

توثيق مشروع

اختبار أمني باستخدام هجوم تخمين كلمة المرور

(Password Attack)

❀ الفصل الأول: المقدمة والخلفية النظرية

1.1 مقدمة المشروع

في ظل التقدم الهائل في عالم التطبيقات والأنظمة الذكية، أصبح أمن التطبيقات جزءًا لا يتجزأ من جودة أي نظام. وبينما تتوسع استخدامات تطبيقات الهواتف الذكية، تزداد كذلك فرص التهديدات الأمنية، وخاصة تلك المتعلقة بسرية البيانات والمصادقة.

جاء هذا المشروع كمحاكاة واقعية لهجوم فعال (Active Attack) من نوع Brute Force Password Attack، يتم تنفيذه على تطبيق موبايل مبني باستخدام Flutter ومربوط بخلفية PHP + MySQL. يهدف هذا المشروع إلى إظهار كيف يمكن تنفيذ هذا النوع من الهجمات فعليًا باستخدام أدوات برمجية عملية، وتحليل نتائجه، مع التأكيد على أهمية وجود حماية فعلية في أنظمة التحقق وتسجيل الدخول.

1.2 هدف المشروع

- محاكاة هجوم تخمين كلمات المرور (Brute Force) بطريقة فعالة وإيجابية.
- استخدام سكربت ذكي يعتمد على واجهة رسومية GUI لتخمين البريد الإلكتروني وكلمات المرور حتى يتم كشف بيانات تسجيل الدخول.
- تجربة سيناريوهات مختلفة تشمل: استهداف مستخدم محدد، تخمين شامل، واستخدام ملفات مساعدة.

1.3 نوع الهجوم

- نوع الهجوم: Password Attack (فعال) (Active)
- طريقة التنفيذ: باستخدام سكربت مخصص بلغة Python مع واجهة رسومية Tkinter
- الهدف: استنزاف آلية تسجيل الدخول واستنتاج بيانات صحيحة للوصول إلى النظام.

1.4 المتطلبات الأساسية

- بيئة العمل:
- نظام تشغيل Windows 10 أو أعلى.
- Python 3.10 أو أحدث.
- Flutter SDK.
- XAMPP (Apache + MySQL).
- البرمجيات المستخدمة:
- Python (مع مكتبات: requests, tkinter)

- Flutter (لتطبيق واجهة المستخدم)
- PHP (لتنفيذ APIs)
- MySQL (لتخزين بيانات المستخدمين)

1.5 الفرق بين الهجمات الإيجابية والسلبية

- الهجمات السلبية (Passive): مثل التنصت والمراقبة دون التأثير على النظام. غير مسموحة كمشروع مستقل في هذا المسار.
- الهجمات الإيجابية (Active): مثل هجومنا، حيث يتم التفاعل مع النظام وتوليد استجابات مختلفة حتى الوصول إلى هدف الاختراق.

❀ الفصل الثاني: إعداد بيئة العمل وتجهيز النظام

2.1 نظرة عامة

تم بناء هذا المشروع من الصفر باستخدام بيئة متكاملة تشمل الواجهة الأمامية (Frontend) بتقنية Flutter، والواجهة الخلفية (Backend) باستخدام PHP، وربطها بقاعدة بيانات MySQL عبر سيرفر محلي XAMPP. بالإضافة إلى ذلك، تم تطوير سكربت تخمين ذكي بلغة Python مزود بواجهة رسومية لتمثيل الهجوم بأسلوب عملي وفعال.

2.2 إعداد بيئة XAMPP

- تم تثبيت برنامج XAMPP وتفعيله لبدء تشغيل خادم Apache وقاعدة بيانات MySQL.

- تم إنشاء قاعدة بيانات جديدة باسم `login_db` عبر phpMyAdmin.

- تم تصميم جدول `users` يتضمن الأعمدة التالية:

- `id`: رقم تسلسلي تلقائي.

- `username`: اسم المستخدم (نصي).

- `email`: البريد الإلكتروني (نصي).

- `password`: كلمة المرور (نصي).

- تم إنشاء ملفات API بلغة PHP تشمل:

- `signup.php`: لتسجيل المستخدمين.

- `login.php`: للتحقق من صحة بيانات الدخول.

2.3 إعداد مشروع Flutter

- تم إنشاء مشروع Flutter جديد باستخدام الأمر:

- تم تصميم صفحات:

- `signup_page.dart`: صفحة إنشاء الحساب.

- `login_page.dart`: صفحة تسجيل الدخول.

- `home_page.dart`: الصفحة الرئيسية بعد الدخول.

- تم استخدام مكتبة `http` في Flutter للتواصل مع خوادم الـ PHP عبر بروتوكول JSON.

2.4 هيكل واجهات الـ API في PHP

- `signup.php`: يستقبل الاسم والبريد وكلمة المرور، ويتحقق من التكرار، ثم يُدخل المستخدم الجديد في الجدول.
- `login.php`: يتحقق من وجود بريد وكلمة مرور متطابقين ضمن السجلات ويُرجع حالة النجاح أو الفشل بصيغة JSON.

2.5 اختبار الاتصال بين التطبيق وواجهة الـ API

- تم تنفيذ أول اختبار ناجح للاتصال من تطبيق Flutter إلى ملف `login.php` عبر الشبكة المحلية باستخدام IP الجهاز المحلي.
- تم التأكد من نجاح الرد باستخدام أدوات مثل Postman و curl قبل الربط النهائي داخل التطبيق.

2.6 إعداد بيئة Python

- تم تثبيت Python 3.10 واستخدام مكتبة `requests` لإرسال الطلبات إلى واجهة API.
- تم تثبيت مكتبة `tkinter` لإنشاء واجهة رسومية للتحكم بالهجوم.
- تم تصميم السكريبت `smart_attack_gui.py` ليقوم بقراءة قائمة الإيميلات وكلمات المرور وتوليد محاولات تخمين تلقائية ذكية مع تسجيل النتائج.

2.7 التحديات والتجارب

- تم اختبار عدة سيناريوهات للوصول إلى هيكل اتصال فعال بين Flutter و PHP.
- تم اختبار نقل التطبيق إلى جوال حقيقي عبر نقطة اتصال Hotspot وربط السيرفر المحلي باستخدام الـ IP الداخلي.
- تمت محاكاة حالات انقطاع الاتصال، أخطاء الشبكة، والردود غير المتوقعة لضمان ثبات النظام.

2.8 التحديات والتجارب

- تم اختبار عدة سيناريوهات للوصول إلى هيكل اتصال فعال بين Flutter و PHP.
- تم اختبار نقل التطبيق إلى جوال حقيقي عبر نقطة اتصال Hotspot وربط السيرفر المحلي باستخدام الـ IP الداخلي.
- تمت محاكاة حالات انقطاع الاتصال، أخطاء الشبكة، والردود غير المتوقعة لضمان ثبات النظام.

✚ بنهاية هذا الفصل، أصبح النظام بكافة مكوناته متكاملًا وجاهزًا لتنفيذ هجوم Brute Force فعلي على واجهة تسجيل الدخول.

✂ الفصل الثالث: تطوير سكربت الهجوم الذكي (Password Attack)

3.1 مقدمة

في هذا الفصل، ننتقل من مرحلة بناء النظام إلى مرحلة اختبار الأمان عبر تنفيذ هجوم حقيقي من نوع "Brute Force" على واجهة تسجيل الدخول. تم تنفيذ الهجوم باستخدام سكربت ذكي بلغة Python مدعوم بواجهة رسومية تفاعلية، يتيح تجربة كاملة لهجوم فعال يحقق الهدف الأمني للمشروع.

3.2 فكرة الهجوم

يعتمد الهجوم على تجربة عدد هائل من تركيبات البريد الإلكتروني وكلمات المرور من أجل الوصول إلى بيانات دخول صحيحة. يتم تنفيذ الهجوم على واجهة `login.php` التي تم إنشاؤها في الفصل السابق.

3.3 أدوات وتقنيات مستخدمة

- اللغة: Python 3.10
- الواجهة الرسومية: Tkinter
- الاتصال بـ API: مكتبة `requests`
- آلية التخمين:
- قراءة كلمات المرور والإيميلات من ملفات نصية (`emails.txt` و `passwords.txt`)
- توليد تلقائي لإيميلات جديدة عند فشل جميع الموجودين
- توليد كلمات مرور بشكل ذكي (أرقام، حروف، أو مزيج) بطول متغير

3.4 خصائص السكربت

- ✓ قابلية تحديد نطاق التخمين (مثلاً: يبدأ من ali1@gmail.com)
- ✓ تحديد عدد المحاولات القصوى أو السماح بالتخمين اللا محدود
- ✓ دعم توليد كلمات مرور وأسماء تلقائية
- ✓ عرض حي لجميع المحاولات داخل نافذة
- ✓ حفظ النتائج الناجحة في ملفات خاصة

- ✓ تسجيل كل محاولة في ملف سجل
- ✓ إمكانية استهداف بريد إلكتروني محدد فقط

3.5 واجهة المستخدم الرسومية

تحتوي الواجهة على:

- مدخل لتحديد بادئة البريد الإلكتروني وكلمة المرور
- تحديد عدد المحاولات
- اختيار نمط توليد كلمات المرور (أرقام/حروف/مزيج)
- تفعيل الوضع المستهدف (استهداف بريد إلكتروني معين فقط)
- سجل حي للمحاولات الفاشلة والناجحة
- عداد محاولات محدث تلقائيًا

3.6 كيفية الاستخدام

1. تشغيل السكريبت `attack_gui.py` باستخدام:
2. إدخال المعطيات المناسبة (أو تركها فارغة ليتم استخدام الوضع التلقائي).
3. الضغط على زر "Start".
4. عند إيجاد تطابق، يتم إيقاف العملية تلقائيًا وعرض نافذة نجاح.

3.7 سيناريوهات التجربة

- تم تنفيذ الهجوم على مستخدم فعلي ضمن قاعدة البيانات وتم التوصل لبياناته الصحيحة بنجاح.
- تم اختبار توليد تلقائي لأكثر من 1000 محاولة دون توقف النظام.
- تم التحقق من أن السكريبت يتجاوز البريد غير الموجود ويبدأ بتوليد محتوى ذكي تلقائي.

3.8 نماذج من نتائج التنفيذ

- ✓ [FILE-SUCCESS] ali1001@gmail.com | 1234ali
- ✓ [AUTO-SUCCESS] test0044@gmail.com | 4455pass
- ✗ Failed: ali1002@gmail.com | 1122ali
- 🔄 Switching to auto-bruteforce mode...

3.9 التحديات التي تم تجاوزها

- ضبط تنسيق JSON بشكل دقيق ليتطابق مع هيكلية واجهة الـ PHP
- تجاوز أخطاء الترميز أو الاستجابات غير المتوقعة من السيرفر
- تأمين استمرار التخمين حتى بعد فشل كل المحاولات من الملفات
- دمج نمط "الاستهداف المباشر" مع النمط العام دون تعارض

✂ بنهاية هذا الفصل، تم اختبار التطبيق فعليًا بهجوم Password Attack إيجابي وفعال يفي بمتطلبات المشروع ويعزز من القيمة الأكاديمية والتقنية للبحث.

الفصل الرابع: تحليل النتائج وتقييم فاعلية الهجوم

4.1 مقدمة

بعد تنفيذ الهجوم الذكي على التطبيق الذي تم تطويره، يتناول هذا الفصل تحليل النتائج التي تم جمعها خلال تنفيذ سيناريوهات متعددة للهجوم، مع تسليط الضوء على فاعليته، مدى دقته، وسلوك التطبيق أثناء التعرض لهجوم من نوع Brute Force.

4.2 طريقة تنفيذ الاختبار

تم تنفيذ الهجوم عبر السكربت الذكي المطور باستخدام Python، حيث تم استهداف واجهة تسجيل الدخول في التطبيق المبني بلغة Flutter والمتصل بقاعدة بيانات PHP MySQL+. وتم استخدام نوعين من التجارب:

- تجربة عشوائية باستخدام توليد تلقائي للإيميلات وكلمات المرور.
- تجربة موجهة (Targeted Mode) باختيار بريد إلكتروني معروف مسبقًا وتخمين كلمة المرور فقط.

4.3 أبرز المؤشرات التي تم قياسها

- عدد المحاولات الكلي قبل النجاح.
- الزمن المستغرق حتى الوصول إلى التطابق الصحيح.
- معدل النجاح عند استخدام بيانات أولية (من الملفات).
- سلوك الخادم (PHP API) من حيث الاستقرار وسرعة الاستجابة.

4.4 نتائج فعالية الهجوم

 ملف السجل (gui_smart_log.txt):

تم تسجيل آلاف المحاولات مع كل محاولة غير ناجحة أو حدث استثنائي.

 ملف النتائج الناجحة (valid_credentials.txt):

تم استخراج بيانات صحيحة فعليًا عبر التخمين، على سبيل المثال:

ali1001@gmail.com |

ali1122 admin@company.com |

4.5 تحليل النتائج

- عند استخدام ملفات `emails.txt` و `passwords.txt`، كانت فرصة النجاح أعلى إذا احتوت الملفات على بيانات واقعية.
- عند تفعيل "الوضع المستهدف"، كانت النتائج أسرع وأكثر دقة لأن مجال التخمين كان محصورًا في كلمة المرور فقط.
- الهجوم لم يتسبب في توقف التطبيق أو انهيار الخادم، مما يعني أن الواجهة الخلفية لا تحتوي على حماية من نوع Rate Limiting أو Captcha.

4.6 تحليل الثغرة الأمنية

- عدم وجود حماية ضد تكرار المحاولات (Brute Force Protection).
- عدم تشفير كلمات المرور في قاعدة البيانات.
- عدم الاعتماد على بروتوكول HTTPS في عملية الإرسال والاستلام.

4.7 التوصيات الأمنية

- تطبيق آلية تأخير بين المحاولات بعد عدد معين من الفشل.
- استخدام مكتبات مثل bcrypt أو Argon2 لتشفير كلمات المرور في قاعدة البيانات.
- تفعيل إشعارات تنبيهية عند تسجيل الدخول من جهاز جديد أو عند تكرار محاولات خاطئة.
- إضافة حماية من نوع ReCAPTCHA على واجهة تسجيل الدخول.

4.8 تقييم الأداة المطورة

- المرونة: تسمح الأداة باختبار جميع السيناريوهات سواء بالتخمين التلقائي أو الاستهداف المباشر.
- الوضوح: الواجهة الرسومية تعرض المحاولات الناجحة والفاشلة بشكل حي.
- قابلية التوسع: يمكن تعديل السكريبت بسهولة لدعم أنظمة تسجيل دخول أخرى.

✂ ختامًا، هذا الفصل يُظهر بوضوح فاعلية السكريبت وقدرته على اختراق أنظمة تسجيل دخول غير محمية بطريقة فعالة، مما يعزز من أهمية هذا المشروع في مجال أمن التطبيقات.

الفصل الخامس: الخاتمة والتوصيات المستقبلية

5.1 الخاتمة

يمثل هذا المشروع نتائجًا لعمل بحثي وعملي مكثف يهدف إلى توظيف مهارات البرمجة والاختبار الأمني في سيناريو حقيقي يُجسد هجوم من نوع Password Attack. تم تطوير تطبيق موبايل بلغة Flutter متصل بخادم PHP وقاعدة بيانات MySQL، ثم تم تنفيذ هجوم فعال على نظام تسجيل الدخول من خلال أداة ذكية تم تطويرها بلغة Python باستخدام واجهة رسومية تسهل عملية التخمين ومراقبة النتائج.

تُظهر نتائج المشروع أن الأنظمة التي لا تطبق سياسات أمان فعالة تكون عُرضة لهجمات بسيطة ولكن فعالة. كما أن تنفيذ الهجوم لم يتطلب أدوات متقدمة أو تقنيات معقدة، بل اعتمد فقط على تصميم ذكي لآلية التخمين والتفاعل مع الـ API، مما يسلط الضوء على سهولة استغلال الأنظمة غير المحمية بشكل كافٍ.

5.2 الدروس المستفادة


- فهم عميق لكيفية عمل هجمات Brute Force والطرق المستخدمة في اكتشاف كلمات المرور.
- القدرة على تطوير تطبيقات موبايل باستخدام Flutter والتعامل مع الخوادم وقواعد البيانات.
- اكتساب خبرة في تحليل وتصميم أدوات اختبار الأمان باستخدام Python وواجهة Tkinter.
- إدراك أهمية دمج عناصر الأمان في كل مرحلة من مراحل تطوير النظام.


5.3 التحديات التي واجهناها


- ضبط التنسيق بين قاعدة البيانات والتطبيق لضمان استقبال وإرسال البيانات بشكل صحيح.
- تجاوز مشاكل صلاحيات الشبكة والبروتوكول أثناء تشغيل الخادم المحلي.
- ضبط توقيتات التخمين وتنسيق المخرجات بطريقة لا تؤثر على أداء الأداة.


- معالجة السيناريوهات المختلفة مثل ملفات البيانات، التوليد التلقائي، والتخمين الموجه.


5.4 التوصيات المستقبلية


 تطوير الأداة لتدعم المزيد من البروتوكولات يمكن توسيع الأداة لتعمل مع أنظمة تستخدم HTTPS أو OAuth، مما يتيح اختبار أكثر واقعية.

 إضافة واجهة تقارير متقدمة عرض تحليلات بيانية لتوزيع المحاولات، عدد النجاحات، الزمن المتوسط للوصول، وغيرها.

 دعم الاختبار على شبكات خارجية (Remote Attack) إضافة ميزة العمل على IP خارجي أو نطاقات فرعية متعددة لتحليل أوسع للحماية على الخوادم العامة.

 تحويل الأداة إلى مكتبة تعليمية يمكن استخدام الكود الحالي كأساس لتعليم الطلاب كيفية تنفيذ اختبارات أمان حقيقية بطريقة احترافية.

 تضمين أدوات الحماية المضادة في المستقبل، يمكن تضمين خيارات لمحاكاة أدوات الحماية مثل Captcha، rate limiting، أو MFA لمحاكاة الهجمات والاختبارات الدفاعية في نفس الوقت.

 بهذا يكون المشروع قد قدم قيمة علمية وعملية في ميدان أمن التطبيقات، حيث تم المزج بين البرمجة، التحليل الأمني، والتفكير الإبداعي للوصول إلى نموذج متكامل يوضح كيف يمكن تنفيذ هجوم، توثيقه، واستخلاص الدروس منه، تمهيداً لتطوير حلول أكثر أماناً في المستقبل.