TufinCSV2SC.py

##
June 11, 2020
TufinCSV2SC.py
Version 1.2
Tested on:
SecureTrack/SecureChange version 19.3 HF1 build 160870 (final)
TufinOS Release 2.18 build 1215 (final)
Paul DeHerrera - paul.deherrera@tufin.com
Pre-Sales Security Engineer - Northwest
Tufin
##

Table of Contents

Disclaimer

TufinCSV2SC.py was written as way to demonstrate how to make API calls into SecureChange using python to launch automated Access Request workflows and provision Firewall policy rules. The script is provided free of charge and intended to be used in non-Production environments for educational/testing purposes. The script is not a Tufin supported product, and may not function as intended. Please contact Paul DeHerrera if you need assistance or have ideas for improvements.

Purpose

This script automates API calls into SecureChange to launch an Access Request workflow using a modified version of the “Replacement rules for export” CSV file exported from the Automatic Policy Generator (APG) in SecureTrack. See the Quick Links section of this document for clarification on the objectives.

Requirements

There are a number of requirements for this script to operate properly.

1. You will need access to a Tufin system with SecureTrack and SecureChange. I used version 19.3 HF1 build 160870 (final) running on TufinOS Release 2.18 build 1215 (final). You will need a user account with adequate privileges to launch a SecureChange workflow.
2. You must create an Access Request workflow in SecureChange. I used the Access Request Workflow template provided in SecureChange and modified the Business Justification input field in Step 1 by removing the check mark next to “Mandatory” next to the Field Type. All other steps in the workflow can be customized to meet your need as the script makes an API call into SecureChange to launch the workflow only.
3. The script was written to execute on a lunix system (I used Ubuntu) with Python v3 installed in /usr/bin/python3 and uses the following shebang line at the top of the script: #!/usr/bin/python3. As such you may need to change the permissions of the script with: *chmod u+x TufinCSV2SC.py*
4. You will need a csv input file with the following minimum fields: Name, Source, Destination, Port, and Protocol. The CSV resulting from the APG “Replacement rules for export” works properly as it provides two additional fields (Hits and Permissiveness), which are not used. **Note:** if you use the APG export file you will need to modify the file to include only the rules section like this:

   "Name","Source","Destination","Port","Protocol","Hits","Permissiveness"

   "Rule 7.2","10.100.200.238/32","192.168.1.68/32","53","UDP","30","1"

   "Rule 7.5","10.100.200.238/32","50.63.243.228/32","80","TCP","8","1"

   "Rule 7.4","10.100.200.238/32","2.21.143.185/32","443","TCP","4","1"

   "Rule 7.3","10.100.200.238/32","192.168.1.88/32","53","UDP","1","1"

   "Rule 7.6","10.100.200.238/32","162.159.200.1/32","123","UDP","1","1"

Execution

It is possible to provide the script with all parameters needed to execute or no parameters at all as defined below. The script will prompt for any or all missing command line arguments with one exception. The script will always prompt for the Tufin password.

This example provides all parameters except for the password:

./TufinCSV2SC.py -o 1 -i data.csv -s 10.6.6.71 -u r -w "Pauls Access Request" -b "Go For it!"

This is the help output:

usage: TufinCSV2SC.py [-h] [-a API_CALL] [-d DEBUG] [-o ONE_TICKET] [-i IN_FILE] [-s SERVER_IP] [-u USERID] [-w WORKFLOW] [-b BUSINESS_JUSTIFICATION]

optional arguments:

-h, --help show this help message and exit

-a API_CALL, --api_call API_CALL

-d DEBUG, --debug DEBUG

-o ONE_TICKET, --one_ticket ONE_TICKET

-i IN_FILE, --in_file IN_FILE

-s SERVER_IP, --server_ip SERVER_IP

-u USERID, --userid USERID

-w WORKFLOW, --workflow WORKFLOW

-b BUSINESS_JUSTIFICATION, --business_justification BUSINESS_JUSTIFICATION

Where:

API_CALL: Flag used to issue the API call (1) or not (0)

– valid values are 1 or 0

DEBUG: Flag used to send debug information to log file (1) or not (0)

– valid values are 1 or 0

ONE_TICKET: Flag used to send all csv data to a single SecureChange ticket (1) or not (0)

– valid values are 1 or 0

IN_FILE: csv file (See Requirements section 4 above)

SERVER_IP: IP address of the Tufin server

USER_ID: credentials for the Tufin server (prompted for password)

WORKFLOW: SecureChange workflow name (See Requirements section 2 above)

BUSINESS_JUSTIFICATION: Text field in Step 1 for Business Justification (optional)

Expected Results

The script will initiate one or more SecureChange Access Request tickets from the data in a CSV file. Any errors encountered should result in an error message.

Source Code

```
#!/usr/bin/python3


##
## June 11, 2020
## TufinCSV2SC.py
## Version 1.2
## Tested on:
##   SecureTrack/SecureChange version 19.3 HF1 build 160870 (final)
##   TufinOS Release 2.18 build 1215 (final)
## Paul DeHerrera - paul.deherrera@tufin.com
## Pre-Sales Security Engineer - Northwest
## Tufin
##


import sys, getopt, csv, json, copy, getpass, argparse, urllib3, datetime, ipaddress
from csv import DictReader
from requests import Session
from ipaddress import IPv4Network



# disable InsecureRequestWarning
urllib3.disable_warnings()


def main():
    ## constants
    cdt = datetime.datetime.now()
    curdatetime = cdt.strftime("%Y%m%d_%H:%M:%S")
```

```
# process command line arguments and gather missing data
## api_call
if args.api_call is None:
    make_api_call = 1
else:
    if args.api_call.isdigit():
        make_api_call = int(args.api_call)
        if ((make_api_call > 1) or (make_api_call < 0)):
            sys.exit("invalid value for -a (api_call). Valid values are 1 or 0")
    else:
        sys.exit("invalid value for -a (api_call). Valid values are 1 or 0")
## debug
if args.debug is None:
    debug = 0
else:
    if args.debug.isdigit():
        debug = int(args.debug)
        if ((debug > 1) or (debug < 0)):
            sys.exit("invalid value for -d (debug). Valid values are 1 or 0")
    else:
        sys.exit("invalid value for -d (debug). Valid values are 1 or 0")

if debug == 1:
    f = open(curdatetime + "_TufinCSV2SC.log", "a")

## one_ticket
if args.one_ticket is None:
    one_ticket = 1
else:
    if args.one_ticket.isdigit():
        one_ticket = int(args.one_ticket)
        if ((one_ticket > 1) or (one_ticket < 0)):
            sys.exit("invalid value for -o (one_ticket). Valid values are 1 or 0")
```

```
    else:
        sys.exit("invalid value for -o (one_ticket). Valid values are 1 or 0")

## input file
if args.in_file is None:
    inputfile = input('Input File: ')
else:
    inputfile = args.in_file

## Tufin server IP
if args.server_ip is None:
    ip = input('Tufin IP: ')
    try:
        ipaddress.ip_address(ip)
    except ValueError:
        sys.exit("invalid value for -s (server_ip).  Must be a valid IPv4 address.")
else:
    try:
        ipaddress.ip_address(args.server_ip)
    except ValueError:
        sys.exit("invalid value for -s (server_ip).  Must be a valid IPv4 address.")
    ip = args.server_ip

## Tufin userid
if args.userid is None:
    user = input('Tufin userid: ')
else:
    user = args.userid

## password
password = getpass.getpass('Tufin password: ')

## SecureChange workflow
```

```
if args.workflow is None:
    workflow = input('Workflow: ')
else:
    workflow = args.workflow

## business justification
if args.business_justification is None:
    business_justification = input('Business Justification: ')
else:
    business_justification = args.business_justification

# show input variables when in debug mode
if debug == 1: f.write("____________________\n")
if debug == 1: f.write('one_ticket: ' + str(one_ticket) + "\n")
if debug == 1: f.write('input file: ' + inputfile + "\n")
if debug == 1: f.write("server ip: " + ip + "\n")
if debug == 1: f.write("user: " + user + "\n")
if debug == 1: f.write("password: ********* \n")
if debug == 1: f.write("workflow: " + workflow + "\n")
if debug == 1: f.write("business justification: " + business_justification + "\n")

# open the input file and append records into array
if debug == 1: f.write("Reading csv records into array...\n")
filearray = []
foundblock = 0
count = 0
with open(inputfile, 'r') as read_obj:
    for row in read_obj:
        if row.startswith('"Name"'):
            foundblock = 1
        if foundblock == 1:
            filearray.append(row)
            if debug == 1: f.write("appended record " + str(count) + "\n")
```

```
        count = count + 1


count = count - 1
# check for data
if (count < 1):
    sys.exit("no data records in " + inputfile)

if debug == 1: f.write("there are " + str(count) + " data records in the csv file...\n")

# securechange workflow json - Access Request template with no Business Justification field
# do not modify the content of ticket_json
if debug == 1: f.write("adding json header...\n")
ticket_json = "{"
ticket_json += "  \"ticket\": {"
ticket_json += "    \"subject\": \"No Subject\","
ticket_json += "    \"workflow\": {"
ticket_json += "      \"name\": \"APG Access Request\""
ticket_json += "    },"
ticket_json += "    \"steps\": {"
ticket_json += "      \"step\": ["
ticket_json += "        {"
ticket_json += "          \"name\": \"Enter your request\","
ticket_json += "          \"tasks\": {"
ticket_json += "            \"task\": ["
ticket_json += "              {"
ticket_json += "                \"fields\": {"
ticket_json += "                  \"field\": ["

if ((count > 1) and (one_ticket == 1)):
    if debug == 1: f.write("building json for one ticket...\n")
    ticket_json += "                    {"
    ticket_json += "                      \"name\": \"Required Access\","
    ticket_json += "                      \"@xsi.type\": \"multi_access_request\","
```

```
ticket_json += "                    \"access_request\": ["
for i in range(1,count + 1):
    if debug == 1: f.write("adding AR" + str(i) + "...\n")
    ticket_json += "                      {"
    ticket_json += "                        \"order\": \"AR" + str(i) + "\","
    ticket_json += "                        \"@xsi.type\": \"accessRuleDTO\","
    ticket_json += "                        \"sources\": {"
    ticket_json += "                          \"source\": ["
    ticket_json += "                            {"
    ticket_json += "                              \"@xsi.type\": \"sourceDTO\","
    ticket_json += "                              \"@type\": \"IP\","
    ticket_json += "                              \"ip_address\": \"192.168.0.0\","
    ticket_json += "                              \"netmask\": \"255.255.255.0\""
    ticket_json += "                            }"
    ticket_json += "                          ],"
    ticket_json += "                          \"@xsi.type\": \"multiSourceDTO\""
    ticket_json += "                        },"
    ticket_json += "                        \"destinations\": {"
    ticket_json += "                          \"destination\": ["
    ticket_json += "                            {"
    ticket_json += "                              \"@xsi.type\": \"destinationDTO\","
    ticket_json += "                              \"@type\": \"IP\","
    ticket_json += "                              \"ip_address\": \"192.168.1.0\","
    ticket_json += "                              \"netmask\": \"255.255.255.0\""
    ticket_json += "                            }"
    ticket_json += "                          ],"
    ticket_json += "                          \"@xsi.type\": \"multiDestinationDTO\""
    ticket_json += "                        },"
    ticket_json += "                        \"services\": {"
    ticket_json += "                          \"service\": ["
    ticket_json += "                            {"
    ticket_json += "                              \"@xsi.type\": \"serviceDTO\","
    ticket_json += "                              \"@type\": \"PROTOCOL\","
    ticket_json += "                              \"protocol\": \"TCP\","
```

```
        ticket_json += "                          \"port\": \"25\""
        ticket_json += "                        }"
        ticket_json += "                      ],"
        ticket_json += "                      \"@xsi.type\": \"multi_service\""
        ticket_json += "                    },"
        ticket_json += "                    \"action\": \"Accept\","
        ticket_json += "                    \"labels\": {"
        ticket_json += "                      \"label\": []"
        ticket_json += "                    },"
        ticket_json += "                    \"use_topology\": true,"
        ticket_json += "                    \"users\": {"
        ticket_json += "                      \"user\": []"
        ticket_json += "                    }"
        if i != count:
            ticket_json += "                  },"
        else:
            ticket_json += "                  }"

    ticket_json += "                ]"

else:
    if debug == 1: f.write("building json for one ticket per csv row...\n")
    ticket_json += "                {"
    ticket_json += "                  \"name\": \"Required Access\","
    ticket_json += "                  \"@xsi.type\": \"multi_access_request\","
    ticket_json += "                  \"access_request\": ["
    ticket_json += "                    {"
    ticket_json += "                      \"order\": \"AR1\","
    ticket_json += "                      \"@xsi.type\": \"accessRuleDTO\","
    ticket_json += "                      \"sources\": {"
    ticket_json += "                        \"source\": ["
    ticket_json += "                          {"
    ticket_json += "                            \"@xsi.type\": \"sourceDTO\","
```

```
ticket_json += "                    \"@type\": \"IP\","
ticket_json += "                    \"ip_address\": \"192.168.0.0\","
ticket_json += "                    \"netmask\": \"255.255.255.0\""
ticket_json += "                  }"
ticket_json += "                ],"
ticket_json += "                \"@xsi.type\": \"multiSourceDTO\""
ticket_json += "              },"
ticket_json += "              \"destinations\": {"
ticket_json += "                \"destination\": ["
ticket_json += "                  {"
ticket_json += "                    \"@xsi.type\": \"destinationDTO\","
ticket_json += "                    \"@type\": \"IP\","
ticket_json += "                    \"ip_address\": \"192.168.1.0\","
ticket_json += "                    \"netmask\": \"255.255.255.0\""
ticket_json += "                  }"
ticket_json += "                ],"
ticket_json += "                \"@xsi.type\": \"multiDestinationDTO\""
ticket_json += "              },"
ticket_json += "              \"services\": {"
ticket_json += "                \"service\": ["
ticket_json += "                  {"
ticket_json += "                    \"@xsi.type\": \"serviceDTO\","
ticket_json += "                    \"@type\": \"PROTOCOL\","
ticket_json += "                    \"protocol\": \"TCP\","
ticket_json += "                    \"port\": \"25\""
ticket_json += "                  }"
ticket_json += "                ],"
ticket_json += "                \"@xsi.type\": \"multi_service\""
ticket_json += "              },"
ticket_json += "              \"action\": \"Accept\","
ticket_json += "              \"labels\": {"
ticket_json += "                \"label\": []"
ticket_json += "              },"
ticket_json += "              \"use_topology\": true,"
```

```
    ticket_json += "                    \"users\": {"
    ticket_json += "                      \"user\": []"
    ticket_json += "                    }"
    ticket_json += "                  }"
    ticket_json += "                ]"

# add business justification if needed
if business_justification != '':
    if debug == 1: f.write("adding business justification to json...\n")
    ticket_json += "                },"
    ticket_json += "                {"
    ticket_json += "                  \"name\": \"Business Justification\","
    ticket_json += "                  \"@xsi.type\": \"text_area\","
    ticket_json += "                  \"text\": \"TEST ME\""
    ticket_json += "                }"
else:
    if debug == 1: f.write("no business justification...\n")
    ticket_json += "                }"

ticket_json += "              ]"
ticket_json += "            }"
ticket_json += "          }"
ticket_json += "        ]"
ticket_json += "      }"
ticket_json += "    }"
ticket_json += "    \"priority\": \"Normal\""
ticket_json += "  }"
ticket_json += "}"

# build the ticket objecta
if debug == 1: f.write("building the ticket object...\n")
```

```
if debug == 1: f.write("ticket_json(\n\n" + ticket_json + "\n\n)")

ticket_template = json.loads(ticket_json)
sess = Session()
sess.auth = (user, password)
sess.headers = {"Accept": "application/json"}
sess.verify = False
ticket = copy.deepcopy(ticket_template)

# assign variables
rcnt = 0
reader = csv.DictReader(filearray)
for row in reader:
    if debug == 1: f.write("reading the array...\n")
    name = row['Name']
    source = row['Source']
    src_ip = str(IPv4Network(source).network_address)
    src_nm = str(IPv4Network(source).netmask)
    destination = row['Destination']
    dst_ip = str(IPv4Network(destination).network_address)
    dst_nm = str(IPv4Network(destination).netmask)
    port = row['Port']
    protocol = row['Protocol']

    # show csv values when in debug mode
    if debug == 1: f.write("\n\nname: " + name + "\n")
    if debug == 1: f.write("source: " + source + "\n")
    if debug == 1: f.write("destination: " + destination + "\n")
    if debug == 1: f.write("port: " + port + "\n")
    if debug == 1: f.write("protocol: " + protocol + "\n")

    if debug == 1: f.write("one_ticket: " + str(one_ticket) + " rcnt: " + str(rcnt) + "\n")
    if one_ticket == 1:
```

```
        if debug == 1: f.write("modify json values for one_ticket! rcnt: " + str(rcnt) + "\n")

        # modify the json to use the csv values

        # workflow

        ticket['ticket']['workflow']['name'] = workflow

        # subject

        ticket['ticket']['subject'] = curdatetime +" csv ticket"

        #source ip

        ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][rcnt]['sources']['source'][0]
['ip_address'] = src_ip

        # source netmask

        ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][rcnt]['sources']['source'][0]
['netmask'] = src_nm

        # destination ip

        ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][rcnt]['destinations']
['destination'][0]['ip_address'] = dst_ip

        # destination netmask

        ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][rcnt]['destinations']
['destination'][0]['netmask'] = dst_nm

        # protocol

        ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][rcnt]['services']['service'][0]
['protocol'] = protocol

        # port

        ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][rcnt]['services']['service'][0]
['port'] = port

        # business justification

        if business_justification != '':

            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][1]['text'] = business_justification

        if rcnt == (count - 1):

            # show the ticket json with the csv values when in debug mode

            if debug == 1: f.write("input json for one_ticket " + name + ":\n")

            if debug == 1: f.write("\n")

            if debug == 1: f.write(json.dumps(ticket))

            if debug == 1: f.write("\n")

            if debug == 1: f.write("_____________________\n")

            # make the API call

            if make_api_call == 1:
```

```
                if debug == 1: f.write("sending one_ticket API call...\n")
                response = sess.post(
                    "https://" + ip + "/securechangeworkflow/api/securechange/tickets",
                    data=json.dumps(ticket),
                    headers={"Content-Type": "application/json"}
                )
                if response.ok:
                    pass  # Ticket should have been created
                    if debug == 1: f.write("one_ticket API call sent!\n")
                else:
                    print("Error occurred:", response.text)
                    if debug == 1: f.write("Error occurred:", response.text + "\n")
            rcnt = rcnt + 1
        else:
            if debug == 1: f.write("modify json values for multiple tickets! rcnt: " + str(rcnt) + "\n")
            # modify the json to use the csv values
            # workflow
            ticket['ticket']['workflow']['name'] = workflow
            # subject
            ticket['ticket']['subject'] = name
            #source ip
            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][0]['sources']['source'][0]
['ip_address'] = src_ip
            # source netmask
            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][0]['sources']['source'][0]
['netmask'] = src_nm
            # destination ip
            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][0]['destinations']['destination']
[0]['ip_address'] = dst_ip
            # destination netmask
            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][0]['destinations']['destination']
[0]['netmask'] = dst_nm
            # protocol
            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][0]['services']['service'][0]
['protocol'] = protocol
```

```
            # port
            ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][0]['access_request'][0]['services']['service'][0]['port']
= port
            # business justification
            if business_justification != '':
                ticket['ticket']['steps']['step'][0]['tasks']['task'][0]['fields']['field'][1]['text'] = business_justification


            # show the ticket json with the csv values when in debug mode
            if debug == 1: f.write("input json for " + name + ":\n")
            if debug == 1: f.write("\n")
            if debug == 1: f.write(json.dumps(ticket))
            if debug == 1: f.write("\n")
            if debug == 1: f.write("____________________\n")
            # make the API call
            if make_api_call == 1:
                if debug == 1: f.write("sending API calls...\n")
                response = sess.post(
                    "https://" + ip + "/securechangeworkflow/api/securechange/tickets",
                    data=json.dumps(ticket),
                    headers={"Content-Type": "application/json"}
                )
                if response.ok:
                    pass  # Ticket should have been created
                    if debug == 1: f.write("API calls sent!\n")
                else:
                    print("Error occurred:", response.text)
                    if debug == 1: f.write("Error occurred:", response.text + "\n")
    read_obj.close()
    if debug == 1: f.close()



if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('-a', '--api_call')
```

```
    parser.add_argument('-d', '--debug')
    parser.add_argument('-o', '--one_ticket')
    parser.add_argument('-i', '--in_file')
    parser.add_argument('-s', '--server_ip')
    parser.add_argument('-u', '--userid')
    parser.add_argument('-w', '--workflow')
    parser.add_argument('-b', '--business_justification')
    args = parser.parse_args()


    main()
```

Quick Links

The following links may help answer some questions. Please feel free to contact me if you need further assistance.

How APG works

https://forum.tufin.com/support/kc/R19-3/#Suite/3343.htm%3FTocPath%3DSecureTrack%2520User%2520Guide%7CAnalyzing%2520Policies%7CAutomatic%2520Policy%2520Generation%7C_____1

Creating an APG Job

https://forum.tufin.com/support/kc/R19-3/#Suite/3344.htm%3FTocPath%3DSecureTrack%2520User%2520Guide%7CAnalyzing%2520Policies%7CAutomatic%2520Policy%2520Generation%7C_____3

Configuring APG Job Results

https://forum.tufin.com/support/kc/R19-3/#Suite/3345.htm%3FTocPath%3DSecureTrack%2520User%2520Guide%7CAnalyzing%2520Policies%7CAutomatic%2520Policy%2520Generation%7C_____4

Viewing and Exporting APG Job Results

https://forum.tufin.com/support/kc/R19-3/#Suite/3355.htm%3FTocPath%3DSecureTrack%2520User%2520Guide%7CAnalyzing%2520Policies%7CAutomatic%2520Policy%2520Generation%7C_____5

SecureTrack REST API Documentation

https://forum.tufin.com/support/kc/R19-3/securetrack/apidoc/#!/

SecureChange/SecureApp REST API Documentation

https://forum.tufin.com/support/kc/R19-3/securechangeworkflow/apidoc/#!/

Creating and Managing Workflows

https://forum.tufin.com/support/kc/R19-3/#Suite/9928.htm%3FTocPath%3DSecureChange%2520User%2520Guide%7CCreating%2520Workflows%7CCreating%2520and%2520Managing%2520Workflows%7C_____0

Configuring Workflow Steps

https://forum.tufin.com/support/kc/R19-3/#Suite/1806.htm%3FTocPath%3DSecureChange%2520User%2520Guide%7CCreating%2520Workflows%7CConfiguring%2520Workflow%2520Steps%7C_____0

Managing Access Requests

https://forum.tufin.com/support/kc/R19-3/#Suite/2299.htm%3FTocPath%3DSecureChange%2520User%2520Guide%7CHandling%2520and%2520Managing%2520Tasks%7CHandling%2520a%2520Task%7CManaging%2520Access%2520Requests%7C_____0