Nexpose Ticketing System Integration Guide for ServiceDesk

Contents

Last Revised: November 19, 2015

RAPID7

Solution Summary

The goal of incident management is to restore normal service operation as quickly as possible following an incident, while minimizing impact to business operations and ensuring quality is maintained. The integration with Nexpose allows customers to create incident tickets based on vulnerabilities found across their systems. With this information in the ServiceDesk platform, said tickets can be assigned to work teams, prioritized and resolved.

The ServiceDesk Ticket Service integration creates reports based upon a scan of user selected sites or tags, depending on the configuration, and then creates tickets either for each machine and / or vulnerability, depending on the ticketing mode selected. On subsequent scans, the existing tickets are updated (and potentially closed) and new tickets are created, based on the delta from the previous scan.

This service can operate in Default, Vulnerability and IP mode. Tickets are not updated in Default mode.

The helper prepares tickets for sending to ServiceDesk, formatting them depending on the selected ticketing mode. It is then responsible for sending new tickets, updating existing tickets and closing the correct tickets from the ServiceDesk service instance.

Installation and Configuration

Please see the Nexpose Ticketing Configuration Guide for how to install and configure the nexpose_ticketing Gem.

Partner Product Configuration

Once all dependencies have been installed, the configuration files need to be edited with the details of the target Nexpose and ServiceDesk clients. To insert the details, open the configuration files under the config folder found in the Gem installation:

- Windows: C:\Ruby<version>\lib\ruby\gems\<version>\gems\nexpose_ticketing\lib\nexpose_ticketing\config
- Linux: /var/lib/gems/<version>/gems/nexpose_ticketing/lib/nexpose_ticketing/config

Your installation folder may differ; please refer to the Ruby documentation for the specific location.

The sites or tags which are to be scanned, as well as the ticketing mode, are defined in ticket_service.config. An example setup of this file can be found in the main Nexpose Ticketing Configuration Guide.

The ServiceDesk ticketing helper requires a local database to store key information about new tickets. This is created automatically during the first run of the Integration. Please set this value in the configuration file to a chosen location.

To use the ServiceDesk integration an API key must be generated. To generate an authentication key for ServiceDesk Plus, you should login as a user, with administrator privileges. Follow the steps mentioned below to generate authentication key for a new user.

1. Go to Admin tab in ServiceDesk Plus
2. Under Users, select Technician
3. Click Add New to create a new technician and fill in the required details.
    a. Enter the name
    b. Enter the required description and specify the privileges required for the technician such as contact information, cost details, department details. Assign the group to technician and select the correct permissions.
    c. Check the Enable login for this technician checkbox. Enter the following details for the technician:
        i. Login Name
        ii. Password
        iii. Password confirmation
        iv. Domain
    d. Select 'Enable Administrator Privileges (SDAdmin)' option
    e. Under API Key Details, click Generate/Regenerate to generate the authentication Key
4. You can also generate authentication key for an existing user by editing the Technician details, and click Generate under API Key details.
5. If you want to generate authentication key for logged in user, select Personalize then 'Generate API Key'.

The log-in details are specified within the servicedesk.config file. Open the file using any text editor and note the following options:

Setting	Description	Sample Value
helper_name	This is the helper class name. This should not be changed	ServiceDeskHelper
rest_uri	This is the location of the REST endpoint on the ServiceDesk instance. Replace with the ServiceDesk base URI.	https://uritoservicedesk:8080/sdpapi/request
api_key	This is the technician API key generated by the user. Used to access the ServiceDesk instance and to send data.	
ticket_db_path	This is the location of the local ticket database path. Replace var with the PATH to the Nexpose ticketing gem.	/var/lib/gems/1.9.1/gems/nexpose_ticketing-0.8.3/lib/nexpose_ticketing/log/servicedesk_ticketdb
requester	The ticket requester.	nexpose
group	The ServiceDesk incident group.	Network

Initial Run

Assuming you've properly configured the Nexpose and ServiceDesk parameters, execute the following command within the 'bin' folder to run the service:

- ruby nexpose_servicedesk

Every time this command is executed, the service will query Nexpose and obtain any new vulnerability information and open tickets accordingly.

To view log information, wait for the service to complete execution and then review the logs located in the **/lib/nexpose_ticketing/log** directory**.**

As part of a continuous ticketing program, it is recommended to run the command daily via a Cron job or Windows task.

Note

If tickets appear to not be updating on a scan > initial scan, please check the ticket history - updates may just appear here, or not, but email integrations-support@rapid7.comeither way.

Troubleshooting

The most common errors when running the script are:

- Configuration errors (such as incorrect indentation or user details).
- Specifying an incorrect or invalid API key
- Specifying a Nexpose user without permission to create reports.
- Not specifying a site or tag.
- Specifying both a tag and a site. The tag will take priority and the site will not be scanned.

If not enabled, enable logging and view the log files after re-running the service. These files will contain any error information and will also contain information statements about what vulnerability data was found in Nexpose and transmitted to ServiceDesk.

If deleting the last_scan_data.csv file to begin a clean initial run for the service, ensure the local database file is also deleted - otherwise it will continue to read tickets from the database rather than start from an initial run. This may cause duplicate rows to be created in the database, or errors to occur when trying to add data.

If everything still fails, please send an email to integrations-support@rapid7.com with the ticket_helper.log and ticket_service.log attached and a description of the issue.

Helper Method Overview

There are several methods implemented in the serviceDesk_Helper common to all helpers, for creating, updating and closing tickets, as well as several ServiceDesk specific methods. Below is an outline of the methods, along with an explanation of when they are called and how they work:

open_database

- Description: This method opens a connection to the local database file.
- Used By: This method is used by add_ticket_to_database, find_ticket_in_database and remove_tickets_from_database methods to open the connection.

add_ticket_to_database

- Description: This method is used to save the ServiceDesk workorderID to the local database. This value is then retrieved when an update or closure of a ticket is to be sent to the ServiceDesk instance. The NXID of the ticket is saved as the key, with the workorderID as the value.
- Used By: This method is called by the submit_ticket method after successfully sending a new ticket to ServiceDesk.

find_ticket_in_database

- Description: This method is used to search for an existing ticket in the local database using the unique NXID provided to the method. This method will use the NXID as the key and return the workOrderID for the ticket in the ServiceDesk instance.
- Used By: This method is called by the prepare_tickets and prepare_close_tickets methods to search for the workOrderID of an existing ticket.

remove_tickets_from_database

- Description: This method is used to remove a workOrderID / NXID pair from the local database. A list of tickets is provided to the method and the entry for each is removed from the database. This process of removing closed tickets allows new tickets which may have the same NXID to be sent in the future e.g. in IP mode, if a ticket for a machine is closed, and this machine later acquires new vulnerabilities, then a new ticket can be sent; or in Vulnerability mode, there are no machines left with a specific vulnerability, so the ticket can be closed - but a new machine is added in a subsequent scan with this vulnerability.
- Used By: This method is called by the close_tickets method after a successful ticket closure has been sent to the ServiceDesk instance.

prepare_create_tickets

- Description: This method is called to choose the correct 'matching fields' for the current ticketing mode when creating new tickets before calling the prepare_tickets method. This value is used to group related vulnerability information together when creating tickets. The matching fields value is chosen based upon the current ticketing mode and how the information is to be grouped per ticket: Individual vulnerability to IP for Default mode; by Individual IP for IP mode; and by Vulnerability for Vulnerability mode.
- Used By: This method is called from the all_site_report, full_site_report and delta_site_new_scan methods in ticket_service.rb to prepare and format new tickets for sending to the ServiceDesk instance.

prepare_tickets

- Description: This method is called to prepare a list of vulnerabilities, converting them into the correct ticket format for sending to the ServiceDesk instance. Using the matching fields variable, this method groups related rows from the CSV file together into a single ticket. For each row corresponding to a new ticket in the CSV file, a new ticket description will be created with the correct values. For every subsequent row that is part of the same ticket, the ticket description will be updated by the common_helper.rb class. When a new matching filed value is encountered, the previous ticket is placed in the array of created tickets, before creating a new ticket for the current row. After all the tickets have been generated, the NXID for the ticket is checked against the local database using the find_ticket_in_database method. If this method returns a workOrderID, then this ticket already exists in ServiceDesk and the method modify_ticket_request is called, passing in the description of the ticket. Else, this is a new ticket and the method create_ticket_request is called instead. This formatted description (in XML), along with the action to take, the NXID and if existing, the workorderID, are put into a new ticket object. This array of tickets is then returned to the parent method.

- Used By: This method is called from the prepare_create_tickets and the prepare_update_tickets methods to prepare a list of vulnerabilities into formatted tickets for sending to ServiceDesk.

create_ticket_request

- Description: This method is used to correctly format an xml request to create a new ticket in ServiceDesk. It takes the ticket title and description as arguments, and places them within an xml message. It also retrieves the requester and group from the servicedesk.config file.

- Used By: This method is called from the prepare_tickets method to create the ticket description for a new ticket.

modify_ticket_request

- Description: This method is used to correctly format an xml request to update an existing ticket in ServiceDesk. It takes the ticket description as an argument and places it within an xml message. It also retrieves the requester and group from the servicedesk.config file.

- Used By: This method is called from the prepare_tickets method to create the ticket description for updating an existing ticket in ServiceDesk.

submit_ticket

- Description: This method is used to submit a new ticket to the ServiceDesk instance. It sends a HTTP post request, passing in the provided ticket description (in XML format) and the API key from the servicedesk.config file. The method either logs the error message returned, or retrieves the new workOrder ID from the response. This workOrderID is stored in the local DB using the add_ticket_to_database method.

- Used By: This method is called from the create_tickets and update_tickets methods to submit new tickets to the ServiceDesk instance.

modify_ticket

- Description: This method is used to update the description of an existing ticket in the ServiceDesk instance. It sends a HTTP post request to the URI of the ticket workOrderID, passing in the new ticket description (in XML format) and the API key from the servicedesk.config file.

- Used By: This method is called from the update_tickets method to update existing tickets in the ServiceDesk instance.

close_ticket

- Description: This method is used to send ticket closure messages to the ServiceDesk instance. It sends a HTTP post request to the URI of the ticket, using the tickets workOrderID and sending a close_request as the operation, along with the API key.
- Used By: This method is called from the close_tickets method to send the ticket closure requests to ServiceDesk.

create_tickets

- Description: This method is used to send new tickets to the ServiceDesk instance. An array of prepared tickets to create is provided to the method. It then calls the submit_ticket method for each ticket in the array.
- Used By: This method is called from the all_site_report, full_site_report and delta_site_new_scan methods in ticket_service.rb to send new tickets to the ServiceDesk service after they have been discovered and prepared.

prepare_update_tickets

- Description: This method is called to choose the correct 'matching fields' for the current ticketing mode when doing an update before calling the prepare_tickets method. This value is used to group related vulnerability information together when creating tickets. The matching fields value is chosen based upon the current ticketing mode and how the information is to be grouped per ticket: by Individual IP for IP mode; and by Vulnerability for Vulnerability mode.
- Used By: This method is called from the delta_site_new_scan method in ticket_service.rb when in IP and vulnerability mode.

update_tickets

- Description: This method is used to send tickets to the ServiceDesk instance on subsequent runs, after the initial tickets have been created in IP and Vulnerability mode. It takes an array of tickets as the parameter and then based upon the action for that ticket, calls the related method: new tickets have an action of create and are sent with the submit_ticket method ; existing tickets have an action of modify and are sent with the modify_ticket method. This method is able to handle both types of tickets.
- Used By: This method is called from the delta_site_new_scan method in ticket_service.rb when in IP and vulnerability mode to update existing tickets in ServiceDesk.

prepare_close_tickets

- Description: This method is used to prepare a list of ticket closures from the CSV of vulnerabilities exported from Nexpose to send to the ServiceDesk instance. For each row of the CSV, the method first generates the NXID for the ticket, before querying the local database file to get the workOrderID for the ticket. If no workOrderID exists, then there is no ticket to close. If there is, then a hash is created to send to ServiceDesk, containing the action for the ticket, to close it, the workOrderID and a generic description, explaining that the ticket service has closed the ticket.
- Used By: This method is called from the delta_site_new_scan method in ticket_service.rb to create ticket closure messages for sending to ServiceDesk. This is used in all three ticketing modes.

close_tickets

- Description: This method is used to send a list of ticket closures to the ServiceDesk instance. Each ticket, already prepared, is sent individually to the close_ticket method. After successfully sending all tickets, this list is passed to the remove_tickets_from_database method to remove the closed tickets from the local database.
- Used By: This method is called from delta_site_new_scan method in ticket_service.rb to send ticket closure messages to ServiceDesk. This is used in all three ticketing modes.