

Let's talk about QRadar Apps: Development & Troubleshooting

IBM SECURITY SUPPORT OPEN MIC



Slides and additional dial in numbers:

<http://ibm.biz/JoinQRadarOpenMic>

NOTICE: BY PARTICIPATING IN THIS CALL, YOU GIVE YOUR IRREVOCABLE CONSENT TO IBM TO RECORD ANY STATEMENTS THAT YOU MAY MAKE DURING THE CALL, AS WELL AS TO IBM'S USE OF SUCH RECORDING IN ANY AND ALL MEDIA, INCLUDING FOR VIDEO POSTINGS ON YOUTUBE. IF YOU OBJECT, PLEASE DO NOT CONNECT TO THIS CALL.

August 23, 2017

Announcements

- QRadar 7.3.0 Patch 4 is being released today (ISO & SFS)!
- Passport Advantage ISOs are for new installations only.
- User Behavior Analytics 2.1.1 is now on the IBM App Exchange.
- Look on the IBM Security App Exchange for new downloadable apps and content (<https://exchange.xforce.ibmcloud.com/hub>).

The screenshot shows the IBM X-Force Exchange interface. At the top, there is a dark navigation bar with the 'IBM X-Force Exchange' logo on the left, a search bar labeled 'Search by Application' in the center, and a notification bell icon with a red '5' badge on the right. Below the navigation bar, the main content area is divided into a left sidebar and a main featured section. The sidebar contains 'Refine By' filters: 'All Applications' (dropdown), 'Categories' (with a 'Clear' link), and 'All Content Types' (dropdown). The categories listed are: Cloud Services (9), Compliance and Reporting (18), Data (15), and Endpoint (29), with a '+ 16 More' link. The main section is titled 'Featured' and displays four app cards. Each card has a QRadar logo, a status badge (PREMIER, UPDATED, or NEW), a title, a description, and attribution to IBM Security, IBM BigFix, or IBM QRadar, all of which are 'IBM Validated'.

Category	Count
Cloud Services	9
Compliance and Reporting	18
Data	15
Endpoint	29

App Name	Status	Developer	Validation
QRadar QRadar Advisor With Watson	PREMIER	By IBM Security	IBM Validated
QRadar IBM BigFix App for QRadar	UPDATED	By IBM BigFix	IBM Validated
QRadar User Behavior Analytics for QRadar	UPDATED	By IBM QRadar	IBM Validated
QRadar IBM QRadar Content for Sysmon	NEW	By IBM QRadar	IBM Validated



Apps and The App Exchange



What is a QRadar App?

IBM QRadar Security Intelligence

admin Help Messages 15 System Time: 10:39 AM

Dashboard Offenses Log Activity Network Activity Assets Reports Risks Vulnerabilities Admin QDI

IBM QRadar Deployment Intelligence Deployment Dashboard Advanced Health Querying Configure Graphs 30 days Health Summary Reports

Enter Hostname or IP Address or Appliance Type or Status to filter

Overview Performance Deployment Overview - 8 Hosts Next Refresh 00:00:07

24h Deployment Health

Health status count: 1 1 6

12h Notifications

Enter Hostname or notification to Filter

- 10:28 AM on Tue, Jun 27 perf_scale_console-primary: License Expired
- 10:27 AM on Tue, Jun 27 perf_scale_console-primary: Maximum active offense reached
- 09:58 AM on Tue, Jun 27 perf_scale_console-primary: Max events reached
- 09:42 AM on Tue, Jun 27 perf_scale_console-primary: Event pipeline dropped events
- 08:40 PM on Mon, Jun 26 perf_scale_ep2: SAR Sentinel: recovered
- 08:32 PM on Mon, Jun 26 perf_scale_ep2: SAR Sentinel: threshold crossed

12h Status Feed - Last 7 days

Enter Hostname or Status or Date/Time to Filter Status Feed

- perf_scale_console-secondary went Standby at 03:33 PM on Fri, Jun 23.
- perf_scale_console-secondary went Offline at 10:32 AM on Fri, Jun 23.
- perf_scale_console-secondary went Standby at 10:28 AM on Fri, Jun 23.
- perf_scale_console-primary went Active at 10:28 AM on Fri, Jun 23.
- perf_scale_console-secondary went Unknown at 10:27 AM on Fri, Jun 23.
- perf_scale_console-secondary went Standby at 10:20 AM on Fri, Jun 23.
- perf_scale_console-secondary went Set_offline at 10:19 AM on Fri, Jun 23.
- perf_scale_ec1 went Active at 09:28 AM on Fri, Jun 23.

Host Status Overview

12h User Activity

12h Top Users By Search Activity

Enter Username to Filter

Username	Count	Running	Error	Cancelled	Max Duration(s)	Avg Duration(s)
admin	1520	0	0	0	791.764	1.029
QDI	94098	0	0	0	7.426	0.014
Total	95608	0	0	0	791.764	0.030

12h Top Users By API Activity

Enter Username to Filter

Username	Successful	Failed
QDI	385320	13681
configservices	221	0
admin	161	1
Total	385702	13682

12h Event Rate

12h Flow Rate

12h Top 5 Hosts - EPS

12h Top 5 Hosts - FPS

12h Top 5 Hosts - EPS

12h Top 5 Hosts - FPS

12h Stored Events by LogSource

12h Stored Events by LogSource type

12h Unknown Events by LogSource

12h Unknown Events by LogSource type

perf_scale_console-primary - 172.16.107.180 - 3199 - Console - Went Active - 95.2 hrs ago - CPU Util:23% - Memory Util:57% - Disk Space Util:5%

perf_scale_ep1 - 172.16.107.181 - 1699 - Eventprocessor - Went Active - 96.2 hrs ago - CPU Util:30% - Memory Util:53% - Disk Space Util:1%

App Validation Process

App validation consists of:

- Review of app name, version and description.
- Review the screen-shots and icons.
- Ensuring the extension zip is the correct structure.
- Thoroughly checking the manifest for all required fields.
- Checking for collisions between app page_scripts and QRadar functions.
- Verify globalization functionality.
- Run static analysis on any python and js code.
- Ensure that the app installs/uninstalls/reinstalls.
- Tests the app's functionality.
- Verify that the docker container can handle the app being started and stopped.
- Verify that the app logs information appropriately.
- Verify any app storage.
- Review all API calls.
- Ensure that there are no hard-coded values.
- Executing security tests.
- Verify any additional packages.

UI App Capabilities

- Tabs unique to your requirements
 - Add a new tab to the QRadar UI.
 - Fully custom UI using flask with HTML.

IBM QRadar Security Intelligence | admin | Help | Messages 7 | System Time: 9:23 AM

Quick insights | Search for User

Monitored Users: 13.9k | **Current High Risk Users: 251** | **Sense Events (last hour): 1.3m** | **Offenses Generated (last hour): 267**

System Score (Last 24 Hours)

Risk Category Breakdown (Last Hour) > User Geography

Recent Sense Offenses

Offense #	User	Event Count	Flow Count	Magnitude
Offense # 340	User: Robert Thomas	58	0	3
Offense # 339	User: Joseph James	49	0	3
Offense # 338	User: Eric Jones	47	0	3
Offense # 337	User: William Jackson	59	0	3
Offense # 336	User: David Taylor	46	0	3

Most Risky Users (Overall Score)

User	Score
Robert Smith	5469
James Smith	4978
Michael Smith	4820
Robert Brown	4722
John Brown	4636
James Johnson	4612
John Johnson	4492
John Smith	4375
Michael Johnson	4025
Robert Williams	3947

Most Suspicious Users (Window Score)

User	Score
Robert Williams	+335
James Johnson	+310
Robert Jones	+300
John Davis	+275
James Brown	+265
John Jackson	+265
James Jones	+255
Robert Smith	+255
William Smith	+245
William Jones	+245

Watchlist

User	Score
Robert Smith	5.5k
Kenneth Anderson	634.5
Frank Harris	370.5

UI App Capabilities

- Dashboard
 - Create a new dashboard tile.
 - Display interesting visualizations.

The screenshot displays the IBM QRadar Security Intelligence dashboard. At the top, there is a navigation bar with tabs for Dashboard, Offenses, Log Activity, Network Activity, Assets, Reports, Risks, Vulnerabilities, Admin, and Deployment. Below the navigation bar, a dropdown menu shows 'Threat and Security Monitoring' as the selected dashboard. To the right of the dropdown are buttons for 'New Dashboard', 'Rename Dashboard', 'Delete Dashboard', and 'Add Item'.

The main content area is divided into several sections:

- Users with the highest risk score:** A table listing users and their risk scores, with a vertical bar on the left indicating the score level. The data is as follows:

User	Risk Score
Marty.Stouffer3	215
Oscar.Grouch3	212
Levar.Burton3	210
Jim.Lehrer3	206
emservice	106
aaa@aa.bb.cc	96
ginny@ibm.com	86
matt@google.com	81
jimmy	78
svc_corpportal_adaut	44
- My Offenses:** A section for viewing personal offenses, currently empty.
- Most Severe Offenses:** A table listing the most severe offenses, with the following entries:

Offense Name
Sense Offense Inject
Sense Offense Inject
Sense Offense Inject
Sense Offense Inject
Sense Offense Inject
- Most Recent Offenses:** A table listing the most recent offenses, with the following entries:

Offense Name
Sense Offense Inject
Sense Offense Inject
Sense Offense Inject
Sense Offense Inject
Sense Offense Inject
- Top Services Denied through Firewalls:** A section for viewing top services denied through firewalls, currently empty.
- Default-IDS / IPS-All: Top Alarm Signatures:** A section for viewing top alarm signatures, currently empty.

There

UI App Capabilities

- Admin Icon
 - Add an icon to the QRadar admin tab.

The screenshot shows the IBM QRadar Security Intelligence Admin interface. The top navigation bar includes: Dashboard, Offenses, Log Activity, Network Activity, Assets, Reports, Risks, Vulnerabilities, Admin, and Deployment. The Admin section is active, showing a sidebar with categories: System Configuration, Data Sources, Remote Networks and Services Configuration, Plug-ins, Risk Manager, Deployment Intelligence, Reference Data Import - LDAP, and User Analytics. The main content area displays 'Remote Networks and Services Configuration' with icons for Remote Networks and Remote Services, and 'Risk Manager' with icons for Configuration Source Management and Device Import. At the bottom, there is a 'Reference Data Import - LDAP' section with a corresponding icon and a 'User Analytics Settings' section with a bar chart icon.

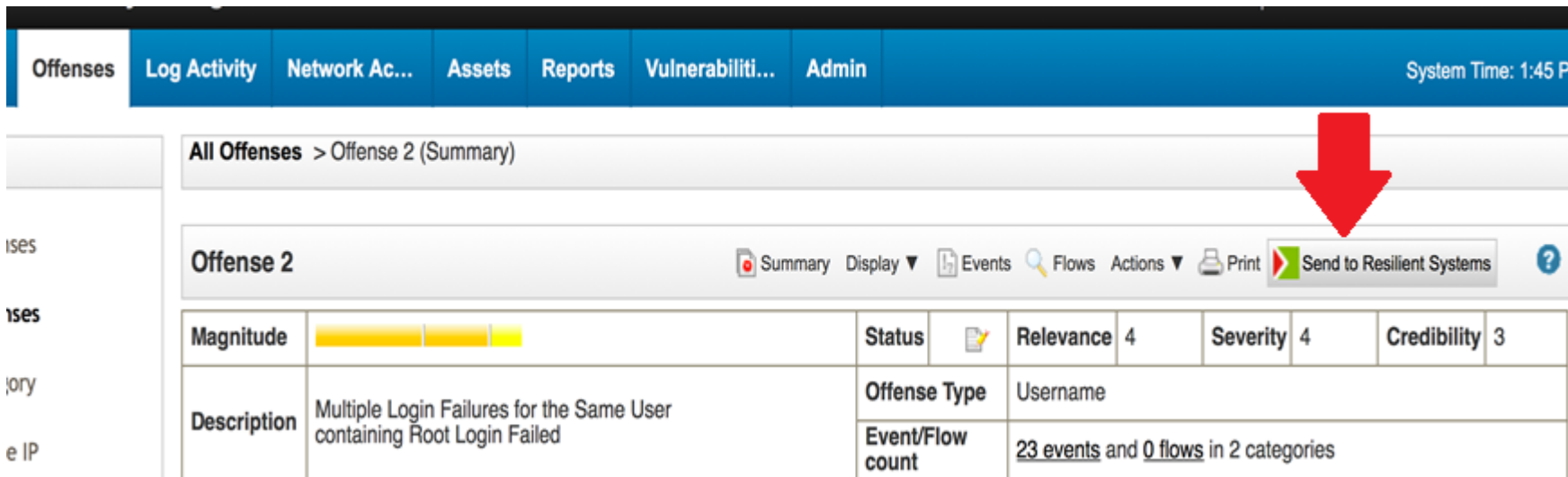
The screenshot shows the 'Reference Data Import (LDAP)' configuration window. The title is 'LDAP Imports'. There are two buttons: '+ Add Import' and 'Configure'. The main content area displays the configuration for an LDAP import with the following details:

Reference Data	IBM_LDAP	Last Poll	Sep 27, 2016, 12:58 PM
Base DN	ou=bluepages.o=ibm.com	Poll Interval	600 minutes
Filter	(workLoc=H6.J)	Paged results	Off
Attribute List	uid, cn, sn, telephoneNumber, primaryUserid, callupname, mail		
Username	anonymous		
Last Updated	Sep 27, 2016, 12:58 PM		



There are also icons for edit and delete, and a 'Poll now' button.

UI App Capabilities

- Toolbar buttons
 - Add a button to any QRadar toolbar.



The screenshot displays the QRadar interface. At the top, a blue navigation bar contains tabs for 'Offenses', 'Log Activity', 'Network Ac...', 'Assets', 'Reports', 'Vulnerabiliti...', and 'Admin'. The system time is shown as 1:45 PM. Below the navigation bar, the breadcrumb 'All Offenses > Offense 2 (Summary)' is visible. The main content area shows 'Offense 2' with a toolbar containing buttons for 'Summary', 'Display', 'Events', 'Flows', 'Actions', 'Print', and 'Send to Resilient Systems'. A large red arrow points to the 'Send to Resilient Systems' button. Below the toolbar is a table with the following data:

Magnitude		Status		Relevance 4	Severity 4	Credibility 3
Description	Multiple Login Failures for the Same User containing Root Login Failed		Offense Type	Username		
		Event/Flow count	23 events and 0 flows in 2 categories			

UI App Capabilities

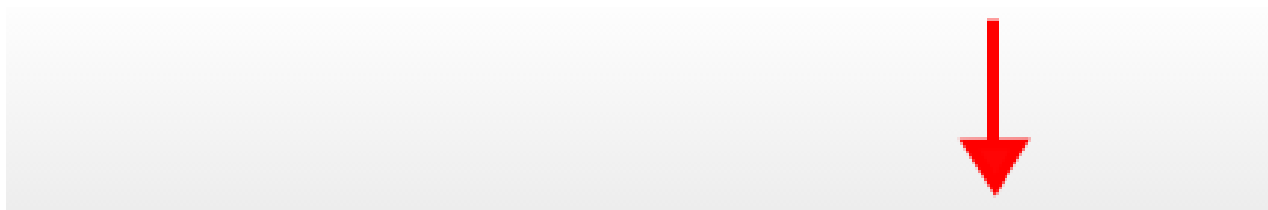
- Right Click Menu
 - Add an option to a right click menu.
- Tooltip
 - Add a hover-over tooltip.

The screenshot shows a table of network logs with columns for ID, timestamp, severity, source IP, and destination IP. Two context menus are overlaid on the table. The first menu, on the left, contains filtering options: 'Filter on Source IP is 172.31.54.212', 'Filter on Source IP is not 172.31.54.212', 'Filter on Source or Destination IP is 172.31.54.212', 'Quick Filter...', and 'More Options...'. The second menu, on the right, contains action options: 'Navigate', 'Information', 'Run Vulnerability Scan', 'Run Forensics Recovery', 'Run Forensics Search', 'Plugin options...', 'Carbon Black Process Search' (highlighted in blue), 'Sync Carbon Black sensor', and 'Quarantine Endpoint from Network'. A red arrow points to the 'Carbon Black Process Search' option.

ID	Timestamp	Severity	Source IP	Destination IP
1	Dec 1, 2015, 3:11:09...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Error	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1
1	Dec 1, 2015, 3:11:07...	Information	172.31.54.212	127.0.0.1

UI App Capabilities

- Custom Columns
 - Add a custom column to the log activity or network activity tables.



	Events	Flows	Star Date	Last Ever	custom col custom javascript
...	36	119	9 ...	12...	id is37,Source IP is10.101.146.25
...	12	48	9 ...	11...	id is62,Source IP is10.101.135.48
...	24	96	9 ...	2...	id is57,Source IP is10.101.145.57
...	24	60	9 ...	6...	id is8,Source IP is10.101.165.174
...	24	57	9 ...	17...	id is42,Source IP is10.101.163.210
...	35	107	9 ...	19...	id is36,Source IP is10.101.138.200
...	23	71	9 ...	28...	id is55,Source IP is10.101.243.200
...	24	120	9 ...	1h...	id is32,Source IP is10.101.174.30
...	24	60	9 ...	2h...	id is17,Source IP is10.101.176.200
...	23	83	9 ...	2h...	id is5,Source IP is10.101.173.34
...	12	12	0	3h	id is25,Source IP is10.103.4.102

UI App Capabilities

- Content Panes
 - Add a HTML frame into either an Offense Summary or an Asset Summary.

The screenshot displays a web application interface with a navigation bar at the top containing tabs for 'Offenses', 'Log Activity', 'Network Activity', 'Assets', 'Reports', 'Risks', and 'Vulnerabilities'. The 'Offenses' tab is active. Below the navigation bar, there is a search and filter area with a search input, 'Save Criteria', 'Actions', and 'Print' buttons. A sidebar on the left lists various filters such as 'Offenses', 'Offenses', 'Category', 'Source IP', 'Destination IP', 'Network', and 'Status'. The main content area shows a table with columns for 'Id', 'Description', and 'Offense Type'. Two red rectangular boxes highlight custom content: 'My Offenses header custom content' at the top and 'My Offenses footer custom content' at the bottom of the main content area.

Content and Content Packs

- Content
 - A UI App can also include QRadar content such as:
 - Rules
 - Custom Properties (event/flow)
 - Log Sources, Log Source Types and Log Source Categories
 - Reports
 - Saved Searches
 - Dashboards
 - Function Groups (i.e. log source groups)
 - Reference Data
 - Historical Correlation Profiles
- Content can also be bundled without a UI app as a Content Extension and shared through the IBM Security App Exchange.



The QRadar SDK



The SDK

- The QRadar SDK contains many functions to help you develop QRadar apps such as:
 - Workspace Creation
 - A single command allows users to create a blank app.
 - Local App Testing
 - Test the basic functions of your app without deploying the app to QRadar.
 - App Deployment
 - Deploy your app to QRadar for testing.
- To install the SDK:
 - We recommend the latest version of Python 2.
 - Download the SDK from the IBM Security App Exchange Developers Forum (<https://developer.ibm.com/qradar>).
 - Unzip the file and run the install script inside (A Windows .bat file and a Unix .sh file in the zip).
 - You should now be able to use the `qradar_app_creator` function.

The `qradar_app_creator` command

All The functions of the SDK are executed by the `qradar_app_creator` command:

- Workspace Creation
 - `qradar_app_creator create -w ~/QRadarApps/com.me.myApp.1.0.0`
- Local App Testing
 - `qradar_app_creator run -w ~/QRadarApps/com.me.myApp.1.0.0`
 - Open your browser to `http://0.0.0.0:5000` to test the UI of your app
- App Deployment
 - `qradar_app_creator package -w ~/QRadarApps/com.me.myapp -p com.me.myapp.zip`
 - `qradar_app_creator deploy -q 10.11.12.13 -u admin -p com.me.myapp.zip`

The App Workspace

- The workspace will contain this file structure:
 - App folder
 - This is the main directory for you app, it contains:
 - qpylib – A python library that your app can use to perform QRadar functions.
 - __init.py__ - The initialization file that starts the flask instance and imports the views and qpylib.
 - static – Static files (.js, .css etc.).
 - templates – Jinja 2 templates.
 - views.py – Flask entry point (routes).
 - store folder
 - This is the Data store for your app (logs, app databases etc.).
 - qradar_appfw_venv folder
 - This folder contains the python virtual environment where the dependencies are installed.
 - manifest.json
 - This file tells QRadar what your app does.
 - run.py
 - This file contains instructions to run the code in the app folder.

▼ demo

▼ app

▶ qpylib

▶ static

▶ templates

__init__.py

views.py

▶ qradar_appfw_venv

▶ store

manifest.json

run.py

App Editor

The screenshot shows the IBM QRadar App Editor interface. The top navigation bar includes "IBM QRadar Security Intelligence" and user options like "admin", "Help", and "Messages". Below this is a secondary navigation bar with tabs for "Dashboard", "Offenses", "Log Activity", "Network Activity", "Assets", "Reports", "Risks", "Vulnerabilities", "Admin", "Hello World", and "Hello World Dev...". The main interface has a dark header with "IBM QRadar App Editor", "File", and "Actions" menus. On the left is a file explorer showing a project named "app" with subfolders like "static" and "templates", and files like "index.html" and "views.py". The main area is a code editor with three tabs: "index.html x", "manifest.json * x", and "__init__.py x". The "manifest.json" file is open, showing a JSON configuration for an application. The code is as follows:

```
1 - {
2   "console_ip": "9.xxx.xxx.xxx",
3   "name": "Hello World",
4   "description": "An Application to display Hello World",
5   "resources": {
6     "memory": 200
7   },
8   "areas": [
9     {
10    "required_capabilities": [
11      "ADMIN"
12    ],
13    "description": "A Hello World example app",
14    "id": "HelloWorld",
15    "text": "Hello World",
16    "url": "index"
17  },
18  {
19    "required_capabilities": [
20      "ADMIN"
21    ],
22    "description": "Application Development",
23    "id": "editorArea_Hello_World",
24    "text": "Hello World Development",
25    "url": "app_editor/editor"
26  }
27 ],
28 "version": "1.0.0",
29 "uuid": "5158c3d1-9035-4d13-a4ac-5e8bdf1221c9",
30 "app_id": 1019
31 }
```



App Development



App Memory Requirements

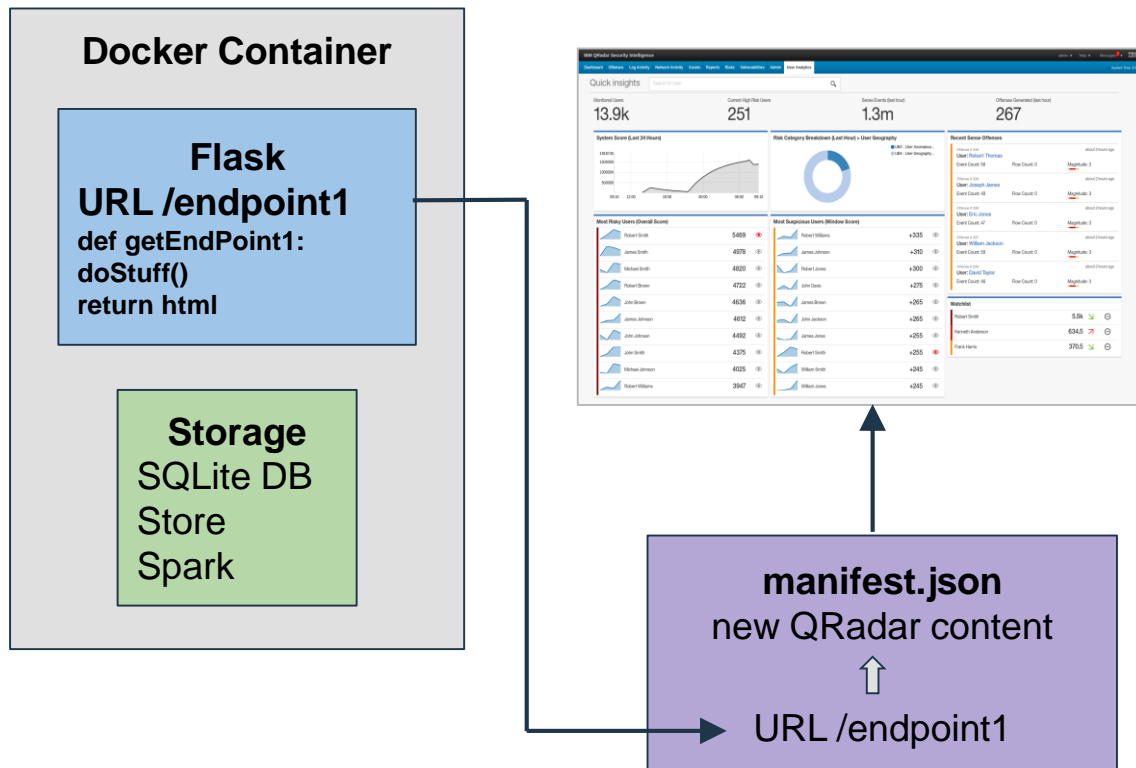
- All QRadar Console appliances have a hard limit of 10% of the overall appliance memory allocated to apps.

Console maximums (physical and VM installations)

- 12.8GB of maximum memory for a 128 GB Console appliance
 - 6.4GB of maximum memory for a 64GB Console appliance
 - 4.8GB of maximum memory for a 48GB Console appliance
- Apps typically take ~200MB; however, some special apps use more resources. For example:
 - QRadar Advisor with Watson takes around ~500MB
 - User Behavior Analytics (UBA) takes ~700MB of memory (without Machine Learning).

High Level App Architecture

- Docker container for each app
 - Storing/processing data
- Flask routes URLs to python methods
 - Return templated html pages
 - Return data (json blobs)
- Manifest maps URLs to QRadar items
 - Area
 - Config page
 - Rightclick
 - Dashboard
- QRadar handles the rest and is the delivery mechanism
 - Proxy (URL's to containers)
 - Provisioning
 - Tabs, dashboard, config page, rightclick etc



Where do App Nodes fit?

- An App Node is a CentOS or RHEL appliances that can host the docker container.
- Applications hosted on an App Node are not restricted to the same memory requirements as the Console appliance.
- App nodes are not QRadar managed hosts and exist outside of the deployment.
- As of QRadar 7.3.0, only one App Node appliance is supported.

Add Node

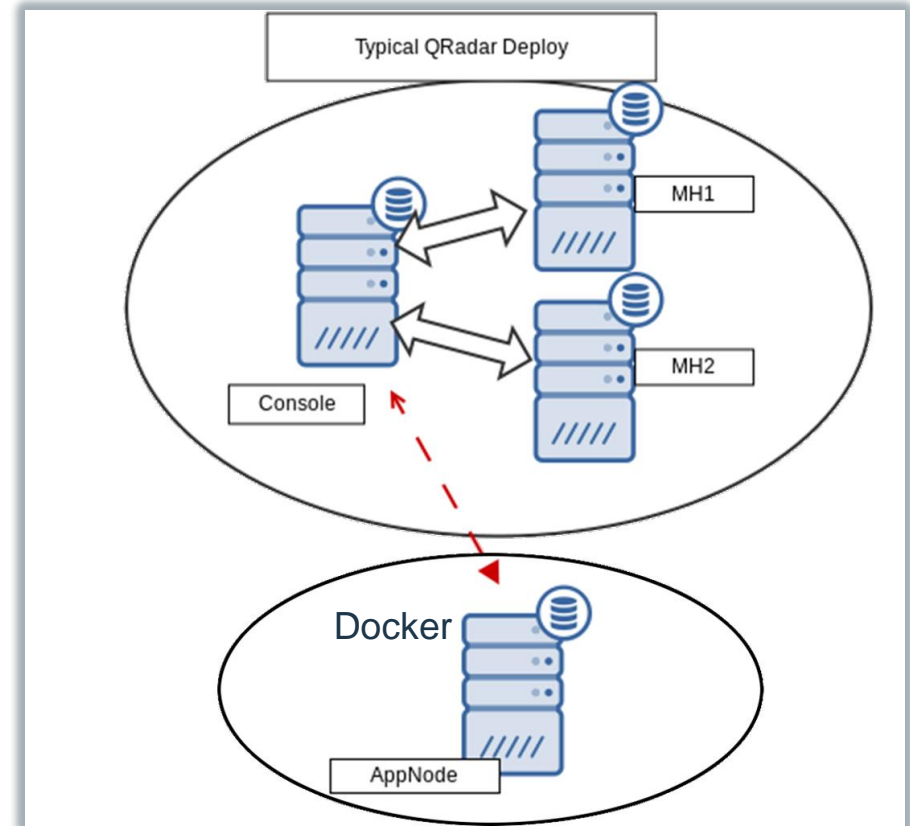
Before you add a Node, please make sure the host has RHEL 7 or Centos 7 installed.
The IP of the host is required as well as user credentials for a user with sudo access.
Hover over label fields for more information.

Node IP:

Node User:

Node Password:

Confirm Password:





- Flask provides easy routing between a URL and your Python code.
- Flask is used primarily through the `route()` decorator which binds a URL to a function:

```
@app.route('/hello')  
def hello():  
    return 'Hello, World'
```

- The `@app.route` defines a URL that is bound to the function definition below it.
- You can also have variable URLs as part of your application:

```
@app.route('/user/<username>')  
def show_user_profile(username):  
    return 'User %s' % username
```

Jinja



- Flask also gives the ability to render Jinja templates.
 - A Jinja template gives you python level control and logic inside HTML.
- To use jinja you need to have a template in the templates folder and call it with flask

```
@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template ('hello.html', name=name)
```

- The above code snippet would try to render the hello.html template which could look like this:

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
    <h1>Hello {{ name }}</h1>
{% else %}
    <h1>Hello, World!</h1>
{% endif %}
```


Qpylib Functions

- Here are the very helpful qpylib functions:
 - **log(message, level='info')** - Creates log messages in the QRadar logs (defaults to info).
 - **set_log_level(log_level='info')** - Overrides the default log level.
 - **get_store_path(relative_path='')** - Returns the path of the store directory.
 - **get_root_path(relative_path='')** - Returns the path of the root directory.
 - **REST(GET, <url>, headers=<headers>, data=<data>, params=<params>, json=<json>, version=<version>)**
 - Function that makes API calls.
 - **get_console_address()** - Returns the IP of the QRadar console.
 - **get_app_base_url()** - Returns the base URL of the app.
 - **get_app_id()** - Returns the numeric id of the app.
 - **get_app_name()** - Returns the name of the app.

App Manifest

- To make the generated manifest readable:
 - mv manifest.json manifest.json.bu
 - python -m json.tool manifest.json.bu > manifest.json

```
{
  "areas": [
    {
      "description": "A Hello World app",
      "id": "QHelloWorld",
      "required_capabilities": [
        "ADMIN"
      ],
      "text": "Hello World",
      "url": "index"
    }
  ],
  "description": "Application to display hello world",
  "name": "Hello World",
  "uuid": "e0372428-6156-4a6a-86bb-6dc0121e306a",
  "version": "1.0"
}
```

- In QRadar make an area
- Call it Hello World
- Link it to '/index' from Flask

Content Export

- Cannot upload app.zip to 'Extension Management' window
- Need content bundle:
`./opt/qradar/bin/contentManagement.pl --action export --content-type 100 --id <id> -t "ZIP"`
- This .zip can be uploaded/installed through the Extension Management page
- Not validated/signed though (warning on install)



QRadar's APIs



Communicating with QRadar through REST APIs



API Documentation

Home

- 6.0
- 5.1
 - /analytics
 - /ariel
 - /asset_model
 - /auth
 - /config
 - /gui_app_framework
 - /help
 - /qvm
 - /reference_data
 - /scanner
 - /siem
 - /local_destination_addre
 - /offense_closing_reason
 - /offenses**
 - /offense id

GET

5.1 - GET - /siem/offenses

Description

Retrieve a list of offenses currently in the syst

Response Description

An array of Offense objects. An Offense objec

- **id** - Number - The ID of the offense.
- **description** - String - The description
- **assigned_to** - String - The user the of
- **categories** - Array of strings - Event ca
- **category_count** - Number - The numl
- **policy_category_count** - Number - Th
- **security_category_count** - Number -

SEC Token and Authorization

- Create a SEC Token from the authorized services UI in the Admin tab.
- Needed for QRadar API calls from Flask (back-end)
- From browser (JS) can use session cookie
- See incident overview app, does all API calls in JS

- Admin/Admin permissions
 - 728+ Ariel is not admin required
 - Everything else is
- Perpetual is best (so app doesn't stop working)

- Hard code into this app for speed (*NOT* the best practice)
- Example code
 - QDI views.py lines ~ 50 - 70
 - Store in config.json

Call QRadar API from Flask

- `import json`
- `from qpylib import qpylib`

```
@app.route('/page4')
def get_page4():
    """
    Returns page filled with values from QRadar API call
    """
    token = 'e4af27ee-cc57-4439-af1b-9770e398e85'
    header = { 'token': token, 'content-type' : 'application/json', 'Version': '7.0' }

    response = qpylib.REST( 'GET', 'api/reference_data/sets', headers=header )
    values = response.json()

    qpylib.log( json.dumps(values, indent=4) )

    return render_template('page4.html', values=values)
```



Advanced Topics



Advanced Topics

- Ingestion
 - Ariel
- Storage
 - Memory
 - Files (config to json)
 - SQLite
- Output
 - LEEF logs
- Performance
 - QRadar API's

Ingestion - Ariel

- Need to use AQL
- Cannot use normal searches (quick filters)
- Searches are dangerous!
 - Limit results, searched records
 - Use paging when pulling back results if necessary

Post search and get search ID

```
token = getToken()
header = { 'SEC' : token, 'content-type' : 'application/json', 'Version': '5.0' }
query = { 'query_expression' : "select * from events limit 10" }

apiResponse = qpylib.REST( 'POST', 'api/ariel/searches', headers=header, params=query)

responseJson = apiResponse.json()
searchid = responseJson['search_id']
```

Example: see QDI app/daemon_methods.py

Ingestion - Ariel

Keep checking search status until it is COMPLETED

```
start_time = time.time()
while True:
    apiResponse = qpylib.REST( 'GET', 'api/ariel/searches/' + searchid, headers=header)
    responseJson = apiResponse.json()

    if responseJson['status'] == "COMPLETED":
        break
    else:
        if time.time() - start_time > TIMEOUT:
            raise Exception('Ariel search did not come back with results within %s seconds' % TIMEOUT )
        time.sleep(2)
```

Get search results

```
apiResponse = qpylib.REST( 'GET', 'api/ariel/searches/' + searchid + '/results', headers=header)
responseJson = apiResponse.json()
events = responseJson['events']
```

Example: see QDI app/daemon_methods.py

Storage

- Memory
 - Limited (200 MB default)
 - Definable in manifest, limited to 10% physical RAM on console
 - App node helps this
- Files (config to store/config.json for example)
 - Use **qpylib.get_store_path('token.json')** and NOT a hard-coded path
 - Makes work both locally and deployed (path problems)
 - See QDI app config.py which reads/writes to file to save SEC token for searching
- SQLite DB
 - schema.sql
 - Init DB before polling
 - DB migration on app update (new columns, tables etc)
 - See UBA app for examples
- Spark, RDD's
 - Barebones container, need to include dependencies (55MB rh6 image, no JRE even)

Output - Create LEEF Logs

```
class EventCreator:
    DSM_PORT = 514
    LEEF_MESSAGE = '{{date}} 127.0.0.1 LEEF:2.0|IBM|Sense|1.0|{{event_name}}|cat={{cat}}\t'
    LEEF_PROPERTIES = {
        "sourceip": "src",
        "destinationip": "dst",
        "sourceport": "srcPort",
        "username": "usrName",
    }

    def __init__(self, logger=None, console_ip=None):
        if logger:
            self.logger = logger
        if console_ip:
            self.DSM_IP = console_ip

    def create_user_score(self, event_data):
        event = self.LEEF_MESSAGE.replace('{{date}}', time.strftime("%b %d %H:%M:%S")) \
            .replace('{{event_name}}', 'User Score') \
            .replace('{{cat}}', 'User Risk')

        event += 'usrName=' + event_data['username'] + '\t'
        # append more properties here

        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.sendto(event, (self.DSM_IP, self.DSM_PORT))
        return
```

Example: see UBA app, event_creator.py

Performance - QRadar API's

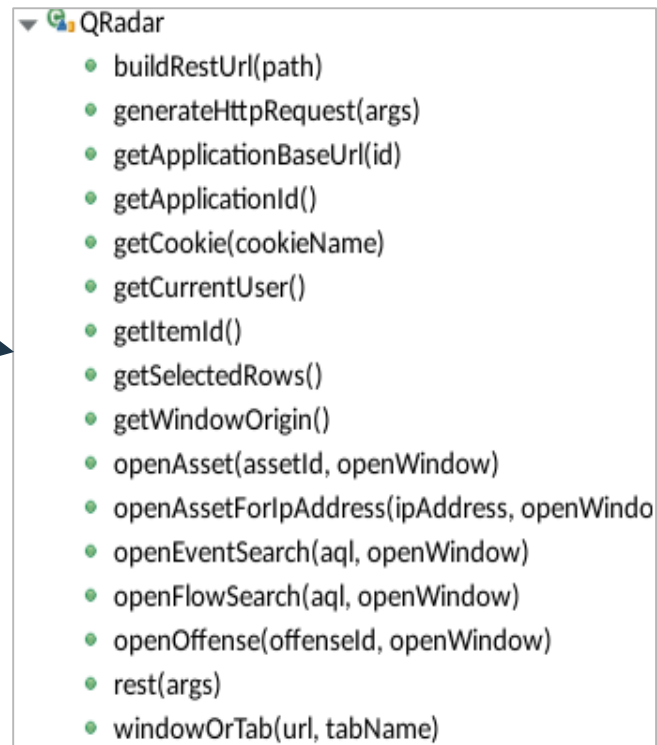
- Ariel
 - Limit results searched
 - Limit records returned
 - Be as specific as possible (logsource, custom property etc)
 - Turn on indexing for custom properties
- Assets
 - Slow right now
 - Expect them to take a long time if you have a full asset model
 - Use filters
- Offenses
 - Returns ALL offenses (closed and open)
 - Kills docker container easily
 - Use filters (filter on 'OPEN' for example)
 - Page results

~~select * from events last 10 hours~~

**select username, starttime
from events
where username is not null
order by starttime desc
limit 100
last 1 hours**

Other Useful Information/Tools

- Can talk to other apps (GET/POST data etc)
 - Call **api/gui_app_framework/applications**
 - Get the app_id (traverse the json, look for app name or similar)
 - URL for app root is:
 - **/console/plugins/<app_id>/app_proxy/**
- New SDK includes QRadar JS lib '**qappfw.js**'
 - Tons of useful functions (open search, asset etc)
- qpylib python module
 - Logging
 - REST methods
 - Token handling



A screenshot of a code editor showing a list of functions from the QRadar JS library. The list is titled 'QRadar' and contains 17 functions, each preceded by a green dot. An arrow from the text 'Tons of useful functions (open search, asset etc)' in the previous block points to this list.

```
QRadar

- buildRestUrl(path)
- generateHttpRequest(args)
- getApplicationBaseUrl(id)
- getApplicationId()
- getCookie(cookieName)
- getCurrentUser()
- getItemId()
- getSelectedRows()
- getWindowOrigin()
- openAsset(assetId, openWindow)
- openAssetForIpAddress(ipAddress, openWindow)
- openEventSearch(aql, openWindow)
- openFlowSearch(aql, openWindow)
- openOffense(offenseId, openWindow)
- rest(args)
- windowOrTab(url, tabName)

```



Troubleshooting & Support



Where do I start with troubleshooting?

Clearing browser cache



A number of issues can be validated by making sure that the browser cache is cleared before you start any investigation or app review.

Use the developer tools to validate API calls (Developers)

All newer browsers support developer tools where you can validate API calls, look at response headers.

The screenshot shows a web browser's developer tools interface. The top part displays a file tree on the left with folders like '/gui_app_framework', '/application_creation_task', and '/applications'. A 'Try It Out!' button is visible in the main area. Below this, the 'Response Code & Request URI' section is shown. The bottom part of the image shows the 'Network' tab with a table of requests. The selected request is a GET for 'applications' with a status of 200 OK. The response headers are expanded, showing 'Access-Control-Allow-Origin: ""' and 'Cache-Control: "no-cache, no-store, must-revalidate"'.

Status	Method	File	Domain	Cause	Type	Transferred	Headers	Cookies	Params	Response	Timings	Security
200	GET	blank.gif	172.16.77.35	img	gif	cached						
200	GET	applications	172.16.77.35	xhr	json	1.29 KB	Request URL: https://172.16.77.35/api/gui_app_framework/applications Request method: GET Remote address: 172.16.77.35:443 Status code: 200 OK Version: HTTP/1.1 Response headers (0.399 KB): Access-Control-Allow-Origin: "" Cache-Control: "no-cache, no-store, must-revalidate"					
200	GET	dojo.css?version=1969177427	172.16.77.35	stylesheet	css	cached						
200	GET	ResizeHandle.css?version=1969177427	172.16.77.35	stylesheet	css	cached						
200	GET	dijit.css?version=1969177427	172.16.77.35	stylesheet	css	cached						
200	GET	BusyButton.css?version=1969177427	172.16.77.35	stylesheet	css	cached						
200	GET	claro.css?version=1969177427	172.16.77.35	stylesheet	css	cached						
200	GET	dijit.css	172.16.77.35	stylesheet	css	cached						

Debugging Deployed Apps

`/opt/qradar/support/qapp_utils.py ls`

```
[root@ouellette106 logs-uba]# /opt/qradar/support/qapp_utils.py ls
Existence of Docker check is complete. Moving on...
PORT    CONTAINER      IMAGE                                     STATUS  appID  NAME
49153   37499fd4bf56   b9b53a43a73b4b4caab5c5e1530c0b41      RUNNING 1003   Deployment Intelligence
49154   1a78fa38ef9f   13b4c3c9464d466291d3774dc85e46fb      RUNNING 1004   Reference Data Import - LDAP
49155   826815381d65   12f882cad1724c44947691bc0fb72d69      RUNNING 1051   User Analytics
```

`/opt/qradar/support/qapp_utils.py connect <appID>`

```
[root@ouellette106 logs-uba]# /opt/qradar/support/qapp_utils.py connect 1003
Existence of Docker check is complete. Moving on...
bash-4.1# tail -n 5 /store/log/app.log
Sep 19 17:33:14 127.0.0.1 [APP_ID/1003][NOT:0000006000][INFO] Poll finished for 'QDI Database Cleanup'
Sep 19 17:33:21 127.0.0.1 [APP_ID/1003][NOT:0000006000][INFO] Running poll for 'Server Info Check'
```

Debugging Deployed Apps

- App Logs
 - **/store/log/**
 - **app.log** (front end, views requests)
 - **poll.log** (back end)
- Can us vi to edit files live in container (html, js, python files)
 - Most changes real-time (html, js) if you reload page
 - Python changes require restarting the container
 - After changes, reload app/containers: **service qdocker restart**
- Get logs with app/docker logs, config, content:
 - **/opt/qradar/support/get_logs.sh -a**

Resource Management (cont)

- Docker containers are currently limited to 200Mb (!)

- **# docker ps**

CONTAINER ID	IMAGE	COMMAND	
CREATED	STATUS	PORTS	NAMES
f6adc016321b	a2949c82f50e4fd8b6f4c5c51377004e:latest	"bash	
start_containe	46 hours ago	Up 46 hours	0.0.0.0:49159-
>5000/tcp	a2949c82f50e4fd8b6f4c5c51377004e	_store_app	

- **# docker stats f6adc016321b**

CONTAINER	CPU %	MEM USAGE/LIMIT	MEM %	NET I/O
f6adc016321b	0.02%	67.93 MiB/100 MiB	67.93%	78.66 KiB/175.3 KiB

Helpful Links



Get information about apps and enablement

<https://developer.ibm.com/qradar>

- Forum links – Ask us questions!
 - <http://ibm.biz/qradarapps> (for app related questions)
 - <http://ibm/biz/qradarappdev> (for development related questions)
- Python: <https://www.python.org>
- Flask: <http://flask.pocoo.org>
- Jinja: <http://jinja.pocoo.org>




THANK YOU

FOLLOW US ON:

 <https://www.facebook.com/IBM-Security-Support/>

 QRadar Forums: <https://ibm.biz/qradarforums>

 [youtube/user/ibmsecuritysupport](https://www.youtube.com/user/ibmsecuritysupport)

 [@askibmsecurity](https://twitter.com/askibmsecurity)

 securityintelligence.com

 xforce.ibmcloud.com

© Copyright IBM Corporation 2016. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

Statement of Good Security Practices: IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a lawful, comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM does not warrant that any systems, products or services are immune from, or will make your enterprise immune from, the malicious or illegal conduct of any party.