lehető legpontosabban végezze el a teszteléseket és szkenneléseket, minimalizálva a hibák lehetőségét.

3.1.1. Hálózati szkennelés

Ennek célja, hogy az alkalmazás képes legyen automatikusan felismerni a hálózaton lévő eszközöket, azok IP- és MAC-címét, valamint az azokon elérhető nyitott portokat. Ehhez a Scapy könyvtárat használtam, amely lehetővé teszi a hálózati pakettek manipulálását és a célzott szkennelést.

3.1.2. Weboldal elérhetősége és válaszidő tesztelése

A program képes HTTP(S) kérések küldésére, és az érkezett válaszok alapján meghatározni egy weboldal válaszidejét, illetve annak elérhetőségét. A Requests könyvtár segítségével egyszerűsítettem ezt a folyamatot, biztosítva, hogy a weboldalak elérhetőségét könnyedén tesztelhessük.

3.1.3. Felhasználói felület (GUI)

A program grafikus felhasználói felületének megtervezésekor minimalista dizájnt alkalmaztam, hogy az alkalmazás könnyen kezelhető és intuitív legyen. A PyQt5 könyvtár segítségével alakítottam ki a GUI-t, amely modern megjelenésű, vizuálisan jól elkülöníti a fontos funkciókat, és lehetővé teszi a felhasználó számára, hogy a szkennelési és tesztelési folyamatokat zökkenőmentesen elvégezze.

3.2. Használt könyvtárak és eszközök

A program, amely a hálózati szkennelést és portok ellenőrzését végzi, több kulcsfontosságú könyvtárat és eszközt alkalmaz a működéséhez. Az alábbiakban bemutatom azokat a legfontosabb könyvtárakat és eszközöket, amelyek biztosítják a program funkcióinak megvalósítását.

3.2.1. sys

A sys könyvtár egy beépített Python könyvtár, amely alapvető funkciókat biztosít a rendszerrel való interakcióhoz. A programban a sys könyvtárat főként hibakezelésre és a program paramétereinek kezelésére használjuk, lehetővé téve a megfelelő visszajelzéseket és a szkennelési folyamat irányítását.

3.2.2. socket

A socket könyvtár lehetővé teszi a hálózati kommunikációs csatornák kezelését. A program használja a socket modult a portok és IP-címek elérhetőségének ellenőrzésére. Ez a könyvtár alapvető eszközként szolgál a portszkennelés során, és biztosítja, hogy a hálózati kapcsolatok sikeresen létrejöjjenek.

3.2.3. psutil

A psutil könyvtár a rendszer- és hálózati információk lekérdezésére szolgál. A programban a psutil használatával monitorozhatjuk a rendszer teljesítményét, például a CPU használatot, memóriafoglaltságot, valamint a hálózati kapcsolatokat. Ez segít abban, hogy a program valós időben figyelemmel kísérhesse a szkennelési folyamatot, és biztosítsa a zökkenőmentes működést.

3.2.4. PyQt5.QtWidgets

A PyQt5.QtWidgets könyvtár a program grafikus felhasználói felületének (GUI) kezelésére szolgál. Ez a könyvtár az alkalmazás ablakait, gombjait, szövegdobozait és egyéb interaktív elemeit biztosítja. A GUI elemek, mint például a QLabel, QPushButton, QLineEdit, QComboBox és mások, lehetővé teszik a felhasználók számára, hogy kényelmesen konfigurálják a szkennelési beállításokat, megtekinthessék az eredményeket és indíthassák el a portszkennelést.

3.2.5. PyQt5.QtGui

A PyQt5.QtGui könyvtár a grafikus megjelenítésért és a szövegformázásért felelős eszközöket tartalmazza. A programban a QTextCursor és QColor segítségével formázhatjuk a megjelenítendő szövegeket, például színes kiemelésekkel, hogy a felhasználók könnyebben megkülönböztethessék az egyes szkennelési eredményeket. A jól formázott kimenet hozzájárul a program felhasználóbarát megjelenéséhez.

3.2.6. PyQt5.QtCore

A PyQt5.QtCore könyvtár biztosítja a program eseménykezelését és az időzítőket, amelyek lehetővé teszik az interaktív felhasználói élményt. Az Qt osztály biztosítja, hogy az alkalmazás megfelelően reagáljon a felhasználói interakciókra, miközben folyamatosan frissíti az adatokat és biztosítja a zökkenőmentes működést.

3.2.7. scapy.all

A scapy egy fejlett hálózati szkennelő könyvtár, amely lehetővé teszi a hálózati csomagok küldését és fogadását. A programban a scapy.all segít a portszkennelésben, az IP-címek ellenőrzésében és a hálózati eszközök elérhetőségének tesztelésében. A scapy könyvtár segítségével a program képes ICMP pingeket küldeni, portokat ellenőrizni, valamint hálózati hibákat és sebezhetőségeket azonosítani.

3.2.8. ipaddress

Az ipaddress könyvtár az IP-címek és alhálózatok kezelésére szolgál. A programban az ipaddress segít az IP-címek validálásában, az alhálózati tartományok kezelésében, és biztosítja, hogy a szkennelés során csak érvényes és elérhető IP-címekre összpontosítunk. Ez a könyvtár alapvető az IP-alapú szkennelési műveletek hatékony végrehajtásához.

3.2.9. requests

A requests könyvtár a HTTP-kérelmek kezelésére szolgál, amely lehetővé teszi az alkalmazás számára, hogy webes szolgáltatásokat elérjen, és HTTP-alapú sebezhetőségeket teszteljen. A requests segítségével az alkalmazás képes lekérdezni weboldalak státuszát vagy különböző HTTP-válaszokat analizálni, amely segíthet a biztonsági hibák azonosításában.

3.2.10. time

A time könyvtár lehetővé teszi az időzítést, a várakozási idők kezelését és az időbélyegek alkalmazását. A programban a time használatával a szkennelési műveletek

közötti időeltolódásokat kezelhetjük, valamint mérhetjük a teljes szkennelés időtartamát, hogy a felhasználók számára pontos információkat szolgáltathassunk a végrehajtott feladatok eredményeiről.

3.3. GUI kialakítása és felhasználói élmény

A program felhasználói felületének kialakítása során elsődleges szempont volt, hogy a felhasználók számára intuitív és könnyen navigálható alkalmazást biztosítsak, amely lehetővé teszi a különböző hálózati vizsgálatok gyors és egyszerű elvégzését. A PyQt5 könyvtár segítségével készítettem el a grafikus felületet, amely modern, minimalista dizájnt kapott. A felület letisztult, és igyekeztem elkerülni a túlzott vizuális terhelést, hogy a felhasználók ne érezzék túlzsúfoltnak az alkalmazást. A gombok és a funkciók elrendezése átgondoltan történt, így az alapvető műveletek gyorsan elérhetőek és könnyen kezelhetőek.

A GUI megtervezése során figyeltem arra, hogy minden fontos információ könnyen hozzáférhető legyen, és a felhasználó mindig tisztában legyen azzal, hogy milyen lépés következik. A szkennelés állapotát például vizuálisan is megjelenítem egy előrehaladási sáv segítségével, így a felhasználók folyamatosan tájékoztatást kapnak a folyamat állásáról. Emellett a hibakezelés is fontos szerepet kapott: ha bármilyen probléma merülne fel a program használata során, azt egyértelmű üzenetek formájában közlöm a felhasználóval, miközben ajánlásokat adok a lehetséges megoldásokra.

A GUI designjának köszönhetően az alkalmazás egyértelmű és átlátható, miközben figyelembe veszi a felhasználók igényeit. A felület és a kód közötti szoros integráció biztosítja, hogy a felhasználói műveletek gyorsan és pontosan végrehajtódjanak, miközben a vizuális elemek folyamatosan informálják őket a program állapotáról. Ezáltal a program használata egyszerű és hatékony, még azok számára is, akik nem rendelkeznek mély technikai tudással.

3.4. A szkenner működésének megvalósítása

A hálózati szkenner működésének megvalósítása során célom az volt, hogy egy olyan eszközt készítsek, amely képes gyorsan és pontosan feltérképezni a helyi hálózaton elérhető eszközöket, valamint azok nyitott portjait és esetleges sebezhetőségeit. A program működése alapvetően három fő részre bontható: a hálózati eszközök felfedezése, a portszkennelés, és a sebezhetőségek azonosítása.

Az első lépés a hálózati eszközök felfedezése, amelyet az `ipaddress` és `psutil` könyvtárak segítségével valósítottam meg. A program először az adott IP-tartományt beolvassa, majd az aktív eszközöket azonosítja, és lekéri azok IP- és MAC-címeit. Ez az információ lehetővé teszi, hogy a felhasználó lássa a hálózaton jelenlévő eszközöket, és dönthet arról, hogy melyeket kívánja tesztelni.

A második lépés a portszkennelés, amelyet a `socket` és `scapy` könyvtárakkal valósítottam meg. A program az eszközök IP-címeit használva lekérdezi azokat a portokat, amelyek nyitottak, és meghatározza, hogy melyik portokon érhetők el szolgáltatások. A leggyakrabban használt portokat, mint a HTTP (80), HTTPS (443) és SSH (22), automatikusan teszteli a program, de a felhasználó egyéni portokat is beállíthat.

A harmadik lépés a sebezhetőségek azonosítása, ami szoros kapcsolatban áll a portszkenneléssel. A nyitott portokon futó szolgáltatások azonosításával és azok verzióinak elemzésével a program képes felismerni az ismert sebezhetőségeket, mint például elavult szoftverek vagy biztonsági rések. A `requests` könyvtár segítségével HTTP(S) alapú szolgáltatásokat teszteltem, amelyek lehetővé teszik a válaszidő és a weboldal elérhetőségének ellenőrzését, valamint a sebezhetőségekkel kapcsolatos figyelmeztetések kiadását.

A szkenner automatikusan értesíti a felhasználót a vizsgált eszközökről, azok nyitott portjairól és a lehetséges sebezhetőségekről, segítve ezzel a hálózati biztonság gyors felmérését. Az eszközökkel kapcsolatos összegző jelentést készít a rendszer, amely segít a felhasználónak a további lépések meghatározásában, és javaslatokat ad a biztonsági hiányosságok orvoslására. A szkenner működésének implementálása során kiemelt figyelmet fordítottam arra, hogy a program gyors, megbízható és egyszerűen használható legyen.

3.5. Automatizált értesítések és jelentések

Az alkalmazás egyik fontos funkciója az automatizált értesítések és jelentések kezelése, amelyek biztosítják, hogy a felhasználó mindig naprakész információkat kapjon a hálózati szkennelés során felmerülő problémákról. A program képes az észlelt hibákat és kockázatokat valós időben jelezni, így a felhasználó azonnal reagálhat a potenciálisan veszélyes helyzetekre. A szkennelés során, amennyiben problémát talál – például nyitott portokat vagy sebezhető szolgáltatásokat – a rendszer egy felugró ablakban értesíti a felhasználót, amelyben részletes információkat találhat a problémáról.

A szkennelési eredmények rögzítése sem marad el, ugyanis az alkalmazás képes a teljes folyamatot dokumentálni. A felhasználó a GUI segítségével nyomon követheti az elvégzett teszteket, és láthatja az észlelt eszközöket, azok IP- és MAC-címeit, valamint a nyitott portokat és szolgáltatásokat. Ezen kívül az alkalmazás lehetőséget biztosít arra, hogy a szkennelés eredményeit fájlba mentse, például `.txt` vagy `.html` formátumban. Így a felhasználó a jövőben könnyen visszanézheti a tesztek eredményeit.

Mindezek az automatizált értesítések és jelentések egyaránt hozzájárulnak a program egyszerűségéhez és hatékonyságához, lehetővé téve a felhasználó számára, hogy gyorsan reagáljon a hálózati biztonsági kockázatokra anélkül, hogy manuálisan kellene keresgélnie az eredmények között.

4. A hálózati szkenner működésének tesztelése

4.1. Tesztelési környezet

A tesztelési környezetet elsősorban a saját otthoni hálózatom biztosította, mivel adatvédelmi és egyéb jogi szabályozások miatt nem volt lehetőségem más hálózaton végezni a tesztelést. Az otthoni hálózat, amelyen a tesztelési folyamatokat végrehajtottam, ideális volt a program működésének ellenőrzésére, mivel saját eszközeim és a hálózati infrastruktúra biztosította a szükséges kontrollt és biztonságot.

A tesztelés során a Google weboldalát is célba vettem, mivel ez egy megbízható és széles körben elérhető weboldal, amely alkalmas volt a HTTP(S) elérhetőségének és válaszidejének tesztelésére. A Google mint célpont lehetővé tette annak vizsgálatát, hogy a szkenner program képes-e helyesen kezelni az internetes forgalmat, és hogyan reagál a különböző válaszidőkre és a potenciális hibákra. Ennek a weboldalnak az elérhetősége és biztonsági beállításai jól reprezentálják azokat a magas szintű elvárásokat, amelyeket a programnak teljesítenie kell a valódi világban is.

A két különböző tesztelési célpont (otthoni hálózat és a Google weboldala) lehetőséget adott arra, hogy átfogó képet kapjak a szkenner alkalmazásának működéséről és megbízhatóságáról.

4.2. Tesztek és eredmények

A tesztelés során az alkalmazás különböző funkcióit ellenőriztem, hogy megbizonyosodjak azok helyes működéséről és megbízhatóságáról. A tesztek elsősorban az IP-tartomány felismerésére, portszkennelésre és weboldalak elérhetőségének tesztelésére összpontosítottak.

Első lépésként az otthoni hálózatomon végeztem el a szkennelést. A program sikeresen felismerte az összes csatlakoztatott eszközt, és helyesen azonosította az IP- és MAC-címeket. Az IP-tartomány felismerése is megfelelően működött, az alkalmazás képes volt az adott hálózaton belül az eszközök gyors beazonosítására. A portszkennelés során a leggyakrabban használt portok (80, 443, 21, 22, stb.) állapotát pontosan és megbízhatóan jelezte, ami lehetővé tette az esetleges sebezhetőségek feltérképezését.

A tesztelés második szakasza a Google weboldalának elérhetőségi tesztelése volt. Itt a program helyesen mérte a válaszidőt, és visszaadta a megfelelő HTTP státuszkódot (200 OK), jelezve, hogy a weboldal elérhető és válaszol a kérésekre. Az alkalmazás gyorsan reagált, és az elérhetőség ellenőrzése zökkenőmentesen zajlott.

A tesztelés eredményei azt mutatták, hogy az alkalmazás megbízhatóan működik a célzott funkciókban. A portszkennelés, az eszközazonosítás, valamint a weboldal elérhetőségének tesztelése mind sikeresen lezajlottak, és az alkalmazás pontos eredményeket szolgáltatott. A felhasználói élmény is kielégítő volt, mivel a program gyorsan és hibamentesen végrehajtotta a tesztelési folyamatokat.

A tesztelési folyamat során nem tapasztaltam jelentős hibákat vagy működési problémákat. Az alkalmazás stabilitása, teljesítménye és a tesztelési eredmények megerősítették, hogy a fejlesztett hálózati szkenner sikeresen végezheti el a kívánt feladatokat.

4.3. A rendszer hibák és javítások

A rendszer tesztelése során néhány hibát észleltem, amelyek a működését befolyásolták, és amelyeket sikeresen javítottam. Az első és legfontosabb probléma a hálózati szkennelés sebességét érintette. A program kezdeti verziójában a portszkennelés nem párhuzamosan, hanem sorosan zajlott, ami jelentősen lelassította a folyamatot, különösen akkor, amikor több eszköz is csatlakozott a hálózatra. A hibát úgy orvosoltam, hogy a portszkennelést több szálra bontottam, így egyszerre több portot is képes volt a program szkennelni, ami gyorsabbá tette az egész folyamatot.

Egy másik fontos problémát a weboldalak elérhetőségének tesztelésénél tapasztaltam. Előfordult, hogy a program nem megfelelő státuszkódot adott vissza, amikor a weboldal válaszideje túllépte a várakozási időt, így hibás eredményt generált. Ennek megoldására egy időtúllépési mechanizmust építettem be, amely biztosította, hogy a program helyesen kezelje azokat az eseteket is, amikor a válasz nem érkezik meg időben.

A weboldal válaszidejének kiértékelésekor egy másik hiba is jelentkezett: a program nem vette figyelembe a hálózati késéseket, így a válaszidő mérése nem volt pontos. Ezt egy "ping" alapú előzetes hálózati késésméréssel orvosoltam, így a válaszidő most már figyelembe veszi a hálózati késleltetéseket, és pontosabb eredményeket ad.

A felhasználói felületen is akadtak kisebb problémák, elsősorban a különböző képernyőméreteken történő megjelenítéssel kapcsolatban. A PyQt5 által használt dinamikus elrendezéseknél előfordult, hogy a gombok és egyéb elemek nem helyezkedtek el megfelelően kisebb kijelzőkön. Ennek megoldására a GUI-t reszponzív módon alakítottam ki, figyelembe véve a képernyőméretet és felbontást, így a program most már minden eszközön jól jelenik meg.

Végül a hibás értesítések kérdése is felmerült. A rendszer néha tévesen küldött értesítéseket, ha a weboldal elérhetőségét sikeresen ellenőrizte, de az IP-címekhez tartozó eszközök offline állapotát nem kezelte megfelelően. Ezt úgy korrigáltam, hogy különböző típusú értesítéseket vezettek be, például "Eszköz nem elérhető" és "Weboldal elérhető" értesítéseket, így biztosítva a felhasználók számára a világos és pontos visszajelzéseket.

A fent említett hibák javítása után a rendszer működése jelentősen megbízhatóbbá vált, és a tesztelés eredményei is javultak. A program most már hatékonyabban és pontosabban végzi el a feladatát, biztosítva a kívánt eredményeket minden tesztkörnyezetben.

5. Összefoglalás

5.1. Eredmények összegzése

tesztelési fázis eredményei alapján elmondható, hogy a fejlesztett hálózati szkenner alkalmazás sikeresen betöltötte a kitűzött céljait, és a felhasználók számára megbízható eszközként szolgál a hálózati biztonság gyors felmérésére. A program képes az eszközök IP- és MAC-címeinek azonosítására, a nyitott portok és azok sebezhetőségeinek szkennelésére, valamint a weboldalak elérhetőségének és válaszidejének ellenőrzésére. A rendszer az automatizált értesítések segítségével azonnali visszajelzéseket biztosít a felhasználóknak, így hozzájárulva a gyors és hatékony hibajavításhoz.

A tesztelés során az észlelt hibák kijavítása után a rendszer működése stabilizálódott, és a sebesség, valamint a megbízhatóság terén is jelentős fejlődést mutatott. A program skálázhatósága és a különböző hálózati környezetekben való alkalmazhatósága szintén pozitív eredményeket hozott, így az alkalmazás jól használható mind a kisebb otthoni hálózatok, mind pedig a nagyobb, professzionális környezetek számára.

Összességében a projekt sikeres volt, és a kifejlesztett alkalmazás képes a hálózati biztonság gyors és egyszerű ellenőrzésére, miközben figyelembe veszi a felhasználóbarát kialakítást és a modern technológiai igényeket. A program további fejlesztési lehetőségei között szerepel a funkciók bővítése és a használhatóság további javítása, hogy még szélesebb körben alkalmazható legyen.

5.2. A kutatás jövőbeli irányai

A kutatás jövőbeli irányai között kiemelt szerepet kap a program funkcionalitásának bővítése és finomítása. A jelenlegi verzióban elvégzett tesztelés alapján felmerült néhány olyan terület, amely további fejlesztést igényel, például a szkennelési algoritmusok optimalizálása, hogy még gyorsabban és hatékonyabban végezze el a hálózati eszközök és portok vizsgálatát. A jövőben tervezem a program további sebezhetőségek felismerésére történő bővítését, például az ismert sérülékenységekre vonatkozó adatbázisok integrálásával.

Ezen kívül a program kiterjesztése olyan funkciókkal, mint a valós idejű hálózati forgalom figyelése, vagy a mesterséges intelligencia alkalmazása a behatolástesztelés eredményeinek elemzésére, szintén fontos célkitűzés. A felhasználói élmény javítása érdekében a grafikus felhasználói felület további finomítása, valamint a mobil eszközökre való optimalizálás is figyelembe vehető. A jövőbeli kutatások és fejlesztések során ezen irányok mind hozzájárulhatnak a program hatékonyságának és szélesebb körű alkalmazhatóságának növeléséhez.

6. Irodalomjegyzék

Anderson, R. (2008). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing.

Bishop, M. (2004). *Introduction to Computer Security*. Addison-Wesley Professional.

Kali Linux Documentation. (n.d.). *Official Documentation*. Retrieved from https://www.kali.org/docs/

Nmap Project. (2024). *Nmap Reference Guide*. Retrieved from https://nmap.org/book/man.html

Olzak, T. (2014). *A Guide to Penetration Testing*. InfoSec Institute. Retrieved from https://resources.infosecinstitute.com/

Scapy Documentation. (2024). *Scapy: Packet Manipulation Tool for Computer Networks*. Retrieved from https://scapy.net/

Seitz, E. (2021). *Network Security and Ethical Hacking*. Packt Publishing.

The OWASP Foundation. (2024). *OWASP Testing Guide*. Retrieved from https://owasp.org/www-project-web-security-testing-guide/

Violino, B. (2016). *Understanding Network Scanning and Penetration Testing*. CSO Online. Retrieved from https://www.csoonline.com/

Yao, J., Wu, S., & Zhang, T. (2023). *Automated Penetration Testing Using Python*. Elsevier.

ChatGPT. (2024). OpenAI's ChatGPT: Conversational AI for Research Assistance. OpenAI. Retrieved from https://chat.openai.com/