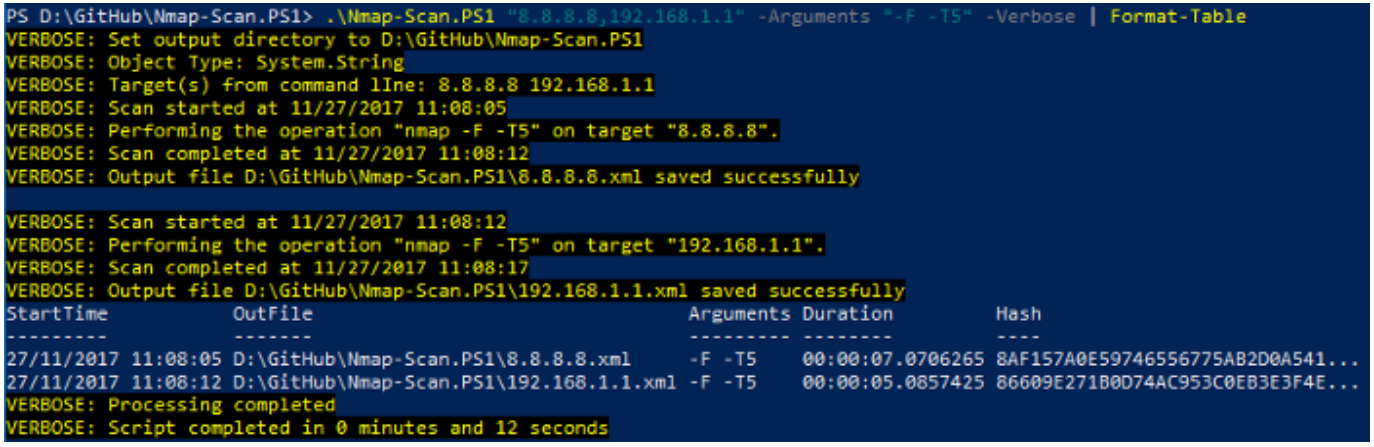
Using Powershell to programmatically run nmap scans

Ethan Follow
Dec 12, 2017 · 2 min read



https://github.com/e-sterling/Nmap-Scan.PS1

Earlier this week I was tasked with running a series of nmap scans across around 130 individual subnets. One of the requirements was each subnet

was scanned and the results output to an XML file for further processing. As we are primarily a Windows environment, and the scan parameters for each subnet are identical, I figured it made sense to write a quick PowerShell script to take in a list of subnets and run nmap against each one in order.

The resulting script could only be described as a quick hack, about ten lines of PowerShell to read a text file and iterate over each line, running the required nmap command and checking to make sure that the XML file actually saved.

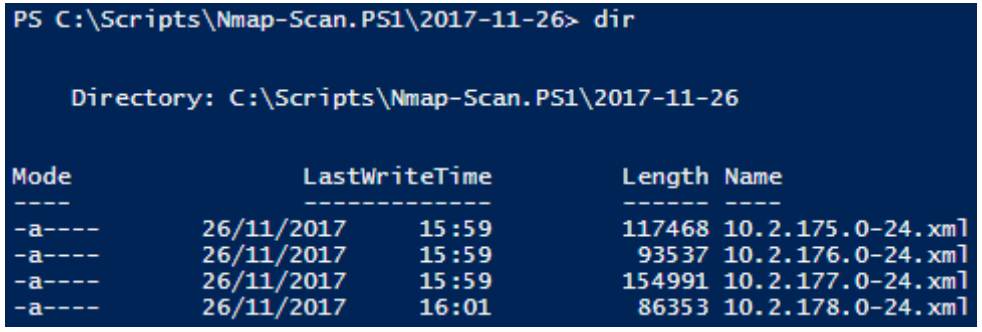
The script did the job, and considering it was only a few minutes work vs. manually running ~130 nmap commands it was perfectly adequate, but I knew I could do better. I'm still a beginner when it comes to PowerShell so this also presented the perfect opportunity to make something useful and learn an thing or two along the way!

Nmap-Scan.ps1

A bit of work later, and after tonnes of StackExchange articles, I've decided to release the resulting script, Nmap-Scan.ps1 to GitHub. Hopefully it might

help someone else in the future if they need to do something similar.The script can take inputs in the form of:

- A simple list of targets
  .\Nmap-Scan.ps1"192.168.0.1/24,scanme.nmap.org"
- A text file (or set of files) containing targets
  *Get-ChildItem *.txt | .\Nmap-Scan.ps1*
- ADComputer objects from Powershell AD Module
  Get-ADComputer -Filter {Name -like "HostName"} | .\Nmap-Scan.ps1*



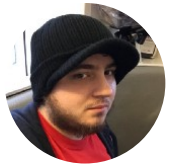
Once provided with a list of targets the parameter *-Arguments* can be used to specify any command line args you want to pass to nmap to control the resulting scan. As standard the argument "-oX" is passed to nmap to output

the results to an XML file. The XML file is given the name [target].XML and is stored by default in the current working dir. The parameter *-OutDir* can be used to specify a directory to output your scan results.

Overall this has been a really fun learning experience, this is the first thing I've ever released open source (and the first time I ever contributed to GitHub!) so it's been great to find out how this all works. There still a few things I'd like to add on to make this script even more useful, but for now if you're interested you can grab a copy on GitHub!

Powershell Nmap Microsoft Infosec

3 claps



WRITTEN BY

Ethan

IT man, Infrastructure Engineer, with a big interest in InfoSec and all the related good stuff

Follow

Write the first response

More From Medium

Related reads

h1–702 CTF — Web Challenge Write Up

Amal Murali
Jul 1, 2018 · 12 min read

1K

Related reads

XSS-Auditor — the protector of unprotected

terjanq in InfoSec Write-ups
Apr 25 · 4 min read ★

322

Related reads

Interlace: A Productivity Tool For Pentesters and Bug Hunters - Automate and Multithread Your...

Luke Stephens (@hakluke)
Feb 3 · 5 min read ★

730