## Exercise with XML and more complicated objects or data structures

This exercise was inspired by <https://isc.sans.edu/forums/diary/Pentest+Time+Machine+NMAP+Powershell+whatever+tool+is+next/20653/>.

We have seen how we can save and import data in CSV format. A CSV file saves simple arrays of objects where each row of the CSV is an object, and each column of the file contains the properties of the objects. More complex data can be stored in eXtended Markup Language (XML). XML is similar to HTML, in that it has tags, but in XML the tag names can be adjusted to suit the data. Nmap can output its scans in XML, so we’ll see how we can import them into PowerShell and manipulate it.

Note: Many languages include support for CSV and XML, or have modules that can be imported to add support. PowerShell is not unique in supporting XML.

Run an Nmap scan of your virtual machine network. You can install Nmap on your Windows host if you haven’t already, or run the scan from your Kali VM. The command for the scan should look something like this.

nmap –sV 192.168.xxx.0/24 –oX nmapout.xml

The –sV argument tells Nmap to do a version scan, xxx in the IP address should be replaced with the numbers for your VM network, and –oX nmapout.xml tells Nmap to save the output in XML format in a file called nmapout.xml.

Open the nmapout.xml file (text editor in Kali, or WordPad (not Notepad) in Windows) and look at the structure. Notepad won’t understand the Linux line breaks and will put all the data on one line, which is why we don’t want to use it.

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE nmaprun>

<?xml-stylesheet href="file:///usr/bin/../share/nmap/nmap.xsl" type="text/xsl"?>

<!-- Nmap 7.01 scan initiated Mon Feb 1 20:08:07 2016 as: nmap -sV -oA nmapout 192.168.217.0/24 -->

<nmaprun scanner="nmap" args="nmap -sV -oA nmapout 192.168.217.0/24" start="1454375287" startstr="Mon Feb 1 20:08:07 2016" version="7.01" xmloutputversion="1.04">

<scaninfo type="syn" protocol="tcp" numservices="1000" services="1,3-4,6-7,9,13,17,19-26,30,32-33,37,42-43,49,53,70,79-85,88-90,99-100,106,109-111,113,119,125,135,139,143-144,146,161,163,179,19  
<snip>

The first information you see will be a long header that tells when the scan was run, what the command line was, what the Nmap version was, and a lot of other stuff. It also lists all the port numbers (services) that Nmap scanned. Since Nmap scans 1,000 ports by default, that list is quite long.

If you look further down the file, you’ll see the data, and the data structure, that the file uses for describing the hosts it found.

<host starttime="1454375289" endtime="1454375368"><status state="up" reason="arp-response" reason\_ttl="0"/>

<address addr="192.168.217.1" addrtype="ipv4"/>

<address addr="00:50:56:C0:00:08" addrtype="mac" vendor="VMware"/>

<hostnames>

</hostnames>

<ports><extraports state="filtered" count="995">

<extrareasons reason="no-responses" count="995"/>

</extraports>

<port protocol="tcp" portid="135"><state state="open" reason="syn-ack" reason\_ttl="128"/><service name="msrpc" product="Microsoft Windows RPC" ostype="Windows" method="probed" conf="10"><cpe>cpe:/o:microsoft:windows</cpe></service></port>

<port protocol="tcp" portid="139"><state state="open" reason="syn-ack" reason\_ttl="128"/><service name="netbios-ssn" product="Microsoft Windows 98 netbios-ssn" ostype="Windows 98" method="probed" conf="10"><cpe>cpe:/o:microsoft:windows\_98</cpe></service></port>

<port protocol="tcp" portid="445"><state state="open" reason="syn-ack" reason\_ttl="128"/><service name="microsoft-ds" product="Microsoft Windows 10 microsoft-ds" ostype="Windows 10" method="probed" conf="10"><cpe>cpe:/o:microsoft:windows\_10</cpe></service></port>

<port protocol="tcp" portid="1027"><state state="open" reason="syn-ack" reason\_ttl="128"/><service name="msrpc" product="Microsoft Windows RPC" ostype="Windows" method="probed" conf="10"><cpe>cpe:/o:microsoft:windows</cpe></service></port>

<port protocol="tcp" portid="1028"><state state="open" reason="syn-ack" reason\_ttl="128"/><service name="msrpc" product="Microsoft Windows RPC" ostype="Windows" method="probed" conf="10"><cpe>cpe:/o:microsoft:windows</cpe></service></port>

</ports>

<times srtt="547" rttvar="311" to="100000"/>

</host>

The XML above shows the data for one host that Nmap detected. A complete dataset would show many hosts. You can see above that the host has several one port**s** record. Each port**s** record has several port records—there is one port record for each open port that Nmap found. In this case, Nmap scanned 1000 ports, and the 995 of those that did not answer were marked as “filtered’, “no response.” The five ports that did answer each have their own port record.

1. Host
   1. starttime
   2. endtime
   3. status
      1. state
      2. reason
      3. reasontttl
   4. address
      1. addr
      2. addrtype
   5. hostnames
   6. ports
      1. extraports
         1. state
         2. count
         3. extrareasons
            1. reason
            2. count
      2. port
         1. protocol
         2. portid
         3. state
            1. state
            2. reason
            3. reasonttl
         4. service
            1. name
            2. product
            3. ostype
            4. method
            5. conf
            6. cpe

cpe

1. Host---record for next host