(Documentation)

أداة فحص المنافذ باستخدام (Bash)

١. مقدمة

- اسم المشروع: أداة فحص المنافذ (Port Scanner)
- الوصف: هذه الأداة هي نسخة من أداة فحص المنافذ الأصلية المكتوبة بلغة بايثون https://github.com/pannagkumaar/PortScanner،
  وقد تم تحويلها إلى سكربت باش. تقوم الأداة بفحص المنافذ المفتوحة على مضيف معين، وتقدم معلومات حول حالة المنفذ والخدمات المحتملة التي تعمل عليه.
- الهدف: كان الهدف من هذا التحويل هو فهم أعمق لكيفية عمل أدوات فحص الشبكات على مستوى النظام، واستكشاف إمكانيات لغة باش في التعامل مع مهام الشبكات.

مقارنة بين أداة فحص المنافذ: بايثون و باش

الميزة	أداة فحص المنافذ (بايثون)	أداة فحص المنافذ (باش)
المرونة والقدرة	عالية. بايثون توفر مكتبات قوية مثل socketو threading للتعامل المعقد مع الشبكات.	محدودة. باش يعتمد على أدوات النظام مثل /dev/tcp وnetcat ، مما يحد من القدرات.
أنواع الفحص	تدعم أنواع فحص متعددة مثل TCP و UDPو SYN scan وغيرها.	تدعم بشكل أساسي فحص TCP. دعم UDP محدود وقد يتطلب أدوات إضافية.
التعامل مع استجابات الخادم	قدرة ممتازة على إرسال طلبات معقدة، واستقبال وتحليل الاستجابات، واستخراج المعلومات) مثل رؤوس.(HTTP	قدرة محدودة على إرسال طلبات بسيطة، واستقبال الاستجابات الأساسية. تحليل الاستجابات المعقدة .
التزامن	دعم ممتاز للتزامن باستخدامthreading ، مما يسمح بفحص سريع للمنافذ المتعددة بالتوازي.	دعم التزامن أكثر تعقيدًا، وغالبًا ما يتم باستخدام أوامر مثل xargs -P أو تشغيل العمليات في الخلفية (&). قد يكون الأداء أقل كفاءة.
الأداء	بشكل عام، أداء جيد، خاصة عند استخدام التزامن بشكل فعال.	الأداء قد يكون أبطأ، خاصة عند فحص عدد كبير من المنافذ أو التعامل مع استجابات معقدة.
الاعتماديات	تعتمد على مكتبات بايثون) مثل socket, threading).	تعتمد على أدوات النظام المتاحة في نظام التشغيل) مثل /dev/tcp, netcat, timeout).
سهولة الاستخدام	سهلة نسبيًا للمبرمجين الذين لديهم خبرة في بايثون.	قد تكون أسهل للمستخدمين الذين لديهم خبرة في استخدام سطر الأوامر وأدوات يونكس.
قابلية التوسع	قابلة للتوسع بدرجة كبيرة وإضافة ميزات معقدة.	قد يكون من الصعب توسيعها لتشمل ميزات معقدة.

مقارنة في فيديو:

رابط الفيديو:

٢. المتطلبات الأساسية

- نظام تشغيل Linux أو أي نظام Unix-like يدعم باش.
- صلاحيات تنفيذ السكربت (يمكن منحها باستخدام الأمر chmod +x portscanner.sh).
- اتصال شبكة للوصول إلى المضيف الهدف.

٣. طريقة الاستخدام

- بناء الجملة:

```
./portscanner.sh <target_ip_or_domain> <start_port-end_port> [options]
```

- المعاملات:
  - <<target_ip_or_domain>>: عنوان IP أو اسم النطاق للمضيف الهدف المراد فحصه.
  - <<start_port-end_port>>: نطاق المنافذ المراد فحصه. يمكن تحديد منفذ واحد أو نطاق (مثال: ٢٠-٨٠).
- الخيارات:
  - <<--timeout=<seconds>>: تحديد المهلة الزمنية لكل محاولة اتصال (الافتراضي: ١ ثانية).
  - --json: إخراج النتائج بصيغة JSON.
  - <<--single-port=<port>>: فحص منفذ واحد فقط بدلاً من نطاق.
  - --help: عرض تعليمات الاستخدام.
- أمثلة:
  - فحص المنافذ من ٢٠ إلى ٨٠ على المضيف ٨,٨,٨,٨ مع مهلة ٢ ثوانٍ:

```
./portscanner.sh 8.8.8.8 20-80 --timeout=2
```

  - فحص المنفذ ٢٢ على المضيف ١٩٢,١٦٨,١,١ وإخراج النتائج بصيغة JSON:

```
./portscanner.sh 192.168.1.1 --single-port=22 --json
```

٤. شرح عمل الأداة

١. معالجة المدخلات:
   - يستقبل السكربت عنوان المضيف ونطاق المنافذ (أو المنفذ المفرد) من المستخدم.
   - يقوم بمعالجة الخيارات التي يمررها المستخدم (مثل المهلة، صيغة الإخراج).
   - يتحقق من صحة نطاق المنافذ المدخل.
٢. فحص المنافذ:
   - يستخدم السكربت حلقة تكرار للمرور على كل منفذ في النطاق المحدد.
   - يحاول إنشاء اتصال TCP مع المنفذ باستخدام /dev/tcp/.
   - إذا نجح الاتصال، يعتبر المنفذ مفتوحًا.
   - يتم استخدام timeout لتحديد المهلة الزمنية لكل محاولة اتصال.
٣. تحديد الخدمات:
   - يحتوي السكربت على قاموس داخلي (SERVICES) يربط أرقام المنافذ الشائعة بأسماء الخدمات المعروفة (مثل HTTP على المنفذ ٨٠، و SSH على المنفذ ٢٢).
   - إذا تم العثور على منفذ مفتوح، يحاول السكربت تحديد الخدمة التي تعمل عليه من خلال القاموس.

٤. اكتشاف الثغرات الأمنية المحتملة:
- يحتوي السكربت على قاموس آخر (VULNS) يربط أرقام المنافذ ببعض الثغرات الأمنية الشائعة المرتبطة بها.
- إذا تم العثور على منفذ مفتوح، يتحقق السكربت مما إذا كانت هناك ثغرات معروفة مرتبطة بهذه الخدمة ويعرض تحذيرًا إذا لزم الأمر.

٥. إخراج النتائج:
- يعرض السكربت النتائج على الشاشة، مع استخدام الألوان لتمييز المنافذ المفتوحة والمغلقة.
- يحفظ السكربت النتائج في ملف نصي (اسم الملف يتضمن التاريخ والوقت).
- يمكن إخراج النتائج بصيغة JSON إذا تم تحديد الخيار --json.

٦. شريط التقدم:
- يعرض السكربت شريط تقدم أثناء الفحص لإعطاء المستخدم فكرة عن مدى التقدم.

٥. هيكل الكود

- المتغيرات:
  - متغيرات لتحديد الألوان في الإخراج (GREEN, RED, YELLOW, إلخ.).
  - متغيرات لتخزين الإعدادات الافتراضية (TIMEOUT, OUTPUT_JSON, إلخ.).
  - متغيرات لتخزين النتائج (RESULTS, OPEN_PORTS).
  - القواميس SERVICES و VULNS لتخزين معلومات الخدمات والثغرات الأمنية.
- الدوال:
  - usage(): تعرض تعليمات الاستخدام.
  - show_progress(): تعرض شريط التقدم.
  - scan_port(): تقوم بفحص منفذ واحد وتحديد حالته والخدمة.
- المنطق الرئيسي:
  - معالجة خيارات سطر الأوامر.
  - التحقق من صحة المدخلات.
  - حلقة التكرار التي تقوم بفحص المنافذ.
  - التعامل مع الإشارات (مثل SIGINT لإيقاف الفحص).
  - إخراج النتائج.

٦. القيود والملاحظات

- يعتمد السكربت على أدوات النظام المتاحة (مثل /dev/tcp/ و timeout).
- قد لا يكون دقيقًا بنسبة ١٠٠٪ في تحديد الخدمات، خاصة إذا كانت الخدمات تستخدم منافذ غير قياسية أو إذا كانت هناك جدران حماية.
- لا يقوم السكربت بإجراء فحص متعمق للثغرات الأمنية؛ إنه يوفر فقط تحذيرات عامة حول الثغرات المحتملة المرتبطة بالمنافذ.
- قد يكون أبطأ من أدوات فحص المنافذ المكتوبة بلغات أخرى (مثل بايثون أو C) بسبب طبيعة باش المفسرة.

٧. التحسينات المستقبلية المحتملة

- إضافة دعم لخيارات فحص إضافية (مثل فحص UDP).
- تحسين دقة تحديد الخدمات.

- إضافة المزيد من المعلومات حول الثغرات الأمنية المحتملة.
- تحسين الأداء.

* إضافة خيار لحفظ النتائج في صيغ مختلفة (مثل CSV).

٨. الخلاصة

- يقدم سكربت باش هذا أداة أساسية لفحص المنافذ.
- على الرغم من قيوده، يمكن أن يكون مفيدًا لمهام فحص الشبكات البسيطة.
- يوضح التحويل من بايثون إلى باش كيفية تنفيذ مهام مماثلة باستخدام أدوات مختلفة.

والحمدالله رب العالمين،،،

اعداد وتطوير:

عبدالرحمن المزاح – عبدالرقيب المشرقي

طارق عزيز سند – احمد مكرم الوجيه

كلية الرازي – يريم

أمن سيبراني – مستوى ثاني

الدفعة الثالثة