Fortinet Technologies (Canada) Inc.

Software Development QA Engineer

Written Interview Questions

November 2015

Instructions

Read the document: Instructions.txt before starting the questions.

Q1. The top-level script for extracting NVD XML files is nvd\_extract.pl in the scripts directory. Running it with the right argument will parse the given XML file and print out the output.

There is an error in the function: NVD::extract(), which is in the imported module, NVD. Find and correct this error so that the command:

perl scripts/nvd\_extract.pl

will print the correct usage information.

Q2. Provide the NVD XML file in the download directory as an argument to the script. Redirect the output to a file in the log directory.

Q3. Download the NVD XML files from the NIST feed for the years 2011 to 2014. The NVD module provided in this task only supports the NVD XML 1.2.1 schema. Either the .zip or .gz files may be downloaded. Extract the downloaded files into the download directory.

The XML file for year 2015 is already in the download directory.

Q4. Modify the script: nvd\_extract.pl to accept any number of NVD XML files from the command line and call the module NVD::extract() on each file. There is no need to make changes to the NVD module for this purpose.

The script should print out the results for all the files provided as input.

Run the script on the files downloaded in Q3 above.

Q5. The output of the script: nvd\_extract.pl includes various fields used as XML tags in the original NVD XML file. It is intended to extract and save the following fields for each entry found into a database:

cve\_id

severity

published

modified

Write a SQL command that could be used to make a new database table, which will contain records for all entries extracted.

For this task, we assume the cve\_id field is always unique.

Q6. Write a new function in NVD named save2db() that:

- accepts the output of NVD::extract(), and the connection

information to a database

- connects to the database specified

- makes a new database table as above

- for each entry record in the output received from

NVD::extract(), save the fields listed in Q5 above

New private functions may be added in NVD and called as required.

You may use any relational database server: MySQL, PostgreSQL, SQLite, etc.

The Perl DBI module may be useful for this purpose.

Q7. Write a simple script called vuln2db.pl similar to nvd\_extract.pl. It should accept any number of NVD XML files. For each given XML file, it should call NVD::extract(), and the new NVD::save2db() function created in Q6, to save all the entries into a database.

You may provide the database connection information as an input to the new script, or as variables inside the script.

Q8. When calling NVD::extract(), determine what may occur in each of the

following cases:

- it is called with an empty file

- it is called with a non-empty file that is not an XML file

- it is called with a file that does not exist

Q9. Provide the minimal changes required to correct any issues observed in Q8 in the top-level scripts: nvd\_extract.pl and the second script created in Q7.

In the case multiple NVD XML files are provided as input, the script should print an error or warning as appropriate, but continue where possible.

Q10. Modify the NVD module to handle any issues observed in Q8 above, such that it will continue or exit gracefully, even without the changes in Q9.

The XML::LibXML module provides error handling support that may be helpful in this respect.

BONUS Questions

The following questions are optional. You may skip (or answer) any or all

questions.

BQ0. A private function called parse\_nvd\_entry() exists in the NVD module. A call to this private function is commented out in extract(). Remove this comment and re-run each of the scripts created or modified in previous questions to ensure they still run correctly.

BQ1. The function parse\_nvd\_entry() in NVD is recursive - it calls itself. Re-write the function to remove recursion.

You may create new private functions and call them as required, as long as none of them is recursive.

BQ2. Modify NVD::save2db() to create a second table, with the following:

cve\_id

name

vendor

Add an autoincrement column as a unique field for each row.

For each entry record in the output of NVD::extract(), as provided to NVD::save2db(), it should still save records to the first table created in Q6. It should also save the product name and vendor, along with the CVE ID, into the new table.

The product name and vendor are both available in the array:

$$vuln{<cve\_id>}{vuln\_soft}{prod}

where <cve\_id> is the current CVE ID. Here is an example array slice, showing name and vendor:

'vuln\_soft' => HASH(0x897b140)

'prod' => ARRAY(0x897b560)

0 HASH(0x897b320)

'name' => 'windows\_2003\_server'

'vendor' => 'microsoft'

'vers' => HASH(0x897b4b8)

'edition' => 'sp2'

'num' => ''

'vers' => ''

1 HASH(0x897b488)

'name' => 'windows\_7'

'prod' => HASH(0x897b440)

'vers' => ARRAY(0x897b650)

0 HASH(0x897b5c0)

'edition' => 'sp1:x64'

'num' => ''

'vers' => ''

1 HASH(0x897b590)

'edition' => 'sp1:x86'

'num' => ''

'vers' => ''

'vendor' => 'microsoft'

2 HASH(0x897b6c8)

'name' => 'windows\_server\_2008'

'prod' => HASH(0x897b4a0)

'vers' => ARRAY(0x897b830)

0 HASH(0x897b7a0)

'edition' => 'sp2'

'num' => ''

'vers' => ''

1 HASH(0x897b770)

'edition' => 'sp1'

'num' => 'r2'

'vers' => ''

'vendor' => 'microsoft'

3 HASH(0x897b848)

'name' => 'windows\_vista'

'prod' => HASH(0x897b6e0)

'vers' => HASH(0x897b920)

'edition' => 'sp2'

'num' => ''

'vers' => ''

'vendor' => 'microsoft'

This is from the following XML fragment:

<vuln\_soft>

<prod name="windows\_2003\_server" vendor="microsoft">

<vers num="" edition="sp2"/>

</prod>

<prod name="windows\_7" vendor="microsoft">

<vers num="" edition="sp1:x64"/>

<vers num="" edition="sp1:x86"/>

</prod>

<prod name="windows\_server\_2008" vendor="microsoft">

<vers num="" edition="sp2"/>

<vers num="r2" edition="sp1"/>

</prod>

<prod name="windows\_vista" vendor="microsoft">

<vers num="" edition="sp2"/>

</prod>

</vuln\_soft>