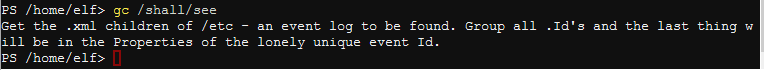
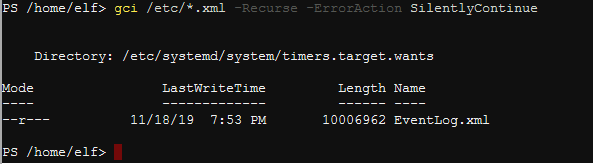
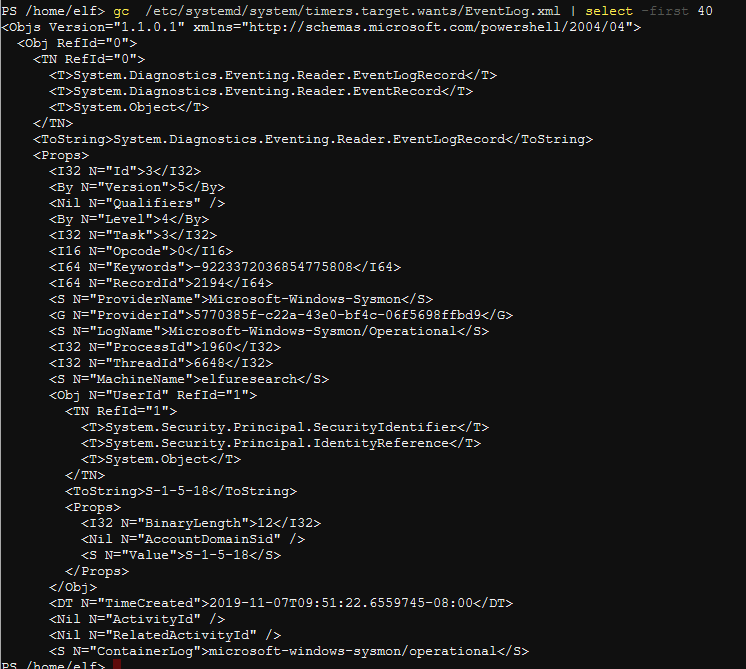
# Christmas Cheer Laser, part 8

Until I see solutions using XML in PowerShell, this is pasted from my solution document as a place holder.



To find the .xml file,  
  
gci /etc/\*.xml -Recurse -ErrorAction SilentlyContinue

This is what part of one event looks like.  


This one was hard. I couldn’t make Powershell XML work for me, so I just wrote a small script. (I’m looking forward to reading the write up for someone that did it directly in XML.) The .Id the hint refers to is in the line <I32 N=”Id”>3</I32>. The I32 part means it is a 32 bit Integer.

$ids =''

$regex = '<I32 N=.Id.>(\d)</I32>'

Get-Content /etc/systemd/system/timers.target.wants/EventLog.xml | ForEach-Object {

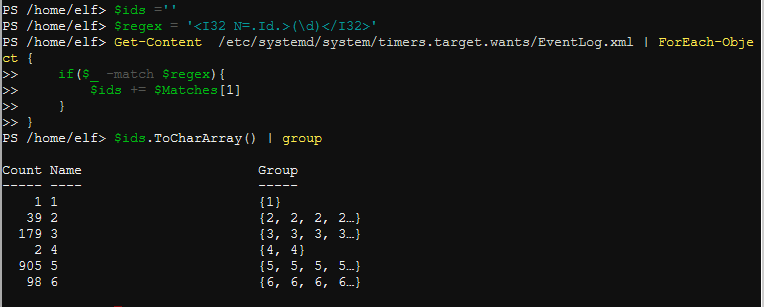
if($\_ -match $regex){

$ids += $Matches[1]

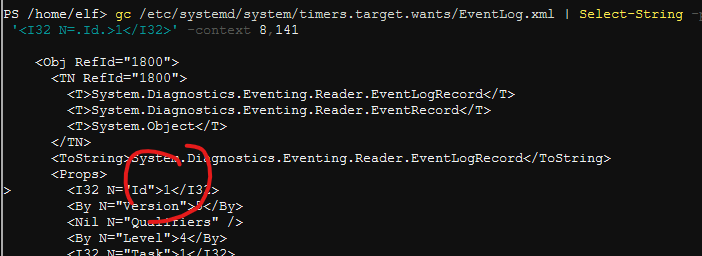
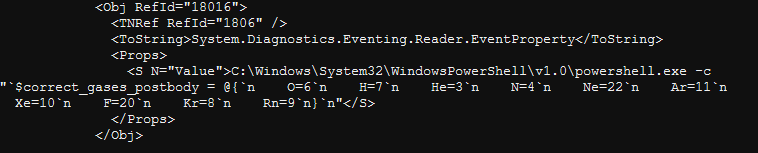
}

}

$ids.ToCharArray() | group   
  
The regular expression in the second line finds the N=”Id” line (I didn’t bother escaping the quotes, and replaced them with the single character wild card, ‘.’) The (\d) saves the number (3 in the example <I32 N=”Id”>3</I32>) in the variable $Matches, which are collected in $ids. To be able to group by the numbers we recover, we have to change the string $ids to an array of characters.



The lonely unique Id referenced in the hint is the number 1 (of course, one is the lonliest number.)

We can find the clue by selecting the .Id we want and looking at lines before and after to get the entire event (-Context 8,141. I played with those numbers until I had the entire event, nothing magic.)  
  
<snip>  
  
gc /etc/systemd/system/timers.target.wants/EventLog.xml | Select-String -pattern '<I32 N=.Id.>1</I32>' -context 8,141

So, we have our third parameter.  
$correct\_gases\_postbody = @{`n O=6`n H=7`n He=3`n N=4`n Ne=22`n Ar=11`n Xe=10`n F=20`n Kr=8`n Rn=9`n}

Note: The backtick character is Powershell’s escape character, so `n is \n, or newline. This is JSON put into a Powershell hash (like a Python dictionary.)

This hint trail has grown cold. To find our fourth parameter we need to back up to the place where we found the runme.elf file. When solving this challenge, it pays to know what OS we are running on. PowerShell Core is available for many OSs. Find the OS, and then ask yourself, “What do I need to do to execute a script/executable I’ve written on this OS?”

## Questions

1. What OS is this terminal runnning?
2. How do you run a brand new script/executable in this OS? (If you need a hint look in /bin. The permissions aren’t in rwx format, but there is still useful information.)
3. What is the value of the last parameter?