



HP Enterprise Security Products

**ServiceNow Integration**

October 14, 2015

Contents

[General information 3](#_Toc432722487)

[Purpose and Scope 3](#_Toc432722488)

[Summary of Tasks Completed 3](#_Toc432722489)

[Customer Profile 3](#_Toc432722490)

[Content 4](#_Toc432722491)

[Installation of content 6](#_Toc432722493)

[Generating a ServiceNow Ticket 7](#_Toc432722496)

# General information

## Purpose and Scope

The purpose of this document is to provide commentary on the details of the Health Check for [Company], Inc. performed by HP ESP Professional Services. This document provides the tasks completed by HP Enterprise Security Products and recommendations to [Company] as next steps. The recommendations are provided for maximizing the value of HP ArcSight solutions based on our experience in working with similar customers and our understanding of the environment.

# Content

## What’s inside the ServiceNowModule.zip file

1. **ArcSightServiceNowIntegrator.py**
   1. Calls the various scripts for the integration process
2. **arcsightIOInterface.py**
   1. Defines the working directories on the ESM manager
      1. /opt/arcsight/manager/archive/exports
         1. Destination where the exported XML from ESM resides.
      2. /opt/arcsight/manager/archive/imports
         1. Location where the XML files are to be ingested by ESM
      3. /opt/arcsight/manager/archive/template
         1. Short-term location where a replica of the exported XML file resides.
            1. Location for XML/Case updates occur prior to be imported into ESM.
   2. **checkFiles2Process** – looks for any exported XML file in the exports directory
   3. **readIncNum** – Reads XML to see if an SNOW Incident # already exists
   4. **readAttackProtocol** – Defines the 4 templates using regex.
      1. 0:Machine scan
      2. 1:Machine reimage
      3. 2:Unauthorized File Sharing Software Removal [software name]
      4. 3:Unauthorized Software Installed [software name]
   5. **readSoftwareName** – Looks for specific software defined in the above templates (2: and 3:)
      1. Looks for strings within the […….]
   6. **readHostName** – looks for Target Asset entry on the Case
      1. defined within the <vulnerabilitySource> entry
   7. **readASCaseId** – Reads the ArcSight Case ID entry
      1. defined within the <displayId> entry
   8. **readUserId** – Looks for the Target User ID field within the case
      1. defined within the <vulnerabilityData> entry
   9. **updateTemplate** & **updateTemplateNewIncNum** – Moves exported XML into the template directory
      1. Renames the new XML with the paired ServiceNow Incident ID
         1. INC1234567.xml
      2. Validates whether same Incident & case already exists anywhere
   10. **updateTemplateIncStatus** – Updates the template xml file with stage ‘closed’ when a matching closed entry are found from ServiceNow
   11. **updateTemplateClosedNotes** – Appends the Closed/Resolution Notes from ServiceNow when ticket has been closed
   12. **updateESMNewIncNum** – sends the template XML into the imports directory to update any new case that’s just opened
       1. Once a ServiceNow ticket has been created, this function updates the same case with the related ServiceNow ticket # located in the External Ticket ID field.
   13. **updateESM** – Renames the XML in preparation for imports
       1. File must be renamed with the following:
          1. “ExternalEventTrackingData-yyyymmddHHMM.xml”
3. **serviceNowInterface.py**
   1. Defines the following functions
      1. **getTemplateInfo** – Each of the 4 templates defined by [Company] that requires a ServiceNow ticket
         1. Machine Scan
         2. Machine Reinstall
         3. Unauthorized File Sharing Software Removal
         4. Unauthorized Software Installed
      2. **createSNOWIncident** – Generates a ServiceNow incident ticket by making a web service SOAP API call into [Company]prod.service-now.com
         1. Defines required fields in ServiceNow for ticket creation:

Impact = 3

Urgency = 2

Priority = 2

Category = High

Location = ‘blank’

User = UserId

Assignment Group = ‘Field Support Services’ | ‘ENTERPRISE\_SERVICEDESK\_ATS’

SubCategory = ‘DART’

Short Description = Title of Template

Description = Body of Template

Business Unit = ‘Corporate’

* + - 1. Parses out the Incident # from the ServiceNow API Response

1. **wget.sh**
   1. script that calls ServiceNow to export custom queries and writes to CSV output & logs to wget.log
      1. Query details:

Subcategory=DART

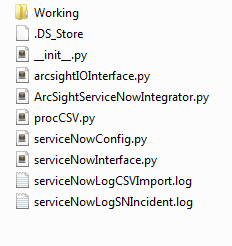
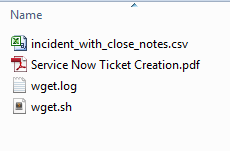
AND(AssignmentGroup = Field Support Services OR ENTERPRISE\_SERVICEDESK\_ATS)

AND Sub Status is NOT NULL

AND Closed Date with the past 1 day.

* 1. **incident\_with\_closed\_notes.csv (headers):**
     1. "number","caller\_id","priority","incident\_state","u\_sub\_status","assignment\_group","assigned\_to","short\_description","sys\_created\_on","u\_resolved","u\_closed\_date","close\_note

1. **procCSV.py**
   1. functions by reading the output file incident\_with\_closed\_notes.csv
   2. Parses the INC#, SubStatus, and ClosedNotes
   3. Appends the XML with these attributes to be updated by ESM
2. **serviceNowConfig.py**
   1. Handles logging of processes
      1. serviceNowLogSNIncident.log
         1. Logs all activities starting with the exporting of the Case.
      2. serviceNowLogCSVImport.log
         1. Logs activities related to when the ‘incident\_with\_closing\_notes.csv’ gets ingested and parsed.

# Installation of content

**Prerequisites**: The following settings and configurations are required for the integration script to work correctly.

* Linux
* Python (v2.6+)
  + SOAPpy Module
* ServiceNow (PROD/DEV/QA) SOAP API account with incident creation permissions

## Extract the ServiceNowModule.zip file to the /opt/arcsight/snow directory on the ESM Manager host.

## Working directories

* 1. /opt/arcsight/snow/serviceNowModule:
     1. \_\_init\_\_.py
     2. ArcSightServiceNowIntegrator.py
     3. arcsightIOInterface.py
     4. serviceNowInterface.py
     5. ProcCSV.py
     6. serviceNowConfig.py

serviceNowLogCSVImport.log

serviceNowLogSNIncident.log

* 1. /opt/arcsight/snow/serviceNowModule/Working
     1. wget.sh

wget.log

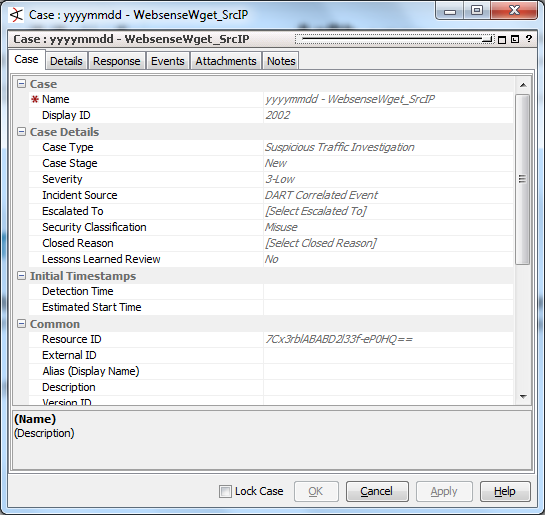
incident\_with\_close\_notes.csv

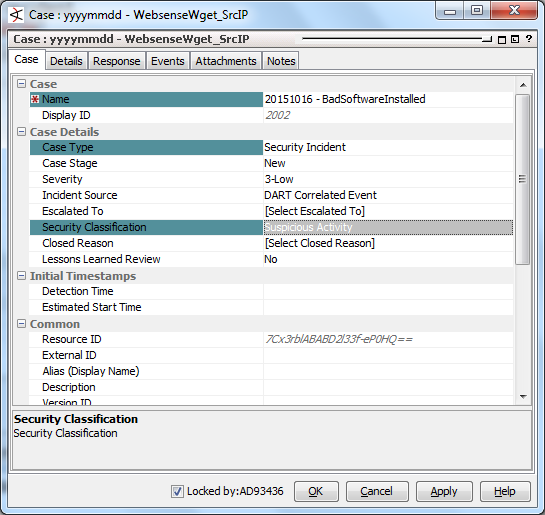
* + 1. Service Now Ticket Creation.pdf
  1. /opt/arcsight/manager/archive/exports (default ESM directory)
  2. /opt/arcsight/manager/archive/imports (default ESM directory)
  3. /opt/arcsight/manager/archive/template (auto-created by script)

1. Set crontab for the following scripts:
   1. ArcSightServiceNowIntegrator.py
      1. Setup to run every 2 - 5 minutes. Run every 2 minutes should be the lowest interval. More frequent runs will assure a quicker ticket creation entry in ServiceNow.
   2. wget.sh
      1. Setup to run once every 30 minutes. This retrieves ServiceNow tickets that have been closed in the past 30 minutes
   3. procCSV.py
      1. Setup to run once every 30 minutes. This should be scheduled to run 10-minute offset **AFTER** the wget.sh task

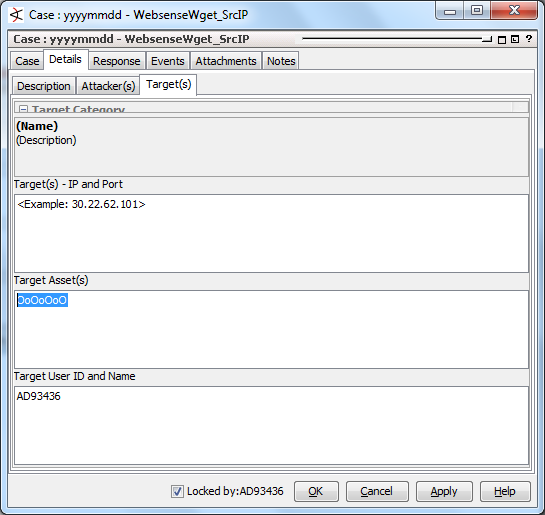
# Generating a ServiceNow Ticket

The procedures below assume that you have already conducted the appropriate SOC Triaging and investigations. Once your investigation calls for a ServiceNow ticket creation based on your Case Creation in ESM, you will need to follow the steps below to assure a successful ticket creation.

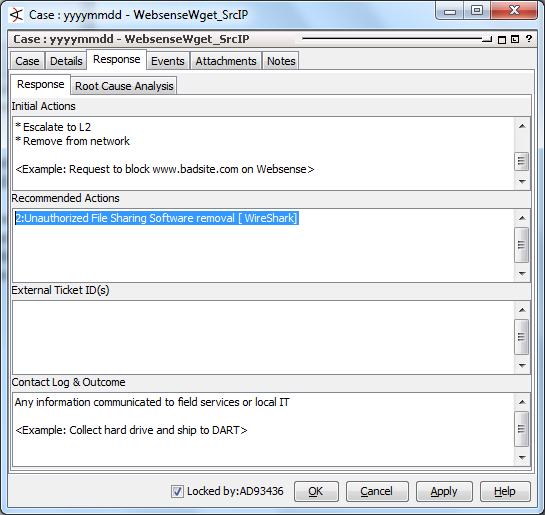
1. Select the Case template that’s most fitting for your incident.
   1. Follow your SOC procedures
   2. 
2. Fill in the name of your Case and the appropriate Case details
   1. A valid Case Name is required
   2. The Case Details entries are irrelevant to the SNOW script.



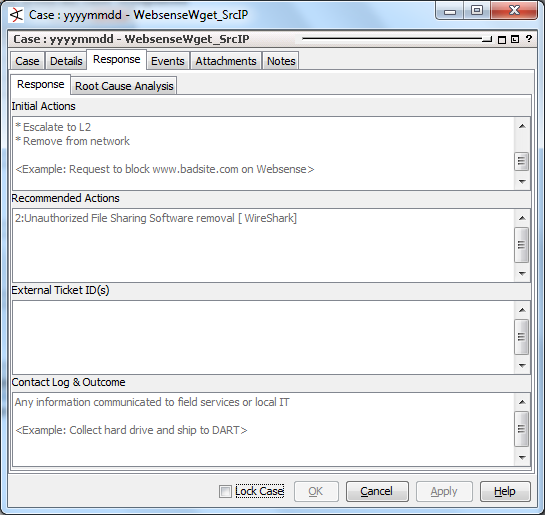
1. **Details Tab** required fields under **TARGETS** tab:
   1. **Target Asset** – This is the Hostname of the target system.
      1. Enter just the hostname (please exclude all angle brackets <OoOoOoO>
   2. **Target User ID & Name** – This field should contain the User’s login ID.
      1. Enter just the USER ID (IN CAPS) and nothing more - AD93436



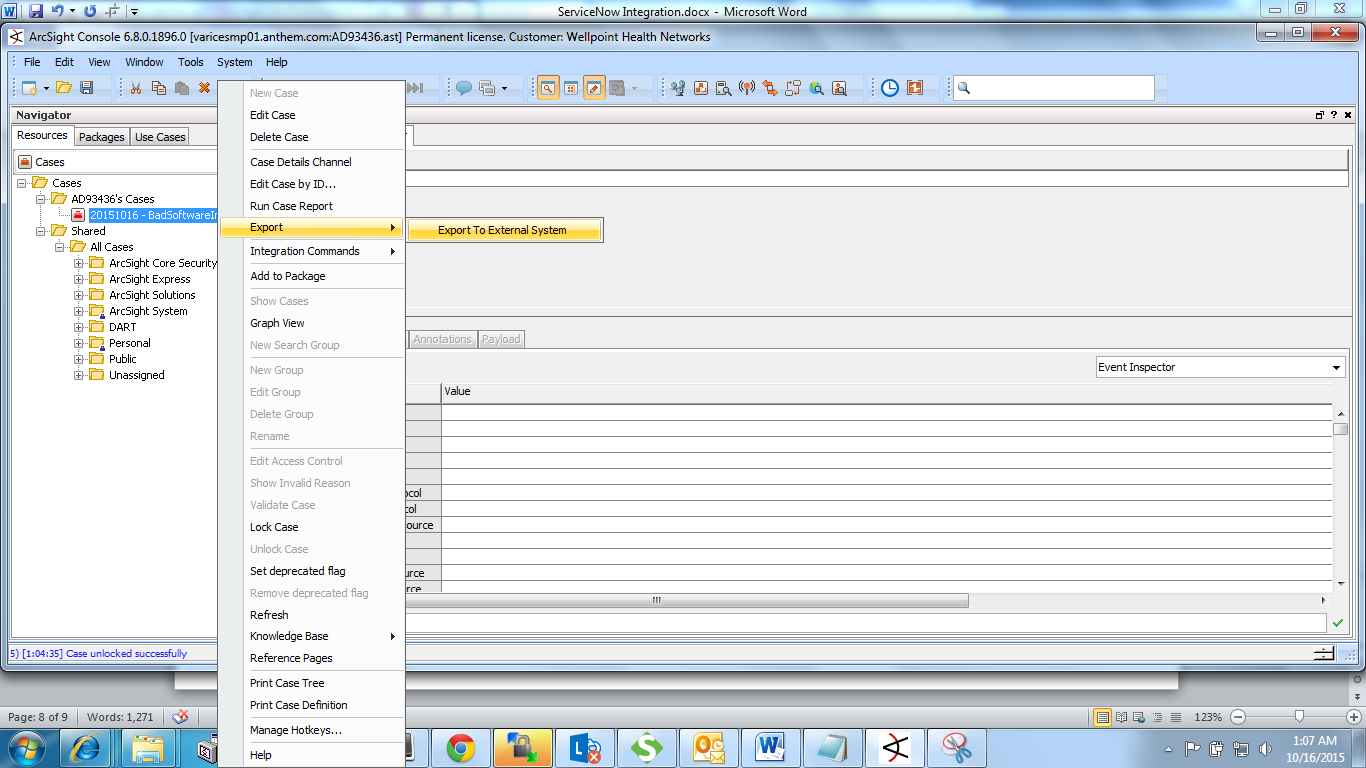
1. **Response Tab** contains one very specific requirement:
   1. **Recommended Actions** section, select one of the following template options below:
      1. 0:Machine scan
      2. 1:Machine reimage
      3. 2:Unauthorized File Sharing Software removal [WireShark]
      4. 3:Unauthorized Software Installed [WireShark]
   2. **External Ticket ID(s)**
      1. Do not enter any INC # in this window. Blank out any entry when creating a case. This field will be populated by the integration process.
      2. If possible, please blank this field from all case templates



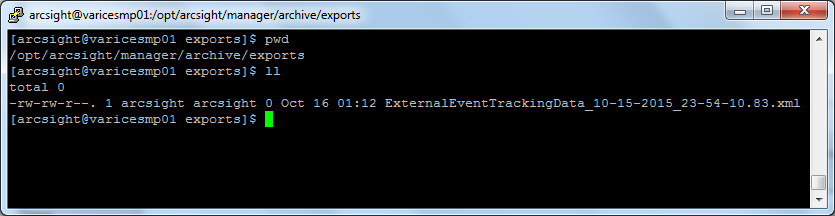
1. Once all appropriate details are entered into the case, **save** and **unlock** the case.
2. If necessary, move the case to the designated folder 🡪 DART Cases



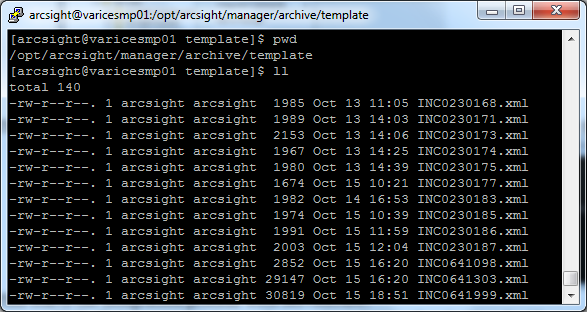
1. You are now ready to Export the case.
2. Right-Click your newly created case 🡪 **Export** 🡪 **Export To External System**



1. This will send your prepackaged case as an XML format. After a few seconds, the file shows up:



1. This starts the integration process with ServiceNow.
   1. XML file’s presence triggers the creation of a new ServiceNow Incident ticket.
   2. ServiceNow returns a response showing the Incident # that got created for the case.
   3. Incident # gets updated in the case you’ve setup.
   4. XML file is duplicated and stored in the template directory



* 1. The file is a renamed file with the actual Incident #.
  2. A query periodically (every 30 minutes) retrieves **Closed** ServiceNow tickets that have **SubCategory=DART** in the tickets.
  3. When a closed incident shows up in the CSV, it will try to match up with one of the XML file.
  4. When it matches, the incident gets updated into the case.
  5. Case is then marked with Case Stage = Closed.
  6. Closed Reason may have 1 of 2 possible options
     1. Legitimately Closed = Resolved
     2. Cancelled by the User = Escalated
  7. At this point, your case should be marked as Closed and can be moved to the appropriate DART Closed Cases directory.