Penetration Testing — mehr Komfort mit nmap

honze Follow
Feb 17, 2017 · 5 min read

Der Netzwerkscanner nmap kommt sehr häufig zum Einsatz, wenn es um Penetration Testing, Vulnerability Scanning, Hardening oder schlicht Inventarisierung geht. Entweder direkt oder durch andere Scanner wie Nessus, OpenVAS, etc. Das Scannen ist eine Kunst für sich und wird in zahlreichen Anleitungen aus den verschiedensten Blickwinkeln betrachtet. Daher spare ich es mir hier auf die eigentlichen Scanoptionen einzugehen. Für den weiteren Verlauf ist dies nicht entscheidend. Der Schwerpunkt liegt auf der Ausgabe und der Weiterverarbeitung von Scanergebnissen.

Keinen Überblick mehr? — Quelle: https://unsplash.com/photos/1-29wyvvLJA

Vorbereitung

Zum besseren Verständnis und zum einfacheren Nachvollziehen gehe ich alles anhand von Beispielen durch.

Mein Setup

- Kali Linux rolling release (mit Patchstand vom Februar 2017)
- nmap 7.40, wobei auch andere Versionen funktionieren sollten
- scanme.nmap.org als Ziel zum Testen

Erster Scan

Mit `nmap scanme.nmap.org` wird das Ziel gescannt und das Ergebnis auf der Konsole ausgegeben:

```
Starting Nmap 7.40 ( https://nmap.org )
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (1.5s latency).
Other addresses for scanme.nmap.org (not scanned):
2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 992 closed ports
PORT STATE SERVICE
22/tcp open ssh
25/tcp filtered smtp
53/tcp filtered domain
80/tcp open http
110/tcp filtered pop3
515/tcp filtered printer
9929/tcp open nping-echo
31337/tcp open Elite

Nmap done: 1 IP address (1 host up) scanned in 94.14 seconds
```

Wie würde jetzt eine Weiterverarbeitung aussehen? Sehr manuell. Für eine ansprechende Aufbereitung (z. B. für einen Bericht) wäre “Copy & Paste” sowie einiges an Formatierung notwendig. Für den Aufruf von anderen Tools zum Scannen von den oben erkannten Services wäre auch Tipparbeit notwendig. Das ist mühsam, fehleranfällig, skaliert nicht und macht wenig Spaß. Es schreit förmlich nach Automatisierung.

Aufbereitung von nmap Scans

Der erste Schritt ist die Ausgabefunktion von nmap zu nutzen. Damit öffnet man die Tür für eine bessere Ergebnissicherung und Anbindung weiterer Tools. Ich verwende gerne die Option `-oA <basename>`, da sie alle wesentlichen Ausgabeformate enthält. Diese sind die normale Ausgabe (.nmap), die grepbare Ausgabe (.gnmap) zur Weiterverarbeitung mit `grep` und die XML-Ausgabe (.xml).

Falls man später noch einmal Details nachlesen möchte, wird man mindestens in einer Dateien fündig. Die XML-Datei ist auf den ersten Blick am schwersten zu lesen. Allerdings können XML-Dateien mit XSL (Extensible Stylesheet Language) formatiert werden und somit

angenehmere und menschenlesbare Ausgaben erzeugt werden. Gleichzeitig bleibt die XML-Datei maschinenlesbar. Eine einfache Möglichkeit bietet die Option `--webxml` was ein Alias für `--stylesheet` `https://nmap.org/data/nmap.xsl` ist. Mit `--stylesheet` können natürlich auch eigene Stylesheets angegeben werden, die z. B. auf die Corporate Identity angepasst wurden.

Beispiel

Mit dem Aufruf `nmap scanme.nmap.org -oA scanme --webxml` wird folgende XML-Datei erzeugt, die im Firefox z. B. so aussieht:

Screenshot der XML-Datei mit Stylesheet im Firefox (Ausschnitt)

Nach dem Öffnen der Datei sind die grauen Zeilen ausgeblendet. D. h. es werden nur die offenen Ports (grün) angezeigt. Mit einem Klick auf “filtered” in der Kopfzeile der Spalte “State” können die gefilterten Ports eingeblendet werden. Dies trägt erheblich zur Übersichtlichkeit bei.

Zenmap

Die GUI zu nmap heißt Zenmap und kann selber Scans ausführen. Allerdings lassen sich mit Zenmap auch alte Scans als XML-Datei einlesen und auch vergleichen. Es können bei Bedarf auch mehrere Scans eingelesen werden, falls z. B. TCP und UDP zusammen dargestellt werden sollen, aber nicht im selben Scan erfasst wurden. Zenmap ist wie nmap in Kali Linux vorinstalliert.

Automatisierte Weiterverarbeitung von nmap Scans

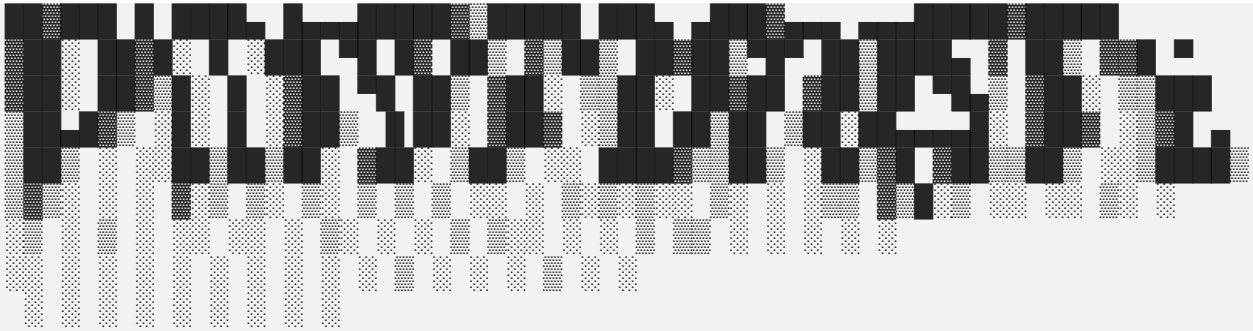
Beim Scannen größerer Netzwerke (>100 Hosts) finden sich häufig mehrere gleichartige Systeme. Die haben dann z. B. einen Webserver, einen FTP-Server und SSH. Nun muss für jeden Service ein Tool gestartet werden, um weitere Aussagen über den Zustand der Systeme treffen zu können. Die Aufrufe der Scanner unterscheiden sich nur durch Name bzw. IP, Port und Ablageort der Ausgaben. Name, IP, Port und Service lassen sich aus der XML-Datei von nmap auslesen. Es schreit erneut nach Automatisierung.

Pwntomate

Für den oben beschriebenen Fall habe ich ein Python-Skript geschrieben, das als Eingabe eine XML-Datei von nmap entgegen nimmt und als Ausgabe ein Bash-Skript liefert. Das Bash-Skript ruft dann alle Scanner auf und legt die Ergebnisse strukturiert nach IP, Port und Tool ab. Hier ein Beispielaufruf für die XML-Datei die ich weiter oben generiert habe.

```
root@kali:~/pentest/pwntomate# ./pwntomate.py -i
../scanme.nmap.org/scanme.xml

_____ _ ____  _  ____________ ___  ___       _____________
```



```
pwntomate 0.0.2#alpha
https://github.com/honze-net/pwntomate


#!/bin/bash
# autogenerated script by pwntomate 0.0.2#alpha
# https://github.com/honze-net/pwntomate
# ./pwntomate.py -i ../scanme.nmap.org/scanme.xml
mkdir -p ~/pwntomate/45.33.32.156/80/dirb
dirb http://45.33.32.156:80 -o
~/pwntomate/45.33.32.156/80/dirb/dirb.txt
mkdir -p ~/pwntomate/45.33.32.156/80/nikto
nikto -host 45.33.32.156 -port 80 -output
~/pwntomate/45.33.32.156/80/nikto/nikto.html
```

Man kann gut erkennen, wie Aufrufe für `dirb` und `nikto` erfolgen. Ähnlich können Aufrufe für alle Arten von Diensten gestartet werden. Das klappt auch, wenn ein Webserver auf Port 21 (eigentlich FTP) läuft, da nmap den Dienst an sich erkennt und sich wenig am Port orientiert.

Der Source Code ist auf GitHub verfügbar. Dort gibt es auch weitere Informationen zu Abhängigkeiten, Benutzung und eine Liste der bisher implementierten Tools. Die Konfigurationseinstellungen für die Tools sind in JSON geschrieben, was es einfacher machen soll neue Tools hinzuzufügen. Hier `dirb` und `nikto` als Beispiel:

```
{
        "toolname": "dirb",
        "command": "dirb http{s}://{ip}:{port} -o
{baseoutputdir}/{ip}/{port}/{toolname}/{toolname}.txt",
        "trigger": ["http", "https", "http-mgmt", "http-alt"],
        "active": true
}


{
        "toolname": "nikto",
        "command": "nikto -host {ip} -port {port} -output
{baseoutputdir}/{ip}/{port}/{toolname}/{toolname}.html",
        "trigger": ["http", "https", "http-mgmt", "http-alt"],
        "active": true
}
```

Zur Vollständigkeit sei gesagt, dass nmap NSE (nmap scripting engine) mit LUA viele interessante Konzepte unterstützt. Allerdings ist das Schreiben solcher LUA-Skripte wesentlich komplizierter als die Verarbeitung von XML-

Dateien. Eine Einführung gibt es unter http://www.scip.ch/?labs.20100507

Ich wünsche viel Spaß beim Ausprobieren. Fragen und Wünsche gerne in die Kommentare oder sonst irgendwie an mich.

Nmap Deutsch Medium Auf Deutsch Security Penetration Testing

50 claps

WRITTEN BY

honze

www.honze.net — 1+1=10 — München

Follow

Write the first response

More From Medium

Top on Medium

10 Bad Habits of Unsuccessful People

Darius Foroux in Forge
Jul 9 · 5 min read ★

46K

Top on Medium

The Uncanny Power of Incompetent Men

Danny Wallace in Forge
Jul 25 · 9 min read ★

5.8K

Top on Medium

Functional Programming? Don't Even Bother, It's a Silly Toy

Ilya Suzdalnitski in Better...
Jul 29 · 13 min read ★

982