Craig ELlrod

senior cloud security architect

Armor API - Python

Linux

Table of Contents

[Introduction 1](#_Toc79666289)

[Prerequisites 1](#_Toc79666290)

[Reference 1](#_Toc79666291)

[Github 2](#_Toc79666292)

[Dependencies 2](#_Toc79666293)

[How it works 2](#_Toc79666294)

[Architecture Diagram 3](#_Toc79666295)

[Step-by-Step 3](#_Toc79666296)

[Step 1 – Download sample code 3](#_Toc79666297)

[Step 2 – Edit the code 3](#_Toc79666298)

[Step 3 – Run the API script 5](#_Toc79666299)

[Step 4 – Sample output 5](#_Toc79666300)

# Introduction

This ARMOR Defense field guide will present a step-by-step method to use the Armor API in a Python script.

## Prerequisites

Python version 3 (Python3)

<https://www.python.org/>

## Reference

Armor API – V2

<https://developer.armor.com/>

<https://docs.armor.com/display/KBSS/Armor+API+Guide>

Armor API Pre-Shared Key – PSK

<https://docs.armor.com/display/KBSS/Pre-Shared+Key+Authentication+Method>

Armor API – Python3

<https://docs.armor.com/display/KBSS/Access+the+Armor+API+System+via+an+API+token+-+Python>

## Github

Github with example scripts

<https://github.com/surfd4wg/armor_script_PYTHON>

Scripts:

* armor\_script.py - ubuntu 20.04, simply calls the API and outputs the result unformatted
* armor\_script\_json.py - ubuntu 20.04, calls the API and outputs pretty json
* armor\_script\_json\_args.py - ubuntu 20.04, calls the API using api key id and secret key as command line arguments

## Dependencies

The following packages must be available on the host system

Python3

## How it works

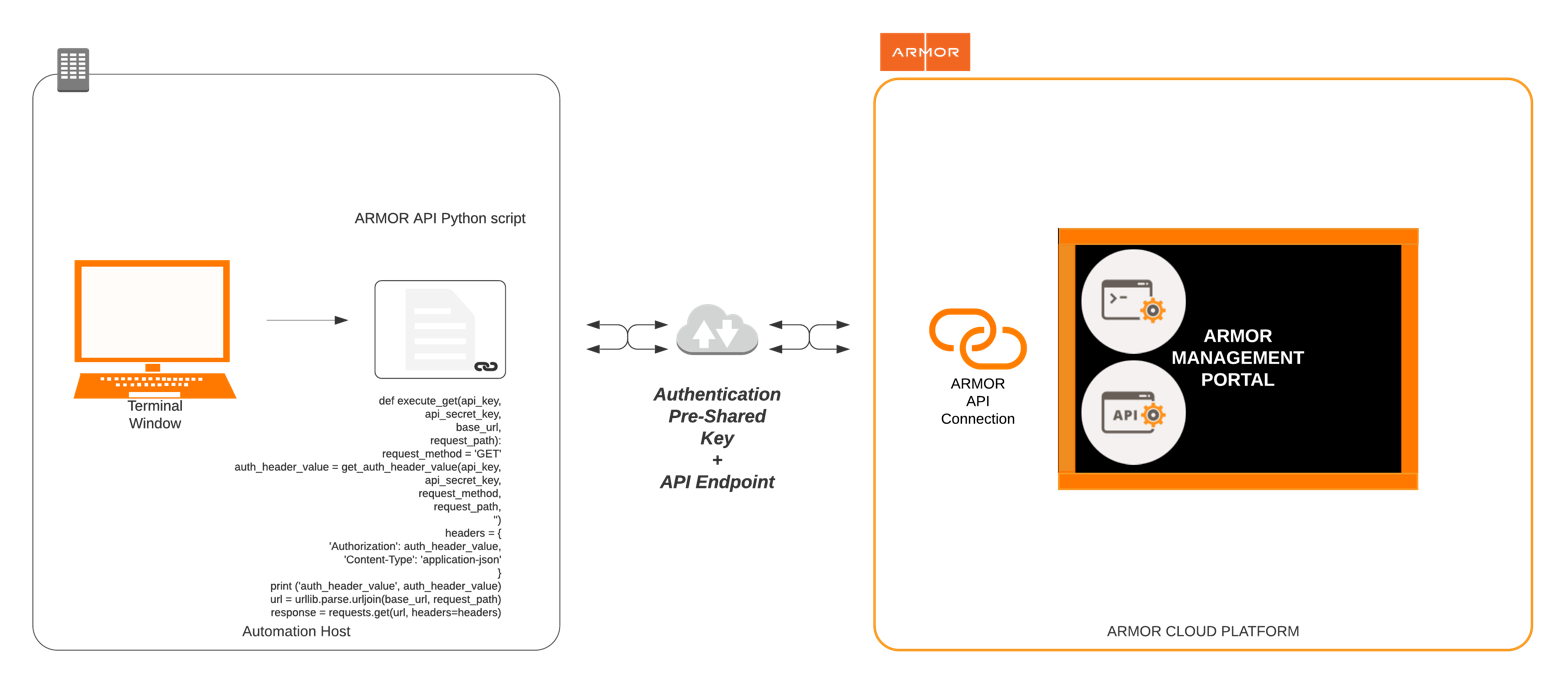
There are three python3 scripts in the Github link. The first script calls the armor API and returns the response unformatted. The second script returns the response and formats they output in pretty json format. The third script takes in the API KEY ID and SECRET KEY as arguments and prints the output in pretty json format. All three scripts do the same thing apart from printing out pretty json and accepting the KEY ID and SECRET KEY as arguments (scripts two and three).

Each script uses the API KEY ID and SECRET KEY obtained through the Armor Management Portal – <https://amp.armor.com>. Each script builds an authentication header, the format can be found at the Armor API Python reference documentation above. Once the authHeader is built, the script uses a python get request to call the HTTP method to the API Endpoint.

The scripts do the following:

* Accepts the API Key Id as input argument 1
* Accepts the Secret Key as input argument 2
* Builds a request string
* Performs a sha512 signature on the request string – used in the authHeader
* Assembles the authHeader
* Calls a get function to access the API endpoint
* Prints the result in pretty json format

## Architecture Diagram



# Step-by-Step

## Step 1 – Download sample code

Download the sample code from github:

<https://github.com/surfd4wg/armor_script_PYTHON>

## Step 2 – Edit the code

Open a terminal to use vi, vim or nano. Alternatively, you can use a code editor such as Visual Studio Code or Atom. Modify the API endpoint to the desired endpoint request. Save the file.

﻿#!/usr/bin/python3

# Authors:

# Craig Ellrod, Joshua Seither

#

# Grab your API Key Id and Secret Key from the Armor Management Portal

# and use them as arguments when calling the python script.

# This script simply calls the armor API and outputs the unformatted response to a .txt file, also

# it outputs the pretty json to a .json file.

# To call a different API endpoint, replace the request\_path='API-ENDPOINT' statement.

# Usage: (Python3)

# ./armor\_script\_json\_args.py API-KEY-ID SECRET-KEY

#

import base64

import hashlib

import hmac

import requests

import sys

import time

import urllib.parse

import uuid

import json

ENCODING = 'utf-8'

def get\_auth\_header\_value(api\_key,

api\_key\_secret,

request\_method,

request\_path,

serialized\_request\_body):

timestamp = int(time.time())

nonce = uuid.uuid4()

# TODO: check if method is not GET and hash/base64 serializedRequestBody

# for GET empty string is enough

request\_body = '' if serialized\_request\_body is None else serialized\_request\_body

request\_data = f'{api\_key}{request\_method}{request\_path}{nonce}{timestamp}{request\_body}'

hash = hmac.new(bytes(api\_key\_secret, ENCODING), bytes(request\_data, ENCODING), hashlib.sha512)

signature = base64.standard\_b64encode(hash.digest()).decode(ENCODING)

print ('signature', signature)

return f'ARMOR-PSK {api\_key}:{signature}:{nonce}:{timestamp}'

def execute\_get(api\_key,

api\_secret\_key,

base\_url,

request\_path):

request\_method = 'GET'

auth\_header\_value = get\_auth\_header\_value(api\_key,

api\_secret\_key,

request\_method,

request\_path,

'')

headers = {

'Authorization': auth\_header\_value,

'Content-Type': 'application-json'

}

print ('auth\_header\_value', auth\_header\_value)

url = urllib.parse.urljoin(base\_url, request\_path)

response = requests.get(url, headers=headers)

#print(response.text)

#print(response.json)

# Build output filename, to later use as input into 'jq'

string=request\_path

endPointOutputFilename=string.replace("/", "\_")+".txt"

endPointOutputFilenameJson=string.replace("/", "\_")+".json"

print ('Output Filename: ', endPointOutputFilename)

print ('Output Filename Pretty Json: ', endPointOutputFilenameJson)

# Write the output to .txt file

open(endPointOutputFilename, 'wb').write(response.content)

# Format the output into pretty json, output to pretty .json file

with open(endPointOutputFilename, 'r') as json\_file:

json\_object = json.load(json\_file)

json\_output=(json.dumps(json\_object, indent=2, separators=(',', ': '), sort\_keys=True))

#print('json\_output: ', json\_output)

open(endPointOutputFilenameJson, 'wb').write(bytes(json\_output, encoding='utf-8'))

if \_\_name\_\_ == '\_\_main\_\_':

api\_key = sys.argv[1]

api\_secret\_key = sys.argv[2]

# Make sure the url does not end with a slash

base\_url = 'https://api.armor.com'

# Make sure the path begins with a slash

**request\_path = '/tickets/list'**

execute\_get(api\_key, api\_secret\_key, base\_url, request\_path)

## Step 3 – Run the API script

On some systems you may have to change the permissions on the script file to allow it to be executed. “chmod +x <script filename>. Replace the API key and Secret key with the ones issued to you through the Armor Management Portal.

$ chmod +x armor\_script\_json\_args.py

$

$ ./armor\_script\_json\_args.py XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX=

## Step 4 – Sample output

The raw output from the Armor API is unformatted JSON. You can send the output through a pretty print parser, such as ‘jq’, and that is built into the example script.

﻿{

"items": [

{

"currentStatus": "Completed",

"issueKeyId": "ATS-235947",

"requestType": {

"description": "",

"id": 65,

"name": "Potential Security Incident"

},

"summary": "Armor - External - AV - Malware Detected and Remediated (Core ID): Sales Demo\_Anywhere\_SE (5595) - 08/12/2021"

},

{

"currentStatus": "Completed",

"issueKeyId": "ATS-235942",

"requestType": {

"description": "",

"id": 65,

"name": "Potential Security Incident"

},

"summary": "Armor - External - AV - Malware Detected and Remediated (Core ID): Sales Demo\_Anywhere\_SE (5595) - 08/12/2021"

},

{

"currentStatus": "Completed",

"issueKeyId": "ATS-235840",

"requestType": {

"description": "",

"id": 65,

"name": "Potential Security Incident"

},

"summary": "Armor - External - AV - Malware Detected and Remediated (Core ID): Sales Demo\_Anywhere\_SE (5595) - 08/12/2021"

},

{

…

…

…