**Developing a scanner tool extension to the web application**

To implement the integration for a new scanning tool, a new sub-directory must be created under the "scanner\_types" directory. This sub-directory may store all helper files/libraries that the implementation may need. There is one file in particular that must exist in this directory, a .py file with the same name as the created sub-directory. This .py file will henceforth be referred to as the scanner module.

The scanner module should implement at least four classes. The first one should extend the Scanner base class defined in models.py. You should name it as (scanner module name)\_Scanner, where (scanner module name) is the name of the sub-directory that you created above.

This Scanner class is a model that will be used to create instances of the scanner tool that you have chosen. For example, if you are writing a scanner module for Nessus, an 'instance' of Nessus would contain the URL address of its API, its API access and secret keys, and other instance-specific information. This class must also define several required methods, which are explained in the skeleton code below.

The Scan class is a model that will be used to create instances of a scan running under the scanner tool. For example, the Scan class for Nessus may include a text field for endpoints (IP addresses), and an integer field for storing the unique identifier of the scan on Nessus.

You may use the skeleton code below to write the scanner module, and refer to the pre-included scanner modules for guidance.

**from** **dd\_downloader.models** **import** Scanner, Scan

**from** **django.db** **import** models #Required to add attributes to your models, such as character or integer fields.

**from** **django.utils** **import** timezone #Required if you would like to update the create\_date, start\_date, and end\_date of your scans

**from** **django** **import** forms #Required for the scan/scanner creation/edit forms.

**import** **logging**

logger = logging.getLogger('dd\_downloader.scanner\_types.Example\_Tool')

**class** **Example\_Tool\_Scanner**(Scanner):

#Required for the web application to recognize this model.

**class** **Meta**:

app\_label='dd\_downloader'

#Required method. Return a path to a template .html file which defines what view you want to see when you view the details of a scanner for this scanner tool.

**@staticmethod**

**def** **get\_scanner\_detail\_template\_path**():

**return** 'dd\_downloader/scanner\_types/Example\_Tool/scanner\_detail.html'

#Required method. Return a path to a template .html file which defines what view you want to see when you view the details of a scan for this scanner tool.

**@staticmethod**

**def** **get\_scan\_detail\_template\_path**():

**return** 'dd\_downloader/scanner\_types/Example\_Tool/scan\_detail.html'

#Required method. Simply return a reference to the scanner creation form defined below.

**@staticmethod**

**def** **get\_scanner\_create\_form\_class**():

**return** Example\_Tool\_Scanner\_Create\_Form

#Required method. Simply return a reference to the scan creation form defined below.

**@staticmethod**

**def** **get\_scan\_create\_form\_class**():

**return** Example\_Tool\_Scan\_Create\_Form

#Any attributes that an instance of Example\_Tool can be defined here.

api\_url = models.CharField('Example\_Tool API URL', max\_length=**200**)

api\_key = models.CharField('Example\_Tool API Key', max\_length=**200**)

#Dummy function to 'create' a scan on an Example\_Tool instance.

**def** **create\_scan**(self, endpoints):

**try**:

scan\_identifier = #communicate with the Example\_Tool API somehow using your own api\_url and api\_key to 'create' a scan with endpoints and return some unique identifier for the scan, using the Requests library or whatever

**except** **Exception** **as** e:

logger.exception('Could not create an Example\_Tool scan!')

**else**:

**return** scan\_identifier

#Dummy function to 'start' a scan on an Example\_Tool instance.

**def** **start\_scan**(self, scan\_identifier):

**try**:

#communicate with the Example\_Tool API to attempt to 'start' the scan with the given identifier.

**except** **Exception** **as** e:

logger.exception('Could not start an Example\_Tool scan!')

**return** False

**else**:

**return** True

#Dummy function to poll a scan on an Example\_Tool instance.

**def** **poll\_scan**(self, scan\_identifier):

**try**:

is\_complete = #communicate with the Example\_Tool API to see if the scan at the given identifier is complete or not

**except** **Exception** **as** e:

logger.exception('Polling an Example\_Tool scan failed')

**else**:

**return** is\_complete

#Dummy function to stop a scan on an Example\_Tool instance.

**def** **stop\_scan**(self, scan\_identifier):

**try**:

#communicate with the Example\_Tool API to attempt ot 'stop' the scan with the given identifier.

**except** **Exception** **as** e:

logger.exception('Could not stop an Example\_Tool scan!')

**return** False

**else**:

**return** True

#Dummy function to 'retrieve' a scan on an Example\_Tool instance.

**def** **retrieve\_scan**(self, scan\_identifier):

**try**:

result = #communicate with the Example\_Tool API to retrieve the results of the scan at the given identifier

**except** **Exception** **as** e:

logger.exception('Retrieving an Example\_Tool scan failed')

**return** None

**else**:

**return** result

**class** **Example\_Tool\_Scan**(Scan):

#Required for the web application to recognize this model.

**class** **Meta**:

app\_label='dd\_downloader'

#Attributes that a scan instance in Example\_Tool may have.

Example\_Tool\_scan\_identifier = models.IntegerField(null=True,default=None) #A unique identifier for a scan object inside an instance of Example\_Tool.

endpoints = models.TextField() #An input of endpoints/files/addresses that you would like to scan using Example\_Tool.

**def** **create**(self):

**if** **not** self.can\_create():

logger.warning('Tried to create non-creatable scan')

**return**

self.status = Scan.CREATING

self.save()

create\_identifier = self.scanner.create\_scan(self.endpoints)

**if** created\_scan\_id:

self.Example\_Tool\_scan\_identifier = create\_identifier

self.status = Scan.CREATED

**else**:

self.status = Scan.ERRORS

self.save()

**def** **start**(self):

**if** **not** self.can\_start():

logger.warning('Tried to start non-startable scan')

**return**

self.status = Scan.STARTING

self.save()

start\_result = self.scanner.start\_scan(self.Example\_Tool\_scan\_identifier)

**if** start\_result:

self.start\_date = timezone.now()

self.status = Scan.IN\_PROGRESS

**else**:

self.status = Scan.ERRORS

self.save()

#The poll function is the only function that the web application will periodically call. It only does so when the scan's status is IN\_PROGRESS.

**def** **poll**(self):

poll = self.scanner.poll\_scan(self.scan\_id)

**if** poll:

logger.info('Nessus poll finished')

self.status = Scan.FINISHED

self.end\_date = timezone.now()

self.save()

**elif** poll **is** None:

logger.warning('Nessus poll ended with error')

self.status = Scan.ERRORS

self.save()

**else**:

logger.info('Nessus poll still in progress')

**def** **retrieve**(self):

**if** **not** self.can\_retrieve():

logger.warning('Tried to retrieve non-retrievable scan')

**return**

self.status = Scan.RETRIEVING

self.save()

file = self.scanner.retrieve\_scan(self.scan\_id)

**if** file **is** None:

logger.warning('File retrieval failed')

self.status = Scan.ERRORS

**else**:

self.save\_result(file)

logger.info('Retrieval successful')

self.status = Scan.RETRIEVED

self.save()

"""

These classes are used to define what the forms for creating a scan/scanner will look like.

For the Scanner creation form, you will want to at least include "scanner\_name" and "notes", or else the user will not be able to set them. For any other fields that you would like the user to be able to set, also include them in the list.

For the Scan creation form, you will want to at least include "scan\_name" and "notes".

"""

**class** **Example\_Tool\_Scanner\_Create\_Form**(forms.ModelForm):

**class** **Meta**:

model = Example\_Tool\_Scanner #Define what model this form class belongs to.

fields = ['scanner\_name', 'api\_url', 'api\_key', 'notes'] #Define the fields you would like to include in a form for the Scanner.

**class** **Example\_Tool\_Scan\_Create\_Form**(forms.ModelForm):

**class** **Meta**:

model = Example\_Tool\_Scan #Define what model this form class belongs to.

fields = ['scan\_name', 'endpoints', 'auto\_create','auto\_start','auto\_retrieve','notes'] #Define the fields you would like to include in a form for the Scan.