DUMPSBOSS.

Automating and Programming Cisco Security Solutions (300-735 SAUTO)

Cisco 300-735

Version Demo

Total Demo Questions: 10

Total Premium Questions: 108

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Software Development and Design	7
Topic 2, Understanding and Using APIs	88
Topic 3, Cisco Security Platforms	11
Topic 4, Infrastructure and Automation	2
Total	108

QUESTION NO: 1 - (SIMULATION)

Fill in the blank to complete the statement with the correct technology.

Cisco_____Investigate provides access to data that pertains to DNS security events and correlations collected by the Cisco security team.

ANSWER: See the explanation for the answer

Explanation:

The correct technology is **Umbrella**.

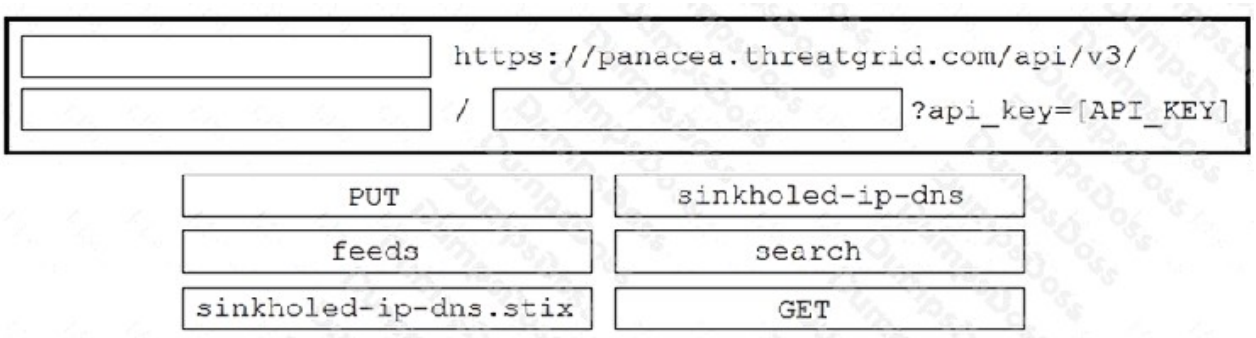
Completed statement:

`Cisco Umbrella Investigate` provides access to data that pertains to DNS security events and correlations collected by the Cisco security team.

Umbrella Investigate is Cisco's threat-intelligence and investigation capability for researching domains, IP addresses, URLs, and related DNS/security-event correlations.

QUESTION NO: 2 - (DRAG DROP)

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.



ANSWER:

Explanation:

The completed request is `GET https://panacea.threatgrid.com/api/v3/feeds/sinkholed-ip-dns.stix?api_key=[API_KEY]`. This correctly uses `GET` because the task is to retrieve a curated intelligence feed rather than create, update, or modify a Threat Grid resource. The `/api/v3/feeds/` portion addresses the Threat Grid API feed collection, which is where curated threat-intelligence feeds are exposed. The requested feed identifier is `sinkholed-ip-dns`, matching the feed named in the task.

Appending `.stix` to that feed identifier requests the result in STIX format. STIX is a standardized threat-intelligence representation that allows indicators and related security intelligence to be consumed by other security tools and automation workflows. The API key is correctly supplied as the `api_key` query parameter, allowing Threat Grid to authenticate and authorize access to the feed. Together, these elements form a read-only, authenticated API request for the specific sinkholed IP and DNS curated feed in the required machine-readable format. The Cisco Live material covering Threat Grid API usage and automation is available in the Cisco Live DEVNET-2164 presentation.

QUESTION NO: 3

```
import requests

URL = 'https://sma.cisco.com:6080/sma/api/v2.0/quarantine/messages/details?quarantineType=spam&device_type=esa'
HEADERS = {'Authorization': 'Basic Y2hlcGFLYWJSQSZe'}

response = requests.get(URL, headers=HEADERS)
```

Refer to the exhibit. A security engineer attempts to query the Cisco Security Management appliance to retrieve details of a specific message.

What must be added to the script to achieve the desired result?

A. Add message ID information to the URL string as a URI.

B. Run the script and parse through the returned data to find the desired message.

C. Add message ID information to the URL string as a parameter.

D. Add message ID information to the headers.

ANSWER: C

Explanation:

Add message ID information to the URL string as a parameter. The Cisco Security Management Appliance message-tracking REST resource accepts filtering criteria as query parameters appended to the resource URL. Supplying the identifier in the request URL lets the appliance apply the message-specific filter while processing the GET request, so the response contains the details for the requested message rather than an unfiltered message-tracking result set. In an HTTP URL, query parameters follow the resource path after a question mark and are sent as part of the request target; the script should therefore construct the message-tracking endpoint with the applicable message-ID query field and value before calling the API. This approach also makes the request deterministic and suitable for automation because the identifier is explicitly represented in the request being made. Cisco publishes API documentation and software resources for the Security Management Appliance through its product support portal, and its Secure Email documentation describes the appliance's centralized message-tracking capabilities. See the Cisco Security Management Appliance support page and the Cisco Secure Email documentation landing page.

QUESTION NO: 4

Which two authentication requirements apply when a Cisco pxGrid client connects to the pxGrid controller? (Choose two.)

A. The client presents a certificate and private key.

B. The client trusts the certificate authority for the pxGrid controller certificate.

C. The client authenticates with an FMC access token.

D. The client supplies an HTTP Basic authentication header for every topic subscription.

E. The client disables certificate validation after registration.

ANSWER: A B

Explanation:

Cisco pxGrid uses mutually authenticated TLS to establish trust between a client and the pxGrid controller. The client must present a valid certificate and private key, and it must trust the certificate authority that issued the controller certificate. This allows the client to validate the controller identity while the controller validates the client identity.

A client certificate alone does not grant unrestricted access to pxGrid services. The pxGrid administrator must approve the client and authorize access to the requested services. Cisco documents certificate-based client registration and mutual TLS in the Cisco pxGrid Developer Guide.

QUESTION NO: 5

```
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())
```

Refer to the exhibit. A Python function named "query" has been developed and the goal is to use it to query the service "com.cisco.ise.session" via Cisco pxGrid 2.0 APIs.

How is the function called, if the goal is to identify the sessions that are associated with the IP address 10.0.0.50?

A. query(config, secret, "getSessionByIpAddress/10.0.0.50", "ipAddress")

B. query(config, "10.0.0.50", url, payload)

C. query(config, secret, url, "10.0.0.50")

D. query(config, secret, url, '{"ipAddress": "10.0.0.50"}')

ANSWER: D

Explanation:

`query(config, secret, url, '{"ipAddress": "10.0.0.50"}')` is the correct invocation because the displayed function accepts the pxGrid configuration, client secret, target service URL, and request payload in that order. The Session Directory service exposed through `com.cisco.ise.session` uses an IP-address lookup request whose input is a JSON object containing the `ipAddress` property. Supplying `{"ipAddress": "10.0.0.50"}` as the payload gives the service the required structured request body while preserving the separately obtained URL for the service operation. The function can then serialize or submit that payload using the authenticated pxGrid connection details represented by `config` and `secret`.

pxGrid is Cisco ISE's framework for securely sharing contextual session information with external applications. A Session Directory query can return session context associated with an endpoint address, allowing an integration to use the returned identity, authorization, and session data in an automated workflow. Cisco's pxGrid documentation describes the service-discovery and authenticated API model used before making service requests, while the ISE API reference documents the available pxGrid services and request formats. See Cisco pxGrid Developer Documentation and Cisco ISE Developer Documentation.

QUESTION NO: 6

For which two programming languages does Cisco offer an SDK for Cisco pxGrid 1.0? (Choose two.)

A. Python

B. Perl

C. Java

D. C

E. JavaScript

ANSWER: C D

Explanation:

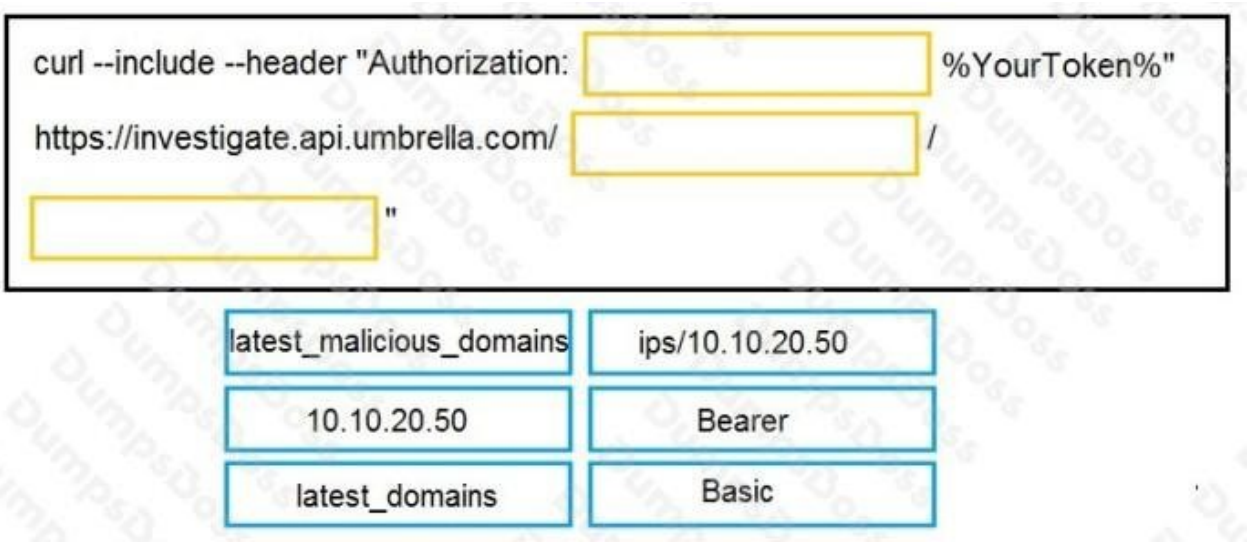
Cisco pxGrid 1.0 provides SDK support for Java and C. The Java SDK enables Java applications to establish the mutually authenticated pxGrid connection, subscribe to published topics, receive context-sharing updates, and publish data through the pxGrid framework. The C SDK provides corresponding capabilities for applications implemented in C, including integrations that may require a native, lower-level client library. These SDKs were supplied to help third-party security products and custom integrations participate in pxGrid's identity and contextual-data exchange without implementing the complete protocol stack independently. Cisco's pxGrid documentation describes the platform's SDK-based integration model and identifies the Java and C SDKs available for pxGrid 1.0. This question specifically concerns the original pxGrid 1.0 SDK offerings, rather than languages that might be usable through later APIs, community libraries, or other Cisco automation interfaces. See Cisco's pxGrid developer documentation and the Cisco ISE pxGrid API reference for pxGrid integration and SDK context.

QUESTION NO: 7 - (DRAG DROP)

DRAG DROP

Drag and drop the code to complete the curl query to the Cisco Umbrella Investigate API for the Latest Malicious Domains for the IP address 10.10.20.50. Not all options are used.

Select and Place:



ANSWER:

Explanation:

The completed request must use the Cisco Umbrella Investigate API's token-based authorization format together with the IP latest-domains resource path. The authorization header is formed as `Authorization: Bearer %YourToken%`. Here, `Bearer` identifies the supplied value as the API access token that Cisco Umbrella expects when it receives an Investigate request. This header lets the service associate the request with the API credentials and permissions for the Umbrella account.

The resource portion begins with `ips/10.10.20.50`. In the Investigate API, the `ips` collection is used when the subject of the investigation is an IP address, and the supplied address identifies the specific IP whose reputation and related intelligence are being requested. Appending `latest_domains` selects the IP intelligence resource that returns recently observed malicious domains related to that address. Together, these path components produce `/ips/10.10.20.50/latest_domains`.

With the API host, the authorized request is therefore directed to `https://investigate.api.umbrella.com/ips/10.10.20.50/latest_domains`. The surrounding quotation marks preserve the entire URL as one curl argument, while `--include` requests that curl display response headers along with the returned content. This arrangement correctly targets the requested data and authenticates it using the required bearer-token convention. Cisco's Investigate API overview describes API authentication and resource usage, and the Latest Malicious Domains for an IP reference documents this IP-focused lookup.

QUESTION NO: 8

Which API is designed to give technology partners the ability to send security events from their platform/service/appliance within a mutual customer's environment to the Umbrella cloud for enforcement?

A. Cisco Umbrella Management API

B. Cisco Umbrella Security Events API

C. Cisco Umbrella Enforcement API

D. Cisco Umbrella Reporting API

ANSWER: B

Explanation:

Cisco Umbrella Security Events API is designed for technology partners that need to submit security-event information generated by their own platform, service, or appliance to Cisco Umbrella. The integration enables a partner product deployed in a shared customer environment to communicate relevant event context to Umbrella, where Umbrella can apply cloud-delivered security enforcement. This is specifically a partner-integration use case: an external security product detects or identifies an event, sends that event through the Security Events API, and Umbrella uses the submitted information as part of its enforcement workflow.

This API therefore connects third-party security telemetry and detections with Umbrella's enforcement capabilities, helping organizations extend protection across products they already operate. It is distinct from general administrative, reporting, or policy-management functions because its purpose is the ingestion of partner-originated security events for enforcement. Cisco's Cloud Security developer documentation identifies the Security Events API as an Umbrella API offering for this type of security-event integration. See the Cisco Umbrella Security Events API documentation and the Cisco Cloud Security developer documentation for API resources and integration guidance.

QUESTION NO: 9

When authenticating to the Cisco Firepower Management Center REST API, which HTTP response headers contain the access token and refresh token that the API client uses for subsequent requests?

A. X-auth-access-token and X-auth-refresh-token

B. X-api-key and X-api-secret

C. Authorization and WWW-Authenticate

D. X-domain-uuid and X-policy-uuid

ANSWER: A

Explanation:

The `X-auth-access-token` and `X-auth-refresh-token` response headers contain the tokens returned after a successful FMC authentication request. The access token is included in the `X-auth-access-token` request header when calling protected FMC API resources. The refresh token can be used to obtain a new access token when needed, without resupplying credentials for every request.

FMC API clients commonly authenticate by sending a POST request to the token-generation endpoint with valid credentials, then preserve the returned token values for the remainder of the automation workflow. The domain UUID returned or retrieved separately is used to scope configuration API requests, but it is not an authentication token. See the Cisco Firepower Management Center API documentation.

QUESTION NO: 10

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: A D

Explanation:

`git checkout -b new_branch` creates a new local branch named `new_branch` and immediately switches the working tree and HEAD to that branch. This is the traditional combined form of creating a branch and checking it out; Git documents `-b <new-branch>` as equivalent to running `git branch <new-branch>` followed by `git checkout <new-branch>`.

`git branch new_branch` also creates a new local branch reference named `new_branch`, using the current HEAD commit as its starting point when no explicit start point is supplied. Unlike the checkout command, it leaves the currently checked-out branch unchanged. Both commands therefore create a branch in the local repository and are valid ways to begin separate work from the current commit. Git's branch documentation describes the command as creating a new branch

head pointing to the current commit by default, while the checkout documentation specifies that `-b` creates the branch before switching to it. See the Git branch documentation and Git checkout documentation.