

➤ **Vendor: Cisco**

➤ **Exam Code: 300-735**

➤ **Exam Name: Automating and Programming Cisco Security Solutions**

➤ **New Updated Questions from [Braindump2go](#) (Updated in [December/2020](#))**

[Visit Braindump2go and Download Full Version 300-735 Exam Dumps](#)

QUESTION 17

The Cisco Security Management Appliance API is used to make a GET call using the URI /sma/api/v2.0/reporting/mail_incoming_traffic_summary/detected_amp?startDate=2016-09-10T19:00:00.000Z&endDate=2018-09-24T23:00:00.000Z&device_type=esa&device_name=esa01. What does this GET call return?

- A. values of all counters of a counter group, with the device group name and device type for web
- B. value of a specific counter from a counter group, with the device name and type for email
- C. value of a specific counter from a counter group, with the device name and type for web
- D. values of all counters of a counter group, with the device group name and device type for email

Answer: D

QUESTION 18

Which two APIs are available from Cisco ThreatGRID? (Choose two.)

- A. Access
- B. User Scope
- C. Data
- D. Domains
- E. Curated Feeds

Answer: CE

QUESTION 19

Which two commands create a new local source code branch? (Choose two.)

- A. git checkout -b new_branch
- B. git branch -b new_branch
- C. git checkout -f new_branch
- D. git branch new_branch
- E. git branch -m new_branch

Answer: AD

QUESTION 20

Which URI string is used to create a policy that takes precedence over other applicable policies that are configured on Cisco Stealthwatch?

- A. /tenants/{tenantId}/policy/system/host-policy
- B. /tenants/{tenantId}/policy/system/role-policy
- C. /tenants/{tenantId}/policy/system
- D. /tenants/{tenantId}/policy/system/{policyId}

Answer: A

QUESTION 21

With Cisco FirePOWER Threat Defense software, which interface mode do you configure to passively receive traffic that passes the appliance?

- A. transparent
- B. routed
- C. passive
- D. inline set
- E. inline tap

Answer: C

QUESTION 22

Drag and Drop Question

Drag and drop the code to complete the script to search Cisco ThreatGRID and return all public submission records associated with cisco.com. Not all options are used.

```
import requests

API_KEY = 'asdf1234asdf1234asdf1234'

QUERY = ' ',

URL = 'https://panacea.threatgrid.com/api/v2/ ' / ' ',

PARAMS={"q":QUERY,"api_key":API_KEY}

request = requests.get(url=URL, params=PARAMS)

print(request.json)
```

submissions	public	query
cisco	search	cisco.com

Answer:

```
import requests

API_KEY = 'asdf1234asdf1234asdf1234'

QUERY = ' cisco.com ',

URL = 'https://panacea.threatgrid.com/api/v2/ search / submissions ',

PARAMS={"q":QUERY,"api_key":API_KEY}

request = requests.get(url=URL, params=PARAMS)

print(request.json)
```

public	query
cisco	

QUESTION 23

Drag and Drop Question

Drag and drop the code to complete the API call to query all Cisco Stealthwatch Cloud observations. Not all options are used.

https://example.observbl.com/api/v3/

/

observations	DELETE	GET
POST	all/	all
obsrv	?query=all	

Answer:

GET
https://example.observbl.com/api/v3/

observations / all

DELETE
POST
all/
obsrv
?query=all

QUESTION 24

Drag and Drop Question

Refer to the exhibit. Drag and drop the elements from the left onto the script on the right that queries Cisco ThreatGRID for indications of compromise.

```
# Threat Grid URL used for collecting samples
tg_url = '_____/_____'

# Parameters for Threat Grid API query
tg_parameters = {'api_key': [_____] ,
                'advanced':'true',
                'state':'succ',
                'q':'_____'}

# Query Threat Grid for samples
request = _____ (tg_url, params=tg_parameters)
```

YOUR_API_CLIENT_ID	hostname
requests.get	uri API request
api/v2/search/submissions	API key
https://panacea.threatgrid.com	query parameters
analysis.threat_score:>=95	requests command

Answer:

```
https://panacea.threatgrid.com
api/v2/search/submissions
YOUR_API_CLIENT_ID
analysis.threat_score:>=95
requests.get
```

QUESTION 25

Drag and Drop Question

Drag and drop the code to complete the curl query to the Umbrella Reporting API that provides a detailed report of blocked security activity events from the organization with an organizationId of "12345678" for the last 24 hours. Not all options are used.

```
curl --include --header "Authorization: Basic %base64string%"
https://reports.api.umbrella.com/v1/ /
/
```

12345678	security-activity
security-activity-events	organizations
organizationId	security-events

Answer:

```
curl --include --header "Authorization: Basic %base64string%"
https://reports.api.umbrella.com/v1/ organizations /
organizationId / security-activity
```

12345678
security-activity-events
security-events

QUESTION 26

Drag and Drop Question

Drag and drop the code to complete the curl command to query the Cisco Umbrella Investigate API for the umbrella popularity list. Not all options are used.

```
curl -H "Authorization: %YourToken%"
"https://investigate.api.umbrella.com/"
```

tophundred	Basic	topmillion
Bearer	topthousand	

Answer:

```
curl -H "Authorization: Bearer %YourToken%"
"https://investigate.api.umbrella.com/topmillion"
```

tophundred

Basic

topthousand

QUESTION 27

Drag and Drop Question

Drag and drop the items to complete the ThreatGRID API call to return a curated feed of sinkholed-ip-dns in stix format. Not all options are used.

https://panacea.threatgrid.com/api/v3/

/ ?api_key=[API_KEY]

PUT

sinkholed-ip-dns

feeds

search

sinkholed-ip-dns.stix

GET

Answer:

GET

https://panacea.threatgrid.com/api/v3/

feeds

/

sinkholed-ip-dns.stix

 ?api_key=[API_KEY]

PUT

sinkholed-ip-dns

search

QUESTION 28

Drag and Drop Question

Refer to the exhibit. A Python function named "query" has been developed, and will be used to query the service "com.cisco.ise.session" via Cisco pxGrid 2.0 APIs.

```
def query(config, secret, url, payload):
    print('query url=' + url)
    print(' request=' + payload)
    handler = urllib.request.HTTPSHandler(context=config.get_ssl_context())
    opener = urllib.request.build_opener(handler)
    rest_request = urllib.request.Request(url=url, data=str.encode(payload))
    rest_request.add_header('Content-Type', 'application/json')
    rest_request.add_header('Accept', 'application/json')
    b64 = base64.b64encode((config.get_node_name() + ':' + secret).encode()).decode()
    rest_request.add_header('Authorization', 'Basic ' + b64)
    rest_response = opener.open(rest_request)
    print(' response status=' + str(rest_response.getcode()))
    print(' response content=' + rest_response.read().decode())
```

Drag and drop the code to construct a Python call to the "query" function to identify the user groups that are associated with the user "fred". Not all options are used.

query(, ,
 ,)

"getUserGroupByUserName", "fred"

url

'{ "userName": "fred" }'

secret

Answer:

query(

"getUserGroupByUserName", "fred"

 ,

secret

 ,

url

 ,

'{ "userName": "fred" }'

)