1) What's the version and year of the Windows machine?
    a) To acquire this information, simply run `Get-ComputerInfo`.
    b) Navigate to the `WindowsProductName` property and its value is **Windows Server 2016** Datacenter.
        i) NOTE: You can narrow down the search specifically for this property by running `Get-ComputerInfo -Property "WindowsProductName"`.
2) Which user logged in last?
    a) Unfortunately, there is no simple way to iterate through all users so you'll need to first use `net users` to acquire all user names and then do `net user USERNAME | findstr "Last"` to get their individual `Last logon` time.
    b) Doing this with each, I find that **Administrator** was the last, which makes sense since that's the user I just logged in as.
3) When did John log onto the system last?
    a) Running `net user John`, we see that he last logged in on **03/02/2019 5:48:32 PM**
4) What IP does the system connect to when it first starts?
    a) We can find this information in the registry. Simply run `reg query hklm\software\microsoft\windows\currentversion\run` and it will be seen in the `UpdateSvc` field. The IP discovered is **10.34.2.3**
5) What two accounts had administrative privileges (other than the Administrator user)?
    a) You can get this information by running `net localgroup Administrators`, which yields two users **Jenny** and **Guest**
6) What's the name of the scheduled task that is malicious?
    a) To get a list of scheduled tasks, run `Get-ScheduledTask`. To know the answer comes from knowing which tasks are not typical.
    b) The task that stood out to me was **Clean file system**
7) What file was the task trying to run daily?
    a) We'll need to assign the `Get-ScheduledTask` output to a variable and then access that variable's `Actions` to see what it does. Here's how:

```
PS C:\Users\Administrator> $task = Get-ScheduledTask -TaskName "Clean file system"
PS C:\Users\Administrator> $task.Actions
```

        i) Looking at the `Execute` property, we see the value **nc.ps1**
8) What port did this file listen locally for?
    a) Fortunately, the port was actually specified in the `Arguments` property above when looking at the previous answer's output. I knew this because `nc.ps1` is really just the Netcat program in a PowerShell script file (.ps1) and the `-l` option is how you specify a port to listen for. The port was **1348**.
9) When did Jenny last logon?

a) Once again, running `net user Jenny` we see they last logged on **Never**, which is odd. Must be the attacker account.

10) At what date did the compromise take place?
    a) This is a really poorly worded question, but I remembered the malicious task was running from the `/` path, which is the same as the `C:` path, so I just ran `ls C\:` and noticed some odd directory timestamps that were all within the same hour on the same date.
        i) The strange date was **03/02/2019**

11) During the compromise, at what time did Windows first assign special privileges to a new logon?
    a) We are going to be looking in the `Security` event logs (as these most likely contain events like privilege changes) and we will be looking for tasks in the `Security Group Management` (which I found out by doing some research) category to find information like this.
    b) Below is how I managed to find the log that contained the answer:

```
Get-EventLog -LogName Security -Before "03/02/2019 16:30:00" -After
"03/02/2019 16:00:00" |
ForEach-Object {
    $message = $_.Message
    $eventTime = $_.TimeGenerated.toString("HH:mm:ss")
    Write-Host "$eventTime - $_ - $message"
} |
findstr "Security Group Management"
```

        i) I used the `Before` and `After` options to narrow down the search to anytime between 4:00 PM and 4:30PM on the weird date we found in the previous answer.
        ii) I wanted to capture the `Message` property of each object acquired in the output because I figured it would contain something similar to “assign special privileges to new logon”
        iii) I needed to display the time all the way down to the second, so I needed to iterate through via `ForEach-Object` because PowerShell does not have a single option to change the time format to `HH:mm:ss` instead of `HH:mm`.
        iv) **To find the answer:** I simply scrolled up from the bottom of the output (Windows displays the logs with the oldest being at the end) and then kept going until I found the first instance of `Special privileges assigned to new logon` and and the timestamp was **03/02/2019 04:04:49 PM**.

12) What tool was used to get Windows passwords?

a) I noticed that a command prompt window kept opening up with the title C:\TMP\mim.exe which is most likely a clue for this question's answer.
b) So I decided to visit C:\TMP and I noticed a singular .txt file along with several executables. I ran type mim-out.txt to see the file contents and the tool name was at the very top of the file. It's called **mimikatz**.

13) What was the attacker's external control and command servers IP?
a) Odds are, the attacker would have modified the `hosts` file and added the C2 IP. Let's display its contents via:

```
type C:\Windows\System32\drivers\etc\hosts
```

b) There's an IP in there that doesn't really fit, which is **76.32.97.132**.

14) What was the extension name of the shell uploaded via the server's website?
a) Odds are, the answers lie in the `/inetpub/wwwroot` directory as this is Windows' equivalent to Linux's `/var/www/` directory and this question has to do with the server's *website*.
b) Visiting the above directory, we see that all the files in there are incredibly suspicious.
i) b.**jsp**
ii) shell.gif
iii) tests.jsp

15) What was the last port the attacker opened?
a) Firstly, I ran the following to get a list of all inbound firewall rules that were not assigned to a group, as an attacker would most likely not be assigned to a group to either avoid detection or their account creation process never involved being assigned to one.

```
Get-NetFirewallRule -Direction Inbound |
Where-Object { $_.Group -eq $null -or $_.Group -eq '' }
```

b) Upon running the above, I found an odd rule with a `DisplayName` of `Allow outside connections for development`, which is very suspicious sounding. Now, let's target this one and list its ports. To do that, we have to run a convoluted command:

```
Get-NetFirewallRule -Direction Inbound |
Where-Object { $_.Group -eq $null -or $_.Group -eq '' } |
Format-Table -Property DisplayName, @{ Name='LocalPort';
Expression={($PSItem | Get-NetFirewallPortFilter).LocalPort}}
```

i) This will show our DisplayName so we can tell which rule had which port and the local port will be displayed, which ends up being **1337**.

ii) I focused on `Inbound` firewall rules because the attacker appears to have attacked the machine remotely, meaning they would want their data to easily move from their remote machine to our compromised one. The easiest way to do that, especially if you have administrative privileges, is to just set a permissive firewall rule.

16) Check for DNS poisoning, what site was targeted?
    a) This answer is simple, the site that was targeted is the same hostname affiliated with the C2 server's IP, which was **google.com**