

---

# **JupiterOne Documentation**

**JupiterOne**

**Jul 09, 2019**



<b>1</b>	<b>Configure Managed Integrations</b>	<b>1</b>
1.1	Other Data . . . . .	1
<b>2</b>	<b>Get started with search</b>	<b>3</b>
2.1	Ask Questions . . . . .	3
2.2	Full Text Search . . . . .	4
2.3	JupiterOne Query Language (J1QL) . . . . .	4
2.4	Combining full text search with J1QL . . . . .	4
<b>3</b>	<b>Navigating the JupiterOne Graphs</b>	<b>5</b>
<b>4</b>	<b>JupiterOne Query Language Tutorial</b>	<b>7</b>
4.1	Part 1 - Simple Root query . . . . .	7
4.2	Part 2 - Infrastructure Analysis . . . . .	9
4.3	Part 3 - User and Access Analysis . . . . .	14
4.4	Part 4 - Cross Account Analysis . . . . .	16
4.5	Part 5 - Endpoint Compliance . . . . .	16
<b>5</b>	<b>How to use filters in the Asset Inventory app</b>	<b>19</b>
5.1	Quick Filters by Class and/or Type . . . . .	19
5.2	Granular Filters by Properties . . . . .	20
<b>6</b>	<b>Alerts</b>	<b>25</b>
6.1	Import Alert Rules from Rule Pack . . . . .	25
6.2	Create Custom Alert Rules . . . . .	26
6.3	Managing Alerts . . . . .	27
6.4	Configure Daily Notification Email . . . . .	27
<b>7</b>	<b>Findings</b>	<b>29</b>
7.1	Managing Findings . . . . .	29
7.2	Create Alerts for Findings . . . . .	31
7.3	Visualizing Findings with J1 Query and Graph . . . . .	31
<b>8</b>	<b>Frequently Asked Questions</b>	<b>33</b>
8.1	How do I get my custom / on-premise data into JupiterOne? . . . . .	33
8.2	Where do these <code>Person</code> entities come from? Why are they not tagged with an integration? . . . . .	33
8.3	How do I add custom properties to my AWS entities from the source? . . . . .	33

8.4	Some AWS resources seem to be missing from the Asset Inventory / Graph. What is going on? . . .	34
8.5	I have a Network marked as “public”, what does that mean? . . . . .	34
8.6	How is it determined if an AWS VPC or Subnet is public? . . . . .	34
8.7	How are Person entities (i.e. employees) created? . . . . .	34
8.8	How can I avoid creating a Person entity for a generic/system user account? . . . . .	34
8.9	I see a user named “Callisto” on my account. Who is that? . . . . .	35
8.10	Endpoint compliance data isn’t appearing as expected. How can I troubleshoot this? . . . . .	35
8.11	How do I search/filter on all AWS entities without enumerating all types? . . . . .	36
<b>9</b>	<b>J1 Queries for AWS Config</b>	<b>37</b>
9.1	ACM Rules . . . . .	37
9.2	EC2 Rules . . . . .	37
9.3	IAM Rules . . . . .	38
9.4	Lambda Rules . . . . .	39
9.5	RDS Rules . . . . .	39
9.6	DynamoDB Rules . . . . .	40
9.7	S3 Rules . . . . .	40
9.8	Other Rules . . . . .	41
<b>10</b>	<b>Using JupiterOne for Active Vulnerability and Threat Monitoring in AWS</b>	<b>43</b>
10.1	Accessing the Findings in the Alerts app . . . . .	43
10.2	Correlation and Alerting . . . . .	45
<b>11</b>	<b>How to configure SAML SSO integration with JupiterOne</b>	<b>47</b>
11.1	Supported Features . . . . .	47
11.2	Configuration Steps . . . . .	47
11.3	Attribute Mappings . . . . .	50
11.4	Removing Users . . . . .	52
11.5	Current Limitations . . . . .	53
<b>12</b>	<b>Detect Suspicious Code Commits in Pull Requests</b>	<b>55</b>
12.1	Enable Detection . . . . .	55
12.2	How does it work? . . . . .	56
12.3	Combine suspicious commits checking and vulnerability checking for CI/CD . . . . .	57
<b>13</b>	<b>JupiterOne Node.js Client and CLI</b>	<b>59</b>
<b>14</b>	<b>Using JupiterOne as a central repository for SecOps and compliance artifacts</b>	<b>61</b>
14.1	TL;DR . . . . .	61
14.2	Security artifacts as code . . . . .	61
14.3	Uploading to JupiterOne . . . . .	66
<b>15</b>	<b>JupiterOne Endpoint Compliance Agent “Power Up”</b>	<b>69</b>
15.1	The Agent . . . . .	69
15.2	Installation . . . . .	69
15.3	Policies . . . . .	70
15.4	Advanced Use Cases . . . . .	70
<b>16</b>	<b>JupiterOne Data Model</b>	<b>71</b>
16.1	Entity . . . . .	71
16.2	Relationships . . . . .	72
16.3	What does this look like? . . . . .	75
<b>17</b>	<b>JupiterOne Data Security</b>	<b>77</b>
17.1	Data Protection . . . . .	77



17.2	External Data Ingestion/Import	78
17.3	Data Ownership and Access	78
17.4	Application Access	79
<b>18</b>	<b>JupiterOne Query Language (J1QL)</b>	<b>81</b>
18.1	Language Features	81
18.2	Basic Keywords	81
18.3	Sorting and Pagination via ORDER BY, SKIP, and LIMIT	84
18.4	Aggregation Functions: COUNT, MIN, MAX, AVG and SUM	84
18.5	Examples	85
18.6	Advanced Notes and Use Cases	87
<b>19</b>	<b>JupiterOne API</b>	<b>91</b>
19.1	Querying Entities and Relationships	91
19.2	Entity Mutations	98
19.3	Relationship Mutations	100
19.4	Building CSV Report	103
19.5	Alert and Rules Operations	103
19.6	Question Operations	107
<b>20</b>	<b>General</b>	<b>111</b>
20.1	Are my assets tracked? How many entities are there?	111
20.2	What are my production information assets and their owners and classification?	111
20.3	What are my production information assets?	112
20.4	What are my production systems and servers?	112
20.5	What are my production data stores and databases?	113
20.6	What are my production resources?	113
20.7	What applications and operating systems are in use?	113
20.8	What are my production applications?	114
20.9	Do I have proper vendor support for my software applications?	114
20.10	Who are the new hires within the last 12 months?	115
20.11	What business applications are we using?	115
20.12	What changed in my environment in the last 24 hours?	115
20.13	What was added to my environment in the last 24 hours?	116
<b>21</b>	<b>Access</b>	<b>117</b>
21.1	Find anything that allows public access to everyone.	117
21.2	Show me the current password policy and compliance status.	117
21.3	Are there external users with access to our systems?	118
21.4	Who has been assigned permissions with administrator/privileged access?	118
21.5	Who has access to what systems/resources?	119
21.6	Who owns which user accounts?	119
21.7	What are the shared/generic/service accounts or access roles? (Including user accounts that are not individually owned)	120
21.8	Did we remove all access from employees who left?	120
21.9	Which user accounts do not have multi-factor authentication enabled?	120
<b>22</b>	<b>Application Development</b>	<b>123</b>
22.1	What are the code repos for a particular application or project?	123
22.2	Were there any Code Repos added in the last 24 hours?	123
22.3	Who are the most recent contributors to this repo?	123
22.4	Which PRs did this developer open in the last 5 days?	124
<b>23</b>	<b>Data</b>	<b>125</b>
23.1	Are there any non-public data stores incorrectly configured with public access to everyone?	125

23.2	Which data stores do not have proper classification tags? . . . . .	125
23.3	What is the inventory of my sensitive data stores? . . . . .	126
23.4	Which production data stores do not have proper classification tags? . . . . .	126
23.5	Is there any known confidential or critical data outside of production? . . . . .	126
23.6	Evidence of data-at-rest encryption for production servers . . . . .	127
23.7	Is my production or PHI/PII data stores encrypted? . . . . .	127
23.8	Is my critical data in production encrypted? . . . . .	128
23.9	Is there unencrypted ePHI or PII? . . . . .	128
<b>24</b>	<b>Endpoints</b>	<b>129</b>
24.1	Whose endpoint is out of compliance? . . . . .	129
24.2	Is there anybody who does not have a user endpoint device (e.g. a laptop or workstation)? . . . . .	129
24.3	What is the configuration and compliance status of my endpoint devices? . . . . .	130
24.4	Is there malware protection for all endpoints? . . . . .	130
24.5	Is there protection for all user endpoints/devices? . . . . .	131
24.6	Is operating system patching and auto update enabled on endpoint hosts? . . . . .	131
24.7	Is application patching and auto update enabled on endpoint hosts? . . . . .	132
24.8	Are my servers and systems protected by hosted-based firewall? . . . . .	132
24.9	Are there security agents monitoring and protecting my endpoint hosts/devices? . . . . .	133
24.10	Is operating system patching and auto update enabled on endpoint hosts? . . . . .	133
24.11	Is application patching and auto update enabled on endpoint hosts? . . . . .	134
24.12	Are my servers and systems protected by hosted-based firewall? . . . . .	134
24.13	What are the approved server/system images? . . . . .	135
24.14	Are all system images updated in the past six months? . . . . .	135
24.15	Which hosts are (or are not) using approved standard images? . . . . .	136
24.16	Which devices have been disposed in the last 12 months? . . . . .	136
<b>25</b>	<b>Governance</b>	<b>137</b>
25.1	What are the corporate security policies and procedures? . . . . .	137
25.2	When was security policies and procedures last updated or reviewed? . . . . .	137
25.3	Who is the appointed security officer? . . . . .	138
25.4	Which are my documented risks? . . . . .	138
25.5	Was there at least one risk assessment performed within the past year? . . . . .	139
25.6	Who are my vendors? Do I have a BAA/DPA/NDA/MSA and SLA/Support Agreement with them? . . . . .	139
<b>26</b>	<b>Infrastructure</b>	<b>141</b>
26.1	What are directly connected to the Internet? . . . . .	141
26.2	What production resources are directly connected/exposed to the Internet/everyone? . . . . .	141
26.3	Are there potential IP collisions among the networks/subnets in my environment? . . . . .	142
26.4	What hosts or devices are connected to my internal networks? . . . . .	142
26.5	Show all inbound SSH firewall rules across my network environments. . . . .	142
26.6	Is inbound SSH allowed directly from an external host or network? . . . . .	143
26.7	What network traffic is allowed between internal and external (i.e. between trusted and untrusted) networks? . . . . .	143
26.8	Is there proper segmentation/segregation of internal networks? . . . . .	143
26.9	Are wireless networks segmented and protected by firewalls? . . . . .	144
26.10	Show listing of network layer firewall protection across all my environments. . . . .	144
26.11	Are there VPN configured for remote access? . . . . .	145
<b>27</b>	<b>Vulnerability Management</b>	<b>147</b>
27.1	What open vulnerabilities do I have? . . . . .	147
27.2	Which applications are vulnerable? . . . . .	147
<b>28</b>	<b>AWS</b>	<b>149</b>
28.1	Overview . . . . .	149

28.2	Integration Instance Configuration . . . . .	149
28.3	Permissions . . . . .	149
28.4	Entities . . . . .	150
28.5	Relationships . . . . .	150
<b>29</b>	<b>JupiterOne Managed Integration for Microsoft Azure</b>	<b>151</b>
29.1	Overview . . . . .	151
29.2	Integration Instance Configuration . . . . .	151
29.3	Entities . . . . .	152
29.4	Relationships . . . . .	152
<b>30</b>	<b>Bitbucket</b>	<b>153</b>
30.1	Overview . . . . .	153
30.2	Integration Instance Configuration . . . . .	153
30.3	Entities . . . . .	153
30.4	Relationships . . . . .	154
<b>31</b>	<b>Carbon Black PSC</b>	<b>155</b>
31.1	Overview . . . . .	155
31.2	Integration Instance Configuration . . . . .	155
31.3	Entities . . . . .	155
31.4	Relationships . . . . .	156
<b>32</b>	<b>GitHub</b>	<b>157</b>
32.1	Overview . . . . .	157
32.2	Integration Instance Configuration . . . . .	157
32.3	Permissions . . . . .	157
32.4	Entities . . . . .	158
32.5	Relationships . . . . .	158
<b>33</b>	<b>Google</b>	<b>159</b>
33.1	Overview . . . . .	159
33.2	Integration Instance Configuration . . . . .	159
33.3	Entities . . . . .	160
33.4	Relationships . . . . .	160
<b>34</b>	<b>HackerOne</b>	<b>161</b>
34.1	Overview . . . . .	161
34.2	Integration Instance Configuration . . . . .	161
34.3	Entities . . . . .	161
34.4	Relationships . . . . .	161
<b>35</b>	<b>jamf</b>	<b>163</b>
35.1	Overview . . . . .	163
35.2	Integration Instance Configuration . . . . .	163
35.3	Entities . . . . .	163
35.4	Relationships . . . . .	163
<b>36</b>	<b>Jira</b>	<b>165</b>
36.1	Overview . . . . .	165
36.2	Integration Instance Configuration . . . . .	165
36.3	Entities . . . . .	165
36.4	Relationships . . . . .	165
<b>37</b>	<b>KnowBe4</b>	<b>167</b>

37.1	Overview . . . . .	167
37.2	Integration Instance Configuration . . . . .	167
37.3	Entities . . . . .	167
37.4	Relationships . . . . .	167
<b>38</b>	<b>Okta</b>	<b>169</b>
38.1	Overview . . . . .	169
38.2	Integration Instance Configuration . . . . .	169
38.3	Entities . . . . .	169
38.4	Relationships . . . . .	169
38.5	Tips . . . . .	170
<b>39</b>	<b>OneLogin</b>	<b>171</b>
39.1	Overview . . . . .	171
39.2	Integration Instance Configuration . . . . .	171
39.3	Entities . . . . .	171
39.4	Relationships . . . . .	171
<b>40</b>	<b>Openshift</b>	<b>173</b>
40.1	Overview . . . . .	173
40.2	Integration Instance Configuration . . . . .	173
40.3	Entities . . . . .	173
40.4	Relationships . . . . .	174
<b>41</b>	<b>SentinelOne</b>	<b>175</b>
41.1	Overview . . . . .	175
41.2	Integration Instance Configuration . . . . .	175
41.3	Entities . . . . .	175
41.4	Relationships . . . . .	176
<b>42</b>	<b>Snyk</b>	<b>177</b>
42.1	Overview . . . . .	177
42.2	Integration Instance Configuration . . . . .	177
42.3	Entities . . . . .	177
42.4	Relationships . . . . .	177
<b>43</b>	<b>Tenable Cloud</b>	<b>179</b>
43.1	Overview . . . . .	179
43.2	Integration Instance Configuration . . . . .	179
43.3	Entities . . . . .	179
43.4	Relationships . . . . .	179
<b>44</b>	<b>Threat Stack</b>	<b>181</b>
44.1	Overview . . . . .	181
44.2	Integration Instance Configuration . . . . .	181
44.3	Entities . . . . .	181
44.4	Relationships . . . . .	182
<b>45</b>	<b>Veracode</b>	<b>183</b>
45.1	Overview . . . . .	183
45.2	Integration Instance Configuration . . . . .	183
45.3	Entities . . . . .	183
45.4	Relationships . . . . .	183
<b>46</b>	<b>Wazuh</b>	<b>185</b>

46.1	Overview . . . . .	185
46.2	Integration Instance Configuration . . . . .	185
46.3	Entities . . . . .	185
46.4	Relationships . . . . .	185
<b>47</b>	<b>Whitehat</b>	<b>187</b>
47.1	Overview . . . . .	187
47.2	Integration Instance Configuration . . . . .	187
47.3	Entities . . . . .	187
47.4	Relationships . . . . .	187
<b>48</b>	<b>AccessKey</b>	<b>189</b>
<b>49</b>	<b>AccessPolicy</b>	<b>191</b>
49.1	admin (boolean) - Optional . . . . .	191
49.2	rules (array of string) - Optional . . . . .	191
49.3	content (string) - Optional . . . . .	191
<b>50</b>	<b>AccessRelationship</b>	<b>193</b>
50.1	_class (string) - Optional . . . . .	193
50.2	permissions (array of string) - Optional . . . . .	193
50.3	accessLevel (array) - Optional . . . . .	194
50.4	protocol (string) - Optional . . . . .	194
50.5	portRange (string) - Optional . . . . .	194
50.6	type (string) - Optional . . . . .	194
<b>51</b>	<b>AccessRole</b>	<b>195</b>
<b>52</b>	<b>Account</b>	<b>197</b>
52.1	production (boolean) - Required . . . . .	197
52.2	accessURL (string) - Optional . . . . .	197
52.3	mfaEnabled (boolean) - Optional . . . . .	197
<b>53</b>	<b>Application</b>	<b>199</b>
53.1	COTS (boolean) - Optional . . . . .	199
53.2	FOSS (boolean) - Optional . . . . .	199
53.3	SaaS (boolean) - Optional . . . . .	199
53.4	external (boolean) - Optional . . . . .	199
53.5	mobile (boolean) - Optional . . . . .	200
53.6	license (string) - Optional . . . . .	200
53.7	licenseURL (string) - Optional . . . . .	200
53.8	productionURL (string) - Optional . . . . .	200
53.9	stagingURL (string) - Optional . . . . .	200
53.10	devURL (string) - Optional . . . . .	201
53.11	testURL (string) - Optional . . . . .	201
53.12	alternateURLs (array of string) - Optional . . . . .	201
<b>54</b>	<b>Assessment</b>	<b>203</b>
54.1	category (string) - Required . . . . .	203
54.2	summary (string) - Required . . . . .	204
54.3	internal (boolean) - Required . . . . .	204
54.4	startedOn (number) - Optional . . . . .	204
54.5	completedOn (number) - Optional . . . . .	204
54.6	reportURL (string) - Optional . . . . .	204
54.7	assessor (string) - Optional . . . . .	204

54.8	assessors (array of string) - Optional . . . . .	204
<b>55</b>	<b>Attacker</b>	<b>205</b>
<b>56</b>	<b>Certificate</b>	<b>207</b>
<b>57</b>	<b>Cluster</b>	<b>209</b>
<b>58</b>	<b>CodeCommit</b>	<b>211</b>
58.1	branch (string) - Required . . . . .	211
58.2	message (string) - Required . . . . .	211
58.3	merge (boolean) - Required . . . . .	211
58.4	versionBump (boolean) - Required . . . . .	211
<b>59</b>	<b>CodeDeploy</b>	<b>213</b>
59.1	jobName (string) - Optional . . . . .	213
59.2	jobNumber (integer) - Optional . . . . .	213
59.3	summary (string) - Optional . . . . .	213
59.4	action (string) - Optional . . . . .	213
59.5	target (string) - Optional . . . . .	214
59.6	production (boolean) - Optional . . . . .	214
<b>60</b>	<b>CodeModule</b>	<b>215</b>
60.1	public (boolean) - Optional . . . . .	215
<b>61</b>	<b>CodeRepo</b>	<b>217</b>
61.1	application (string) - Optional . . . . .	217
61.2	project (string) - Optional . . . . .	217
61.3	public (boolean) - Optional . . . . .	217
<b>62</b>	<b>CodeReview</b>	<b>219</b>
62.1	title (string) - Required . . . . .	219
62.2	summary (string) - Optional . . . . .	219
62.3	state (string) - Optional . . . . .	219
<b>63</b>	<b>Configuration</b>	<b>221</b>
<b>64</b>	<b>Control</b>	<b>223</b>
<b>65</b>	<b>ControlPolicy</b>	<b>225</b>
65.1	category (string) - Optional . . . . .	225
65.2	rules (array of string) - Optional . . . . .	225
65.3	content (string) - Optional . . . . .	226
<b>66</b>	<b>CryptoKey</b>	<b>227</b>
<b>67</b>	<b>DataObject</b>	<b>229</b>
67.1	category (string) - Optional . . . . .	229
67.2	format (string) - Optional . . . . .	229
67.3	classification (string) - Required . . . . .	229
67.4	location (string) - Optional . . . . .	230
67.5	PII (boolean) - Optional . . . . .	230
67.6	PHI (boolean) - Optional . . . . .	230
67.7	PCI (boolean) - Optional . . . . .	230
67.8	encryptionRequired (boolean) - Optional . . . . .	230
67.9	encrypted (boolean) - Optional . . . . .	230

67.10	public (boolean) - Optional	230
<b>68</b>	<b>DataStore</b>	<b>231</b>
68.1	location (string) - Optional	231
68.2	encryptionRequired (boolean) - Optional	231
68.3	encryptionAlgorithm (string) - Optional	231
68.4	encryptionKeyRef (string) - Optional	231
68.5	encrypted (boolean) - Optional	232
68.6	public (boolean) - Optional	232
68.7	hasBackup (boolean) - Optional	232
<b>69</b>	<b>Database</b>	<b>233</b>
69.1	location (string) - Optional	233
69.2	encryptionRequired (boolean) - Optional	233
69.3	encrypted (boolean) - Optional	233
69.4	classification (string) - Required	233
<b>70</b>	<b>Deployment</b>	<b>235</b>
70.1	desiredSize (number) - Optional	235
70.2	currentSize (number) - Optional	235
70.3	maxSize (number) - Optional	235
<b>71</b>	<b>Device</b>	<b>237</b>
71.1	category (string) - Required	237
71.2	hardwareVendor (string) - Required	237
71.3	hardwareModel (string) - Required	237
71.4	hardwareVersion (string) - Optional	238
71.5	hardwareSerial (string) - Required	238
71.6	assetTag (string) - Optional	238
71.7	platform (string) - Optional	238
71.8	osDetails (string) - Optional	238
71.9	osName (string) - Optional	238
71.10	osVersion (string) - Optional	238
71.11	userEmails (array of string) - Optional	239
71.12	location (string) - Optional	239
71.13	cost (number) - Optional	239
71.14	value (number) - Optional	239
71.15	BYOD (boolean) - Required	239
71.16	status (string) - Optional	239
<b>72</b>	<b>Document</b>	<b>241</b>
<b>73</b>	<b>Domain</b>	<b>243</b>
<b>74</b>	<b>Entity</b>	<b>245</b>
74.1	name (string) - Required	245
74.2	displayName (string) - Required	245
74.3	summary (string) - Optional	245
74.4	description (string) - Optional	245
74.5	classification (string) - Optional	246
74.6	criticality (integer) - Optional	246
74.7	risk (integer) - Optional	246
74.8	trust (integer) - Optional	246
74.9	complianceStatus (number) - Optional	246
74.10	status (string) - Optional	246

74.11	active (boolean) - Optional	247
74.12	public (boolean) - Optional	247
74.13	validated (boolean) - Optional	247
74.14	temporary (boolean) - Optional	247
74.15	createdOn (number) - Optional	247
74.16	updatedOn (number) - Optional	247
74.17	expiresOn (number) - Optional	247
74.18	webLink (string) - Optional	247
74.19	owner (string) - Optional	248
74.20	tag.* (string) - Optional	248
74.21	tags (array of string) - Optional	248
74.22	notes (array of string) - Optional	248
<b>75</b>	<b>Finding</b>	<b>249</b>
75.1	assessment (string) - Optional	249
75.2	status (string) - Optional	249
75.3	severity (string) - Required	249
75.4	priority (string) - Optional	250
75.5	score (number) - Optional	250
75.6	impact (string) - Optional	250
75.7	exploitability (number) - Optional	250
75.8	vector (string) - Optional	250
75.9	stepsToReproduce (array of string) - Optional	250
75.10	recommendation (string) - Optional	250
75.11	targets (array of string) - Optional	250
75.12	targetDetails (array of string) - Optional	250
75.13	remediationSLA (integer) - Optional	251
75.14	blocksProduction (boolean) - Optional	251
75.15	open (boolean) - Required	251
75.16	production (boolean) - Required	251
75.17	public (boolean) - Required	251
75.18	validated (boolean) - Optional	251
75.19	references (array of string) - Optional	251
<b>76</b>	<b>Firewall</b>	<b>253</b>
76.1	category (array of string) - Required	253
76.2	isStateful (boolean) - Optional	253
<b>77</b>	<b>Framework</b>	<b>255</b>
77.1	name (string) - Required	255
77.2	displayName (string) - Required	255
77.3	summary (string) - Optional	255
77.4	description (string) - Optional	255
77.5	standard (string) - Required	256
77.6	version (string) - Required	256
<b>78</b>	<b>Function</b>	<b>257</b>
78.1	image (string) - Optional	257
78.2	version (string) - Optional	257
78.3	runtime (string) - Optional	257
78.4	memorySize (string) - Optional	257
78.5	codeSize (string) - Optional	258
78.6	codeHash (string) - Optional	258
78.7	trigger (string) - Optional	258
78.8	handler (string) - Optional	258



<b>79 Gateway</b>	<b>259</b>
79.1 category (array of string) - Required	259
79.2 function (array of string) - Required	259
79.3 public (boolean) - Required	260
<b>80 Group</b>	<b>261</b>
<b>81 Host</b>	<b>263</b>
81.1 hostname (string) - Required	263
81.2 ipAddress (string) - Optional	263
81.3 publicDnsName (string) - Optional	263
81.4 privateDnsName (string) - Optional	264
81.5 publicIpAddress (string) - Optional	264
81.6 privateIpAddress (string) - Optional	264
81.7 ipAddresses (array of string) - Optional	264
81.8 ipv6Addresses (array of string) - Optional	264
81.9 macAddress (string) - Optional	264
81.10 platform (string) - Optional	264
81.11 osDetails (string) - Optional	265
81.12 osName (string) - Optional	265
81.13 osVersion (string) - Optional	265
81.14 macAddresses (array of string) - Optional	265
81.15 isPhysical (boolean) - Optional	265
<b>82 HostAgent</b>	<b>267</b>
82.1 function (array of string) - Required	267
<b>83 Image</b>	<b>269</b>
<b>84 Incident</b>	<b>271</b>
84.1 category (string) - Required	271
84.2 severity (string) - Required	271
84.3 impacts (array of string) - Optional	272
84.4 reportable (boolean) - Required	272
84.5 reporter (string) - Optional	272
84.6 postmortem (string) - Optional	272
<b>85 Internet</b>	<b>273</b>
85.1 displayName (string) - Optional	273
85.2 CIDR (string) - Optional	273
85.3 CIDRv6 (string) - Optional	273
85.4 public (boolean) - Optional	273
<b>86 IpAddress</b>	<b>275</b>
86.1 dnsName (string) - Optional	275
86.2 publicIpAddress (string) - Optional	275
86.3 privateIpAddress (string) - Optional	275
86.4 ipVersion (integer) - Optional	276
<b>87 Key</b>	<b>277</b>
87.1 fingerprint (string) - Optional	277
87.2 material (string) - Optional	277
87.3 usage (string) - Optional	277
<b>88 Metadata</b>	<b>279</b>

88.1	_accountId (string) - Required	279
88.2	_id (string) - Required	279
88.3	_key (string) - Required	279
88.4	__iconPath (string) - Optional	279
88.5	_class (string) - Required	280
88.6	_type (string) - Required	280
88.7	_integrationName (string) - Optional	280
88.8	_integrationDefinitionId (string) - Optional	280
88.9	_integrationInstanceId (string) - Optional	280
88.10	_createdOn (number) - Required	280
88.11	_createdBy (string) - Optional	280
88.12	_beginOn (number) - Required	280
88.13	_endOn (number) - Optional	281
88.14	_updatedBy (string) - Optional	281
88.15	_lastSeenOn (number) - Required	281
88.16	_version (integer) - Required	281
88.17	_latest (boolean) - Optional	281
88.18	_deleted (boolean) - Optional	281
88.19	vendorManaged (boolean) - Optional	281
88.20	inUse (boolean) - Optional	281
88.21	ignore (boolean) - Optional	281
<b>89</b>	<b>Module</b>	<b>283</b>
89.1	public (boolean) - Optional	283
<b>90</b>	<b>Network</b>	<b>285</b>
90.1	environment (string) - Required	285
90.2	CIDR (string) - Required	286
90.3	CIDRv6 (string) - Optional	286
90.4	public (boolean) - Required	286
90.5	internal (boolean) - Required	286
90.6	wireless (boolean) - Optional	286
<b>91</b>	<b>NetworkInterface</b>	<b>287</b>
91.1	macAddress (string) - Optional	287
91.2	dnsName (string) - Optional	287
91.3	publicIpAddress (string) - Optional	287
91.4	privateIpAddress (string) - Optional	288
91.5	ipVersion (integer) - Optional	288
<b>92</b>	<b>Organization</b>	<b>289</b>
92.1	_type (string) - Optional	289
92.2	website (string) - Optional	289
92.3	emailDomain (string) - Optional	290
92.4	external (boolean) - Optional	290
<b>93</b>	<b>PR</b>	<b>291</b>
93.1	title (string) - Required	291
93.2	summary (string) - Optional	291
93.3	state (string) - Required	291
93.4	source (string) - Required	292
93.5	target (string) - Required	292
93.6	repository (string) - Required	292
93.7	approved (boolean) - Optional	292
93.8	validated (boolean) - Optional	292

<b>94 PasswordPolicy</b>	<b>293</b>
94.1 minLength (integer) - Optional	293
94.2 requireSymbols (boolean) - Optional	293
94.3 requireNumbers (boolean) - Optional	293
94.4 requireUppercase (boolean) - Optional	293
94.5 requireLowercase (boolean) - Optional	294
94.6 maxAgeDays (integer) - Optional	294
94.7 minAgeMins (integer) - Optional	294
94.8 historyCount (integer) - Optional	294
94.9 preventReset (boolean) - Optional	294
94.10 expiryWarningDays (integer) - Optional	294
94.11 hardExpiry (boolean) - Optional	294
94.12 excludeUsername (boolean) - Optional	294
94.13 excludeAttributes (array of string) - Optional	294
94.14 excludeCommonPasswords (boolean) - Optional	295
94.15 lockoutAttempts (integer) - Optional	295
94.16 autoUnlockMins (integer) - Optional	295
94.17 requireMFA (boolean) - Optional	295
<b>95 Person</b>	<b>297</b>
95.1 firstName (string) - Required	297
95.2 lastName (string) - Required	297
95.3 middleName (string) - Optional	297
95.4 email (array of string) - Required	297
95.5 title (string) - Optional	298
95.6 phone (array of string) - Optional	298
95.7 address (string) - Optional	298
95.8 employeeId (string) - Optional	298
95.9 employeeType (string) - Optional	298
95.10 userIds (array of string) - Optional	298
95.11 manager (string) - Optional	298
95.12 managerId (string) - Optional	298
95.13 managerEmail (string) - Optional	299
<b>96 Policy</b>	<b>301</b>
96.1 title (string) - Required	301
96.2 summary (string) - Required	301
96.3 author (string) - Optional	301
96.4 content (string) - Required	301
96.5 applicable (boolean) - Optional	302
96.6 adopted (boolean) - Optional	302
<b>97 Procedure</b>	<b>303</b>
97.1 title (string) - Required	303
97.2 summary (string) - Required	303
97.3 author (string) - Optional	303
97.4 content (string) - Required	303
97.5 control (string) - Optional	304
97.6 applicable (boolean) - Optional	304
97.7 adopted (boolean) - Optional	304
<b>98 Process</b>	<b>305</b>
98.1 state (string) - Optional	305
<b>99 Project</b>	<b>307</b>

99.1	key (string) - Optional . . . . .	307
99.2	productionURL (string) - Optional . . . . .	307
99.3	stagingURL (string) - Optional . . . . .	307
99.4	devURL (string) - Optional . . . . .	308
99.5	testURL (string) - Optional . . . . .	308
99.6	alternateURLs (array of string) - Optional . . . . .	308
<b>100</b>	<b>Record</b>	<b>309</b>
<b>101</b>	<b>RecordEntity</b>	<b>311</b>
101.1	name (string) - Required . . . . .	311
101.2	displayName (string) - Required . . . . .	311
101.3	summary (string) - Optional . . . . .	311
101.4	description (string) - Optional . . . . .	311
101.5	classification (string) - Optional . . . . .	312
101.6	category (string) - Optional . . . . .	312
101.7	webLink (string) - Optional . . . . .	312
101.8	content (string) - Optional . . . . .	312
101.9	open (boolean) - Optional . . . . .	313
101.10	public (boolean) - Optional . . . . .	313
101.11	production (boolean) - Optional . . . . .	313
101.12	approved (boolean) - Optional . . . . .	313
101.13	approvedOn (number) - Optional . . . . .	313
101.14	approvers (array of string) - Optional . . . . .	313
101.15	reporter (string) - Optional . . . . .	313
101.16	reportedOn (number) - Optional . . . . .	313
101.17	createdOn (number) - Optional . . . . .	314
101.18	updatedOn (number) - Optional . . . . .	314
<b>102</b>	<b>Relationship</b>	<b>315</b>
102.1	_class (string) - Optional . . . . .	315
102.2	displayName (string) - Optional . . . . .	316
102.3	webLink (string) - Optional . . . . .	316
102.4	isValidated (boolean) - Optional . . . . .	316
102.5	isTemporary (boolean) - Optional . . . . .	316
102.6	isGroupLayout (boolean) - Optional . . . . .	317
102.7	tag.* (string) - Optional . . . . .	317
102.8	tags (array of string) - Optional . . . . .	317
<b>103</b>	<b>Requirement</b>	<b>319</b>
103.1	title (string) - Required . . . . .	319
103.2	summary (string) - Optional . . . . .	319
103.3	state (string) - Optional . . . . .	319
<b>104</b>	<b>Resource</b>	<b>321</b>
<b>105</b>	<b>Review</b>	<b>323</b>
105.1	title (string) - Required . . . . .	323
105.2	summary (string) - Optional . . . . .	323
105.3	state (string) - Optional . . . . .	323
<b>106</b>	<b>Risk</b>	<b>325</b>
106.1	assessment (string) - Optional . . . . .	325
106.2	category (string) - Optional . . . . .	325
106.3	probability (integer) - Required . . . . .	325

106.4 impact (integer) - Required . . . . .	326
106.5 score (integer) - Required . . . . .	326
106.6 details (string) - Optional . . . . .	326
106.7 mitigation (string) - Optional . . . . .	326
106.8 status (string) - Required . . . . .	326
<b>107Root</b>	<b>327</b>
107.1 displayName (string) - Optional . . . . .	327
<b>108Rule</b>	<b>329</b>
108.1 category (string) - Optional . . . . .	329
108.2 content (string) - Optional . . . . .	329
<b>109Ruleset</b>	<b>331</b>
109.1 category (string) - Optional . . . . .	331
109.2 rules (array of string) - Optional . . . . .	331
109.3 content (string) - Optional . . . . .	331
<b>110Scanner</b>	<b>333</b>
110.1 category (array of string) - Required . . . . .	333
<b>111Service</b>	<b>335</b>
111.1 category (array of string) - Required . . . . .	335
111.2 endpoints (array of string) - Required . . . . .	335
<b>112Site</b>	<b>337</b>
112.1 category (array of string) - Optional . . . . .	337
112.2 location (string) - Optional . . . . .	337
112.3 hours (string) - Optional . . . . .	338
112.4 secured (boolean) - Optional . . . . .	338
112.5 restricted (boolean) - Optional . . . . .	338
<b>113Task</b>	<b>339</b>
<b>114Team</b>	<b>341</b>
114.1 email (string) - Optional . . . . .	341
<b>115Training</b>	<b>343</b>
<b>116User</b>	<b>345</b>
116.1 username (string) - Required . . . . .	345
116.2 email (string) - Optional . . . . .	345
116.3 shortLoginId (string) - Optional . . . . .	345
116.4 mfaEnabled (boolean) - Optional . . . . .	345
<b>117UserGroup</b>	<b>347</b>
117.1 email (string) - Optional . . . . .	347
<b>118Vendor</b>	<b>349</b>
118.1 category (string) - Required . . . . .	349
118.2 website (string) - Optional . . . . .	350
118.3 departments (array of string) - Optional . . . . .	350
118.4 emailDomain (string) - Optional . . . . .	350
118.5 mainContactName (string) - Optional . . . . .	350
118.6 mainContactEmail (string) - Optional . . . . .	350
118.7 mainContactPhone (string) - Optional . . . . .	350

118.8	mainContactAddress (string) - Optional	351
118.9	admins (array of string) - Optional	351
118.10	breachResponseDays (integer) - Optional	351
118.11	linkToNDA (string) - Optional	351
118.12	linkToMSA (string) - Optional	351
118.13	linkToSLA (string) - Optional	351
118.14	linkToBAA (string) - Optional	351
118.15	linkToDPA (string) - Optional	351
118.16	linkToVTR (string) - Optional	352
118.17	linkToISA (string) - Optional	352
118.18	statusPage (string) - Optional	352
<b>119</b>	<b>Vulnerability</b>	<b>353</b>
119.1	category (string) - Required	353
119.2	status (string) - Optional	353
119.3	severity (string) - Required	354
119.4	priority (string) - Optional	354
119.5	score (number) - Optional	354
119.6	impact (number) - Optional	354
119.7	exploitability (number) - Optional	354
119.8	vector (string) - Optional	354
119.9	impacts (array of string) - Optional	354
119.10	remediationSLA (integer) - Optional	354
119.11	blocking (boolean) - Required	355
119.12	open (boolean) - Required	355
119.13	production (boolean) - Required	355
119.14	public (boolean) - Required	355
119.15	validated (boolean) - Optional	355
119.16	references (array of string) - Optional	355
<b>120</b>	<b>Weakness</b>	<b>357</b>
120.1	category (string) - Optional	357
120.2	exploitability (string) - Optional	357
120.3	references (array of string) - Optional	357
<b>121</b>	<b>Workload</b>	<b>359</b>
121.1	image (string) - Optional	359
121.2	fqdn (string) - Optional	359
<b>122</b>	<b>JupiterOne 2018.10 Release</b>	<b>361</b>
122.1	New Features	361
122.2	Improvements	361
122.3	Bug Fixes	361
<b>123</b>	<b>JupiterOne 2018.11 Release</b>	<b>363</b>
123.1	New Features	363
123.2	Improvements	363
<b>124</b>	<b>JupiterOne 2018.12 Release</b>	<b>365</b>
124.1	New Features	365
124.2	Improvements	365
<b>125</b>	<b>JupiterOne 2018.13 Release</b>	<b>367</b>
125.1	New Features	367
125.2	Improvements	367

<b>126JupiterOne 2018.14 Release</b>	<b>369</b>
126.1 New Features . . . . .	369
126.2 Improvements and Bug Fixes . . . . .	371
<b>127JupiterOne 2019.15 Release</b>	<b>373</b>
127.1 New Features . . . . .	373
127.2 Improvements and Bug Fixes . . . . .	375
<b>128JupiterOne 2019.16 Release</b>	<b>377</b>
128.1 New Features . . . . .	377
128.2 Improvements and Bug Fixes . . . . .	378
<b>129JupiterOne 2019.17 Release</b>	<b>379</b>
129.1 New Features . . . . .	379
129.2 Improvements and Bug Fixes . . . . .	379
<b>130JupiterOne 2019.18 Release</b>	<b>381</b>
130.1 New Features . . . . .	381
130.2 Improvements and Bug Fixes . . . . .	382
<b>131JupiterOne 2019.19 Release</b>	<b>383</b>
131.1 New Features . . . . .	383
131.2 Improvements and Bug Fixes . . . . .	384
<b>132JupiterOne 2019.20 Release</b>	<b>387</b>
132.1 New Features . . . . .	387
132.2 Improvements and Bug Fixes . . . . .	389
<b>133JupiterOne 2019.21 Release</b>	<b>391</b>
133.1 New Features . . . . .	391
133.2 Improvements and Bug Fixes . . . . .	392
<b>134JupiterOne 2019.22 Release</b>	<b>393</b>
134.1 New Features . . . . .	393
134.2 Improvements and Bug Fixes . . . . .	393
134.3 Additional Notes . . . . .	394
<b>135JupiterOne 2019.23 Release</b>	<b>395</b>
135.1 New Features . . . . .	395
135.2 Improvements and Bug Fixes . . . . .	395
<b>136JupiterOne 2019.24 Release</b>	<b>397</b>
136.1 New Features . . . . .	397
136.2 Improvements and Bug Fixes . . . . .	399
<b>137JupiterOne 2019.25 Release</b>	<b>401</b>
137.1 New Features . . . . .	401
137.2 Early Access / Beta Features . . . . .	401
137.3 Improvements and Bug Fixes . . . . .	403





---

## Configure Managed Integrations

---

You will need to have data in the JupiterOne platform to take advantage of its capabilities. The more data, the more powerful these capabilities become.

There are over a dozen managed integrations available out-of-the-box for turnkey configuration. More are added regularly.

Each integration may have a slightly different mechanism for authentication and configuration, as required by the provider. For example, the AWS integration uses an IAM Role and Assume Role Trust policies for access. Other integrations may use an API key/token, OAuth, or Basic Auth.

This recording below shows an example of how to configure an AWS integration.

configure-aws-integration

For details on other integrations, please see their corresponding documentation page under the **Managed Integrations** section.

### 1.1 Other Data

Additionally, you can upload data outside of these managed integrations using the JupiterOne [API Client](#) or [CLI](#). This allows you to centrally track, monitor and visualize any of your data such as on-premise systems and security / compliance artifacts.



---

### Get started with search

---

You can quickly search and get insight across your entire digital environment integrated with JupiterOne, right here from the Landing Zone. There are three modes of search:

1. **Ask questions** by typing in any keywords to search across all packaged/saved questions
2. **Full text search** across all entities based on their property values
3. **JupiterOne query language (J1QL)** for precise querying of entities and relationships

Results can be toggled in four different display modes: **Table**, **Graph**, **Raw JSON**, or **Pretty JSON**.

*Note that for performance reasons, search results are limited to return up to 250 items. If you believe something is missing from a large result set, try tuning the query to generate more precise results.*

## 2.1 Ask Questions

Just start typing any keyword (or combination of keywords) such as these (without quotes):

- compliance
- access
- traffic
- ssh
- data encrypted
- production

Or ask a question like:

- Who are my vendors?
- What lambda functions do I have in AWS?
- What is connected to the Internet?
- Who has access to ... ?

## 2.2 Full Text Search

Put your keywords in quotes (e.g. “keyword”) to start a full text search. For example,

- “0123456789012” will likely find an AWS Account entity with that account ID
- “sg-123ab45c” will find an AWS EC2 Security Group with that group ID
- “Charlie” will find a Person and/or User with that first name

## 2.3 JupiterOne Query Language (J1QL)

The JupiterOne Query Language (J1QL) is used here for searching for anything across all of your entities and relationships.

To start, understand the basic query structure:

```
FIND {class or type of Entity1} AS {alias1}
  WITH {property}={value} AND|OR {property}={value}
  THAT {relationship_verb} {class or type of Entity2} AS {alias2}
  WHERE {alias1}.{property} = {alias2}.{property}
```

For example:

- Find User that IS Person
- Find Firewall that ALLOWS as rule (Network|Host) where rule.ingress=truee and rule.fromPort=22
- Find \* with tag.Production='true' (note the wildcard \* here)

The query language is case insensitive except for the following:

- TitleCase Entity keyword after Find and the {relationship verb} will search for entities of that **Class**. (e.g. CodeRepo)
- lowercase Entity keyword after Find and the {relationship verb} will search for entities of that **Type**. An entity type with more than one word is generally in snake\_case. (e.g. github\_repo)
- Entity property names and values, and alias names defined as part of the query, are case sensitive.

Checkout the [J1QL query tutorial](#) and the [complete J1QL documentation](#) with more advanced examples.

## 2.4 Combining full text search with J1QL

You can also start with a full text search and then use J1QL to further filter the results from the initial search. For example:

```
"Administrator" with _class='AccessPolicy' that ASSIGNED (User|AccessRole)
```

## CHAPTER 3

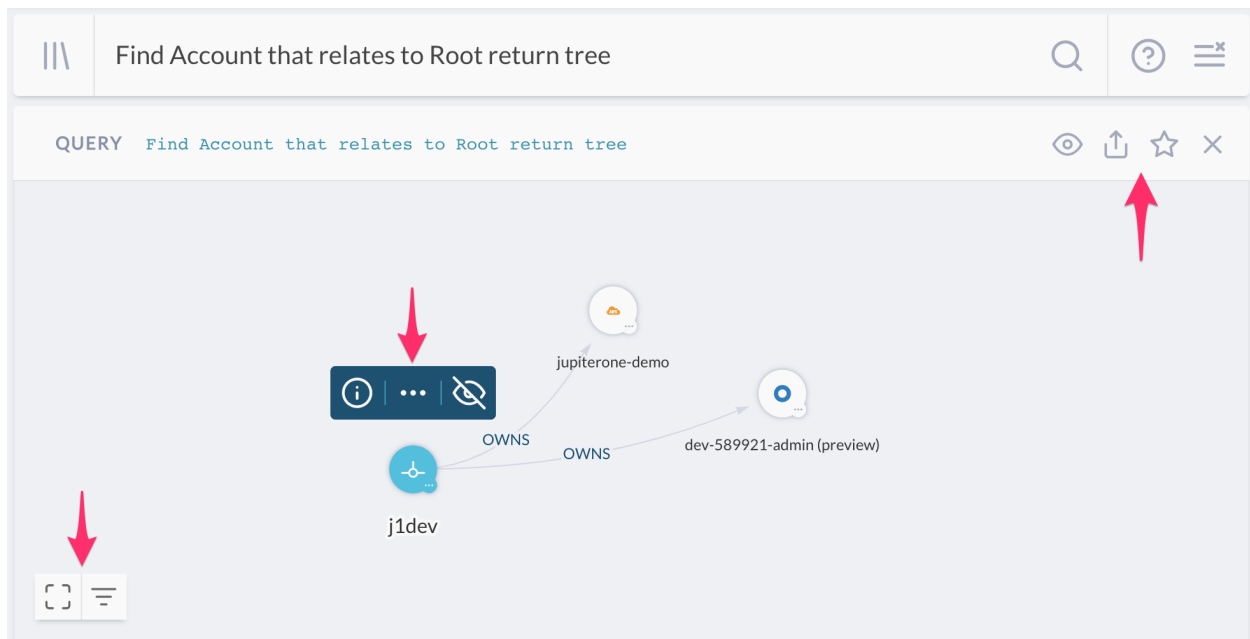
### Navigating the JupiterOne Graphs

JupiterOne is built on a data-driven graph platform. For the story that inspired us to build it, check out [this blog](#).

JupiterOne query language (J1QL) is designed to traverse this graph and return a sub-graph – or data from the nodes (i.e. entities) and edges (i.e. relationships) of a sub-graph. You can view and interact with the sub-graph from any J1QL query result.

This guide focuses on interacting with the graph component. For more details on J1QL, check out the [J1QL tutorial](#) and [technical doc](#).

This screenshot below shows an example result graph from a query in the Landing app:

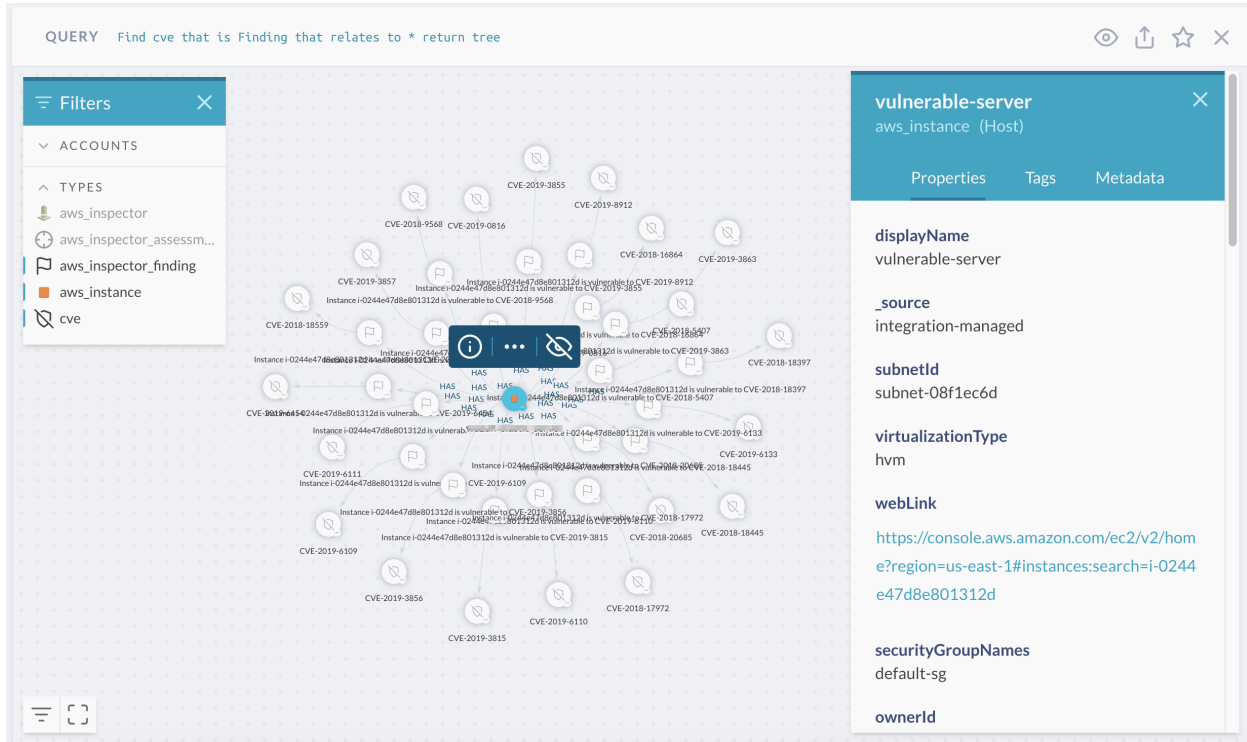


The first set of controls on the upper right corner does the following:

Selecting any node (i.e. entity) on the graph will bring up a set of controls right on top of it that allows you to interact with the node. They serve the following functions:

The last set of controls are at the bottom left corner of the graph, and they do the following:

Here's a screenshot of a graph with the **property panel** and **filter panel** open:



### Zoom and Move

The stand-alone **Galaxy / Graph Viewer** app uses the same sets of controls.

**That's it!** Now go explore! Check out the [J1QL tutorial](#) if you haven't yet.

---

## JupiterOne Query Language Tutorial

---

Querying can be the most challenging, yet the most fun and rewarding part of the JupiterOne experience. Once you become familiar with the query language, we are certain that you will find yourself uncovering all sorts of previously undiscovered insight from your data.

The JupiterOne Query Language (aka “J1QL”) is a query language for finding the entities and relationships within your digital environment. J1QL blends together the capabilities of asking questions, performing full text search, or querying the complex entity-relationship graph.

There are plenty of pre-packaged queries you can easily use in the **Landing** app or browse in **Query Library**. This tutorial focuses instead on helping you construct custom queries yourself.

This tutorial builds on the [full J1QL documentation](#) using some common use cases.

### 4.1 Part 1 - Simple Root query

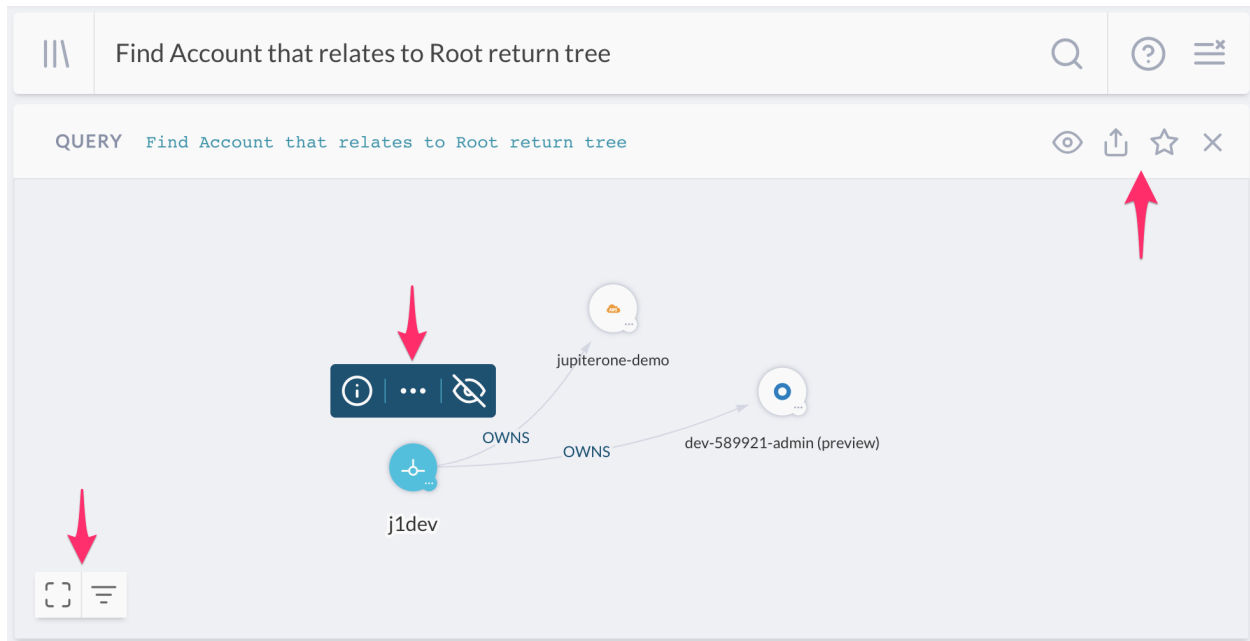
First, let’s try this query:

```
Find Account that relates to Root return tree
```

Please note the noun that immediately follows the verb is case sensitive:

- A `TitleCase` word tells the query to search for entities of that **class** (e.g. `Account`, `Firewall`, `Gateway`, `Host`, `User`, `Root`, `Internet`, etc.);
- A `snake_case` word tells the query to search for entities of that **type** (e.g. `aws_account`, `aws_security_group`, `aws_internet_gateway`, `aws_instance`, `aws_iam_user`, `okta_user`, `user_endpoint`, etc.)

You should get a result that looks like this (the `return tree` part of the query tells it to show the graph view by default):



The selected node in the above example is the special `Root` node, which represents your organization. Depending on the number of integration configurations you have, you'll see different number of accounts connected, showing that the `Root` entity `OWNS` these `Account` entities.

See the three sets of controls in the result panel. Starting from top right to bottom left –

The first set of controls (next to the query) allows you to:

- Switch views between **Table**, **Graph**, **Raw JSON**, and **Pretty JSON**.
- Share the query – shows a popup box with the weblink to copy and share.
- Save the query – give it a title, description, and optionally some tags to save it to your own query library.
- Close / remove this result panel from the page.

The second set of controls (above the selected entity node) allows you to:

- Show the detailed properties, tags and metadata of the selected entity.
- Expand the entity to see more of its connected neighbors - this will bring in additional data that may not have been returned by the original query, allowing you to further the search and analysis.
- Hide the selected entity node from the graph view - once you've hidden an entity, an unhide button will show up in the third set of controls at the bottom left of the graph, allowing you to unhide all currently hidden entities.

The last set of controls (at the bottom left corner) allows you to:

- Toggle the full screen mode.
- Opens up the filter panel to show/hide entities in the graph by account or entity type.
- Unhide all currently hidden entities (not shown in the above screenshot – it only shows up when there is at least one hidden entity).

See more details on the graph controls in [this doc](#).



## 4.2 Part 2 - Infrastructure Analysis

Examples in this section require at least one AWS integration configuration.

If you've configured an AWS integration, you are now ready to try something a lot more interesting. Type in, or copy/paste the following query:

### 4.2.1 2a - SSH Key Usage Examples

```
Find AccessKey with usage='ssh'
```

This should find a set of `aws_access_key` entities used for SSH access into your EC2 instances, assuming you have some of those and they are configured to allow SSH access.

You can also query by the entity type instead of its class. The following query will get you the same result - unless you also have SSH Keys you've added from other integrations (non-AWS) or from the UI / API.

```
Find aws_key_pair
```

Now expand the search a little bit with the following:

```
Find Host as h
  that uses AccessKey with usage='ssh' as k
  return
    h.tag.AccountName,
    h._type,
    h.displayName,
    h.instanceId,
    h.region,
    h.availabilityZone,
    h.publicIpAddress,
    h.privateIpAddress,
    h.platform,
    h.instanceType,
    h.state,
    k._type,
    k.displayName
```

This finds the `Host` entities that **USES** each `AccessKey` and returns a set of specific properties. You can add or remove properties returned as desired.

Note the keyword `that` is what tells the query to traverse the graph to find connections/relationships between entities, followed by a *verb* that represents :) the relationship class.

Also keep in mind you can switch to the **Graph** view to get a more visual result, and continue to drill down interactively.

Again, you can query using the more specific entity types. For example:

```
Find aws_instance that uses aws_key_pair
```

Or mix and match them:

```
Find Host that uses aws_key_pair
```

Note that the relationship keyword/verb is *not* case sensitive.

## 4.2.2 2b - EBS Volume Examples

First, let's see if there are any unencrypted EBS volumes:

```
Find aws_ebs_volume with encrypted != true
```

Note in the above query, the `with` keyword binds to the entity noun immediately to its left, and allows you to filter results on that entity's property values.

If the above query finds some unencrypted EBS volumes, it'll be interested to see what's using them:

```
Find Host that uses aws_ebs_volume with encrypted != true
```

You can view the `aws_ebs_volume` entities and their relationships in the **Graph** mode, and further inspect the properties on each entity node or relationship edge. You can also expand to see more connected entities and relationships.

Are these actively in use? And in production?

```
Find Host with active = true and tag.Production = true
  that uses aws_ebs_volume with encrypted != true
```

What subnets are these instances in? Let's also just return a few key properties from type of entities related in this search:

```
Find Network as n
  that has Host as h
  that uses aws_ebs_volume with encrypted != true and tag.Production = true as e_
↪return
  n.displayName, h._type, h.displayName, e.displayName, e.encrypted
```

OK. How about any EBS Volumes *not* actively in use? Perhaps some of them can be removed...

```
Find aws_ebs_volume that !uses Host
```

You may notice the above query feels backwards. That's okay. The query will work the same way regardless of the direction of relationship. Because the query by default returns all properties from the initial set of entities, it is sometimes easier to reverse the query direction so that you get the data you're looking for more easily.

Technically, `Find Host that !uses aws_ebs_volume as v return v.*` may feel more correct, but it is definitely a bit more to type out.

## 4.2.3 2c - Unencrypted Data

There are many types of data stores you may have in AWS. For example, **EBS Volumes, S3 Buckets, RDS Clusters and Instances, DynamoDB Tables, Redshift Clusters**, to name a few. You likely want them to be encrypted if they store confidential data.

How do you find out if that's the case?

```
Find (aws_s3_bucket|aws_rds_cluster|aws_db_instance|aws_dynamodb_table|aws_redshift_
↪cluster) with encrypted!=true
```

The above query will certainly do the job, but it's quite complicated. This is where the abstract class labeling automatically assigned by JupiterOne serves its purpose. Querying by class makes it a whole lot simpler:

```
Find DataStore with encrypted != true
```

Now, you can start adding a few property filters to make the results much more focused, to help cut down the noise or to prioritize remediation. For example:

```
Find DataStore with
  encrypted != true and
  tag.Production = true and
  (classification = 'confidential' or classification = 'restricted')
```

#### 4.2.4 2d - Tagging Resources

As you can see from some of the earlier examples, tagging resources can be very useful operationally. That's why we highly recommend tagging your resources at the source. These tags will be ingested by JupiterOne and you can use them in your custom queries.

By default, the packaged queries provided by JupiterOne, as seen in the **Query Library** from the **Landing** app and used in the **Compliance** app, rely on the following tags:

- Classification
- Owner
- PII or PHI or PCI (boolean tags to indicate data type)
- AccountName
- Production

All custom tags ingested by JupiterOne integrations are prefixed with `tag.<TagName>`. They need to be used as such in the query.

The `Classification` and `Owner` tags are automatically captured as properties so they can be used directly in the query without the `tag. prefix` - in all lower case: `classification = '...'` or `owner= '...'`.

The `tag.AccountName (string)` and `tag.Production (boolean)` tags can be added by JupiterOne as part of the Advanced Options in each integration configuration.

#### 4.2.5 2e - Network Resources and Configurations

You may have a number of questions to ask or confirm about your network resources and their configurations. Here are a few examples.

Let's start with finding a few network resources and their connections:

```
Find (Gateway|Firewall) with category='network'
  that relates to *
  return tree
```

Keep in mind you can toggle the result back to **Table** view if you'd like.

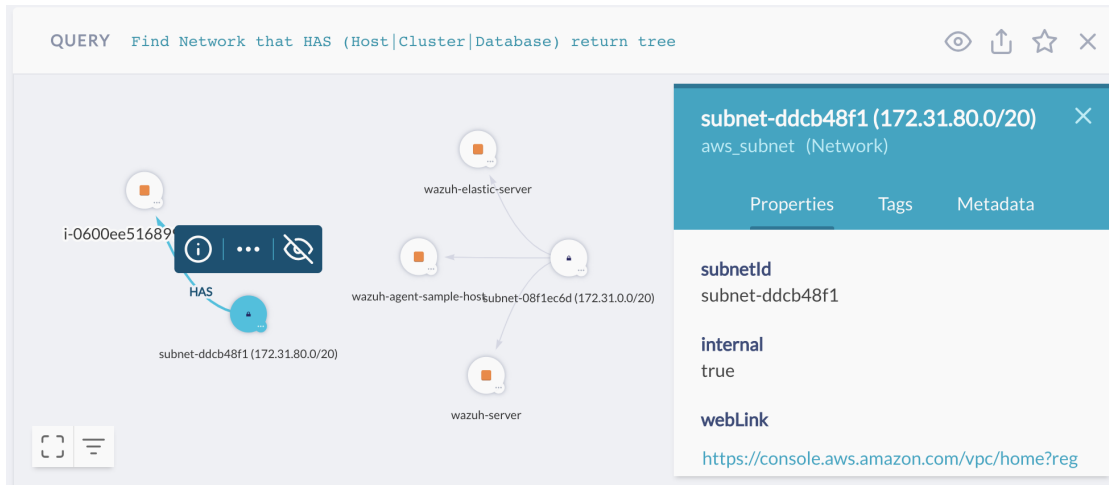
How about networks and subnets?

```
Find Network that contains Network return tree
```

Or resources in a VPC:

```
Find Network that has (Host|Cluster|Database) return tree
```

The result looks like this (you may have a lot more going on than what's shown here from the demo environment):

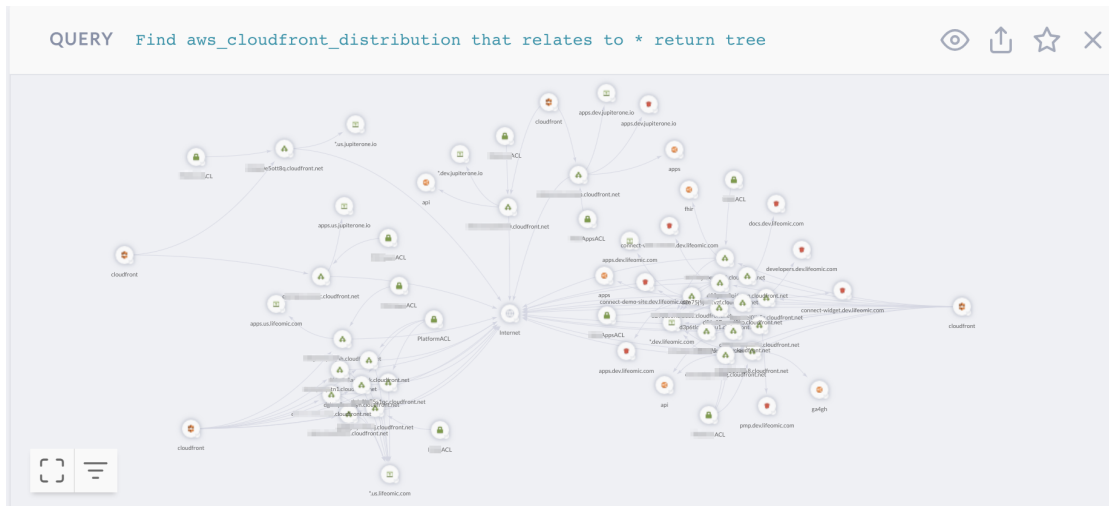


Note that the properties panel for the selected `aws_subnet` has a `webLink` that will allow you to quickly get to the source directly in the AWS web console.

In AWS, you most likely have set up **CloudFront distributions** to distribute traffic to your API Gateways or static websites hosted in S3. What does that look like?

```
Find aws_cloudfront_distribution that relates to * return tree
```

Here, the result looks a little busier, from a J1 account with multiple AWS integration configurations and quite a few `aws_cloudfront_distribution` entities and relationships.



This graph shows you the **origins** connected to the **distributions**: both **S3 buckets** (for static website/contents) and **API Gateways**. Additionally, the graph shows you the **ACM Certificate** being used by them and the **WAF ACL**, if any, configured to protect them.

Keep in mind you can select any entity node in the graph to inspect its detailed properties, or find a web link to quick get to the source in AWS web console.

If you use **AWS Transfer for SFTP**, you can find the **Transfer Servers**, **Users**, which **IAM Roles** are assigned to them, and which **S3 Buckets** the users have access to.

```
Find aws_account
  that HAS aws_transfer
  that HAS Host
  that HAS User
  that RELATES TO *
  return tree
```

You'll get a visual that looks like this:



## 4.2.6 2f - Serverless Functions

Are you using serverless (lambda functions)? If you are, here are a few things that may help you see how they are set up.

Let's start with a listing of your lambda functions:

```
Find aws_lambda_function
```

Simple. Now, what triggers each function?

```
find aws_lambda_function as function
  that TRIGGERS * as trigger
  return
    trigger._type, trigger.displayName, trigger.arn, trigger.webLink, function.
    ↪functionName, function.arn, function.webLink
```

Are there lambda functions with access to resources in a VPC?

```
Find aws_lambda_function that has aws_vpc return tree
```

The above query will give you a visual graph of the lambda functions and the VPC they are configured to run inside.

It is actually a best practice to **not** run lambda functions without access to a VPC unless they need direct access to resources within one – for example, EC2 instances, RDS databases, or Elastic-Search/ElastiCache.

**Is inbound SSH allowed directly from an external host or network?**

```
Find Firewall as fw
  that ALLOWS as rule (Host|Network)
    with internal=false or internal=undefined as src
  where rule.ingress=true and (rule.fromPort<=22 and rule.toPort>=22)
  return
    fw._type,
    fw.displayName,
    rule.fromPort,
    rule.toPort,
    src.displayName,
    src.ipAddress,
    src.CIDR
```

Notice the above query uses `where` to filter the property values of the relationship. You can use both `with` and `where` to filter property values of entities. See the [full JIQL documentation](#) for more details.

Also keep in mind you can toggle to **Graph View** to see the above results more visually and interactively.

### What production resources are directly connected/exposed to the Internet/everyone?

```
Find (Internet|Everyone)
  that relates to *
    with tag.Production=true and _class!='Firewall' and _class!='Gateway'
  return tree
```

### What are my network layer resources?

```
Find (Firewall | Gateway) with category='network'
```

### What about Security Group protection?

```
Find aws_security_group that PROTECTS aws_instance return tree
```

Pro Tip: selecting an edge in the graph to see the security group rule details (i.e. properties on the edge)

## 4.3 Part 3 - User and Access Analysis

Once you have an Okta or OneLogin integration configured, try some of these example queries yourself.

### 4.3.1 3a - IdP users and access

*Examples in this section require an identity provider integration (Okta or OneLogin)*

#### Are there system accounts do not belong to an individual employee/user?

```
Find User that !is Person
```

User entities in JupiterOne are automatically mapped to a corresponding `Person` (`_type: 'employee'`) entity, when there is at least one Identity Provider (IdP) integration configuration - such as Okta or OneLogin.

Pro Tip 1: Set the `userType` property of the user profile in your IdP account to `'system'` or `'generic'` or `'bot'` will prevent JupiterOne from creating a `Person` entity for that user.

**Pro Tip 2:** Set the username of your `aws_iam_user` or other non-IdP users to be the email address of a Person / employee will allow JupiterOne to automatically map that User to its corresponding Person. Alternatively, you can add an email tag to your `aws_iam_user` for the mapping to work.

#### Which active user accounts do not have multi-factor authentication enabled?

```
Find User with active = true and mfaEnabled != true
that !(ASSIGNED|USES|HAS) mfa_device
```

Depending on the specific IdP integration, a User entity may have a relationship mapping to an `mfa_device` instead of the `mfaEnabled` flag directly as a property.

Therefore, the above query finds all User entities with the `active` flag but not the `mfaEnabled` flag set to true on its properties, and additionally, checks for the existence of an relationship between that User and any `mfa_device` assigned or in use.

#### Are there users accessing my 'AWS' application without using MFA?

```
Find User with active = true and mfaEnabled != true
that ASSIGNED Application with displayName = 'Amazon Web Services'
```

Replace the string value of the `displayName` to check for another application.

You can also use `shortName = 'aws'`, which will check for all AWS application instances, if you have more than one AWS SAML app configured with your IdP.

#### Find all contractors and external users in the environment.

```
Find User that IS Person that !EMPLOYS Root
```

The above query finds user accounts belong to any individual not directly employed by your organization (Root entity).

```
Find User as u that IS Person as p
where u.userType='contractor' or p.employeeType='contractor'
```

The above query finds contractor users.

### 4.3.2 3b - Cloud users and access

*Examples in this section require at least one AWS integration configuration.*

#### Who has been assigned full Administrator access in AWS?

```
find (aws_iam_role|aws_iam_user|aws_iam_group)
that ASSIGNED AccessPolicy with policyName='AdministratorAccess'
```

#### Which IAM roles are assigned which IAM policies?

```
find aws_iam_role as role
that ASSIGNED AccessPolicy as policy
return
  role._type as RoleType,
  role.roleName as RoleName,
  policy._type as PolicyType,
  policy.policyName as PolicyName
```

### 4.3.3 3c - Combined users and access across all environments

Examples in this section work best when there are both IdP and AWS integration configurations enabled in JupiterOne.

#### Who has access to what systems/resources?

```
Find (User|Person) as u
  that (ASSIGNED|TRUSTS|HAS|OWNS)
    (Application|AccessPolicy|AccessRole|Account|Device|Host) as a
  return
    u.displayName, u._type, u.username, u.email,
    a._type, a.displayName, a.tag.AccountName
  order by u.displayName
```

## 4.4 Part 4 - Cross Account Analysis

Many examples in this section requires both Okta and AWS integration configurations in JupiterOne, as well as an AWS SAML app configured in your Okta account. Some queries work best when you have multiple AWS configurations.

#### Who has access to my AWS accounts via single sign on (SSO)?

```
Find User as U
  that ASSIGNED Application as App
  that CONNECTS aws_account as AWS
  return
    U.displayName as User,
    App.tag.AccountName as IdP,
    App.displayName as ssoApplication,
    App.signOnMode as signOnMode,
    AWS.name as awsAccount
```

#### Are there assume role trusts from one AWS account to other external entities?

```
Find aws_account
  that HAS aws_iam
  that HAS aws_iam_role
  that TRUSTS (Account|AccessRole|User|UserGroup) with _source='system-mapper'
  return tree
```

Note from the above query, `_source='system-mapper'` is an indicator that the trusted entity is not one ingested by an integration configuration, rather, mapped and created by JupiterOne during the analysis of Assume Role policies of the IAM roles in your account(s). Therefore, these entities are most likely external.

For example, you will most definitely see the JupiterOne integration IAM role with a TRUSTS relationship to the JupiterOne AWS account.

## 4.5 Part 5 - Endpoint Compliance

Examples in this section require the activation of at least one JupiterOne Endpoint Compliance Agent - powered by Stethoscope app.

#### Do I have local firewall enabled on end-user devices?



```
Find HostAgent as agent
  that MONITORS user_endpoint as device
return
  device.displayName,
  device.platform,
  device.osVersion,
  device.hardwareModel,
  device.owner,
  agent.firewall,
  agent.compliant,
  agent._type,
  agent.displayName
```

### Whose endpoints are non-compliant?

```
Find Person as person
  that OWNS (Host|Device) as device
  that MONITORS HostAgent with compliant!=true as agent
return
  person.displayName,
  person.email,
  device.displayName,
  device.platform,
  device.osVersion,
  device.hardwareModel,
  device.owner,
  agent.compliant,
  agent._type,
  agent.displayName
```

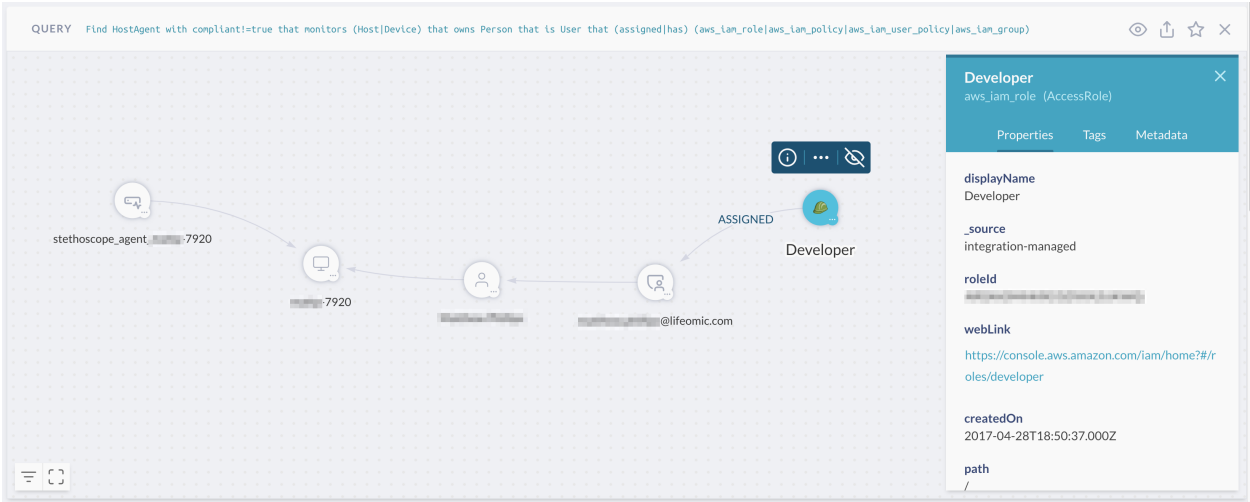
### What applications do those users have access to?

```
Find HostAgent with compliant!=true
  that MONITORS (Host|Device)
  that OWNS Person
  that IS User
  that Assigned Application
return tree
```

### Out of those above, any of them have access to AWS?

```
Find HostAgent with compliant!=true
  that MONITORS (Host|Device)
  that OWNS Person
  that IS User
  that (ASSIGNED|HAS) (aws_iam_role|aws_iam_policy|aws_iam_user_policy|aws_iam_group)
return tree
```

The resulting graph may look like this:



device-aws-access

noncompliance-

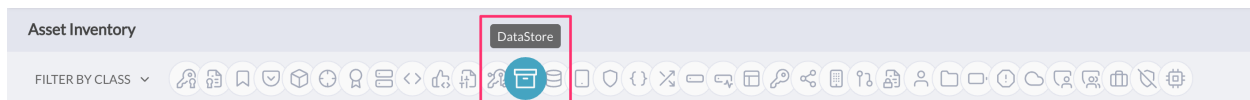
## How to use filters in the Asset Inventory app

There are two ways to filter the thousands of digital assets (i.e. Entities) you may have from the Asset Inventory app:

- **Quick Filters by Class and/or Type**
- **Granular Filters by Properties**

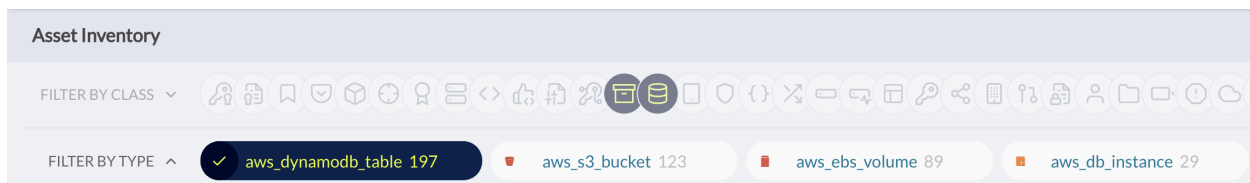
### 5.1 Quick Filters by Class and/or Type

You can quickly filter the entities/assets by **Class**, by selecting one or multiple of the icons that represent each class. The tooltip displays the class label when you move over it:



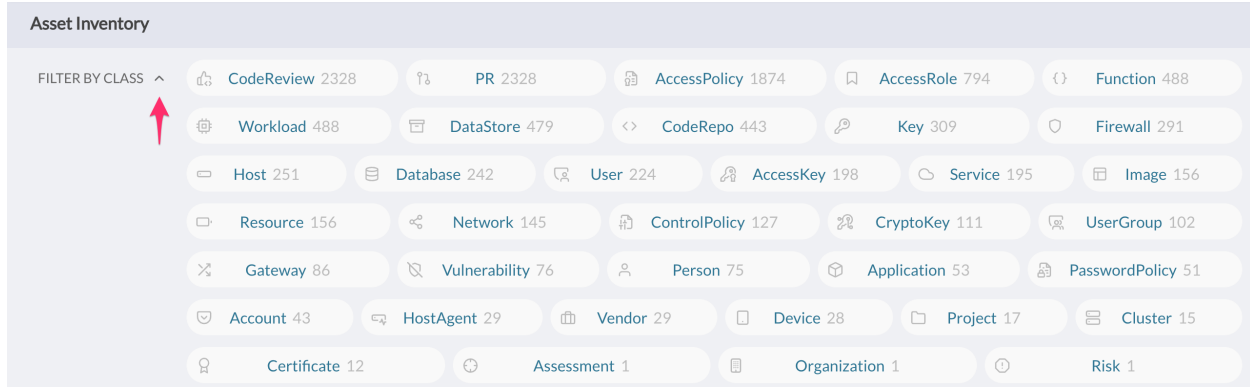
*The **Class** of an entity is an abstract label that defines what the entity is within the concept of security operations. For more details, see the [JupiterOne Data Model documentation](#).*

Once you select one or more class, you can further filter the entities/assets by **Type**:

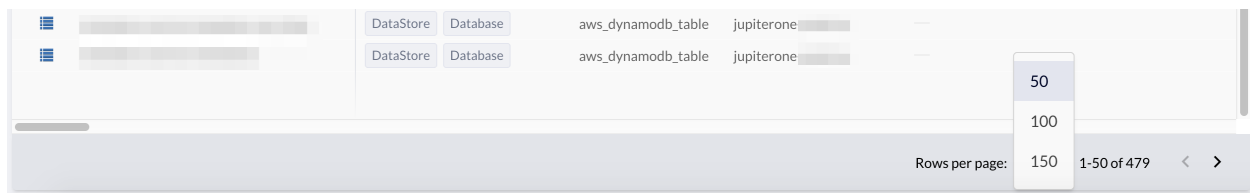


*The **Type** of an entity represents the specific type of entity as defined by its source. For more details, see the [JupiterOne Data Model documentation](#).*

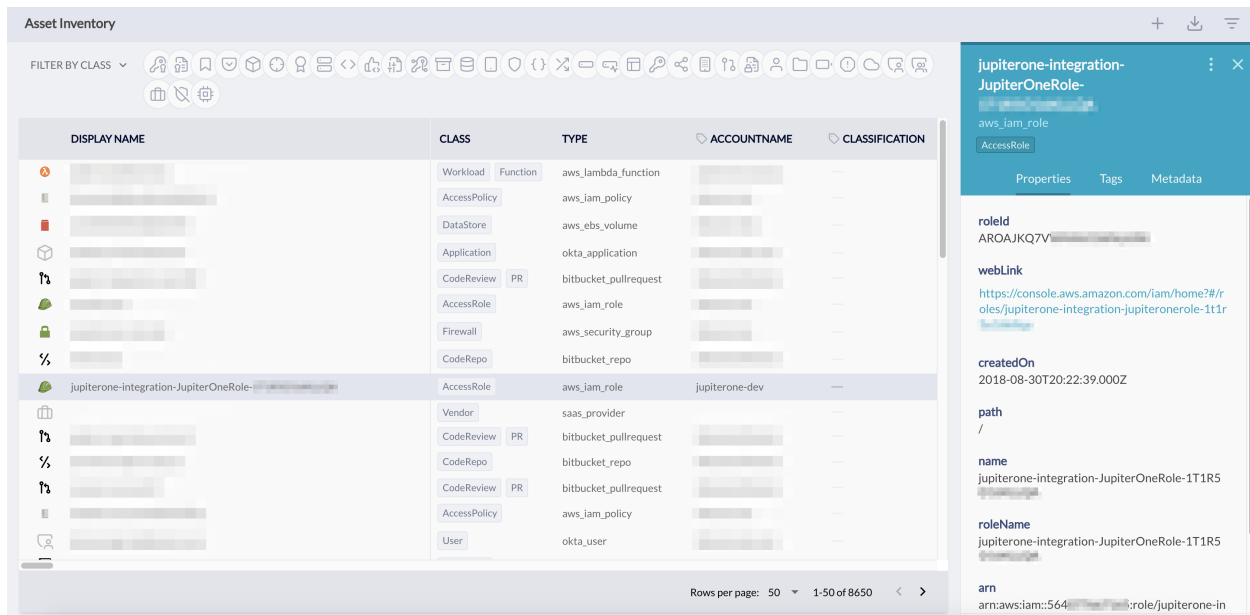
You can also expand the Class filter to get a more detailed, dashboard-like view of the entites/assets with a count for each class.



The data will respond correspondingly to the selection in the table below the quick filters. Note the pagination control at the bottom of the table:



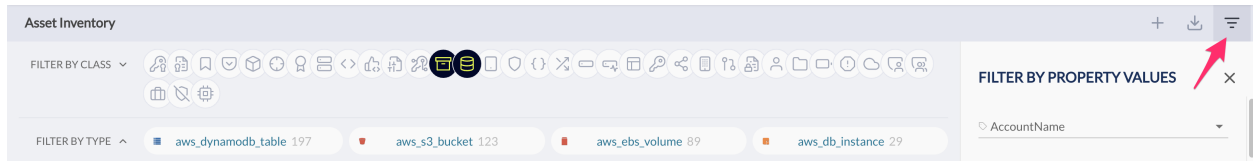
Selecting an entity in the table will bring up its detailed properties in a side panel on the right.






## 5.2 Granular Filters by Properties


You can apply granular filters by selecting specific property values.

Open up the **Filter Panel** using the control icon near the top right corner:



Look for the property or properties you'd like to filter on to select one or multiple values to apply the filter. Clicking on a previously selected value from the property dropdown box will unselect it.



**FILTER BY PROPERTY VALUES** 

bucketName

classification

critical

141 internal

62 critical

12 public

10 Internal

9 confidential

createdOn

databaseName

defaultEncryptionEnabled

displayName

**Tips:** We recommend selecting Class/Type using the quick filter first, before apply more granular property filters. This will reduce the number of properties/values and make it a lot easier for selection.





JupiterOne allows you to configure alert rules using any J1QL query for continuous auditing and threat monitoring. This is done in the **Alerts** app.

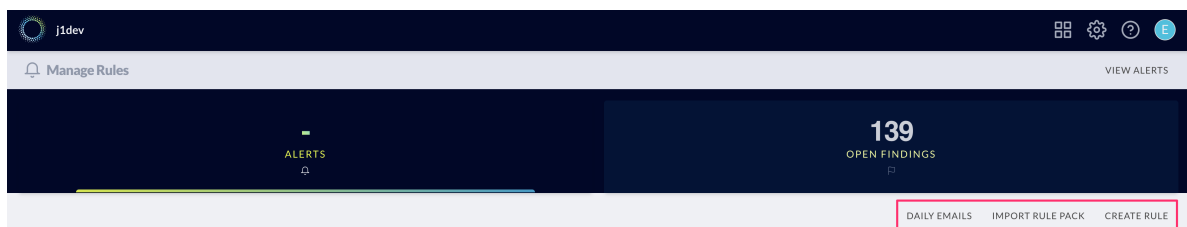
## 6.1 Import Alert Rules from Rule Pack

You will need to have at least one active alert rule to trigger any alert. The easiest way to add some rules is to import rule packs, following these steps:

1. Go to **Manage Rules** from the Alerts app



2. Click **Import Rule Pack** action button



3. This will bring up the **Import Rules from Rule Pack** modal window, where you can select the rule packs or individual rules within a rule pack. Click **Save** to import the selected rules.

Import Rules From Rule Pack

IMPORT YOUR OWN RULE PACK (JSON) X

☒ AwsConfig

▼

☒ AwsThreat

▼

☒ CommonAlerts

^

☒ high-severity-finding - Findings with a severity of High or a numeric severity rating higher than 7 that were new within the last 24 hours.

☒ prod-resources-with-high-severity-finding - Production resources impacted by high severity findings

☒ prod-resources-with-critical-finding - Production resources impacted by critical findings

☒ unencrypted-critical-data-stores - Unencrypted data store with classification label of 'critical' or 'sensitive' or 'confidential' or 'restricted'

CANCEL

SAVE

## 6.2 Create Custom Alert Rules

Creating your own custom alert rule is easy:

1. Go to **Manage Rules** from the Alerts app
2. Click **Create Rule** action button to bring up the modal window
3. Enter the following details for the custom rule and hit **SAVE**:
  - **Name**
  - **Description**
  - **Severity** (select from drop down list)
  - **Query** (any J1QL query)

Create Rule

SHOW ADVANCED X

Name

Severity

CRITICAL

Description

Query

```
Find DataStore with classification='critical' and unencrypted=true as d return d.tag.AccountName as Account, d.displayName as UnencryptedDataStores, d._type as Type, d.encrypted as Encrypted
```

CANCEL

SAVE

The custom rule will be added and be evaluated daily. If the query you have specified in the rule returns at least one match, it will trigger an alert.

## 6.3 Managing Alerts

The alert rules are evaluated *daily* by default, or at the custom interval – *hourly* or *every 30 minutes* – you have specified for a specific rule.

Active alerts that matched the evaluation criteria of the alert rules will show up in the **Alerts** app in a data grid that looks like this:

TYPE	SEVERITY	ALERT TITLE / MESSAGE	COUNT	LAST ALERTED ON	
Alert	HIGH	s3-bucket-replication-enabled S3 buckets should enable cross-region replication.		05/3/19 7:01 am	DISMISS
Alert	HIGH	config-rule-noncompliant AWS Config rule evaluation found non-compliant resource configurations.		05/3/19 7:01 am	DISMISS
Alert	INFO	acm-certificate-expiration-check ACM certificate set to expire within 30 days.		05/3/19 6:58 am	DISMISS
Alert	HIGH	s3-bucket-versioning-enabled S3 buckets should enable versioning.		05/3/19 6:55 am	DISMISS
Alert	MEDIUM	unclassified-data-stores Data stores without a classification tag assigned		05/3/19 6:48 am	DISMISS
Alert	HIGH	s3-bucket-replication-enabled S3 buckets should enable cross-region replication.		05/2/19 7:01 am	DISMISS

- Click on an individual alert row will expand it to show the alert details.
- Click on the **DISMISS** button to dismiss an alert.

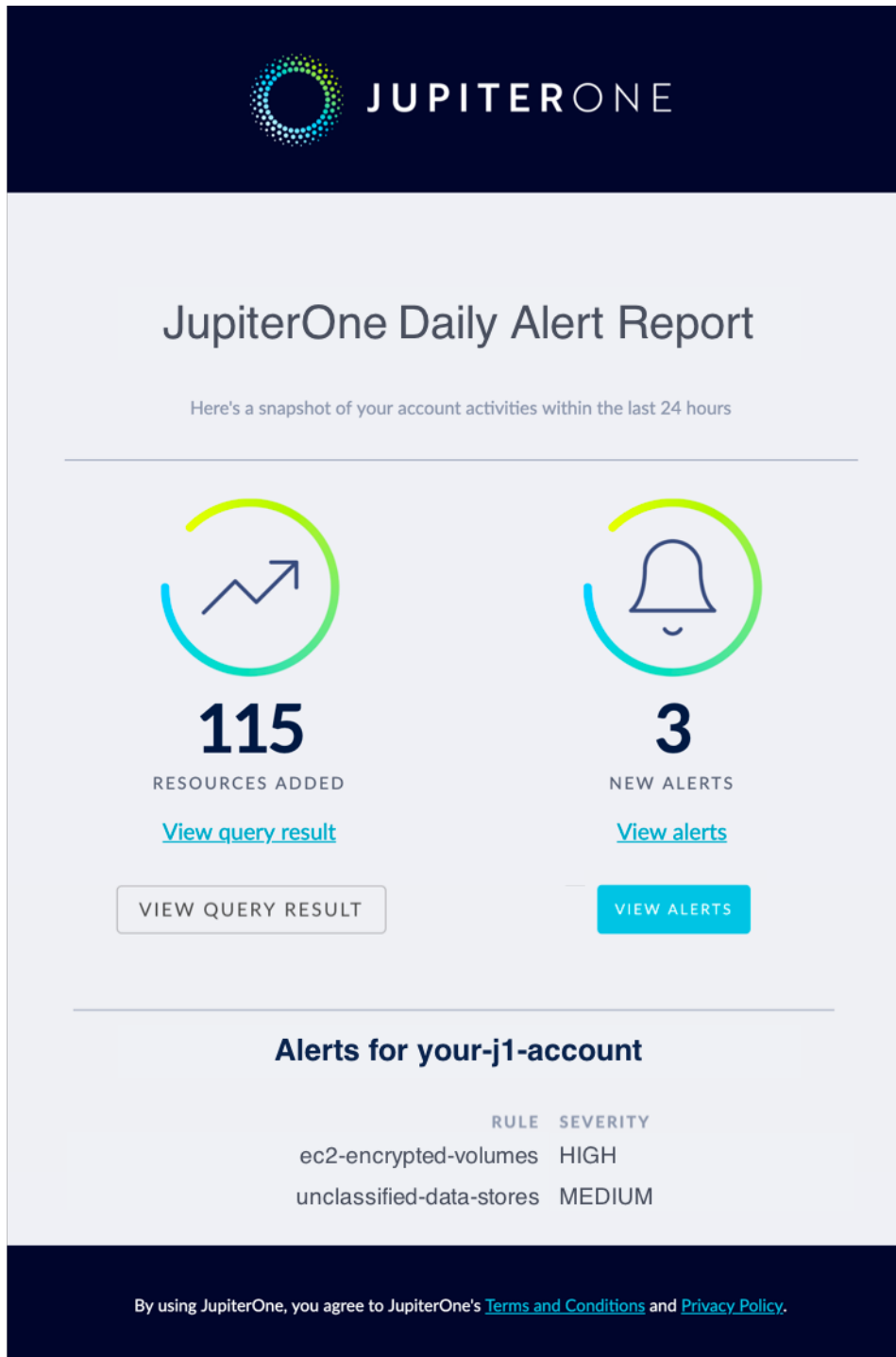
## 6.4 Configure Daily Notification Email

To receive daily notification of new/active alerts, select:

- Manage Rules**
- Daily Emails**

- Enter the email addresses of the users or teams in the **Recipients** field

Your **JupiterOne Daily Alert Report** will look like this:



To ensure delivery of these reports, please whitelist `@jupiterone.io` and `@us.jupiterone.io` in your email configuration.

JupiterOne provides a centralized repository and dashboard to let you easily manage security findings from different sources, including:

- AWS Inspector findings
- AWS GuardDuty findings
- Veracode static and dynamic analysis findings
- WhiteHat application security findings
- Tenable Cloud scanning findings
- HackerOne report findings
- CVEs and other vulnerability findings
- Manual penetration testing findings (imported via API - see [this guide](#))

*More vulnerability scanner integrations are being added. Current roadmap includes: Rapid7, Qualys, Bugcrowd, White Source, Source Clear, and Snyk.*

## 7.1 Managing Findings

Consolidated findings can be accessed in the **Alerts** app, under the **Findings** tab. The header tab shows a total count of currently open findings. Selecting it will bring you to the detailed findings view:

GETTING STARTED

Findings

57 ALERTS

216 OPEN FINDINGS

Type	Severity	Finding Title / Description	Created On
aws_inspector_finding	CRITICAL	Instance i-0244e47d8e801312d is not compliant with rule 4.1... Description Monitor changes to file permissions, attributes, ownership and group. The parameters in this section track changes for system calls that affect file permissions and attributes. The chmod, fchmod and fchmodat system calls affect the permissions associated with a file. The chown, fchown, fchownat and lchown system calls affect owner and group attributes on a file. The setxattr, lsetxattr, fsetxattr (set extended file attributes) and removexattr, lremovexattr, fremovexattr (remove extended file attributes) control extended file attributes. In all cases, an audit record will only be written for non-system user ids (audit >= 1000) and will ignore Daemon events (audit = 4294967295). All audit records will be tagged with the identifier "perm_mod." Rationale Monitoring for changes in file attributes could alert a system administrator to activity that could indicate intruder activity or policy violation.	04/11/19 7:53 pm
aws_inspector_finding	CRITICAL	Instance i-0244e47d8e801312d is not compliant with rule 4.2... Description Log files stored in /var/log/ contain logged information from many services on the system, or on log hosts others as well. Rationale It is important to ensure that log files have the correct permissions to ensure that sensitive data is archived and protected.	04/11/19 7:53 pm
aws_guardduty_finding	MEDIUM	Outbound portscan from EC2 instance i-031f61bc80ca33ceb. EC2 instance i-031f61bc80ca33ceb is performing outbound port scans against remote host 10.51.17.149.	04/20/19 2:25 am
aws_guardduty_finding	MEDIUM	Outbound portscan from EC2 instance i-09a81dbcd8d99ebb2. EC2 instance i-09a81dbcd8d99ebb2 is performing outbound port scans against remote host 10.51.12.142.	04/1/19 12:28 pm

JupiterOne will automatically map resources impacted by or related to each finding based on the available attributes from the finding source.

Selecting a finding from the list will show you a graph of those relationships. This allows you to visualize the context to further analyze the finding's impact and to determine a course of action for remediation.

TYPE	SEVERITY	FINDING TITLE / DESCRIPTION	CREATED ON
aws_inspector_finding	CRITICAL	Instance i-0244e47d8e801312d is not compliant with rule 5.2... Description The PermitUserEnvironment option allows users to present environment options to the ssh daemon. Rationale Permitting users the ability to set environment variables through the SSH daemon could potentially allow users to bypass security controls (e.g. setting an execution path that has ssh executing trojan'd programs)	04/11/19 9:54 pm

QUERY Find Finding with \_key="z0RksU00LiLb5F6R5SWTMD8V2n4zsqhVR8sv8uQhLE=" that relates to \* return tree

wazuh-elastic-server

aws\_instance (Host)

Properties Tags Metadata

displayName

wazuh-elastic-server

\_source

integration-managed

subnetid

subnet-08f1ec6d

virtualizationType

hvm

On instance i-0244e47d8e801312d, TCP port 9200 which is associat...  
On this instance, TCP port 9200, which is associated with Elasticsearch, is reachable from the internet with no process listening. The instance i-0244e47d8e801312d is located in VPC vpc-43fe713a and has an attached ENI eni-0157f38c02ff051db which uses network ACL acl-9eab9fe7. The port is reachable from the internet through Security Group sg-008200348cb9961b2 and IGW igw-bafcd8dc

## 7.2 Create Alerts for Findings

You can create custom alert rules to notify you on certain findings, using JIQL to filter and correlate.

### 7.2.1 Examples:

The following three rules are included in the J1 **Common Alerts** Rule Pack:

- **high-severity-finding**

*Alerts on Findings with a severity of High or a numeric severity rating higher than 7 that were new within the last 24 hours.*

```
Find Finding with
  (severity='High' or severity='high' or numericSeverity>7) and
  _createdOn > date.now-24hours
```

- **prod-resources-with-high-severity-finding**

*Alerts when Production resources are impacted by high severity findings.*

```
Find (Host|DataStore|Application|CodeRepo|Account|Service|Network)
  with tag.Production=true
  that has Finding with severity=('High' or 'high') or numericSeverity=(7 or 8)
```

- **prod-resources-with-critical-finding**

*Alerts when Production resources are impacted by critical findings.*

```
Find (Host|DataStore|Application|CodeRepo|Account|Service|Network)
  with tag.Production=true
  that has Finding with severity=('Critical' or 'critical') or numericSeverity=(9
  ↪or 10)
```

The following rule is included in the J1 **AWS Threat** Rule Pack:

- **aws-guardduty-inspector-finding-instance-correlation**

*Identifies vulnerable EC2 instances (i.e. with medium or higher rated open Inspector finding) that are also targets of suspicious activities (i.e. with medium or higher rated open GuardDuty finding).*

```
Find aws_guardduty_finding with numericSeverity>5 and open=true as guardduty
  that relates to aws_instance as i
  that has aws_inspector_finding with numericSeverity>5 and open=true as inspector
  return i.*, guardduty.*, inspector.*
```

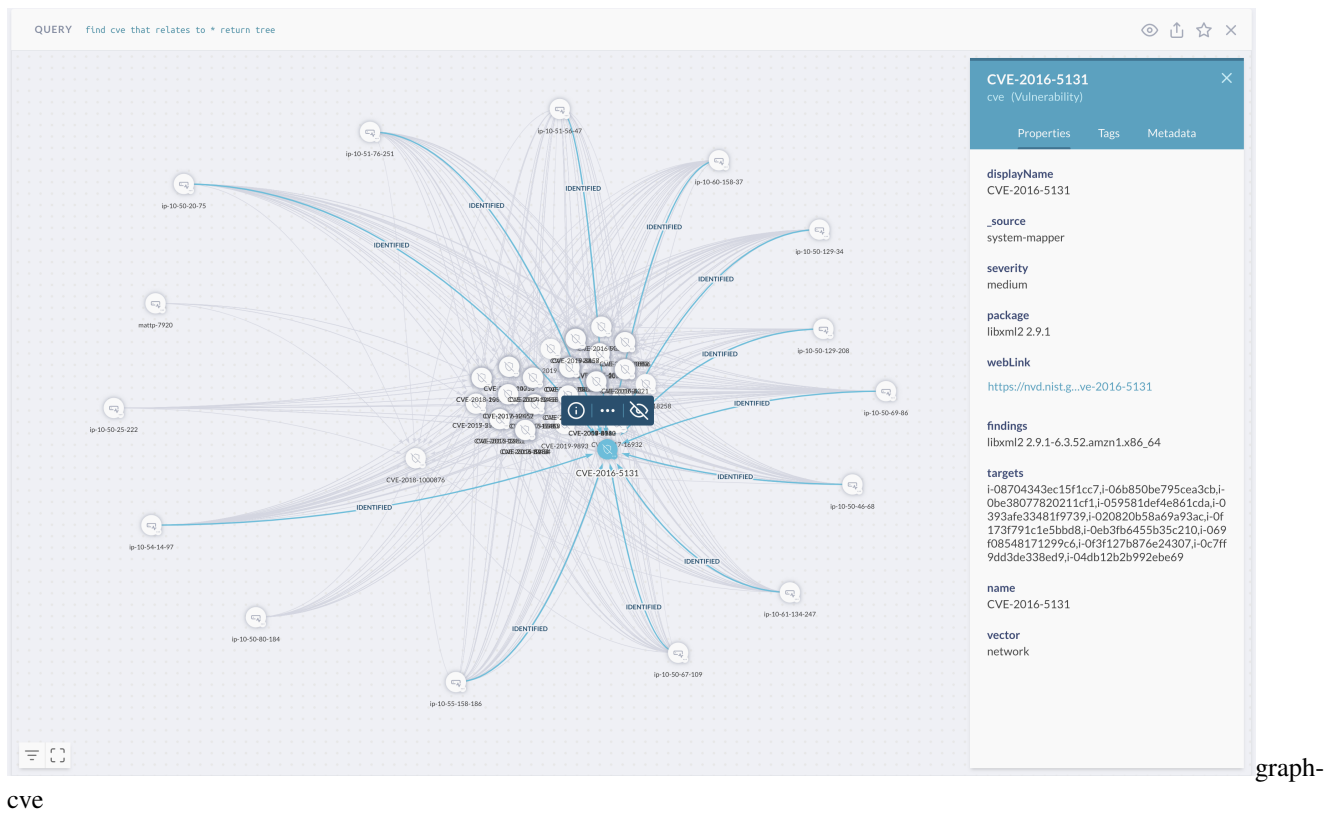
## 7.3 Visualizing Findings with J1 Query and Graph

You can execute JIQL queries to generate graph visualizations that help you analyze the relationships among findings, the agents/scanners/services that identified them, and the resources they impact.

Here's an example:

```
Find cve that relates to (Host|HostAgent) with active=true return tree
```

This will give you a visual like this (you may need to move the nodes around to adjust their positioning):





---

### Frequently Asked Questions

---

#### 8.1 How do I get my custom / on-premise data into JupiterOne?

JupiterOne’s asset inventory, search and visualization supports any data imported that follows the *reference data model*, not limited to data ingested by managed integrations.

This is easily done via the *API or CLI*. Each entity object can be represented in a few lines of JSON or YAML code. The *J1 API Client or CLI* can create/update them to your JupiterOne account. You can also develop a script to run on a schedule (e.g. via a cron job) or integrate into your DevOps automation.

#### 8.2 Where do these `Person` entities come from? Why are they not tagged with an integration?

The `Person` entities are “mapped” from `User` entities. They are considered “shared” entities that multiple integrations can map to and contribute properties to. For example, a `Person` can be created by a Google integration (from a `google_user`). Or from a Github User, AWS IAM User, etc.

The `Person` entities represent actual individuals in the organizations, whereas the `User` entities are the logical user accounts within each digital environment/account (i.e. from an integration).

#### 8.3 How do I add custom properties to my AWS entities from the source?

You can add custom properties by tagging your AWS resources. AWS supports tags for most resources. All tags will be ingested as entity properties. Each tag will have a prefix `tag_` followed by the tag name as the entity property name.

You can then build queries using these tag properties. For example:

```
Find aws_instance with tag.Environment='staging'
```

## 8.4 Some AWS resources seem to be missing from the Asset Inventory / Graph. What is going on?

This is most commonly caused by incorrect or insufficient permissions. Check the IAM policy assigned to the IAM role used by JupiterOne in your AWS account. You can find details on the required permissions by going to

**Integrations Configuration > Add AWS Configuration >** and clicking on the **Setup Instructions** button.

Or they can be found on the [jupiterone-aws-integration](#) project on Github.

## 8.5 I have a Network marked as “public”, what does that mean?

The `public` property on a Network entity means the network is publicly accessible. A publicly accessible network could be either internal or external. There is an `internal` property to indicate whether that is the case.

A network that is not an entity ingested from an integration is determined to be potentially an external network, with `internal=undefined`. When such a network (or host) has a public IP address or CIDR, it is set to be `public=true`.

An internal network - that is, a Network entity ingested from an integration, such as an `aws_subnet` or `aws_vpc` - is set to `internal=true`. An internal network may be determined to be publicly accessible by the integration with certain conditions that are specific to each type of integration.

## 8.6 How is it determined if an AWS VPC or Subnet is public?

An `aws_vpc` or `aws_subnet` is determined to be publicly accessible – i.e. `public=true` – only when the following conditions are met:

- The VPC has an Internet Gateway that connects it to the Internet
- The VPC or subnet has a Route in the Route Table to external networks
- The VPC or subnet has a Network ACL that allows traffic to/from external networks

## 8.7 How are Person entities (i.e. employees) created?

A `Person` entity is created by the “mapper” process – when a `User` entity is ingested/updated from an identity provider integration (e.g. Okta, OneLogin, Google), a `Person` entity is “mapped” with the user’s information (first and last name, email address, etc.).

## 8.8 How can I avoid creating a Person entity for a generic/system user account?

Certain properties are used to determine if the user is a system user or an actual individual. This depends on the integration.

For **Okta**, you can set the `userType` property for the user to one of the following to avoid it being mapped to a `Person`:

- `bot`
- `generic`
- `service`
- `system`

## 8.9 I see a user named “Callisto” on my account. Who is that?

“Callisto [callisto@jupiterone.io](mailto:callisto@jupiterone.io)” is the account for JupiterOne Support. The Support User is by default added to a new account during free trial, proof-of-concept evaluation, or initial account onboarding. This is to facilitate better support and training on using the platform.

- The support user can be removed by an account administrator at any time, should you determine that ongoing regular support is no longer needed.
- You have the option and administrative privilege to add the support user back at any time, when support is needed in the future.

## 8.10 Endpoint compliance data isn’t appearing as expected. How can I troubleshoot this?

For the Stethoscope-powered compliance data to report successfully:

- Endpoint devices must be running Stethoscope-app.
- Endpoint devices must be running the powerup agent.
- The powerup agent must be successfully activated.

Stethoscope-app and the powerup agent are both installed via the same powerup installation bundle. You should check to see that the Stethoscope giraffe icon is present in the device’s system tray. If it is not, you will need to launch Stethoscope-app. You can verify that the powerup agent is running as a system service by checking for its process. On MacOS or Linux, issue a command like `ps -ef | grep jl-endpoint-agent` to verify that there is indeed an agent service running.

To verify that the powerup agent is successfully activated, you can perform a manual scan from within a shell terminal:

```
/opt/jlendpointagent/bin/jl-endpoint-agent scan --verbose
```

The output of this scan will indicate success if your agent has been successfully activated and can communicate securely with JupiterOne. If this is unsuccessful, you should resend an activation email to this device from the JupiterOne Powerup administration UI for Endpoint Compliance, and perform the activation step as indicated in the email. If you suspect there may have been a problem with installation, or if other errors persist, please try re-downloading and executing the installation script, and performing the activation step.

Additional diagnostic information may be available in the device’s system log. You can search this for “jupiterone” or “ECA:” to quickly filter the results.

## 8.11 How do I search/filter on all AWS entities without enumerating all types?

For example, you may want to identify if a certain tag is present across all entities from AWS. You can do this by using the special metadata `_integrationName`, like this:

```
Find * with _integrationName="AWS" and tag.ABC=undefined
```

You may also want to limit this query to filter out irrelevant entities by class. For example:

```
Find * with
  _integrationName="AWS" and
  tag.ABC=undefined and
  _class!='Finding'
```

---

## J1 Queries for AWS Config

---

AWS Config is a service provided by AWS that can be used to evaluate the configuration settings of your AWS resources. This is typically done by enabling AWS Config rules in one or multiple of your AWS accounts to represent your ideal configuration settings.

There are a few downsides of AWS Config:

- It can easily cost \$500 to \$1000+ per account per month depending on the number of resources in the account and number of rules you have configured.
- It is hard to fine tune AWS Config rules with the tags and other contextual data to reduce false positives.
- Setting up alerts and notifications requires additional configuration using SNS or CloudWatch.

Fortunately, almost all of evaluation of AWS Config rules can be done by a simple J1QL query/alert in JupiterOne. Additionally, each query can easily be modified to include tags or even relationship context.

Here are some examples.

### 9.1 ACM Rules

#### **acm-certificate-expiration-check**

Ensures ACM Certificates in your account are marked for expiration within the specified number of days.

```
Find aws_acm_certificate with expiresOn < date.now + 30days
```

OR

```
Find Certificate with expiresOn < date.now + 30days
```

### 9.2 EC2 Rules

#### **ec2-instances-in-vpc**

Ensure all EC2 instances run in a VPC.

```
Find aws_instance with vpcId=undefined
```

### ec2-volume-inuse-check

Checks whether EBS volumes are attached to EC2 instances.

```
Find aws_ebs_volume that !USES aws_instance
```

### encrypted-volumes

Checks whether the EBS volumes that are in an attached state are encrypted.

```
Find aws_instance as i that USES aws_ebs_volume with encrypted!=true as v
return
    i.tag.AccountName, i.name, i.instanceId, i.state, i.region, i.webLink,
    v.volumeId, v.encrypted, v.webLink
```

### restricted-ssh

Checks whether the incoming SSH traffic for the security groups is accessible.

*With AWS Config, this rule is compliant when the IP addresses of the incoming SSH traffic in the security groups are restricted. This rule applies only to IPv4.*

```
Find aws_security_group as sg that ALLOWS as rule * as src
where
    rule.ingress=true and rule.ipProtocol='tcp' and
    (rule.fromPort<=22 and rule.toPort>=22)
return
    sg.displayName,
    rule.ipProtocol, rule.fromPort, rule.toPort,
    src.displayName, src.ipAddress, src.CIDR
```

## 9.3 IAM Rules

### iam-root-access-key-check

Ensure root AWS account has MFA enabled.

```
Find aws_account with _source!='system-mapper' and AccountMFAEnabled!=1
```

### iam-password-policy

Ensure the account password policy for IAM users meets the specified requirements.

```
Find aws_iam_account_password_policy with
    requireUppercase != true or
    requireLowercase != true or
    requireSymbols != true or
    requireNumbers != true or
    minLength < 8 or
    maxAgeDays > 90 or
    historyCount < 12
```

*Adjust the above values to match your organization policy. You can also separate each into its own query.*

**iam-user-no-policies-check**

Ensure that none of your IAM users have policies attached.

*IAM users should inherit permissions from IAM groups or roles.*

```
Find aws_iam_user that assigned (aws_iam_user_policy|aws_iam_policy)
```

The `aws_iam_user_policy` in the above query specifies an inline policy whereas the `aws_iam_policy` is a managed policy.

## 9.4 Lambda Rules

**lambda-function-public-access-prohibited**

Checks whether the AWS Lambda function policy attached to the Lambda resource prohibits public access.

**lambda-function-settings-check**

```
Find aws_lambda_function with runtime='nodejs6.10'

Find aws_lambda_function with timeout < 5 or timeout > 300

Find aws_lambda_function with memorySize <= 128 or memorySize >= 1024

Find aws_lambda_function with role = '<role_arn>'
```

You can of course adjust any of the values in the above example queries.

**Note that `nodejs6.10` is End-of-Life (EOL) as of April 2019.** The first query above is an excellent check to ensure you have migrated all of your lambda functions to `nodejs8.10`.

## 9.5 RDS Rules

**db-instance-backup-enabled**

Checks whether RDS DB instances have backups enabled.

```
Find (aws_db_instance|aws_rds_cluster) with BackupRetentionPeriod=undefined
```

*Optionally, the rule checks the backup retention period and the backup window.*

```
Find (aws_db_instance|aws_rds_cluster) with
  BackupRetentionPeriod=undefined or
  BackupRetentionPeriod<30
```

**rds-snapshots-public-prohibited**

Checks if Amazon Relational Database Service (Amazon RDS) snapshots are public. The rule is non-compliant if any existing and new Amazon RDS snapshots are public.

**rds-storage-encrypted**

Checks whether storage encryption is enabled for your RDS DB instances.

```
Find (aws_db_instance|aws_rds_cluster) with encrypted!=true
```

See a visual graph of which RDS cluster/instance is using which KMS key with the following query:

```
Find (aws_db_instance|aws_rds_cluster) that uses aws_kms_key return tree
```

You can easily extend the query to cover other data stores and check for their encryption status across the board:

```
Find DataStore with encrypted!=true
```

The above query covers **RDS instances/clusters, S3 buckets, EBS volumes, DynamoDB tables, Redshift clusters** all at once.

This is often combined with some tagging to reduce false positives. For example:

```
Find DataStore with encrypted!=true and  
  (classification='critical' or classification='confidential')
```

## 9.6 DynamoDB Rules

### dynamodb-throughput-limit-check

Checks whether provisioned DynamoDB throughput is approaching the maximum limit for your account. By default, the rule checks if provisioned throughput exceeds a threshold of 80% of your account limits.

```
n/a
```

## 9.7 S3 Rules

### s3-bucket-public-read-prohibited

Checks that your Amazon S3 buckets do not allow public read access.

```
Find aws_s3_bucket that ALLOWS as grant Everyone where grant.permission='READ'
```

Or, to return certain specific properties about the bucket and the rule:

```
Find aws_s3_bucket as bucket that ALLOWS as grant everyone  
  where grant.permission='READ'  
  return  
    bucket.displayName, bucket.tag.AccountName,  
    grant.permission, grant.granteeType, grant.granteeURI
```

### s3-bucket-public-write-prohibited

Checks that your Amazon S3 buckets do not allow public write access.

```
Find aws_s3_bucket that ALLOWS as grant Everyone where grant.permission='WRITE'
```

### s3-bucket-replication-enabled

Checks whether S3 buckets have cross-region replication enabled.

```
Find aws_s3_bucket with replicationEnabled != true or destinationBuckets = undefined
```

### s3-bucket-server-side-encryption-enabled

Checks whether server side encryption is enabled for your S3 buckets.



```
Find aws_s3_bucket with encrypted=false and defaultEncryptionEnabled=false
```

**s3-bucket-ssl-requests-only**

Checks whether S3 buckets have policies that require requests to use Secure Socket Layer (SSL/TLS).

```
n/a
```

**s3-bucket-logging-enabled**

Checks whether logging is enabled for your S3 buckets.

```
Find aws_s3_bucket with loggingEnabled != true
```

**s3-bucket-versioning-enabled**

Checks whether versioning is enabled for your S3 buckets. Optionally, the rule checks if MFA delete is enabled for your S3 buckets.

```
Find aws_s3_bucket with versioningEnabled != true or mfaDelete != true
```

## 9.8 Other Rules

**cloudtrail-enabled**

Ensure CloudTrail is enabled.

```
Find aws_account that !HAS aws_cloudtrail
```



---

# Using JupiterOne for Active Vulnerability and Threat Monitoring in AWS

---

Active threats within an organization's AWS environments typically arise from these two main sources:

1. System and application vulnerabilities on EC2 instances.
2. Malicious network activities, API activities and resource operations.

AWS provides two native services – [AWS Inspector](#) and [AWS GuardDuty](#) – to address the above, respectively.

Inspector performs automated scans of active EC2 instances to identify exposure and vulnerabilities.

GuardDuty continuously analyzes network events (VPC Flow Logs and DNS logs) and API events (CloudTrail logs) to identify malicious/unauthorized activity and behavior.

JupiterOne integrates with both AWS Inspector and GuardDuty out-of-the-box to provide a consolidated UI to manage, visualize and correlate the findings from these services.

## 10.1 Accessing the Findings in the Alerts app

You can see all **open findings** in the **Alerts** app.

j1dev

GETTING STARTED

Findings

57

ALERTS

216

OPEN FINDINGS

Type	Severity	Finding Title / Description	Created On
aws_inspector_finding	CRITICAL	Instance i-0244e47d8e801312d is not compliant with rule 4.1... Description Monitor changes to file permissions, attributes, ownership and group. The parameters in this section track changes for system calls that affect file permissions and attributes. The chmod , fchmod and fchmodat system calls affect the permissions associated with a file. The chown , fchown , fchownat and lchown system calls affect owner and group attributes on a file. The setattr , lsetattr , fsetattr (set extended file attributes) and removexattr , lremovexattr , fremovexattr (remove extended file attributes) control extended file attributes. In all cases, an audit record will only be written for non-system user ids (auid >= 1000) and will ignore Daemon events (auid = 4294967295). All audit records will be tagged with the identifier "perm_mod." Rationale Monitoring for changes in file attributes could alert a system administrator to activity that could indicate intruder activity or policy violation.	04/11/19 7:53 pm
aws_inspector_finding	CRITICAL	Instance i-0244e47d8e801312d is not compliant with rule 4.2... Description Log files stored in /var/log/ contain logged information from many services on the system, or on log hosts others as well. Rationale It is important to ensure that log files have the correct permissions to ensure that sensitive data is archived and protected.	04/11/19 7:53 pm
aws_guardduty_finding	MEDIUM	Outbound portscan from EC2 instance i-031f61bc80ca33ceb. EC2 instance i-031f61bc80ca33ceb is performing outbound port scans against remote host 10.51.17.149.	04/20/19 2:25 am
aws_guardduty_finding	MEDIUM	Outbound portscan from EC2 instance i-09a81dbcd8d99ebb2. EC2 instance i-09a81dbcd8d99ebb2 is performing outbound port scans against remote host 10.51.12.142.	04/1/19 12:28 pm

findings

Expanding a finding will give you a visual graph showing the resources the selected finding is related to. You can interact with the graph and drill down to see additional relationships and context to perform further analysis.

TYPE	SEVERITY	FINDING TITLE / DESCRIPTION	CREATED ON
aws_inspector_finding	CRITICAL	Instance i-0244e47d8e801312d is not compliant with rule 5.2... Description The PermitUserEnvironment option allows users to present environment options to the ssh daemon. Rationale Permitting users the ability to set environment variables through the SSH daemon could potentially allow users to bypass security controls (e.g. setting an execution path that has ssh executing trojan'd programs)	04/11/19 9:54 pm
<div>QUERY Find Finding with _key="z0RksU001LuLb5F6R5S5WTD0BV2n4zsqhVR8sv8uQWLe=" that relates to * return tree</div> <div><pre>graph LR; F[Instance i-0244e47d8e801312d is not compliant with rule 5.2...] -- HAS --&gt; I[wazuh-elastic-server]; R[Run - Assessment - Template - Default - All - Rules - 2019-04-12T00:52:20.495Z] --&gt; I;</pre></div> <div><div>wazuh-elastic-server</div><div>aws_instance (Host)</div><div>Properties Tags Metadata</div><div>displayNamewazuh-elastic-server</div><div>_sourceintegration-managed</div><div>subnetidsubnet-08f1ec6d</div><div>virtualizationTypehvm</div></div> <div>On Instance i-0244e47d8e801312d, TCP port 9200 which is associat... On this instance, TCP port 9200, which is associated with Elasticsearch, is reachable from the internet with no process listening. The instance i-0244e47d8e801312d is located in VPC vpc-436e713a and has an attached ENI eni-0157f38c02f051db which uses network ACL act-9eab9fe7. The port is reachable from the internet through Security Group sg-008200348cb9961b2 and IGW igw-bafc80dc</div>			
aws_inspector_finding	INFO		04/11/19 9:54 pm

findings

Note that JupiterOne also integrates with several other security scanners, including **Tenable**, **Veracode**, and **WhiteHat**. All security findings are aggregated in the above centralized dashboard for easy management, filtering and reporting. You can also import **manual pen test findings**. See [this doc](#) for more

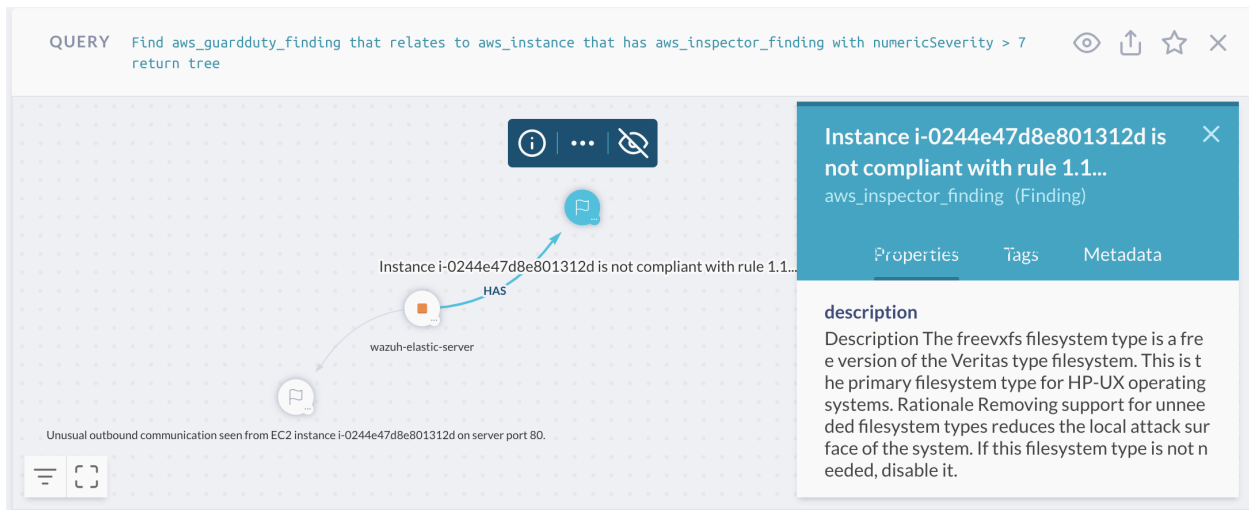
details.

## 10.2 Correlation and Alerting

Using J1QL, you can easily perform correlation of Inspector and GuardDuty findings and the resources they impact. For example, the following query identifies suspicious activities against any EC2 instance that also has high severity vulnerability findings.

```
Find aws_guarddduty_finding that relates to aws_instance
  that has aws_inspector_finding with numericSeverity > 7 return tree
```

Here is an example graph returned by the above query:



### inspector-finding-correlation

You can correlate Inspector and GuardDuty findings with other security scanner findings as well, if you have those integrations configured.

You can also set up alerts and receive notification on certain findings. For example, the following query can be used to set up an alert rule for high risk findings that impact production resources:

```
Find (Host|DataStore|Application|CodeRepo|Account|Service|Network)
  with tag.Production=true
  that has Finding with severity='High' or numericSeverity>7
```

See this alert rule in the JupiterOne `common-alerts` rule pack on [Github](#).



---

## How to configure SAML SSO integration with JupiterOne

---

Single Sign On is supported via a custom authentication client configured within a JupiterOne account. This feature is available to all enterprise customers upon request. To request SSO integration to be enabled for your account, please open a support ticket or contact your technical account manager.

### 11.1 Supported Features

- **SP-initiated SSO**

Service Provider Initiated (SP-initiated) SSO means when SAML authentication is initiated by the Service Provider (SP). This is triggered when the end user tries to access a resource in JupiterOne or login directly to the JupiterOne account.

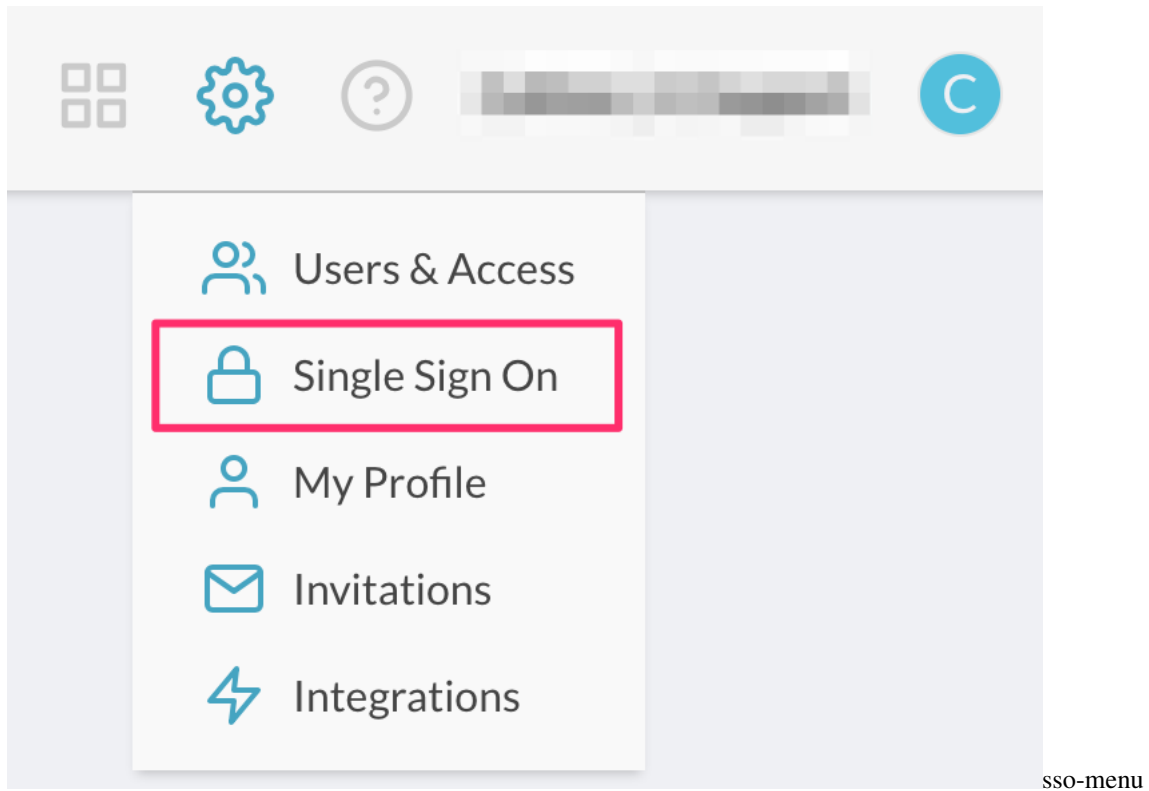
- **JIT (Just In Time) Provisioning**

Users are created/updated on the fly using the SAML attributes sent as part of the SAML response coming from the Identity Provider (IdP). The user is created during initial login to JupiterOne and updated during subsequent logins.

*IdP-initiated SSO is currently unsupported due to a limitation of Amazon Cognito.*

### 11.2 Configuration Steps

1. Log in to your JupiterOne account – your user must be a member of the *Administrators* group.
2. Go to the **Single Sign On** setup from the configurations menu.



3. Click on **Configure**.



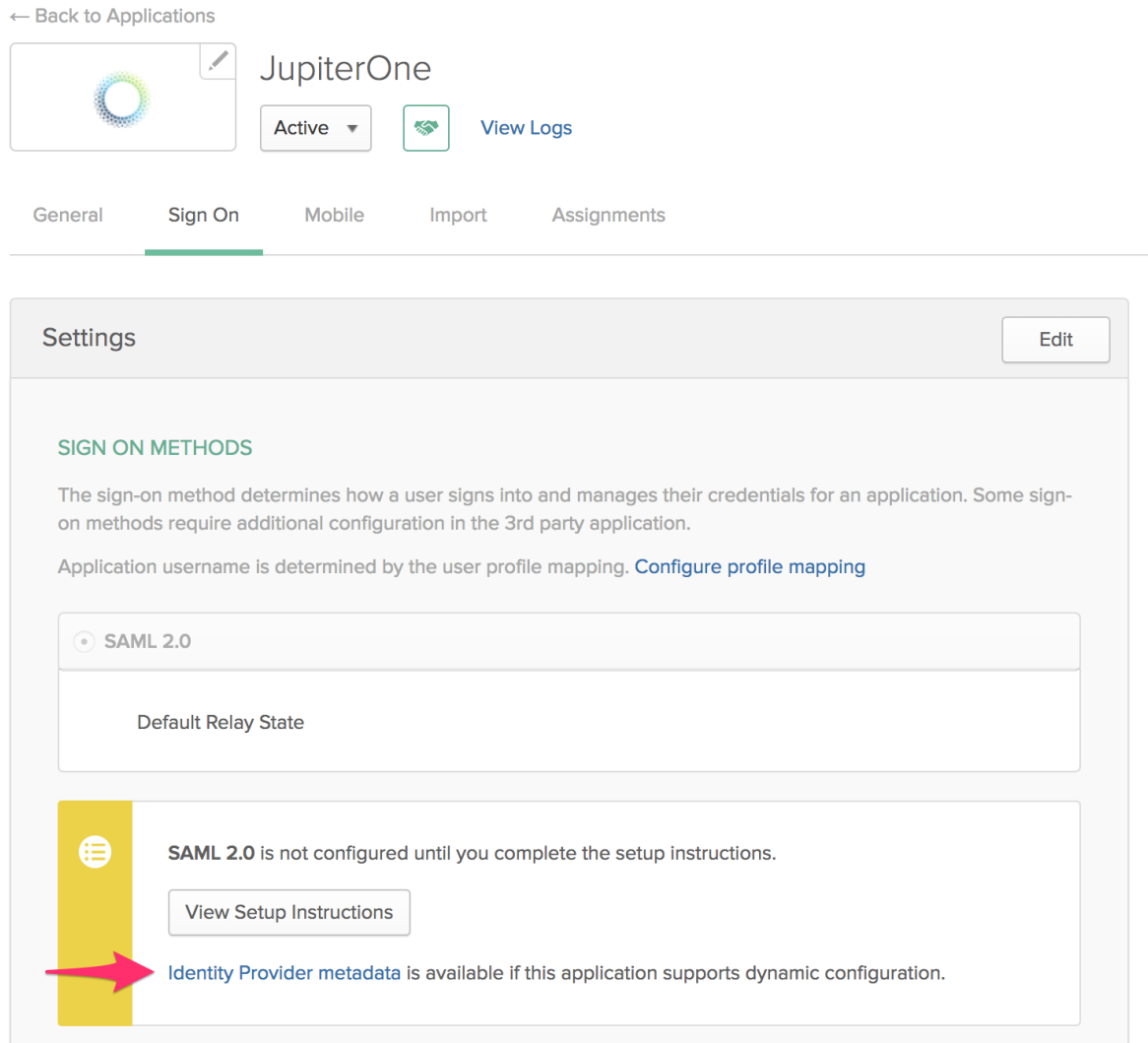
4. In the client configuration screen, copy the following two variables to be used when adding JupiterOne as an application in your SAML IdP account:
  - **Single Sign On URL**
  - **Audience URI (SP Entity ID)**
5. In your IdP Account, add a new SAML Application and name it “JupiterOne”.
  - Copy/paste the previous two variable values in the SAML settings.
  - Use the same **Single Sign On URL** string value for **Recipient URL** and **Destination URL**.
  - Leave the **Default Relay State** blank.
  - Select *EmailAddress* for **Name ID Format**.
  - Select *Email* or *Username* for **Application Username**.



- See next section for details on **Attribute Mappings**.

6. Complete setup of the SAML application within your IdP account, and copy the **Identity Provider Metadata** link.

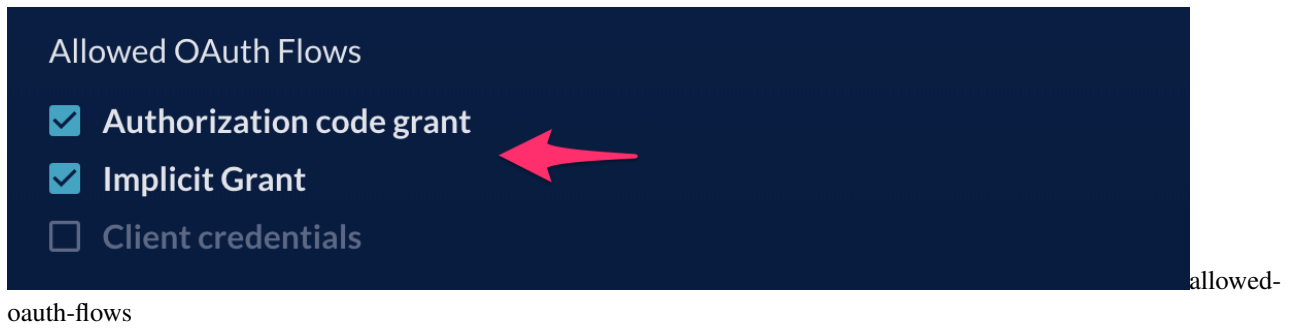
In Okta, this link can be found on the **Sign On** tab of the application, under **View Setup Instructions**, as shown below:



okta-

idp-metadata

7. Go back to **JupiterOne Auth Client Settings** screen, paste the above link to the **SAML Metadata Document URL** field.
8. Enter a **Client Name**, such as “Okta”.
9. Check **Authorization code grant** and **Implicit Grant** under “Allowed OAuth Flows”.



Save and you are all set. Next time you access your JupiterOne account via the vanity URL (e.g. [https://your\\_company.apps.us.jupiterone.io](https://your_company.apps.us.jupiterone.io)), you should be redirected to your SAML IdP for authentication.

## 11.3 Attribute Mappings

The following attribute mappings are supported:

- `email`: User's email address
- `family_name`: User's last name
- `given_name`: User's first name
- `name`: User's display name
- `group_names`: Dynamically assigns user to specified groups within JupiterOne. Use a comma to separate multiple group names (without spaces). Users without `group_names` mapping are assigned to the **Users** group within your JupiterOne account by default.

Here's an example of attribute mapping configuration in Okta:

ATTRIBUTE STATEMENTS (OPTIONAL) [LEARN MORE](#)

Name	Name format (optional)	Value
email	Basic	user.email
family_name	Basic	user.lastName
given_name	Basic	user.firstName
name	Basic	user.displayName
group_names	Basic	appuser.jupiterone_groups

Add Another

attribute-mappings

We highly recommend adding a custom *group attribute* to the JupyterOne app profile in your IdP account (e.g. Okta). This is typically added using the **Profile Editor** for the app. You can name the attribute something like `jupyterone_groups`.

Below is an example within Okta:

Profile Editor [← Back to profiles](#)


JupyterOne User

Display name: JupyterOne User

Description:

Variable name: `jupyterone_1`

Edit



JupyterOne

---

Attributes

[+ Add Attribute](#) [Map Attributes](#)

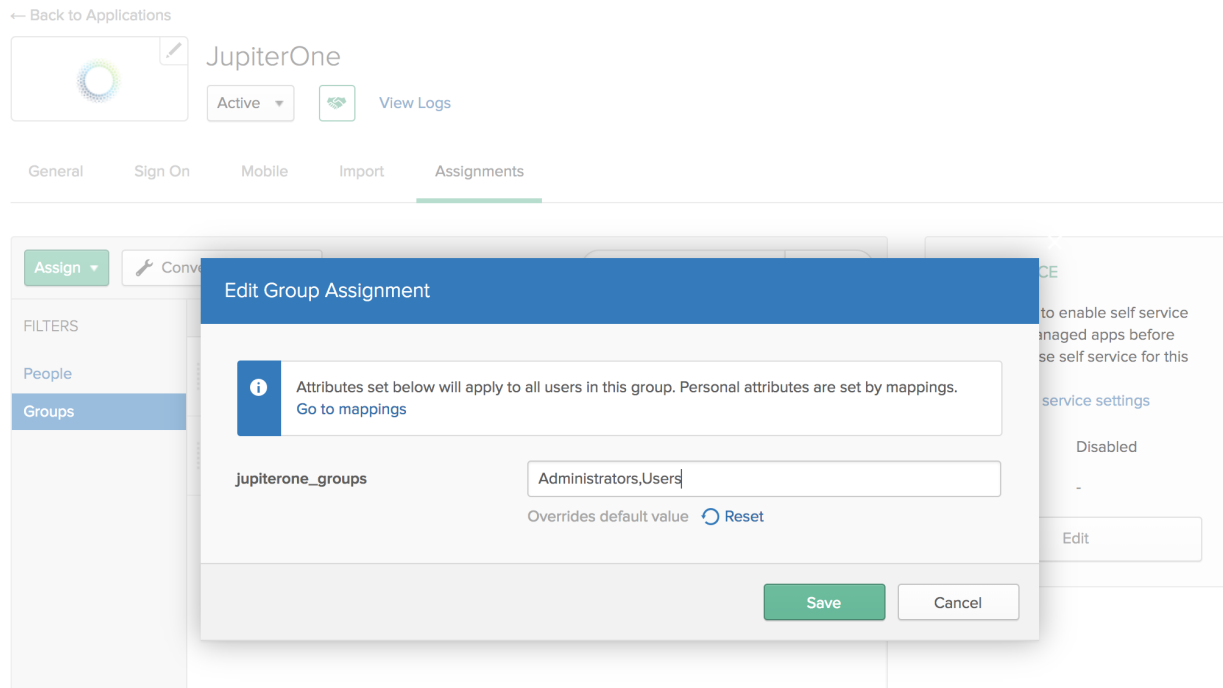
FILTERS	Display Name	Variable Name	Data Type	Attribute Type	
All	Username	userName	string	Base	<span>ⓘ</span> <span>×</span>
Base	jupyterone_groups	jupyterone_groups	string	Custom	<span>✎</span> <span>×</span>
Custom					

okta-

app-profile-editor

You can then use this custom app attribute to assign group memberships to your users based on their IdP group assignments. The actual value for the attribute is typically configured on the group(s) assigned to the app.

Below is an example within Okta:

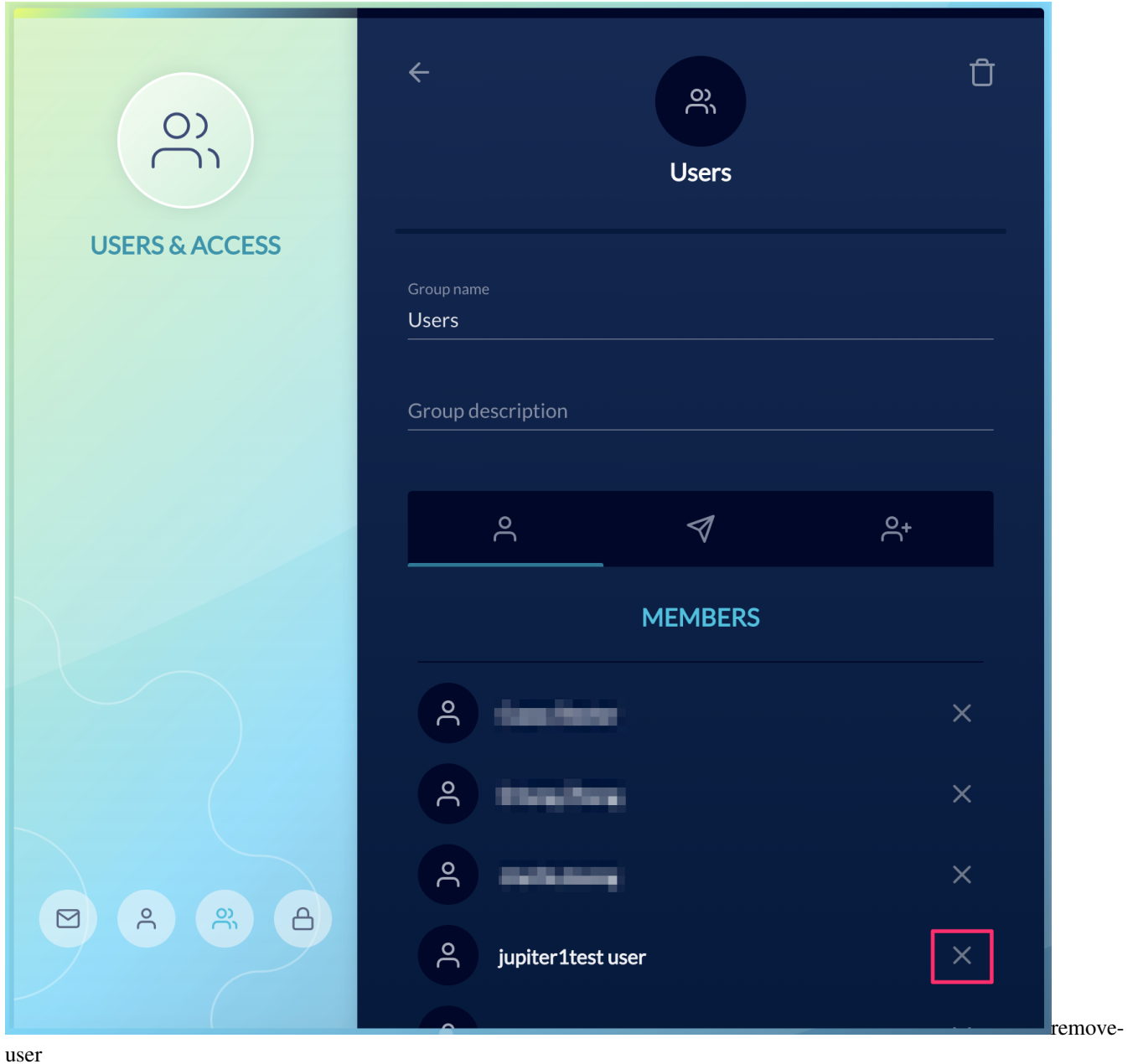


app-group-assignment

Note that provisioning users with `group_names` attribute mapping is *OPTIONAL*. Users without `group_names` mapping are assigned to the **Users** group within your JupiterOne account by default.

## 11.4 Removing Users

When you unassign / remove a user from the JupiterOne app within your IdP, the user will no longer be able to log in to your JupiterOne account because the authentication happens with your IdP. However, the user memberships will remain in the Groups. You can manually remove them from the groups within JupiterOne.



## 11.5 Current Limitations

### 11.5.1 IdP-initiated sign on flow is not supported

JupiterOne uses Amazon Cognito service to manage authentication including SSO. Cognito currently does *not* support IdP-initiated sign on. That is, you will *not* be able to click on the app icon on your IdP account (e.g. JumpCloud, Okta, OneLogin). Instead, you will need to initiate single sign on by going to your JupiterOne account URL:

```
https://<your_j1_account_id>.apps.us.jupiterone.io
```

This will redirect to your configured SSO provider for authentication.

You can find your J1 account id by running the following query:

```
Find jupiterone_account as a return a.accountId
```

### Workaround

If your SSO provider supports configuring a “Bookmark” app, you can workaround this limitation by doing the following:

- **Hide** the app icon to users for the configured JupiterOne SAML SSO app
- Configure a **Bookmark** app with your JupiterOne account URL and assigned it to the same users/groups that have been assigned the JupiterOne SAML app

---

## Detect Suspicious Code Commits in Pull Requests

---

Security of software development and code is more important than ever. JupiterOne is capable of detecting suspicious code commits in a git pull request (PR) in two ways:

- Commits self-approved by the code author
- Commits made by a user unknown to the organization

### 12.1 Enable Detection

For the detection to work, you will need to:

- Enable Pull Request (PR) and commit analysis in the integration configuration in JupiterOne.

*This feature is currently supported on Bitbucket integration. Github support is coming soon.*

- Configure branch permissions in your git source control system to prohibit directly committing to the main branch (e.g. `master`) and to require pull request reviews before merging.

*This option is typically found under the repo settings. This allows PR analysis to catch the suspicious activities.*

When enabled, JupiterOne sets the `approved` and `validated` flags on each merged PR entity.

You can run a J1QL query to detect “PRs with suspicious activities”:

```
Find PR with approved = false or validated = false
```

The screenshot displays the JupiterOne web interface. At the top, there's a search bar with the query `find PR with approved=false or validated=false`. Below this, a table lists search results for pull requests. The table has columns for class, type, displayName, updated\_on, comment\_count, account\_uid, AccountName, webLink, description, source, title, and r. Two results are shown, both of type `bitbucket_pullrequest` from the `bitbucket-integration-demo` account.

class	type	displayName	updated_on	comment_count	account_uid	AccountName	webLink	description	source	title	r
CodeReview, PR	bitbucket_pullrequest	bitbucket-integration-demo/1	2019-05-07T18:26:28.894Z	0	{f52fd1cf-2acc-4d1c-9717-41ea891417d5}	jupiteronedemo	https://bitbucket.org/jupiteronedemo/bitbucket-integration-demo/pull-requests/1	...	greeting	Add a greeting	b ir d
CodeReview, PR	bitbucket_pullrequest	bitbucket-integration-demo/2	2019-05-07T19:18:59.914Z	0	{f52fd1cf-2acc-4d1c-9717-41ea891417d5}	jupiteronedemo	https://bitbucket.org/jupiteronedemo/bitbucket-integration-demo/pull-requests/2	...	remove-smiley	Remove smiley >J	b ir d

Below the table, a relationship diagram shows a user `Isaac Williams` with an `OPENED` relationship to `bitbucket-integration-demo/2`. This entity has a `HAS` relationship to `bitbucket-integration-demo/1`, which in turn has a `HAS` relationship to `bitbucket-integration-demo`. A sidebar on the right shows the details for `bitbucket-integration-demo/1`, including its displayName, source, updated\_on, account\_uid, and webLink.

You can also set up an alert with the above query. You can also integrate this analysis into your DevOps pipeline to check for suspicious commits in PRs before deploying code to production.

## 12.2 How does it work?

### 12.2.1 Detecting self-approved commits

At the time of integration execution, or when requested via the API, JupiterOne will analyze the activities on a merged PR to determine if there is any code commit on the PR that was not approved by someone other than the code author.

*Isn't this already configured via branch protection/permissions?*

Consider the following scenario:

- Bob writes some code and commits them to a feature branch
- Bob opens a PR with those changes and requests review from Alice
- Alice makes another commit to the same branch and updates the PR
- Alice approves the PR

The PR is considered approved by a reviewer because Bob opened the PR and Alice reviewed it. However, Alice technically approved her own code associated with the commit she made to the branch after Bob opened the PR.

JupiterOne will detect this condition and sets the `approved` flag on the PR entity to `false`.

The commit hash of the detected suspicious commit is added to the `commitsNotApproved` list property.



### 12.2.2 Detecting commits by unknown/external authors

Additionally, JupiterOne checks the commit author against known bitbucket users that are part of your organization. If a commit was made by an unknown/external author, JupiterOne sets the `validated` flag on the PR entity to `false`.

The commit hash of the detected suspicious commit is added to the `commitsByUnknownAuthor` list property.

## 12.3 Combine suspicious commits checking and vulnerability checking for CI/CD

You can use the following JIQL query to detect open vulnerability findings that are associated with certain code repos, and use this in conjunction with the PR analysis query previously discussed to make automated decisions for promoting code to production in your CI/CD pipeline.

For example, you can query JupiterOne via API for:

```
Find Finding with open=true and severity=('Critical' or 'High')
  that relates to CodeRepo with name='my-new-project'

Find PR with id=55 as PR that relates to CodeRepo with name='my-new-project'
  return PR.approved, PR.validated
```

And block production deploy if the first query above returns a finding or if the second query returns `false` for `approved` or `validated` status.



## CHAPTER 13

---

### JupiterOne Node.js Client and CLI

---

JupiterOne is a data driven platform. It is easy to add your own data that is not covered by out-of-the-box managed integrations.

We provide a node.js API client wrapper and a CLI utility on [Github](#).

The CLI supports uploading entities in either JSON or YAML format. This guide - [Using JupiterOne as a central repository for SecOps and compliance artifacts](#) provides an example use case.



---

# Using JupiterOne as a central repository for SecOps and compliance artifacts

---

JupiterOne integrates with and consolidates data from several security and compliance solutions right out of the box (for example, ingesting security assessments and findings from AWS Inspector, GuardDuty, Veracode, WhiteHat, and more).

However, there will inevitably be operational and compliance artifacts produced outside of automated tools, such as **Assessments** performed manually (E.G. risk assessments or penetration tests) and the **Findings** and **Risks** identified by those assessments.

These efforts are typically documented in unstructured formats (Word or PDF) or are maintained in a separate repository such as a governance, risk and compliance (GRC), or vulnerability management system/software/service (VMS).

JupiterOne serves as a lightweight GRC and VMS so that no separate, siloed tools are needed, allowing teams to manage security and compliance artifacts as code.

## 14.1 TL;DR

*Ok. Here's some example code. Dive in!*

<https://github.com/JupiterOne/secops-automation-examples>

## 14.2 Security artifacts as code

Instead of writing security documents in Word, which are difficult to track and maintain, you should create and store artifacts and records as code. You can then easily upload these artifacts to JupiterOne for querying and reporting. Check out the examples below!

Note that the following properties are common across all entity types:

- `entityId`

The JupiterOne API does not require this property. If it is not provided, JupiterOne will create a new entity for the document. If it is provided, JupiterOne will update the existing entity for that id.

- `entityKey`

This property is required and must be unique. The JupiterOne entity creation API will update any existing entity with an identical key.

- `entityType`, `entityClass`, `name`, `displayName`

These properties are required.

All other properties listed in the examples are recommended but not required.

You can create documents to upload to JupiterOne in either JSON or YAML format. We use YAML in the examples below because it makes dealing with long, multi-line text much easier.

### 14.2.1 Assessment Entity Example

```

---
- entityId:
  entityKey: assessment:hipaa:2018
  entityType: risk_assessment
  entityClass: Assessment
  properties:
    name: HIPAA Risk Assessment 2018
    displayName: company-hipaa-risk-assessment-2018
    summary: 2018 Annual HIPAA Risk Assessment
    description:
      (sample text)
      Organization's security and compliance team assessed policies, controls
      and procedures to ensure they meet and exceed
      the requirements specified by HIPAA privacy rule and security rule.
    details:
      additional report details
    category: risk-assessment
    status: complete
    assessors:
      - security.staff@yourcompany.com
      - internal.audit@yourcompany.com
    open: false
    classification: confidential
    completedOn: 2018-07-23
    reportURL: <link to full report>
    webLink: <link to full report>

- entityId:
  entityKey: assessment:pentest:2019q1
  entityType: penetration_test
  entityClass: Assessment
  properties:
    name: internal-pen-test-2019q1
    displayName: Company Internal Penetration Test 2019Q1
    summary: Company Internal Penetration Test Q1 2019 conducted between Mar 18th -
↩Mar 29th
    description:
      (sample text)
      Performed a thorough security assessment of the company product line.

```

(continues on next page)

(continued from previous page)

```

    Scope includes product A, B and C.
details:
    additional report details
category: penetration-testing
status: complete
assessors:
    - pen.testers1@yourcompany.com
    - pen.testers2@yourcompany.com
open: false
classification: confidential
completedOn: 2019-04-05

```

The above example contains an array of two assessment objects - one HIPAA Risk Assessment and one Internal Penetration Test. If there is a more detailed report stored elsewhere, such as on your company's SharePoint or Google Docs account, you can link to it using the `reportURL` and `webLink` properties. The `webLink` property is known by the JupiterOne UI and will render a hyperlink.

We recommend also writing a full report in Markdown and storing that in source code control. The `reportURL` / `webLink` in that case will be something like this:

```

https://bitbucket.org/yourorg/security-assessments/src/master/2018/hipaa-risk-
↪assessment-report.md

```

When you specify the email address(es) of the assessor(s), JupiterOne looks up those individuals (the `Person` entities) and creates the following mapped relationship:

```

Person (with matching email address) - PERFORMED -> Assessment

```

## 14.2.2 Risk Entity Example

```

---
- entityId:
  entityKey: risk:endpoint-management-gaps
  entityType: technical_risk
  entityClass: Risk
  properties:
    name: Endpoint management gaps
    displayName: Endpoint management gaps
    summary: Lack of visibility on how user endpoint systems/devices are configured
    description:
      (sample text)
      Endpoint systems should be configured according to the company's IT and
      security standards. Because currently all enduser systems (e.g. laptops)
      are self managed, there is a lack of centralized visibility into how
      each system is configured and if they meet the compliance requirements

    details:
      'Systems should be configured with at least the following:'

      1. Disk encryption enabled
      2. Screensaver protection/screen lock on
      3. Local firewall enabled
      4. Remote login disabled
      5. Auto install OS security patches enabled

```

(continues on next page)

(continued from previous page)

```
6. (if it is Windows) Has Windows Defender or equivalent malware protection.↵
↵running

category: technical
threats: malware
targets: enduser devices
probability: 2
impact: 2
score: 4
status: open
reporter: security@yourcompany.com
open: true
mitigation:
jiraKey: SEC-112
webLink: https://yourcompany.atlassian.net/browse/SEC-112
```

**Notes:**

- The Risk score = probability times impact
  - Both probability and impact are numeric values, between 0-3. (you may choose to use a different scale)
  - Probability rating:
    - \* 3: high/certain
    - \* 2: medium/likely
    - \* 1: low/unlikely
    - \* 0: none/negligible
  - Impact rating:
    - \* 3: high/severe
    - \* 2: medium/moderate
    - \* 1: low/minor
    - \* 0: none/insignificant
- Example valid Risk status:
  - accepted
  - mitigated
  - transferred
  - reported
  - planned
  - acknowledged
  - prioritized
- The Risk is considered open unless it is accepted, mitigated or transferred status.
- When uploaded to JupiterOne, Risks will automatically map to an employee/Person using the email address specified in the reporter property:



```
Person (with matching email address) - REPORTED -> Risk
```

- Similarly, specify the assessment name in the `assessment` property to create the following mapping:

```
Assessment (with matching name) - IDENTIFIED -> Risk
```

- The `webLink` property is optional.
- Note the `jiraKey` property and the `webLink` URL in the example point to a Jira issue since Jira is used to track the workflow of this Risk item.

### 14.2.3 Finding Entity Example

A vulnerability finding is similar to a risk item:

```
---
- entityId:
  entityKey: finding:pentest:2019q1:appcode-1
  entityType: pentest_finding
  entityClass: Finding
  properties:
    name: XSS in application {appname}
    displayName: XSS in application {appname}
    summary: Stored cross side scripting identified in application {appname}
    targets:
      - appname
    description:
      description of the finding
    stepsToReproduce:
      - '1 - Sign in to application... navigate to page...'
      - '2 - Enter <script>alert(1)</script> in textbox...'
      - '3 - Hit save...'
    impact:
      Attacker may store malicious javascript...
    recommendation:
      Perform input validation in the code...
    severity: high
    priority: 2
    remediationSLA: 30
    status: open
    assessment: internal-pen-test-2019q1
    open: true
    classification: confidential
    jiraKey: SEC-99
    webLink: https://yourcompany.atlassian.net/browse/SEC-99
```

Again, the `assessment` property here is used to connect the finding to the assessment that identified it.

Additionally, if the `targets` property contains one or more entries that match the name of an Application/CodeRepo/Project entity, this finding will be linked to that matching entity, so that you can easily run a query like:

```
Find (Application|CodeRepo|Project) that has Finding with severity='high'
```

Also note the `remediationSLA` property. This specifies the number of days your team has left to address this finding per your company policy.

## 14.3 Uploading to JupiterOne

Once you have created your artifacts, you can easily upload them to JupiterOne using the CLI. Just follow these three simple steps:

1. Obtain an API Key from your JupiterOne account
2. Install JupiterOne client/CLI:

```
npm install @jupiterone/jupiterone-client-nodejs -g
```

1. Upload the artifacts (entities) to your account on JupiterOne:

```
export J1_API_TOKEN={api_key}
j1 -o create --entity -a {accountId} -f ./risks.yml
j1 -o create --entity -a {accountId} -f ./assessments.yml
j1 -o create --entity -a {accountId} -f ./findings.yml
```

If you have several YAML files to upload, you might use a command like:

```
export J1_API_TOKEN={api_key}
find . -name \*.yaml | while read yaml; do j1 -o create --entity -a <j1_account_id> -f
↪$yaml; done
```

We highly recommended you use a source code control system such as Github or Bitbucket to maintain these artifacts. This way, you can easily set up your CI system (e.g. Travis CI or Jenkins) to run the above commands and automatically keep the entities up to date in JupiterOne with every approved code change (i.e. when a PR is merged into master).

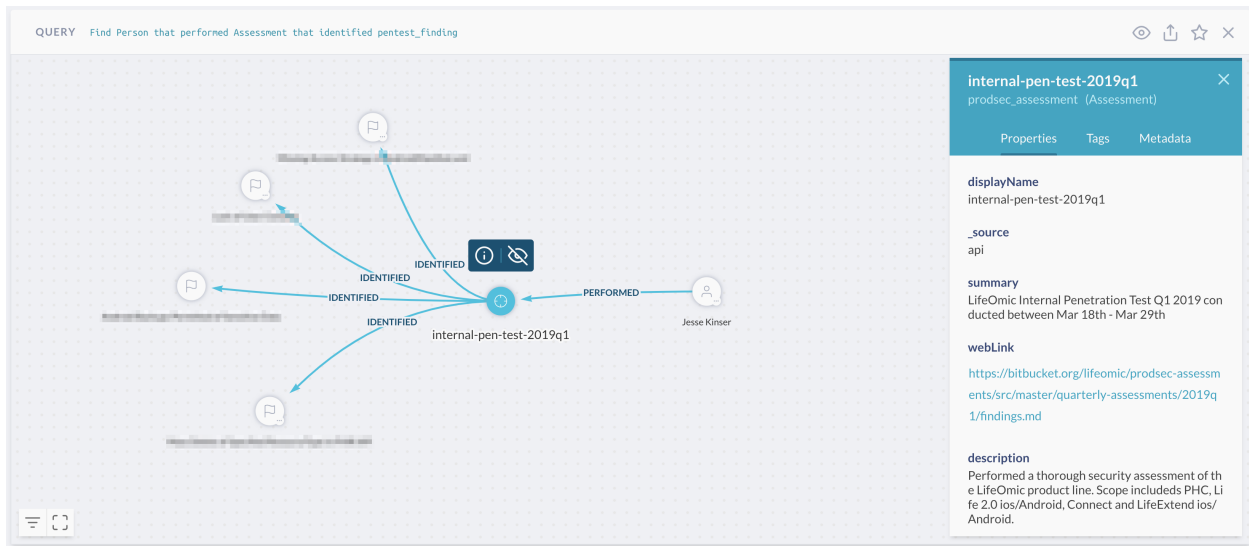
### 14.3.1 Reporting and Visualization

You can see and export these Assessments, Risks, and Findings from the Asset Inventory app in JupiterOne or query and visualize them on the Landing page.

#### Query:

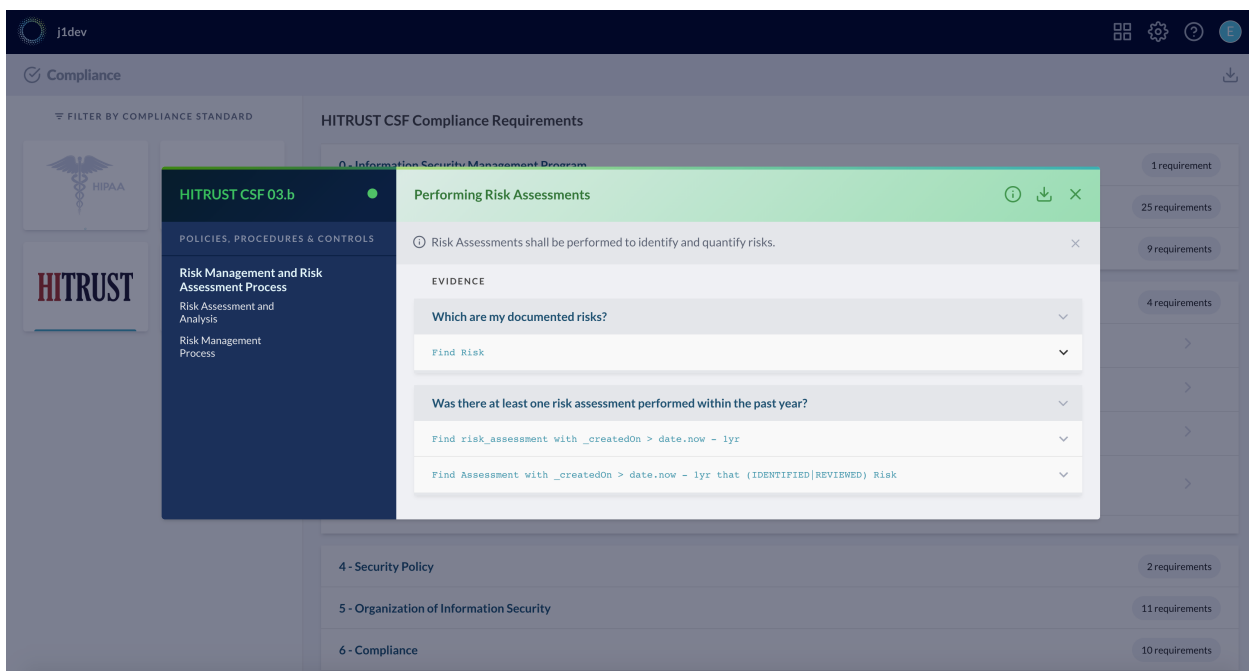
```
Find Person that performed Assessment that identified (Risk|Finding) return tree
```

#### Graph:



findings

Lastly, these artifacts are automatically tracked and mapped to the supported compliance requirements as evidences for conducting the necessary assessments.



assessments

## PDF:

You can use a simple utility to query J1 for an assessment, its related findings, and generate a PDF report. An example can be found in the `secops-automation-examples` Github repo:

<https://github.com/JupiterOne/secops-automation-examples>



---

## JupiterOne Endpoint Compliance Agent “Power Up”

---

JupiterOne is not an endpoint security solution. Rather, it is a graph platform designed for security operations and compliance. So, why are we even talking about an endpoint agent?

The JupiterOne team’s internal security operations is a highly distributed and self-managed. We needed a lightweight approach to ensure that users have correctly configured their own endpoint devices (i.e. laptops) and prompt them to remediate if a non-compliant configuration is detected.

Read [this blog](#) for more on our journey in solving endpoint compliance in a cloud-first landscape.

We are offering this endpoint agent as a “Power Up” to all JupiterOne customers.

### 15.1 The Agent

The endpoint agent has two components:

1. **Stethoscope-app** - an open-source solution by Netflix. This is a community project and it is *not officially supported* by the JupiterOne team. See the [Github project](#) for more details.
2. **J1 integration agent** - a native agent binary to connect Stethoscope-app with JupiterOne as the management backend for policy and configuration status reporting.

### 15.2 Installation

Installing and activating the JupiterOne endpoint compliance agent follows a self-service approach.

1. A JupiterOne administrator needs to send an activation email to users:
  - From the JupiterOne web UI, go to **Settings** (the gear icon near the top right), and then **Power Ups** for the Settings menu.
  - Select **JupiterOne Endpoint Compliance Agent** from the Power Ups menu.
  - Select one of three options to **Send Activations** to:

- All users
- User devices without Endpoint Agent
- Specify emails individually

Note the first two options requires you to have an identity provider (IdP) integration configured in JupiterOne so that the platform knows about the users of your organization. Example IdP integrations include Okta, OneLogin, Google G Suite, JumpCloud.

2. Users should then follow the simple instructions in the activation email to download, install and activate the endpoint agent.

## 15.3 Policies

For details on how to customize your endpoint compliance policy, see the documentation [here](#):

Supported Practices

## 15.4 Advanced Use Cases

For organizations using an automated package deployment tool such as SaltStack or Jamf, we are often asked if the JupiterOne power up agent can be included in the auto-rollout from the deployment tool. This is technically doable (see below for more details below), but not officially supported at the moment.

Technically the Stethoscope agent installation and JupiterOne integration process has three moving parts:

1. **Stethoscope-app**: You would likely need to build this yourself from the code in this repo. It supports an optional “publish”/distribution URL for distributing updates. The Stethoscope-app build that we ship with our installer is configured to pull updates from our S3 bucket location. Since you intend to roll-out updates via your deployment tool, you would likely not use this mechanism. Once built without a publishing configuration, and installed at a certain version of Stethoscope, that version would remain until you replace it.
2. Our native agent binary for JupiterOne integration: This is bundled into the installer (.pkg for macOS, .msi for Windows, or a .sh script for Linux), which can be downloaded from the download link within an JupiterOne endpoint activation email (see above). We could provide this to you for distribution with your own Stethoscope builds, or you could run the packaged installer and pull the binary agent from that. It is a data-driven GraphQL client that doesn’t change often.
3. A one-time activation step that is performed using the agent binary in a CLI mode. This is automatically done via the installer wizard, but can certainly be done in the deployment using a command of the form:

```
# macOS example:
jl-endpoint-agent-darwin activate --email <your.email> --account <your J1 account_
↪name> --code <one-time-use activation code>
```

Ordinarily, these account codes are generated at email-send time, using the send email feature of the administrative Endpoint Compliance Power Up UI. Contact JupiterOne Support to pre-generate a CSV of activation codes for a list of your email addresses. This activation step registers each particular endpoint device with JupiterOne, sending along the device-specific UUID along with the activation information provided at the command line, and generates an API token used to retrieve the Stethoscope policy and upload scan results to J1.

---

## JupiterOne Data Model

---

The **JupiterOne Data Model** is a reference model used to describe digital resources and the complex interconnections among all the resources in a technology organization as an **entity-relationship graph**.

The data model is defined by a set of Entities and their Relationships. It represents a reference model, not a strict or rigid structure.

### 16.1 Entity

An Entity is a node/vertex in the graph that represents a resource within your digital infrastructure.

#### 16.1.1 Class and Type of an Entity

Each Entity has a specific **type** that defines what that entity is, and is assigned one or more higher level **class** that represents a more abstract categorization or labeling of the entity in the perspective of security and technical operations.

##### Type

The **type** property represents the specific type that entity is as defined by the source. For example, an AWS resource may be of type `aws_instance` or `aws_s3_bucket` or `aws_iam_user`.

##### Class

The **class** of an entity is considered an abstract, super-type that defines what that entity is within the general framework of IT and security operations. In the above example, an `aws_instance` entity has a class of `Host`, while an `aws_s3_bucket` is a `DataStore`, and an `aws_iam_user` a `User`.

## 16.1.2 Common Entity Properties

Most Entities will have the following common properties.

## 16.1.3 Class Specific Entity Properties

Each specific class of Entity also has its own defined properties. For example, a `Person` entity will have properties including `firstName` and `lastName`, while a `Device` entity may have properties such as `hardwareVendor`, `hardwareModel`, and `hardwareSerial`.

## 16.1.4 Custom Properties

Entities can also have custom properties that are specific to the type of that entity, defined by the source system where the resource belongs to, or defined by the individual or team managing the resource.

## 16.1.5 Defined Entities

Here is a list of reference entities defined by the JupiterOne Data Model, each with its own defined set of properties in addition to the shared common properties:

### Special Entities

There are three special entities defined. These are singleton entities.

## 16.2 Relationships

A **Relationship** is the edge between two Entity nodes in the graph. The `_class` of the relationship should be, in most cases, a generic descriptive verb, such as `HAS` or `IMPLEMENTS`.

Relationships can also carry their own properties.

For example, `CodeRepo -- DEPLOYED_TO -> Host` may have `version` as a property on the `DEPLOYED_TO` relationship. This represents the mapping between a code repo to multiple deployment targets, while one deployment may be of a different version of the code than another. Storing the version as a relationship property allows us to void duplicate instances of the code repo entity to be created to represent different versions.

Relationships have the same metadata properties as entities, which are managed by the integration providers.

### 16.2.1 Example defined Relationships between abstract Entity Classes

#### HAS / CONTAINS

Account	-- HAS ->	User
Account	-- HAS ->	UserGroup
Account	-- HAS ->	AccessRole
Account	-- HAS ->	Resource
CodeRepo	-- HAS ->	Vulnerability
Host	-- HAS ->	Vulnerability
Organization	-- HAS ->	Site

(continues on next page)



(continued from previous page)

Organization	-- HAS ->	Organization (e.g. a business unit)
Application	-- HAS ->	Vulnerability
CodeRepo	-- HAS ->	Vulnerability
Host	-- HAS ->	Vulnerability
Service	-- HAS ->	Vulnerability
Site	-- HAS ->	Network
Site	-- HAS ->	Site
UserGroup	-- HAS ->	User
Network	-- CONTAINS ->	Host
Network	-- CONTAINS ->	Database
Network	-- CONTAINS ->	Network (e.g. a subnet)

**IS / OWNS**

User	-- IS ->	Person
Vulnerability	-- IS ->	Vulnerability (e.g. a Snyk Vuln IS a CVE)
Person	-- OWNS ->	Device

**EXPLOITS / IMPACTS**

Vulnerability	-- EXPLOITS ->	Weakness
Vulnerability	-- IMPACTS ->	CodeRepo   Application

**USES**

Host	-- USES ->	Resource (e.g. aws_instance USES aws_ebs_volume)
------	------------	--

**CONNECTS / TRIGGERS / EXTENDS**

Application	-- CONNECTS ->	Account
Gateway	-- CONNECTS ->	Network
Gateway	-- TRIGGERS ->	Function
HOST	-- EXTENDS ->	Resource

**IMPLEMENTS / MITIGATES**

Procedure	-- IMPLEMENTS ->	Policy
Control	-- IMPLEMENTS ->	Policy
Control	-- MITIGATES ->	Risk

**MANAGES**

Person	-- MANAGES ->	Person
Person	-- MANAGES ->	Organization
Person	-- MANAGES ->	Team

(continues on next page)

(continued from previous page)

User	-- MANAGES ->	Account
User	-- MANAGES ->	UserGroup
ControlPolicy	-- MANAGES ->	Control
AccessPolicy	-- MANAGES ->	AccessRole

## EVALUATES / MONITORS / PROTECTS

ControlPolicy	-- EVALUATES ->	<any entity>
HostAgent	-- MONITORS ->	Host
HostAgent	-- PROTECTS ->	Host

## TRUSTS

AccessRole	-- TRUSTS ->	AccessRole
AccessRole	-- TRUSTS ->	Service
AccessRole	-- TRUSTS ->	Account

## ASSIGNED

User	-- ASSIGNED ->	Application
User	-- ASSIGNED ->	AccessRole
UserGroup	-- ASSIGNED ->	AccessRole

## IDENTIFIED / PERFORMED / COMPLETED

Person	-- PERFORMED ->	Assessment
Person	-- COMPLETED ->	Training
Assessment	-- IDENTIFIED ->	Risk
Assessment	-- IDENTIFIED ->	Vulnerability

## PROVIDES

Vendor	-- PROVIDES ->	Service
--------	----------------	---------

## CONTRIBUTES\_TO

User	-- CONTRIBUTES_TO ->	CodeRepo
------	----------------------	----------

## OPENED

User	-- OPENED ->	CodeReview (i.e. PR)
------	--------------	----------------------





---

## JupiterOne Data Security

---

This document describes in detail the data JupiterOne ingests and how your data is protected on our platform.

### 17.1 Data Protection

#### 17.1.1 Encryption

Data is fully encrypted both at rest and in transit. This includes all of your account and user data, as well as operational data imported/ingested into the JupiterOne platform.

**Data in Transit** is encrypted via TLSv1.2 or later, using SHA-256 with 2048-bit RSA Encryption or equivalent strength cypher.

*Production Domains:* `*.apps.us.jupiterone.io` is the associated production URL that the SSL/TLS certificate has been issued to.

**Data at Rest** is hosted in our production AWS environments, using the managed RDS/Neptune, DynamoDB, and S3 services. All database instances, tables, and S3 buckets with customer data have server-side encryption enabled, using AWS KMS for key management. KMS encryption keys are scheduled to rotate annually.

In addition to encryption, managed backup is enabled for the database clusters. For S3 buckets, cross-account replication is enabled to back up data to a different region for disaster recovery. All backup data is fully encrypted the same way as its source.

#### 17.1.2 Multi-tenancy

JupiterOne is a multi-tenancy, software-as-a-service platform hosted in AWS. Customer data is logically partitioned/segreated by software via a unique `accountId` associated with every piece of data. Access to data is restricted to within each unique account for users granted proper access to that account. This is a standard pattern used by cloud infrastructure and SaaS providers.

## 17.2 External Data Ingestion/Import

JupiterOne ingests data from external sources and connected environments primarily via the APIs provided by the target environment/service provider. Objects from these external environments and their corresponding metadata, including configuration properties and tags but never the actual data content, are ingested as “entities”. The entity properties and tags are used to perform analysis to build out “relationships” among ingested entities. These entities and relationships are the **JupiterOne CORE Data Model**.

JupiterOne then uses this data model to inventory for and provide insight into your digital infrastructure across all of your connected environments.

More information on the JupiterOne Data Model can be found [here](#).

For more details on data ingested for each managed integration, see their corresponding documentation in the **Integrations** section.

### 17.2.1 Access Permissions Needed to Integrated Environments

Access to your environments is needed in order to ingest data, or to enable workflow automation (future capability).

In general, JupiterOne only requires read-only, security-auditor-type access permissions to your environments. Additionally, this read-only access only applies to configurations and meta data, not the actual data content. For example, we do **NOT** read S3 objects data from a connected AWS account, or the actual source code of a connected Bitbucket/Github account.

Additional level of access may be needed for workflow automation. For example, integration with Jira to automatically create an issue when a new Vulnerability finding is added; or to post to a Slack channel/user to send a security alert notification.

You are always in control of the actual permissions granted for each integration. More details of the access permissions required for each managed integration can be found in its corresponding documentation listed above.

### 17.2.2 Custom Data Import

Additionally, JupiterOne supports the ability for you to add custom data by

- Manually adding entities via Web UI in the Asset Inventory app;
- Adding bulk number of entities via CSV import; or
- Adding custom entities via custom integrations using the public API.

## 17.3 Data Ownership and Access

You retain full ownership of all data that is ingested via integrations, API or manual importing/creation. Data is stored in JupiterOne’s production environment in AWS, protected via encryption and replication as specified in the first section.

### 17.3.1 Infrastructure and Operational Access

JupiterOne infrastructure is built on a **Zero Trust** security model, where access to production is *highly restricted*.

The production environment is virtually “air-gapped” such that there is no SSH, “bastion host”, or VPN connectivity into the production systems to prevent unintended network access to databases and other production servers. We

do not allow internal access to production data by any JupiterOne team member. All necessary operational support and maintenance jobs are performed via automation where the automation code is fully documented, reviewed, and approved, ensuring end-to-end traceability.

Our production environment incorporates multiple layers of security monitoring, using JupiterOne itself as well as third party security solutions. Additionally, our software development includes rigorous code analysis and continuous testing practices to ensure we proactively identify any security vulnerability. Our infrastructure-as-code operational model and automated change management process allows us to deploy security patches within minutes of identification and remediation of an issue.

You can review our published [security model](#) and corresponding [policies and procedures](#) for more details on our operational, infrastructure, and software development security.

## 17.4 Application Access

Access to the JupiterOne application and your accounts/data on the platform is enabled over HTTPS, through either the JupiterOne web apps or the public APIs.

*Note:* `*.us.jupiterone.io` is the current production domain.

### 17.4.1 User Logins

Each user has a unique user login to the JupiterOne platform and apps. Users may be invited to one or multiple organizational accounts on JupiterOne.

#### Password Policy

Users are required to select a strong password meeting the following password policy requirements in order to create a login and authenticate to the system:

- Minimum of 8 characters
- Must contain an uppercase letter
- Must contain a lowercase letter
- Must contain a number
- Must contain a special character

#### Single Sign On (SSO)

JupiterOne currently supports single sign on (SSO) via:

- Google
- SAML

#### Multi-Factor Authentication (MFA) / Two-Step Verification (2SV)

Multi-Factor Authentication (MFA) or Two-Step Verification (2SV) is strongly recommended for all users on the JupiterOne platform. This needs to be enabled and configured via your SSO provider (Google or your SAML IdP such as Okta or OneLogin).

### 17.4.2 Access Control

In order to support potential complex access control use cases, JupiterOne platform implements Attribute Based Access Control (ABAC).

A good general overview of ABAC is sections 1 and 2 of NIST's [Guide to Attribute Based Access Control](#). The absolute basics of ABAC are that you have a subject (e.g. a user) who wants to perform some operation (e.g. download) on an object (e.g. a file) in some environment. The subject, object and environment all have attributes (i.e. key/value pairs), and there are policies that control the privileges (i.e. what operations the subject can perform) given the attributes.

Access policies defined in JupiterOne are associated with a **User Group** and **Users** are invited/added as members to one or more groups.

- A *Read-Only* access policy is predefined and associated with the default **Users** group.
- A *Full-Access* policy is also predefined and associated with the default **Administrators** group.
- The ability to customize and add granular access control policies is to be released in 1Q2019.

### 17.4.3 API Access

JupiterOne API is available at: <https://api.us.jupiterone.io/>

We use [OAuth 2.0](#) for authorization, which means in order to access data a user must authenticate and the requesting app must be authorized. Implicit grant, authorization code, and client credentials flows are supported. Authorization code is recommended for web apps, which involves utilizing both the authorize and token API resources. When using the authorization code grant flow, it is also recommended to use Proof Key for Code Exchange (PCKE) to mitigate authorization code intercept attacks. Contact us if building a native app which can securely perform client credentials flow.

Additionally, each user on the platform can create an API key that can be passed along with request to act on behalf of that user.

*Note: the UI for self-service configuration of OAuth and user API key is targeted to be available in 1Q2019.*

### 17.4.4 Support Access to Your JupiterOne Account(s)

A JupiterOne Support User is by default added to a new account during free trial, proof-of-concept evaluation, or initial account onboarding. This is to facilitate better support and training on using the platform.

- The support user's login can either be the individual Security Engineer/Architect designated to your account (e.g. `firstname.lastname@jupiterone.io`) or the general support login (i.e. `callisto@jupiterone.io`).
- The support user can be removed by an account administrator at any time, should you determine that ongoing regular support is no longer needed.
- You have the option and administrative privilege to add the support user back at any time, when support is needed in the future.



---

## JupiterOne Query Language (J1QL)

---

The JupiterOne Query Language (aka “J1QL”) is a query language for querying data stored by JupiterOne. The execution of a J1QL query will seamlessly query full text search, entity-relationship graph, and any other future data stores as needed. By design, the query language does not intend to make these data store boundaries obvious to query authors.

### 18.1 Language Features

- Seamlessly blend full-text search and graph queries
- Language keywords are case-insensitive
- Inspired by SQL and Cypher and aspires to be as close to natural language as possible
- Support for variable placeholders
- Return **entities**, **relationships**, and/or traversal **tree**
- Support for sorting via `ORDER BY` clause (currently only applies to the starting entities of traversal)
- Support for pagination via `SKIP` and `LIMIT` clauses (currently only applies to the starting entities of traversal)
- Multi-step graph traversals through relationships via `THAT` clause
- Aliasing of selectors via `AS` keyword
- Pre-traversal filtering using property values via `WITH` clause
- Post-traversal filtering using property values or union comparison via `WHERE` clause
- Support aggregates including `COUNT`, `MIN`, `MAX`, `AVG` and `SUM`.

### 18.2 Basic Keywords

`FIND` is followed by an **Entity** `class` or `type` value.

The value is case sensitive in order to automatically determine if the query needs to search for entities by the `class` or the `type`, without requiring authors to specifically call it out.

Entity `class` is stored in `TitleCase` while `type` is stored in `snake_case`.

A wildcard `*` can be used to find *any entity*.

For example:

- `FIND User` is equivalent to `FIND * with _class='User'`
- `FIND aws_iam_user` is equivalent to `FIND * with _type='aws_iam_user'`

Note that using the wildcard at the beginning of the query without any pre-traversal filtering – that is, `FIND * THAT ...` without `WITH` (see below) – may result in long query execution time.

`WITH` is followed by **property name and values** to filter entities.

Supported operators include:

- `=` or `!=` for **String** value, **Boolean**, **Number**, or **Date** comparison.
- `>` or `<` for **Number** or **Date** comparison.

Note:

- The property names and values are *case sensitive*.
- **String** values must be wrapped in either single or double quotes - `"value"` or `'value'`.
- **Boolean**, **Number**, and **Date** values must *not* be wrapped in quotes.
- The `undefined` keyword can be used to filter on the absence of a property. For example: `FIND DataStore with encrypted=undefined`

`AND`, `OR` for multiple property comparisons are supported.

For example:

```
FIND DataStore WITH encrypted = false AND tag.Production = true
FIND user_endpoint WITH platform = 'darwin' OR platform = 'linux'
```

- You can filter multiple property values like this (similar to `IN` in SQL):

```
FIND user_endpoint WITH platform = ('darwin' OR 'linux')
Find Host WITH tag.Environment = ('A' or 'B' or 'C')
Find DataStore WITH classification != ('critical' and 'restricted')
```

`THAT` is followed by a **Relationship verb**.

The verb is the `class` value of a **Relationship** – that is, the edge between two connected entity nodes in the graph. This relationship verb/class value is stored in ALLCAPS, however, it is *case insensitive* in the query, as the query language will automatically convert it.

The predefined keyword `RELATES TO` can be used to find *any* relationship between two nodes. For example:

```
FIND Service THAT RELATES TO Account
```

`( | )` can be used to select entities or relationships of different class/type.

For example, `FIND (Host|Device) WITH ipAddress='10.50.2.17'` is equivalent to and much simpler than the following:

```
FIND * WITH
  (_class='Host' OR _class='Device') AND ipAddress='10.50.2.17'
```

It is fine to mix entity class and type values together. For example:

```
FIND (Database|aws_s3_bucket)
```

It can be used on Relationship verbs as well. For example:

```
FIND HostAgent THAT (MONITORS|PROTECTS) Host
```

Or both Entity and Relationships together. For example:

```
FIND * THAT (ALLOWS|PERMITS) (Internet|Everyone)
```

AS is used to define an aliased selector.

Defines an aliased selector to be used in the WHERE or RETURN portion of a query. For example:

- **Without** selectors: `FIND Firewall THAT ALLOWS *`
- **With** selectors: `FIND Firewall AS fw THAT ALLOWS * AS n`

Selectors can also be defined on a relationship:

- `FIND Firewall AS fw THAT ALLOWS AS rule * AS n`

WHERE is used for post-traversal filtering or union (requires selector)

From the example above:

```
FIND Firewall as fw that ALLOWS as rule * as n
  WHERE rule.ingress=true AND
    (rule.fromPort=22 or rule.toPort=22)
```

The following examples joins the properties of two different network entities, to identify if there are multiple networks in the same environment using conflicting IP spacing:

```
FIND (Network as n1 | Network as n2)
  WHERE n1.CIDR = n2.CIDR
```

RETURN is used to return specific entities, relationships, or properties

By default, the entities and their properties found from the start of the traversal is returned. For example, `Find User that IS Person` returns all matching `User` entities and their properties, but not the related `Person` entities.

To return properties from both the `User` and `Person` entities, define a selector for each and use them in the RETURN clause:

```
FIND User as u that IS Person as p
  RETURN u.username, p.firstName, p.lastName, p.email
```

Wildcard can be used to return all properties. For example:

```
FIND User as u that IS Person as p
  RETURN u.*, p.*
```

A side effect of using wildcard to return all properties is that all metadata properties associated with the selected entities are also returned. This may be useful when users desire to perform analysis that involves metadata.

Keep in mind the keywords are *case insensitive*.

## 18.3 Sorting and Pagination via ORDER BY, SKIP, and LIMIT

ORDER BY is followed by a `selector.field` to indicate what to sort.

SKIP is followed by a number to indicate how many results to skip.

LIMIT is followed by a number to indicate how many results to return.

In the example below, the query sorts users by their username, and returns the 15th-20th users from the sorted list.

```
FIND Person as u WITH encrypted = false
ORDER BY u.username SKIP 10 LIMIT 5
```

## 18.4 Aggregation Functions: COUNT, MIN, MAX, AVG and SUM

It is useful to be able to perform calculations on data that have been returned from the graph. Being able to perform queries to retrieve a count, min, max or perform other calculations can be quite valuable and gives users more ways to understand their data.

The ability to perform aggregations are exposed as **Aggregating Functions**. These are functions that can be applied to a given set of data that was requested via the RETURN clause.

The following aggregating functions are supported:

- `count(selector)`
- `count(selector.field)`
- `min(selector.field)`
- `max(selector.field)`
- `avg(selector.field)`
- `sum(selector.field)`

The keywords are *case insensitive*.

A few examples:

```
find
  bitbucket_team as team
  that relates to
  bitbucket_user as user
return
  team.name, count(user)
```

```
find
  bitbucket_team as team
  that relates to
  bitbucket_user as user
return
  count(user), avg(user.age)
```

See more details and examples *below*.

Future development:

There are plans to support the following aggregations:

- `count (*)` - for determining the count of all other entities related to a given entity.

## 18.5 Examples

More example queries are shown below.

These examples, and same with all packaged queries provided in the JupiterOne web apps, are constructed in a way to de-emphasize the query keywords (they are *case insensitive*) but rather to highlight the relationships – the operational context and significance of each query.

### 18.5.1 Simple Examples

```
/* Find any entity that is unencrypted */
Find * with encrypted = false

/* Find all entities of class DataStore that are unencrypted */
Find DataStore with encrypted = false

/* Find all entities of type aws_ebs_volume that are unencrypted */
Find aws_ebs_volume with encrypted = false
```

### 18.5.2 Query with relationships

```
/* return just the Firewall entities that protects public-facing hosts */
Find Firewall that PROTECTS Host with public = true

/* return Firewall and Host entities that matched query */
Find Firewall as f that PROTECTS Host with public = true as h RETURN f, h

/* return all the entities and relationships that were traversed as a tree */
Find Firewall that PROTECTS Host with public = true RETURN tree
```

### 18.5.3 Full-text search

```
/* find any and all entities with "127.0.0.1" in some property value */
Find "127.0.0.1"

/* the `FIND` keyword is optional */
"127.0.0.1"

/* find all hosts that have "127.0.0.1" in some property value */
Find "127.0.0.1" with _class='Host'
```

### 18.5.4 Negating relationships

It's useful to know if entities do not have a relationship with another entity. To achieve this, relationships can be negated by prefixing a relationship with an exclamation point: `!`.

```
Find User that !IS Person

/* This also applies to any relationships */
Find User that !RELATES TO Person
```

This finds EBS volumes that are not in use. The query finds relationships regardless of the edge direction, therefore the !USES in the below query translates more directly as “**is not used by**”.

```
Find aws_ebs_volume that !USES aws_instance
```

It is important to note that the above query returns `aws_ebs_volume` entities. If the query were constructed the other way around –

```
Find aws_instance that !USES aws_ebs_volume
```

– it would return a list of `aws_instances`, if it does not have an EBS volume attached.

### 18.5.5 More complex queries

Find critical data stored outside of production environments.

This assumes you have the appropriate tags (Classification and Production) on your entities.

```
Find DataStore with tag.Classification='critical'
  that HAS * with tag.Production='false'
```

Find all users and their devices without the required endpoint protection agent installed:

```
Find Person that has Device that !protects HostAgent
```

Find incorrectly tagged resources in AWS:

```
Find * as r
  that RELATES TO Service
  that RELATES TO aws_account
  where r.tag.AccountName != r.tag.Environment
```

If your users sign on to AWS via single sign on, you can find out who has access to those AWS accounts via SSO:

```
Find User as U
  that ASSIGNED Application as App
  that CONNECTS aws_account as AWS
  RETURN
    U.displayName as User,
    App.tag.AccountName as IdP,
    App.displayName as ssoApplication,
    App.signOnMode as signOnMode,
    AWS.name as awsAccount
```

### 18.5.6 Using metadata

Filtering on metadata can often be useful in performing security analysis. The example below is used to find network or host entities that did *not* get ingested by an integration instance. In other words, these are entities that are likely “external” or “foreign” to the environment.

```
Find (Network|Host) with _IntegrationInstanceId = undefined
```

The following example finds all brand new code repos created within the last 48 hours:

```
Find CodeRepo with _beginOn > date.now-24hr and _version=1
```

For more details on metadata properties, see the [JupiterOne Data Model](#) documentation.

## 18.6 Advanced Notes and Use Cases

### 18.6.1 How aggregations are applied

There are three different ways for aggregations to be applied

- on the customer's subgraph (determined by the traversal that is run)
- on a portion of the customer's subgraph relative to a set of entities (groupings)
- on data for a single entity

The way aggregations happen are determined by what is requested via the query language's `return` clause.

#### Aggregations relative to a subgraph

If all selectors are aggregations, then all aggregations will be scoped to the entire traversal that the user has requested and not tied to individual entities.

Ex. `return count(user), count(team)`

#### Aggregations relative to a grouping

If selectors are provided that do not use an aggregation function, they will be used as a *grouping key*. This key will be used to apply the aggregations relative to the data chosen.

Ex. `return user, count(team)`

#### Aggregations relative to a single entity

If aggregations are provided that use the same selector as the grouping key, then aggregations will be scoped to values on each individual entity.

Ex. `return user, count(user._classes)`

### Aggregations Examples

#### The Simple Case

For example, with the following query,

```
find
  bitbucket_team as team
  that relates to
  bitbucket_user as user
return
  team.name, count (user)
```

the result will be:

```
{
  "type": "table",
  "data": [
    { "team.name": "team1", "count (user)": 25 },
    { "team.name": "team2", "count (user)": 5 }
  ]
}
```

In this case, the `team.name` acts as the key that groups aggregations together. So `count (user)` finds the count of users relative to each team.

### Multiple grouping keys

When there are return selectors that are not aggregating functions, the aggregating functions will be performed relative to the identifier that it is closer to in the traversal.

Example:

```
find
  bitbucket_project as project
  that relates to
  bitbucket_team as team
  that relates to
  bitbucket_user as user
return
  project.name, team.name, count (user)
```

The `count (user)` aggregation will be performed relative to the team, because the `team` traversal is closer to the `user` traversal in the query.

Example result:

```
{
  "type": "table",
  "data": [
    { "project.name": "JupiterOne", "team.name": "team1", "count (user)": 25 },
    { "project.name": "JupiterOne", "team.name": "team2", "count (user)": 5 },
    { "project.name": "Windbreaker", "team.name": "team2", "count (user)": 5 }
  ]
}
```

If the `return` statement is changed to this:

```
return
  project.name, count (user)
```

The `count (user)` aggregation will be performed relative to the project.



Example result:

```
{
  "type": "table",
  "data": [
    { "project.name": "JupiterOne", "count (user)": 50 },
    { "project.name": "Windbreaker", "count (user)": 5 }
  ]
}
```

### Examples relative to a single entity

If a selector is specified and an aggregating function is applied to that selector's source identifier in some way, aggregations will happen locally to the element.

Example:

```
find
  bitbucket_project as project
  that relates to
  bitbucket_team as team
  that relates to
  bitbucket_user as user
return
  project.name, count (project.aliases), team.name, count (user)
```

Example result:

```
{
  "type": "table",
  "data": [
    {
      "project.name": "JupiterOne",
      "count (project.aliases)": 1,
      "team.name": "team1",
      "count (user)": 25
    },
    {
      "project.name": "JupiterOne",
      "count (project.aliases)": 1,
      "team.name": "team2",
      "count (user)": 5
    },
    {
      "project.name": "Windbreaker",
      "count (project.aliases)": 5,
      "team.name": "team2",
      "count (user)": 5
    }
  ]
}
```



# CHAPTER 19

---

## JupiterOne API

---

JupiterOne platform exposes a number of public GraphQL endpoints.

**Base URL:** `https://api.us.jupiterone.io`

**Endpoint for query and graph operations:** `/graphql`

**Endpoint for alert and rules operations:** `/rules/graphql`

An experimental [node.js client](#) and [CLI](#) can be found on [Github](#).

## 19.1 Querying Entities and Relationships

**Endpoint:** `/graphql`

This query will allow you to run JIQL queries for fetching data.

```
query JIQL($query: String!, $variables: JSON, $dryRun: Boolean) {  
  queryV1(query: $query, variables: $variables, dryRun: $dryRun) {  
    type  
    data  
  }  
}
```

Variables:

```
{  
  "query": "find Person with _type=${type}",  
  "variables": {  
    "type": "employee"  
  },  
  "dryRun": true  
}
```

NOTE: there's also a `queryV1Tree` variant that has nice types for use when displaying graph data.

```
query J1QL($query: String!, $variables: JSON, $dryRun: Boolean) {
  queryV1Tree(query: $query, variables: $variables, dryRun: $dryRun) {
    type
    data {
      vertices {
        id
        edges {
          id
        }
      }
    }
  }
}
```

Variables:

```
{
  "query": "find Person with _type=${type} return tree",
  "variables": {
    "type": "employee"
  },
  "dryRun": true
}
```

variables

### 19.1.1 Fetching graph data

This query will be used for fetching graph data.

Note: ATM a canned query for IAM Role data is run. No input variables need to be provided.

```
query testQuery {
  queryGraph {
    vertices {
      id
      entity {
        _id
        _key
        _type
        _accountId
        _integrationName
        _integrationDefinitionId
        _integrationInstanceId
        _version
        _createdOn
        _beginOn
        _endOn
        _deleted
        displayName
      }
      properties
    }
    edges {
      id
      toVertexId
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    fromVertexId
    relationship {
      _id
      _key
      _type
      _accountId
      _integrationName
      _integrationDefinitionId
      _integrationInstanceId
      _version
      _createdOn
      _beginOn
      _endOn
      _deleted
      _fromEntityKey
      _toEntityKey
      displayName
    }
    properties
  }
}
}

```

### 19.1.2 Retrieving a single vertex by Id

This query will be used for fetch a vertex by it's id.

```

query VertexQuery($id: String!, $filters: VertexFilters) {
  vertex(id: $id, filters: $filters) {
    id
    entity {
      _id
      _key
      _type
      _accountId
      _integrationName
      _integrationDefinitionId
      _integrationInstanceId
      _version
      _createdOn
      _beginOn
      _endOn
      _deleted
      displayName
    }
    properties
  }
}

```

Variables:

```

{
  "id": "<a vertex id>",
  "filters": {
    "_id": "<an entity id>",

```

(continues on next page)

(continued from previous page)

```
"_key": "<an entity key>",
"_type": ["<a entity type>"],
"_class": ["<a entity class>"]
}
}
```

NOTE: Only one of the variables (`id` or `filters`) are required. Specifying both is allowed but is somewhat redundant unless you want to assert that the vertex with a specific `id` exists with a specific entity property.

`filters` is “well defined” right now (all allowed fields are shown in the variables above) but can be tweaked to allow for arbitrary properties in the future.

### 19.1.3 Fetching neighbors of a vertex

The `Vertex` type allows vertex and edge neighbors up to a certain depth to be retrieved using the `neighbors` field. The return type of the `neighbors` resolver is the same as that of a graph query.

```
query VertexQuery($id: String!, $depth: Int) {
  vertex(id: $id) {
    id
    entity {
      displayName
    }
    neighbors(depth: $depth) {
      vertices {
        id
        entity {
          displayName
        }
      }
      edges {
        id
        relationship {
          displayName
        }
      }
    }
  }
}
```

Variables:

NOTE: The depth that is supplied must be a value between 1 and 5 (inclusive)

```
{
  "id": "<a vertex id>",
  "depth": 5
}
```

### 19.1.4 Retrieving a edge by Id

This query will be used for fetch a vertex by it's id.

```

query VertexQuery($id: String!) {
  edge(id: $id, label: $id, filters: $id) {
    id
    relationship {
      _id
      _key
      _type
      _accountId
      _integrationName
      _integrationDefinitionId
      _integrationInstanceId
      _version
      _createdOn
      _beginOn
      _endOn
      _deleted
      _fromEntityKey
      _toEntityKey
      displayName
    }
    properties
  }
}

```

Variables:

```

{
  "id": "<an edge id>",
  "label": "<edge label>",
  "filters": {
    "_id": "<a relationship id>",
    "_key": "<a relationship key>",
    "_type": "<a relationship type>",
    "_class": "<a relationship class>"
  }
}

```

NOTE: Only one of the variables (id, label or filters) are required. Specifying a label and filters when an id is present is somewhat redundant but can be used to assert that the edge with a specific id exists with additional constraints.

Much like with the vertex query, filters is “well defined” right now (all allowed fields are shown in the variables above) but can be tweaked to allow for arbitrary properties in the future.

### 19.1.5 Fetching the count of entities via a `_type` and/or `_class`

For fetching the count of the latest entities the `_id`, `_key`, `_type` and `_class` fields can be supplied as filters. This query only counts the latest versions of entities matching the filter criteria.

```

query testQuery($filters: VertexFilters, $filterType: FilterType) {
  entityCount(filters: $filters, filterType: $filterType)
}

```

Note: Use field aliases to request the counts of multiple different entities. Also, the `filterType` argument is optional and defaults to the value `AND`

```
query testQuery {
  Users: entityCount(filters: { _class: ["User"] }, filterType: "AND")
  Repos: entityCount(filters: { _class: ["CodeRepo"] }, filterType: "OR")
}
```

Example result:

```
{
  "User": 40,
  "CodeRepo": 153
}
```

### 19.1.6 Fetching the count of all types and classes

```
query testQuery {
  allEntityCounts
}
```

Note: This resolver uses the JSON scalar as the return type.

Example result:

```
{
  "typeCounts": {
    "iam_user": 12,
    "iam_managed_policy": 10,
    "iam_role_policy": 10
  },
  "classCounts": {
    "User": 12,
    "AccessPolicy": 20
  }
}
```

### 19.1.7 Fetching the count of all types and classes

```
query testQuery ($classes: [String], filterType: FilterType) {
  typeCounts (classes: $classes, filterType: $filterType)
}
```

Note: This resolver uses the JSON scalar as the return type.

If OR is specified as the filter type, all of the types between the classes will be returned. By default, the query ANDs the classes and returns only the count of entities that have *all* of the specified classes.

Example result:

```
{
  "iam_user": 12,
  "iam_managed_policy": 10,
  "iam_role_policy": 10
}
```



### 19.1.8 Vertex full-text search

```

query testQuery($query: String!, $size: Int, $after: String) {
  queryText(query: $query, size: $size, after: $after) {
    vertices {
      id
      entity {
        _source
        _id
        _key
        _type
        _class
        _accountId
        _integrationName
        _integrationDefinitionId
        _integrationInstanceId
        _version
        _createdOn
        _beginOn
        _endOn
        _deleted
        displayName
      }
      properties
    }
    total
    pageInfo {
      endCursor
      hasNextPage
    }
  }
}

```

Variables:

```

{
  "query": "127.0.0.1"
}

```

### 19.1.9 Listing vertices via a `_type` and/or `_class`

For fetching the count of the latest entities the `_id`, `_key`, `_type` and `_class` fields can be supplied as filters. This query only returns the latest versions of entities matching the filter criteria.

```

query testQuery($filters: VertexFilters, $filterType: FilterType, $after: String) {
  listVertices(filters: $filters, filterType: $filterType, after: $after) {
    vertices {
      id
      entity {
        // entity details here
      }
      properties
    }
    total
    pageInfo {
      endCursor
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    hasNextPage
  }
}

```

Note: the `filterType` argument is optional and defaults to the value `AND`

Variables:

```

{
  "filters": {
    "_type": ["<an entity type>"],
    "_class": ["<an entity class>"]
  },
  "filterType": "<AND or OR>",
  "after": "the value of pageInfo.endCursor"
}

```

Example result

```

{
  "vertices": [
    {
      "id": "some-id",
      "entity": {
        "displayName": "Laptop-2345"
      }
    }
  ],
  "total": 1,
  "pageInfo": {
    "endCursor": "some-base64-cursor",
    "hasNextPage": true
  }
}

```

## 19.2 Entity Mutations

**Endpoint:** `/graphql`

### 19.2.1 Create Entity

```

mutation CreateEntity (
  $entityKey: String!
  $entityType: String!
  $entityClass: String!
  $timestamp: Long
  $properties: JSON
) {
  createEntity (
    entityKey: $entityKey,
    entityType: $entityType,
    entityClass: $entityClass,

```

(continues on next page)

(continued from previous page)

```

    timestamp: $timestamp,
    properties: $properties
  ) {
    entity {
      _id
      ...
    }
    vertex {
      id,
      entity {
        _id
        ...
      }
      properties
    }
  }
}

```

Variables:

```

{
  "entityKey": "<an entity key>",
  "entityType": "<an entity type>",
  "entityClass": "<an entity class>",
  "timestamp": 1529329792552,
  "properties": {
    // Custom properties on the Entity
    ...
  }
}

```

## 19.2.2 Updating Entity

```

mutation UpdateEntity (
  $entityId: String!
  $timestamp: Long
  $properties: JSON
) {
  updateEntity (
    entityId: $entityId,
    timestamp: $timestamp,
    properties: $properties
  ) {
    entity {
      _id
      ...
    }
    vertex {
      id,
      entity {
        _id
        ...
      }
      properties
    }
  }
}

```

(continues on next page)

(continued from previous page)

```
}  
}
```

Variables:

```
{  
  "entityId": "<an entity Id (entity._id)>",  
  "timestamp": 1529329792552,  
  "properties": {  
    // Custom properties to get updated  
    ...  
  }  
}
```

### 19.2.3 Deleting Entity

```
mutation DeleteEntity (  
  $entityId: String!  
  $timestamp: Long  
) {  
  deleteEntity (  
    entityId: $entityId,  
    timestamp: $timestamp,  
  ) {  
    entity {  
      _id  
      ...  
    }  
    vertex {  
      id,  
      entity {  
        _id  
        ...  
      }  
      properties  
    }  
  }  
}
```

Variables:

```
{  
  "entityId": "<an entity Id (entity._id)>",  
  "timestamp": 1529329792552  
}
```

## 19.3 Relationship Mutations

**Endpoint:** /graphql

### 19.3.1 Create Relationship

```
mutation CreateRelationship (
  $relationshipKey: String!
  $relationshipType: String!
  $relationshipClass: String!
  $fromEntityId: String!
  $toEntityId: String!
  $timestamp: Long
  $properties: JSON
) {
  createRelationship (
    relationshipKey: $relationshipKey,
    relationshipType: $relationshipType,
    relationshipClass: $relationshipClass,
    fromEntityId: $fromEntityId,
    toEntityId: $toEntityId,
    timestamp: $timestamp,
    properties: $properties
  ) {
    relationship {
      _id
      ...
    }
    edge {
      id
      toVertexId
      fromVertexId
      relationship {
        _id
        ...
      }
      properties
    }
  }
}
```

Variables:

```
{
  "relationshipKey": "<a relationship key>",
  "relationshipType": "<a relationship type>",
  "relationshipClass": "<a relationship class>",
  "fromEntityId": "<the _id of the from entity>",
  "toEntityId": "<the _id of the to entity>",
  "timestamp": 1529329792552,
  "properties": {
    // Custom properties on the relationship
    ...
  }
}
```

### 19.3.2 Update Relationship

```
mutation UpdateRelationship (
  $relationshipId: String!
  $timestamp: Long
  $properties: JSON
) {
  updateRelationship (
    relationshipId: $relationshipId,
    timestamp: $timestamp,
    properties: $properties
  ) {
    relationship {
      _id
      ...
    }
    edge {
      id
      toVertexId
      fromVertexId
      relationship {
        _id
        ...
      }
      properties
    }
  }
}
```

Variables:

```
{
  "relationshipId": "<a relationship Id (relationship._id)>",
  "timestamp": 1529329792552,
  "properties": {
    // Custom properties to get updated
    ...
  }
}
```

### 19.3.3 Delete Relationship

```
mutation DeleteRelationship (
  $relationshipId: String!
  $timestamp: Long
) {
  deleteRelationship (
    relationshipId: $relationshipId,
    timestamp: $timestamp,
  ) {
    relationship {
      _id
      ...
    }
    edge {
      id
      toVertexId
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    fromVertexId
    relationship {
      _id
      ...
    }
    properties
  }
}
}

```

Variables:

```

{
  "relationshipId": "<a relationship Id (relationship._id)>",
  "timestamp": 1529329792552
}

```

## 19.4 Building CSV Report

**Endpoint:** /graphql

```

mutation BuildCsv(
  $filters: VertexFilters
  $propertyFilters: JSON
  $filterType: FilterType
) {
  buildCsv(
    filters: $filters
    propertyFilters: $propertyFilters
    filterType: $filterType
  ) {
    stateFileUrl
  }
}

```

Variables:

```

{
  "filters": {
    "_type": ["<an entity type>"],
    "_class": ["<an entity class>"]
  },
  "filterType": "<AND or OR>",
  "propertyFilters": {
    ...
  }
}

```

## 19.5 Alert and Rules Operations

**Endpoint:** /rules/graphql

## 19.5.1 Create an alert rule

```
mutation CreateQuestionRuleInstance (
  $instance: CreateQuestionRuleInstanceInput!
) {
  createQuestionRuleInstance (
    instance: $instance
  ) {
    id
    name
    description
    version
    pollingInterval
    question {
      queries {
        query
        version
      }
    }
    operations {
      when
      actions
    }
    outputs
  }
}
```

variables:

```
{
  "instance": {
    "name": "unencrypted-prod-data",
    "description": "Data stores in production tagged critical and unencrypted",
    "version": "v1",
    "pollingInterval": "ONE_DAY",
    "outputs": [
      "alertLevel"
    ],
    "operations": [
      {
        "when": {
          "type": "FILTER",
          "version": 1,
          "condition": [
            "AND",
            [ "queries.unencryptedCriticalData.total", "!=" , 0 ]
          ]
        },
        "actions": [
          {
            "type": "SET_PROPERTY",
            "targetProperty": "alertLevel",
            "targetValue": "CRITICAL"
          },
          {
            "type": "CREATE_ALERT"
          }
        ]
      }
    ]
  }
}
```

(continues on next page)



(continued from previous page)

```

    ]
  }
],
"question": {
  "queries": [
    {
      "query": "Find DataStore with (production=true or tag.Production=true) and
↪classification='critical' and encrypted!=true as d return d.tag.AccountName as
↪Account, d.displayName as UnencryptedDataStores, d._type as Type, d.encrypted as
↪Encrypted",
      "version": "v1",
      "name": "unencryptedCriticalData"
    }
  ]
}
}
}
}

```

Note that the recommended interval for query based alert rules (aka a question) is ONE\_DAY. Supported intervals are THIRTY\_MINUTES, ONE\_HOUR, and ONE\_DAY.

### 19.5.2 Update an alert rule

```

mutation UpdateQuestionRuleInstance (
  $instance: UpdateQuestionRuleInstanceInput!
) {
  updateQuestionRuleInstance (
    instance: $instance
  ) {
    id
    name
    description
    version
    pollingInterval
    question {
      queries {
        query
        version
      }
    }
    operations {
      when
      actions
    }
    outputs
  }
}

```

variables:

```

{
  "instance": {
    "id": "b1c0f75d-770d-432a-95f5-6f59b4239c72",
    "name": "unencrypted-prod-data",
    "description": "Data stores in production tagged critical and unencrypted",

```

(continues on next page)

(continued from previous page)

```

    "version": "v1",
    "pollingInterval": "ONE_DAY",
    "outputs": [
      "alertLevel"
    ],
    "operations": [
      {
        "when": {
          "type": "FILTER",
          "version": 1,
          "condition": [
            "AND",
            [ "queries.unencryptedCriticalData.total", "!=", 0 ]
          ]
        },
        "actions": [
          {
            "type": "SET_PROPERTY",
            "targetProperty": "alertLevel",
            "targetValue": "CRITICAL"
          },
          {
            "type": "CREATE_ALERT"
          }
        ]
      }
    ],
    "question": {
      "queries": [
        {
          "query": "Find DataStore with (production=true or tag.Production=true) and_
↪classification='critical' and encrypted!=true as d return d.tag.AccountName as_
↪Account, d.displayName as UnencryptedDataStores, d._type as Type, d.encrypted as_
↪Encrypted",
          "version": "v1",
          "name": "unencryptedCriticalData"
        }
      ]
    }
  }
}

```

Note that the only difference here for update is the "id" property associated with the rule instance. All settings of a rule instance can be modified.

### 19.5.3 Delete an alert rule

```

mutation DeleteRuleInstance ($id: ID!) {
  deleteRuleInstance (
    id: $id
  ) {
    id
  }
}

```

variables:

```
{
  "id": "b1c0f75d-770d-432a-95f5-6f59b4239c72"
}
```

Note that deleting an alert rule this way will **not** dismiss active alerts already triggered by this rule. It is recommended to **Disable** a rule in the alerts app UI instead of deleting one.

### 19.5.4 Trigger an alert rule on demand

```
mutation EvaluateRuleInstance ($id: ID!) {
  evaluateRuleInstance (
    id: $id
  ) {
    outputs {
      name
      value
    }
  }
}
```

variables:

```
{
  "id": "b1c0f75d-770d-432a-95f5-6f59b4239c72"
}
```

## 19.6 Question Operations

**Endpoint:** /graphql

### 19.6.1 Create a Question

```
mutation CreateQuestion($question: CreateQuestionInput!) {
  createQuestion(question: $question) {
    id
    title
    description
    queries {
      name
      query
      version
    }
    variables {
      name
      required
      default
    }
    compliance {
      standard
      requirements
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    accountId
    integrationDefinitionId
  }
}

```

variables:

```

{
  "question": {
    "title": "What are my production resources?",
    "tags": ["SecOps"],
    "description": "Returns a list of all production entities.",
    "queries": [
      {
        "name": "prodresources",
        "query": "Find * with tag.Production=true"
      }
    ],
    "compliance": [
      {
        "standard": "HITRUST CSF",
        "requirements": ["10.k"]
      }
    ]
  }
}

```

**Notes on “named queries”:**

- name field is optional
- name should be a single word without special characters
- queries named good, bad, and unknown are used to determine gaps/issues and to perform continuous compliance assessment

**19.6.2 Update a question**

```

mutation UpdateQuestion($id: ID!, $update: QuestionUpdate!) {
  updateQuestion(id: $id, update: $update) {
    id
    title
    description
    queries {
      name
      query
      version
    }
    variables {
      name
      required
      default
    }
    compliance {
      standard
      requirements
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    }
    accountId
    integrationDefinitionId
  }
}

```

variables:

```

{
  "id": "sj3j9f0j2ndlsj300swdjfjs",
  "update": {
    "title": "What are my production resources?",
    "tags": ["SecOps"],
    "description": "Returns a list of all production entities.",
    "queries": [
      {
        "name": "prodresources",
        "query": "Find * with tag.Production=true"
      }
    ],
    "compliance": [
      {
        "standard": "HITRUST CSF",
        "requirements": ["10.k"]
      }
    ]
  }
}

```

Note that the only difference here for update is the "id" property associated with the question.

### 19.6.3 Delete a question.

```

mutation DeleteQuestion($id: ID!) {
  deleteQuestion(id: $id) {
    id
    title
    description
    queries {
      query
      name
      version
    }
    variables {
      name
      required
      default
    }
    tags
    accountId
    integrationDefinitionId
  }
}

```

variables:

```
{  
  "id": "slj3098s03j-i2ojd0j2-sjkkdjf"  
}
```

### 20.1 Are my assets tracked? How many entities are there?

Returns the current count of total assets/entities tracked in JupiterOne - either automatically ingested via integrations or manually entered through the Asset Inventory app or API.

**Tags:** compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

#### 20.1.1 Queries

- Find \* **as** e **return** count (e)

#### 20.1.2 Compliance Mappings

**CIS Controls:** 1.1, 1.2, 1.4, 1.5, 2.1, 2.3, 2.4, 2.5

**HITRUST CSF:** 07.a

### 20.2 What are my production information assets and their owners and classification?

Returns a list of Application, Code Repo, Workload, Function, Host, Device, Database, Data Store entities along with their owner and classification.

**Tags:** compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

## 20.2.1 Queries

- Find (Application|CodeRepo|Workload|Function|Host|Device|Database|DataStore) **as** `asset` **return** `asset._class, asset._type, asset.displayName, asset.tag, AccountName, asset.owner, asset.classification`

## 20.2.2 Compliance Mappings

**CIS Controls:** 1.4, 1.5

**HITRUST CSF:** 07.a

**PCI DSS:** 2.4

## 20.3 What are my production information assets?

Returns a list of production Applications, Code Repos, Workloads, Functions, Hosts, Devices, Databases, and Data Stores.

**Tags:** compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 20.3.1 Queries

- Find (Application|CodeRepo|Workload|Function|Host|Device|Database|DataStore) **with** `tag.Production=true`

### 20.3.2 Compliance Mappings

**CIS Controls:** 1.4, 1.5

**HITRUST CSF:** 07.a

**PCI DSS:** 2.4

## 20.4 What are my production systems and servers?

Returns a list of production Workloads, Functions, and Hosts.

**Tags:** compliance, HIPAA, HITRUST CSF, PCI DSS

### 20.4.1 Queries

- Find (Workload|Function|Host) **with** `tag.Production=true`



## 20.4.2 Compliance Mappings

**CIS Controls:** 1.4, 1.5

**HITRUST CSF:** 07.a

**PCI DSS:** 2.4

## 20.5 What are my production data stores and databases?

Returns a list of production Databases and Data Stores.

**Tags:** compliance, HIPAA, HITRUST CSF, PCI DSS

### 20.5.1 Queries

- Find (Database|DataStore) **with** tag.Production=true

### 20.5.2 Compliance Mappings

**CIS Controls:** 1.4, 1.5

**HITRUST CSF:** 07.a

**PCI DSS:** 2.4

## 20.6 What are my production resources?

Returns a list of all production entities.

**Tags:** SecOps

### 20.6.1 Queries

- Find \* **with** tag.Production=true

## 20.7 What applications and operating systems are in use?

Returns a list of software applications and operating systems.

**Tags:** SecOps, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 20.7.1 Queries

- Find Application

- Find Host `with platform!=undefined as h return h.platform, h.platformName, h.osName, h.osVersion, h.osDetails ORDER BY h.platform`

## 20.7.2 Compliance Mappings

**CIS Controls:** 2.3

**HITRUST CSF:** 07.a

**PCI DSS:** 2.4

## 20.8 What are my production applications?

Returns a list of production Applications.

**Tags:** SecOps, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 20.8.1 Queries

- Find Application `with tag.Production=true`

### 20.8.2 Compliance Mappings

**CIS Controls:** 2.1

**HITRUST CSF:** 07.a

**PCI DSS:** 2.4

## 20.9 Do I have proper vendor support for my software applications?

Returns a list of applications and their vendors. Vendors should have support agreement and/or SLA attached.

**Tags:** compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 20.9.1 Queries

- Find Application `as app that CONNECTS Account that RELATES TO Vendor as v return app.displayName as app, v.name as vendor, v.linkToSLA, v.linkToMSA`
- Find Application that RELATES TO Vendor
- Find Application

## 20.9.2 Compliance Mappings

**CIS Controls:** 2.2

**HITRUST CSF:** 05.i

**PCI DSS:** 2.4

## 20.10 Who are the new hires within the last 12 months?

Returns all employees added in the last 12 months.

**Tags:** compliance, HIPAA, HITRUST CSF

### 20.10.1 Queries

- Find employee **with** `_createdOn > date.now-12months`

### 20.10.2 Compliance Mappings

**HITRUST CSF:** 02.a, 02.b, 02.c, 02.e

For each of the new hire, you should provide supporting evidence to meet requirements for pre-hire screening and onboarding. Links to these evidence may be added to each employee/Person entity (e.g. linking to a SharePoint document or a Jira issue).

## 20.11 What business applications are we using?

Finds all application entities that does not have associate code repos. It is assumed that an application with code repos is a commercial-facing application or part of your custom development.

**Tags:** SecOps

### 20.11.1 Queries

- Find Application that `!has CodeRepo`

## 20.12 What changed in my environment in the last 24 hours?

Find all entities that were updated with a timestamp within the last 24 hours.

**Tags:** SecOps

### 20.12.1 Queries

- Find \* **with** `_beginOn > date.now-24hrs`

## 20.13 What was added to my environment in the last 24 hours?

Find all entities that were created within the last 24 hours.

**Tags:** SecOps

### 20.13.1 Queries

- Find \* **with** \_createdOn > date.now-24hrs

### 21.1 Find anything that allows public access to everyone.

Returns all entities that have an 'ALLOWS' permission directly to the global 'everyone' entity.

**Tags:** access, SecOps

#### 21.1.1 Queries

- Find Everyone that ALLOWS \* **return** tree
- Find Everyone that ALLOWS \* **as** resource **return** resource.tag.AccountName, resource.  
→\_type, resource.name, resource.classification, resource.description, resource.  
→webLink

### 21.2 Show me the current password policy and compliance status.

Returns all password policies and details. The second query finds all ControlPolicy entities with 'password' as a search string and the entity resources that each matched ControlPolicy evaluates – this works if you have AWS Config enabled to evaluate your account password policy.

**Tags:** access, compliance, HIPAA, HITRUST CSF

#### 21.2.1 Queries

- Find PasswordPolicy

- `'password' with _class='ControlPolicy' as p` that evaluates `* as e return p.`  
↳ `displayName as Policy, e.displayName as TargetEnv, p.compliant as Compliant, p.`  
↳ `inputParameters as Details`

## 21.2.2 Compliance Mappings

**HITRUST CSF:** 01.d, 01.p, 01.r

## 21.3 Are there external users with access to our systems?

Returns all User entities that are a Person (i.e. users accounts owned by an individual) who is not employed by your organization (i.e. the Root entity). Note that the query finds relationships bidirectionally. *!EMPLOYS* here translates to 'is not employed by'. The second query returns user accounts owned by contractors.

**Tags:** access, compliance, HIPAA, HITRUST CSF

### 21.3.1 Queries

- Find User that IS Person that !EMPLOYS Root
- Find User `as u` that IS Person `as p` where `u.userType='contractor' or p.`  
↳ `employeeType='contractor'`

### 21.3.2 Compliance Mappings

**HITRUST CSF:** 01.j, 05.i

Organizations must show due diligence managing the information security risks posed by external parties. This includes identifying and managing the access to data/systems by external parties such as service providers and contractors.

## 21.4 Who has been assigned permissions with administrator/privileged access?

Returns policies with admin access and the entities that are assigned each policy. Note that in most cases, integrations set the 'admin' boolean to true if the policy name contains the keyword 'admin'.

**Tags:** access, SecOps, compliance, CIS Controls, HITRUST CSF, PCI DSS

### 21.4.1 Queries

- Find AccessPolicy `with admin=true as policy` that ASSIGNED `* as e return policy.`  
↳ `displayName, policy.webLink, e.displayName, e.webLink`

## 21.4.2 Compliance Mappings

**CIS Controls:** 4.1

**HITRUST CSF:** 01.c

**PCI DSS:** 7.1, 7.3, 8.1, 8.3, 8.7

## 21.5 Who has access to what systems/resources?

Returns all users and their access.

**Tags:** access, SecOps, compliance, HIPAA, HITRUST CSF

### 21.5.1 Queries

- Find (User|Person) **as** u that (ASSIGNED|TRUSTS|HAS|OWNS)   
 ↳ (Application|AccessPolicy|AccessRole|Account|Device|Host) **as** a **return** u.  
 ↳ displayName, u.\_type, u.username, u.email, a.\_type, a.displayName, a.tag.  
 ↳ AccountName order by u.displayName

### 21.5.2 Compliance Mappings

**HITRUST CSF:** 01.e, 01.v

Access should be reviewed at least quarterly and whenever an employee's status changes.

## 21.6 Who owns which user accounts?

Returns all User entities (i.e. user accounts) that are mapped to a Person.

**Tags:** access, SecOps, compliance, HIPAA, HITRUST CSF

### 21.6.1 Queries

- Find User that IS Person

### 21.6.2 Compliance Mappings

**HITRUST CSF:** 01.e

Access should be reviewed at least quarterly and whenever an employee's status changes.

## 21.7 What are the shared/generic/service accounts or access roles? (Including user accounts that are not individually owned)

Returns all AccessRoles (e.g `aws_iam_role`) that trusts a service (i.e. can be assumed/used by a service). Additionally, the second query returns all User entities (i.e. user accounts) that are NOT mapped to a Person.

**Tags:** `access`, `SecOps`

### 21.7.1 Queries

- Find AccessRole that TRUSTS Service
- Find User with `mfaEnabled != true` that !IS Person

## 21.8 Did we remove all access from employees who left?

Returns any User entity (i.e. user account) that is mapped to a Person no longer employed by your organization (Root). If access is properly configured and mapped in JupiterOne, this query should return nothing.

**Tags:** `access`, `SecOps`, `compliance`, `HIPAA`, `HITRUST CSF`

### 21.8.1 Queries

- Find User that IS Person that !EMPLOYS Root

### 21.8.2 Compliance Mappings

**HIPAA:**

**HITRUST CSF:** 02.i

## 21.9 Which user accounts do not have multi-factor authentication enabled?

Returns all user entities that do not have the *mfaEnabled* property set to true and have no MFA device assigned/in use.

**Tags:** `access`, `SecOps`, `compliance`, `CIS Controls`, `PCI DSS`

### 21.9.1 Queries

- Find User with `mfaEnabled != true` that !(ASSIGNED|USES|HAS) mfa\_device
- Find User **with** `mfaEnabled = true`



- `Find User that (ASSIGNED|USES|HAS) mfa_device`

## 21.9.2 Compliance Mappings

**CIS Controls:** 4.5, 12.11, 16.3

**PCI DSS:** 8.2, 8.3



## 22.1 What are the code repos for a particular application or project?

Returns all code repos connected to a given Application or Project. You will need to edit this Application/Project name to match yours.

**Tags:** app, dev, DevOps

### 22.1.1 Queries

- Find CodeRepo that relates to (Application|Project) **with** name='JupiterOne'

## 22.2 Were there any Code Repos added in the last 24 hours?

Returns all code repos whose first version was created within the last 24 hours.

**Tags:** app, dev, DevOps

### 22.2.1 Queries

- Find CodeRepo **with** \_beginOn > date.now-24hr **and** \_version=1

## 22.3 Who are the most recent contributors to this repo?

Returns the authors of the last five pull requests to a give code repo. Replace the repo name with the name of the repo you are searching for.

**Tags:** app, dev, DevOps

### 22.3.1 Queries

- Find User **as** u that OPENED PR **as** PR that HAS CodeRepo **with** name='repo-name' **as** **↪**repo **return** u.displayName, u.username, PR.displayName, PR.name, PR.\_createdOn, **↪**repo.name ORDER BY PR.\_createdOn LIMIT 5

## 22.4 Which PRs did this developer open in the last 5 days?

Returns a list of pull requests opened by the given developer. Replace the full text search string (at the very beginning of query in quotes) with the name or github/gitlab/bitbucket username of the developer.

**Tags:** app, dev, DevOps

### 22.4.1 Queries

- 'Charlie' that OPENED PR **with** \_createdOn > date.now - 5days **as** PR **return** PR.  
**↪**displayName, PR.name

## 23.1 Are there any non-public data stores incorrectly configured with public access to everyone?

Find all Data Stores that are marked publicly accessible or have an ALLOWS relationship to everyone, unless the data store is specifically tagged as 'public' per data classification.

**Tags:** data, SecOps

### 23.1.1 Queries

- Find DataStore **with** (classification!='public' **or** classification=undefined) that **↳** ALLOWS everyone
- Find DataStore **with** (classification!='public' **or** classification=undefined) **and** **↳** public=true

## 23.2 Which data stores do not have proper classification tags?

Find Data Stores across my entire environment that are not tagged with classification.

**Tags:** data, SecOps

### 23.2.1 Queries

- Find DataStore **with** classification='' **or** classification=undefined

## 23.3 What is the inventory of my sensitive data stores?

Find Data Stores that are tagged as ‘sensitive’ or ‘confidential’ or ‘critical’.

**Tags:** data, SecOps, compliance, CIS Controls, HITRUST CSF, PCI DSS

### 23.3.1 Queries

- Find DataStore **with** classification='sensitive' **or** classification='confidential' **↪**  
**↪ or** classification='critical'
- Find DataStore **with** (classification='' **or** classification=undefined) **and** **↪**  
**↪** (production=true **or** tag.Production=true)

### 23.3.2 Compliance Mappings

**CIS Controls:** 13.1

**HITRUST CSF:** 07.d, 07.e

## 23.4 Which production data stores do not have proper classification tags?

Find Data Stores in production that are not tagged with classification.

**Tags:** data, SecOps

### 23.4.1 Queries

- Find DataStore **with** (classification='' **or** classification=undefined) **and** **↪**  
**↪** (production=true **or** tag.Production=true)

## 23.5 Is there any known confidential or critical data outside of production?

Returns a list of Data Stores tagged with ‘confidential’ or ‘critical’ classification label outside of production environments. Confidential or critical data should remain inside production environments.

**Tags:** data, SecOps

### 23.5.1 Queries

- Find DataStore **with** (classification='confidential' **or** classification='critical') **↪**  
**↪ and** (tag.Production!=true **or** production!=true)

- Find DataStore **with** (classification='confidential' **or** classification='critical')  
 ↳ that RELATES TO (Account|Service) **with** (tag.Production!=true **or** production!  
 ↳ =true)

## 23.6 Evidence of data-at-rest encryption for production servers

Returns all data volumes (disks) attached to production hosts and their encryption status.

**Tags:** data, compliance, HIPAA, HITRUST CSF

### 23.6.1 Queries

- Find Host **with** (tag.Production=true **or** production=true **or** tag.ePHI=true **or** tag.  
 ↳ PHI=true **or** tag.PII=true) **as** h that uses DataStore **with** encrypted=true **as** d  
 ↳ **return** h.tag.AccountName **as** Account, h.displayName **as** Hostname, d.displayName  
 ↳ **as** EncryptedDisks, d.encrypted **as** Encrypted
- Find Host **with** (tag.Production=true **or** production=true **or** tag.ePHI=true **or** tag.  
 ↳ PHI=true **or** tag.PII=true) **as** h that uses DataStore **with** encrypted!=true **as** d  
 ↳ **return** h.tag.AccountName **as** Account, h.displayName **as** Hostname, d.displayName  
 ↳ **as** UnencryptedDisks, d.encrypted **as** Encrypted

### 23.6.2 Compliance Mappings

**HITRUST CSF:** 06.d, 07.e

Data volumes containing ePHI must be encrypted. If unencrypted disks are being used, as returned by the second query, you must remediate.

## 23.7 Is my production or PHI/PII data stores encrypted?

Returns a list of Data Stores (such as AWS S3 buckets) tagged as production or as containing ePHI/PHI/PII data and their encryption status.

**Tags:** data, compliance, HIPAA, HITRUST CSF

### 23.7.1 Queries

- Find DataStore **with** (production=true **or** tag.Production=true **or** tag.ePHI=true **or**  
 ↳ tag.PHI=true **or** tag.PII=true) **and** encrypted=true **as** d **return** d.tag.AccountName  
 ↳ **as** Account, d.displayName **as** EncryptedDataStores, d.\_type **as** Type, d.encrypted  
 ↳ **as** Encrypted
- Find DataStore **with** (production=true **or** tag.Production=true **or** tag.ePHI=true **or**  
 ↳ tag.PHI=true **or** tag.PII=true) **and** encrypted!=true **as** d **return** d.tag.AccountName  
 ↳ **as** Account, d.displayName **as** UnencryptedDataStores, d.\_type **as** Type, d.  
 ↳ encrypted **as** Encrypted

## 23.7.2 Compliance Mappings

**HITRUST CSF:** 06.d, 07.e

Data stores containing ePHI must be encrypted. If unencrypted data stores are found, as returned by the second query, you must remediate.

## 23.8 Is my critical data in production encrypted?

Returns a list of Data Stores (such as AWS S3 buckets) in that are tagged as 'critical' in production environments and their encryption status. Replace the classification label to match your organization's data classification model/policy.

**Tags:** data, SecOps

### 23.8.1 Queries

- Find DataStore **with** (production=true **or** tag.Production=true) **and** classification=  
→ 'critical' **and** encrypted=true **as** d **return** d.tag.AccountName **as** Account, d.  
→ displayName **as** EncryptedDataStores, d.\_type **as** Type, d.encrypted **as** Encrypted
- Find DataStore **with** (production=true **or** tag.Production=true) **and** classification=  
→ 'critical' **and** encrypted!=true **as** d **return** d.tag.AccountName **as** Account, d.  
→ displayName **as** UnencryptedDataStores, d.\_type **as** Type, d.encrypted **as** Encrypted

## 23.9 Is there unencrypted ePHI or PII?

Returns any Data Store tagged as ePHI that is not encrypted.

**Tags:** data, SecOps

### 23.9.1 Queries

- Find DataStore **with** (tag.PHI=true **or** tag.ePHI=true **or** tag.PII=true) **and**   
→ encrypted=false



### 24.1 Whose endpoint is out of compliance?

Find employees whose endpoint device did not meet your defined configuration compliance.

**Tags:** endpoint, SecOps

#### 24.1.1 Queries

- Find Person that OWNS Device that (MONITORS|MANAGES|PROTECTS) HostAgent **with** `↪compliant=false`

### 24.2 Is there anybody who does not have a user endpoint device (e.g. a laptop or workstation)?

Find employees who do not have an endpoint device being mapped and tracked in the system.

**Tags:** endpoint, SecOps

#### 24.2.1 Queries

- Find Person that !OWNS (user\_endpoint|laptop|workstation|desktop)

## 24.3 What is the configuration and compliance status of my endpoint devices?

Returns all endpoint Devices and their relevant compliance status, such as disk encryption, host firewall, auto-update, and screensaver protection. Secondly, returns hosts or devices that do not have either an endpoint configuration or compliance agent protection.

**Tags:** endpoint, compliance, HIPAA, HITRUST CSF

### 24.3.1 Queries

- Find HostAgent **with** compliant=true that (MONITORS|MANAGES) (user\_↵endpoint|workstation|laptop|desktop|tablet)
- Find HostAgent **with** compliant=false that (MONITORS|MANAGES) (user\_↵endpoint|workstation|laptop|desktop|tablet)
- Find (user\_endpoint|workstation|laptop|desktop|tablet) that !(MONITORS|MANAGES) ↵↵HostAgent with function='endpoint-compliance' or function='endpoint-configuration'

### 24.3.2 Compliance Mappings

**HITRUST CSF:** 01.x, 01.y, 06.d, 10.h

Systems shall be configured according to a current security baseline. Use full-disk encryption to protect the confidentiality of information on laptops and other mobile devices. Also, enable local host firewall, auto install of OS patches/updates, and screen lock with password protection.

## 24.4 Is there malware protection for all endpoints?

Returns all endpoint Devices and their anti-malware Host Agents. Secondly, returns devices that do not have anti-malware agent protection.

**Tags:** endpoint, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.4.1 Queries

- Find HostAgent **with** function='anti-malware' **as** a that PROTECTS (user\_↵endpoint|workstation|laptop|desktop|server) **as** h **return** a.displayName, h.displayName, h.owner
- Find (user\_endpoint|workstation|laptop|desktop|server) that !PROTECTS HostAgent ↵↵with function='anti-malware'

## 24.4.2 Compliance Mappings

**CIS Controls:** 8.1

**HITRUST CSF:** 10.h

**PCI DSS:** 5.1

## 24.5 Is there protection for all user endpoints/devices?

Returns all user endpoints and their Host Agents. Secondly, returns user endpoints that do not have any monitoring or protection by a host agent.

**Tags:** endpoint, compliance, HIPAA, HITRUST CSF

### 24.5.1 Queries

- Find HostAgent that (PROTECTS|MANAGES|MONITORS) user\_endpoint
- Find user\_endpoint that !(PROTECTS|MANAGES|MONITORS) HostAgent

### 24.5.2 Compliance Mappings

**HITRUST CSF:** 01.g

## 24.6 Is operating system patching and auto update enabled on endpoint hosts?

Returns all user endpoints that has either enabled or disabled automatic operating system updates in two lists.

**Tags:** endpoint, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.6.1 Queries

- Find HostAgent **with** automaticOsUpdates='ON' **and** automaticSecurityUpdates='ON' **as**   
 ↪ agent that (PROTECTS|MONITORS|MANAGES) user\_endpoint **as** device **return** device.   
 ↪ displayName, device.owner, agent.automaticOsUpdates, agent.   
 ↪ automaticSecurityUpdates
- Find HostAgent **with** automaticOsUpdates='OFF' **or** automaticSecurityUpdates='OFF' **as**   
 ↪ agent that (PROTECTS|MONITORS|MANAGES) user\_endpoint **as** device **return** device.   
 ↪ displayName, device.owner, agent.automaticOsUpdates, agent.   
 ↪ automaticSecurityUpdates

## 24.6.2 Compliance Mappings

**CIS Controls:**

**HITRUST CSF:** 01.x, 01.y

**PCI DSS:**

## 24.7 Is application patching and auto update enabled on endpoint hosts?

Returns all user endpoints that has either enabled or disabled automatic application updates in two lists.

**Tags:** endpoint, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.7.1 Queries

- Find HostAgent **with** automaticAppUpdates='ON' **as** agent that  
→ (PROTECTS|MONITORS|MANAGES) user\_endpoint **as** device **return** device.displayName,   
→ device.owner, agent.automaticAppUpdates
- Find HostAgent **with** automaticAppUpdates='OFF' **as** agent that  
→ (PROTECTS|MONITORS|MANAGES) user\_endpoint **as** device **return** device.displayName,   
→ device.owner, agent.automaticAppUpdates

## 24.7.2 Compliance Mappings

**CIS Controls:**

**HITRUST CSF:** 01.x, 01.y

**PCI DSS:**

## 24.8 Are my servers and systems protected by hosted-based fire-wall?

Returns all user endpoints that has local firewall turned on or off in two lists.

**Tags:** infra, host, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.8.1 Queries

- Find HostAgent **with** firewall='ON' **as** agent that (PROTECTS|MONITORS|MANAGES) user\_  
→ endpoint **as** device **return** device.displayName, device.owner, agent.firewall
- Find HostAgent **with** firewall!='ON' **as** agent that (PROTECTS|MONITORS|MANAGES) user\_  
→ endpoint **as** device **return** device.displayName, device.owner, agent.firewall

## 24.8.2 Compliance Mappings

**CIS Controls:**

**HITRUST CSF:** 01.x, 01.y

**PCI DSS:** 1.4

## 24.9 Are there security agents monitoring and protecting my endpoint hosts/devices?

Returns all endpoint Hosts or Devices and their Host Agents. Secondly, returns devices that do not have any monitoring or protection by a host agent.

**Tags:** endpoint, compliance, HIPAA, HITRUST CSF

### 24.9.1 Queries

- Find HostAgent **as** a that (PROTECTS|MANAGES|MONITORS) (Host|Device) **as** h **return** a.  
↪ displayName, a.\_type, a.function, h.displayName, h.owner
- Find (Host|Device) with \_type!='mapped\_entity' that !(PROTECTS|MANAGES|MONITORS) ↪  
↪ HostAgent

## 24.9.2 Compliance Mappings

**HITRUST CSF:** 09.ab

## 24.10 Is operating system patching and auto update enabled on endpoint hosts?

Returns all endpoint Hosts that has either enabled or disabled automatic operating system updates in two lists.

**Tags:** endpoint, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.10.1 Queries

- Find (Host|HostAgent) **with** automaticOsUpdates='ON' **and** automaticSecurityUpdates=  
↪ 'ON'
- Find (Host|HostAgent) **with** automaticOsUpdates='OFF' **or** automaticSecurityUpdates=  
↪ 'OFF'

## 24.10.2 Compliance Mappings

**CIS Controls:** 3.4

**HITRUST CSF:** 01.y, 10.m

**PCI DSS:** 6.2

## 24.11 Is application patching and auto update enabled on endpoint hosts?

Returns all endpoint Hosts that has either enabled or disabled automatic application updates in two lists.

**Tags:** endpoint, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.11.1 Queries

- Find (Host|HostAgent) **with** automaticAppUpdates='ON'
- Find (Host|HostAgent) **with** automaticAppUpdates='OFF'

### 24.11.2 Compliance Mappings

**CIS Controls:** 3.5

**HITRUST CSF:** 01.y, 10.m

**PCI DSS:** 6.2

## 24.12 Are my servers and systems protected by hosted-based firewall?

Lists Firewall instances and the Hosts they each protect. Additionally, to identify gaps, returns a list of active Host or Device entities that do not have local firewall enabled or a PROTECTS relationship connection to a Firewall entity.

**Tags:** infra, host, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.12.1 Queries

- Find Firewall **as** f that PROTECTS Host **as** h **return** f.displayName **as** firewall, h.  
↪ displayName **as** host
- Find (Host|Device) **with** firewall='ON'
- Find (Host|Device) with firewall!='ON' and active=true that !PROTECTS Firewall

## 24.12.2 Compliance Mappings

**CIS Controls:** 9.4

**HITRUST CSF:** 07.a, 09.ab, 10.h

Implement host-based / local firewalls to monitor and prevent unauthorized access attempts. The organization shall maintain information systems according to a current baseline configuration and configure system security parameters to prevent misuse. The operating system shall have in place supporting technical controls such as antivirus, file integrity monitoring, host-based (personal) firewalls or port filtering tools, and logging as part of their baseline.

**PCI DSS:**

## 24.13 What are the approved server/system images?

Lists all system images. Standard approved system images should be used to build servers and hosts. Images should be updated regularly to include the latest security patches and application/OS updates.

**Tags:** infra, host, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.13.1 Queries

- Find Image

## 24.13.2 Compliance Mappings

**CIS Controls:** 5.1, 5.2

**HITRUST CSF:** 10.h

**PCI DSS:** 2.2

## 24.14 Are all system images updated in the past six months?

Lists all system images that have (or have not) been updated in the past 6 months.

**Tags:** infra, host, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.14.1 Queries

- Find Image **with** createdOn > date.now - 6 months
- Find Image **with** createdOn < date.now - 6 months

## 24.14.2 Compliance Mappings

**CIS Controls:** 5.1, 5.2

**HITRUST CSF:** 10.h

**PCI DSS:** 2.2

## 24.15 Which hosts are (or are not) using approved standard images?

Lists all server and container instances using approved standard images and those that are not, in two listings.

**Tags:** infra, host, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 24.15.1 Queries

- Find (aws\_instance|docker\_container|server) **as** h that USES Image **as** i **return** h.\_type, h.displayName, h.tag.AccountName, i.\_type, i.displayName
- Find (aws\_instance|docker\_container|server) with active=true that !USES Image

### 24.15.2 Compliance Mappings

**CIS Controls:** 5.1, 5.2

**HITRUST CSF:** 10.h

**PCI DSS:** 2.2

## 24.16 Which devices have been disposed in the last 12 months?

Returns a list of devices with a 'disposed' status and last updated within 12 months.

**Tags:** compliance, HIPAA, HITRUST CSF

### 24.16.1 Queries

- Find Device **with** status='disposed' **and** \_beginOn > date.now-24hrs

### 24.16.2 Compliance Mappings

**HITRUST CSF:** 08.k



### 25.1 What are the corporate security policies and procedures?

Find all security policies and procedures.

**Tags:** compliance, HIPAA, HITRUST CSF

#### 25.1.1 Queries

- Find security\_policy
- Find security\_procedure **as** procedure that IMPLEMENTS security\_policy **as** policy\_  
↪ **return** policy.displayName, procedure.displayName order by policy.displayName

#### 25.1.2 Compliance Mappings

**HITRUST CSF:** 04.a

### 25.2 When was security policies and procedures last updated or reviewed?

Find all security policies and procedures by date, and the ones that have not been reviewed or updated in the past year.

**Tags:** compliance, HIPAA, HITRUST CSF

## 25.2.1 Queries

- Find (security\_policy|security\_procedure) **as** p **return** p.displayName **as**   
↳ PolicyProcedureName, p.updatedOn **as** lastUpdatedOn
- Find (security\_policy|security\_procedure) **with** (reviewedOn < date.Now - 1yr **and**   
↳ updatedOn < date.Now - 1yr)

## 25.2.2 Compliance Mappings

HITRUST CSF: 04.b

## 25.3 Who is the appointed security officer?

Find the Person who implements the security program or is assigned the security leadership role.

**Tags:** compliance, HIPAA, HITRUST CSF

### 25.3.1 Queries

- Find Person that (IMPLEMENTS|ASSIGNED) Procedure **with** id='cp-role-assignment'

### 25.3.2 Compliance Mappings

HITRUST CSF: 02.d, 05.a

## 25.4 Which are my documented risks?

Return all documented risks.

**Tags:** compliance, HIPAA, HITRUST CSF

### 25.4.1 Queries

- Find Risk

### 25.4.2 Compliance Mappings

HITRUST CSF: 03.a, 03.b, 03.c, 03.d

Formal risk assessments shall be conducted at least annually or with major product/organization/system changes. As a result of the assessment, any identified risk should be documented and tracked in a risk register.

## 25.5 Was there at least one risk assessment performed within the past year?

Return all risk assessments performed with a createdOn timestamp in the past year; and secondly returns all risks identified by the assessments.

**Tags:** compliance, HIPAA, HITRUST CSF

### 25.5.1 Queries

- Find risk\_assessment with \_createdOn > date.now - 1yr
- Find Assessment with \_createdOn > date.now - 1yr that (IDENTIFIED|REVIEWED) Risk

### 25.5.2 Compliance Mappings

**HITRUST CSF:** 03.b, 03.d

Formal risk assessments shall be conducted at least annually or with major product/organization/system changes. As a result of the assessment, any identified risk should be documented and tracked in a risk register.

## 25.6 Who are my vendors? Do I have a BAA/DPA/NDA/MSA and SLA/Support Agreement with them?

Returns a list of Vendors and their properties, including links to NDA, BAA, MSA, SLA, support agreement and vendor security review report, if available.

**Tags:** compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 25.6.1 Queries

- Find Vendor

### 25.6.2 Compliance Mappings

**CIS Controls:** 2.2

**HITRUST CSF:** 05.i

**PCI DSS:** 2.4



## 26.1 What are directly connected to the Internet?

Find all the entities with an edge directly connected to the Internet

**Tags:** infra, network, SecOps

### 26.1.1 Queries

- Find (Internet|everyone) that relates to \* **return** tree

## 26.2 What production resources are directly connected/exposed to the Internet/everyone?

Find all production entities, except for firewalls and gateways, with an edge directly connected to the Internet or everyone

**Tags:** infra, network, SecOps, compliance, PCI DSS

### 26.2.1 Queries

- Find (Internet|Everyone) that relates to \* **with** tag.Production=true **and** \_class!=  
↪ 'Firewall' **and** \_class!='Gateway' **as** resource **return** resource.tag.AccountName, ↪  
↪ resource.\_type, resource.name, resource.classification, resource.description

### 26.2.2 Compliance Mappings

PCI DSS: 1.3

## 26.3 Are there potential IP collisions among the networks/subnets in my environment?

Find any two Network entities within the same account or service that have identical IP CIDR address.

**Tags:** infra, network, SecOps

### 26.3.1 Queries

- Find Network **as** n1 that has (Service|Account) **as** env that has Network **as** n2 where   
↪ n1.CIDR=n2.CIDR **return** n1.displayName, n1.CIDR, n1.region, n2.displayName, n2.  
↪ CIDR, n2.region, env.displayName, env.tag.AccountName order by env.tag.  
↪ AccountName

## 26.4 What hosts or devices are connected to my internal networks?

Lists Host and Device entities that are connected to (i.e. relates to) internal Network entities.

**Tags:** infra, network, compliance, HIPAA, HITRUST CSF

### 26.4.1 Queries

- Find (Host|Device) that relates to Network **with** internal=true

### 26.4.2 Compliance Mappings

**HITRUST CSF:** 01.k

## 26.5 Show all inbound SSH firewall rules across my network environments.

Returns ingress firewall rules that match port 22 and the incoming source.

**Tags:** infra, network, SecOps

### 26.5.1 Queries

- Find Firewall **as** fw that **ALLOWS as** rule \* **as** src where rule.ingress=true **and**   
↪ (rule.fromPort=22 **or** rule.toPort=22) **return** fw.displayName, rule.fromPort, rule.  
↪ toPort, src.displayName, src.ipAddress, src.CIDR

## 26.6 Is inbound SSH allowed directly from an external host or network?

Returns ingress firewall rules that include port 22 in the allowed range from an external host or network.

**Tags:** infra, network, SecOps

### 26.6.1 Queries

- Find Firewall **as** fw that **ALLOWS as** rule (Host|Network) **with** internal=false **or**   
 ↳ internal=undefined **as** src where rule.ingress=true **and** (rule.fromPort<=22 **and**   
 ↳ rule.toPort>=22) **return** fw.displayName, rule.fromPort, rule.toPort, src.   
 ↳ displayName, src.ipAddress, src.CIDR

## 26.7 What network traffic is allowed between internal and external (i.e. between trusted and untrusted) networks?

Find all Firewall entities and rules that allow traffic to/from an external Network or Host.

**Tags:** infra, network, SecOps, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 26.7.1 Queries

- Find Firewall **as** fw that **ALLOWS as** r (Network|Host) **with** internal=undefined **or**   
 ↳ internal=false **as** n **return** fw.tag.AccountName, fw.\_type, fw.displayName, fw.   
 ↳ description, r.ipProtocol, r.fromPort, r.toPort, n.displayName, n.CIDR, n.   
 ↳ ipAddress

### 26.7.2 Compliance Mappings

**CIS Controls:** 12.2

**HITRUST CSF:** 09.m

**PCI DSS:** 1.2

## 26.8 Is there proper segmentation/segregation of internal networks?

Find all internal networks and show their purpose, environment and associated network-layer security gateway/firewall protection.

**Tags:** infra, network, compliance, CIS Controls, HIPAA, HITRUST CSF, PCI DSS

### 26.8.1 Queries

- Find Network **with** internal=true **as** n that (HAS|CONTAINS|CONNECTS|PROTECTS)   
 ↳ (Gateway|Firewall) **with** category='network' **as** g **return** n.displayName **as** Network,   
 ↳ n.\_type **as** NetworkType, n.CIDR **as** CIDR, n.tag.AccountName **as** Account, n.   
 ↳ internal **as** Internal, g.displayName **as** Gateway, g.\_type **as** GatewayType

## 26.8.2 Compliance Mappings

**CIS Controls:** 12.1

**HITRUST CSF:** 01.m

**PCI DSS:** 1.1

## 26.9 Are wireless networks segmented and protected by firewalls?

Find all wireless networks and show their connected router/gateway and firewall.

**Tags:** infra, network, compliance, HIPAA, HITRUST CSF

### 26.9.1 Queries

- Find Network **with** wireless=true **as** n that (HAS|CONTAINS|CONNECTS|PROTECTS) **↳**  
↳ (Gateway|Firewall) **with** category='network' **as** g that **↳**  
↳ (CONNECTS|ALLOWS|PERMITS|DENIES|REJECTS) **as** r \* **return** n.displayName **as** Network,  
↳ n.\_type **as** NetworkType, n.cidr **as** CIDR, n.environment **as** Environment, g.  
↳ displayName **as** Gateway, g.\_type **as** GatewayType, r.\_class, r.ipProtocol, r.  
↳ fromPort, r.toPort

### 26.9.2 Compliance Mappings

**HITRUST CSF:** 09.m

## 26.10 Show listing of network layer firewall protection across all my environments.

Lists Firewall instances and the Networks they each protects.

**Tags:** infra, network, compliance, HIPAA, HITRUST CSF

### 26.10.1 Queries

- Find Firewall **as** f that PROTECTS Network **as** n **return** f.displayName **as** firewall, n.  
↳ displayName **as** network

### 26.10.2 Compliance Mappings

**HITRUST CSF:** 07.a, 09.m

Organizations shall implement controls to ensure the security of information in networks, and the protection of connected services from unauthorized access.



## 26.11 Are there VPN configured for remote access?

Lists Host, Device, or Network entities that contains the keyword 'vpn' in its properties.

**Tags:** infra, network, vpn, compliance, HIPAA, HITRUST CSF

### 26.11.1 Queries

- `'vpn' with _class='Host' or _class='Device' or _class='Network'`

### 26.11.2 Compliance Mappings

**HITRUST CSF:** 01.j, 09.s

Virtual private networks (VPN) shall be implemented for remote access into internal systems and network environments.



---

## Vulnerability Management

---

### 27.1 What open vulnerabilities do I have?

Returns Vulnerability findings that are still active (i.e. with a status that is open/pending).

**Tags:** vuln-mgmt, compliance, HIPAA, HITRUST CSF

#### 27.1.1 Queries

- Find Vulnerability **with** active=true

#### 27.1.2 Compliance Mappings

**HITRUST CSF:** 10.m

### 27.2 Which applications are vulnerable?

Returns Applications and their open (i.e. active) Vulnerability findings except low severity ones.

**Tags:** vuln-mgmt, compliance, HIPAA, HITRUST CSF

#### 27.2.1 Queries

- Find (Application|Project|CodeRepo) **as** app that has Vulnerability **with** severity>2,   
↪ **and** active=true **as** vuln **return** app.name **as** AppName, vuln.name **as** Vulnerability,   
↪ vuln.severity **as** Severity, vuln.priority **as** Priority

## 27.2.2 Compliance Mappings

**HITRUST CSF:** 10.m

### 28.1 Overview

JupiterOne provides a managed integration with Amazon Web Services. The integration connects directly to AWS APIs to obtain infrastructure metadata and analyze resource relationships. Customers authorize read-only, security audit access by establishing an IAM trust relationship that allows JupiterOne to assume a role in their account.

Information is ingested from all AWS regions that do not require additional contractual arrangements with AWS. Please submit a JupiterOne support request if you need to monitor additional regions.

### 28.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the customer's `roleArn` to assume in order to read infrastructure information through AWS APIs. The role is configured to require an `externalId`; this also must be maintained in the instance configuration.

Detailed setup instructions and a pre-built CloudFormation Stack are provided in the application and maintained in the public [JupiterOne AWS CloudFormation](#) project on Github.

### 28.3 Permissions

The AWS integration requires security auditor permissions into the target AWS account, as defined by a combination of the [SecurityAudit](#) IAM policy managed by AWS, and a few additional `List*`, `Get*`, and `Describe*` permissions missing from the AWS managed policy. The exact policy and permission statements can be found in the public [JupiterOne AWS CloudFormation](#) project on Github.

## 28.4 Entities

The following entity resources and their meta data (not actual contents) are ingested when the integration runs:

## 28.5 Relationships

The following relationships are created/mapped:

### 28.5.1 Basic relationships within the integration instance account/resources

### 28.5.2 Connections to broader entity resources

### 28.5.3 Advanced mappings

The AWS integration performs analysis of security group rules, IAM policies, and assume role trust policies to determine the following mapping:

### 28.5.4 ProTips and Best Practices

- Tag your resources with the following tags:
  - `Classification`
  - `Owner`
  - `PII or PHI or PCI` (boolean to indicate data type)
- Use email address as the `username` for your **IAM Users**, or tag them with `Email` tag, so that they can be automatically mapped to a `Person` (i.e. `employee`) entity.
- Configure tagging as part of your integration configuration (in JupiterOne), under **Advanced Options**, to tag the
  - `AccountName` and
  - `Production flag`, if applicable.
- Configure your integration name to be the same as your AWS account alias.

---

# JupiterOne Managed Integration for Microsoft Azure

---

## 29.1 Overview

JupiterOne provides a managed integration for Microsoft Azure. The integration connects directly to Azure APIs to obtain account metadata and analyze resource relationships. Customers authorize access by ... and providing that credential to JupiterOne.

## 29.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires credentials of the App which is registered with Azure AD. You need:

1. Go to your Azure portal
2. Navigate to [App registrations](#)
3. Create a new app
4. Navigate to Overview page of the new app.
5. Get Application (client) ID and pass it as a `AZURE_CLOUD_LOCAL_EXECUTION_CLIENT_ID` environment variable
6. Get Directory (tenant) ID and pass it as `AZURE_CLOUD_LOCAL_EXECUTION_DIRECTORY_ID` environment variable
7. Navigate to the Certificates & secrets section.
8. Create a new client secret.
9. Store generated token and pass it as `AZURE_CLOUD_LOCAL_EXECUTION_CLIENT_SECRET` environment variable
10. Navigate to API permissions section
11. Grant `Group.Read.All` and `User.Read.All` permissions

12. Grant admin consent for this directory for the permissions above.

Check this instruction for additional information: <https://docs.microsoft.com/en-us/graph/auth-v2-service>

### 29.3 Entities

The following entity resources are ingested when the integration runs:

### 29.4 Relationships

The following relationships are created/mapped:



### 30.1 Overview

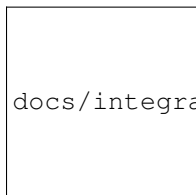
JupiterOne provides a managed integration with Bitbucket. The integration connects directly to Bitbucket APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating a Bitbucket OAuth App in their account and providing the app credentials to JupiterOne.

### 30.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the customer's Bitbucket OAuth App `clientId` and `clientSecret` to authenticate requests to the Bitbucket REST APIs. The integration requires Read access to the target Account, Team Membership, Projects, and Repositories.

See the following screenshot for an example configuration within a Bitbucket Team Settings, note the required and optional settings.



docs/integrations/bitbucket/../../../../assets/integration-bitbucket-oauth-consumer-settings.png

BitBucket OAuth Example Config

*Pull requests read permission is needed to ingest PRs. The PR entities serve as code review records for security and compliance.*

### 30.3 Entities

The following entity resources are ingested when the integration runs:

## 30.4 Relationships

The following relationships are created/mapped:

### 30.4.1 Basic relationships within the integration instance account/resources

```
|| - | bitbucket_team HAS bitbucket_project | bitbucket_team HAS bitbucket_user |  
bitbucket_project HAS bitbucket_repo | bitbucket_repo HAS bitbucket_pull_request |  
bitbucket_user OPENED bitbucket_pull_request
```

### 31.1 Overview

JupiterOne provides a managed integration with Carbon Black (Cb) Predictive Security Cloud (PSC). The integration connects directly to Carbon Black PSC APIs to obtain configuration about its device sensors/agents, starting with Cb Defense sensors. Customers authorize access by creating a Connector and an API Key in their target PSC account and providing that credential to JupiterOne.

### 31.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the following three parameters for API authentication:

- **Site** (`site`): The part immediately follows `defense-` in your Carbon Black PSC / CbDefense account URL. For example, if you access your account at `https://defense-prod05.conferdeploy.net/`, the `site` is `prod05`
- **API Key** (`apiKey`): Go to **Settings > Connectors** from the web console of your Carbon Black account, then click on **Add Connector** button, give it a *Name*, select **API** for the *Connector Type* to create a connector. The **API Key** is displayed to you on screen.
- **Connector ID** (`connectorId`): Once a *Connector* is created, you will see the **Connector ID** on the list.

### 31.3 Entities

The following entity resources are ingested when the integration runs:

## 31.4 Relationships

The following relationships are created/mapped:

```
| Relationships | | ----- | | carbonblack_psc_account HAS  
cbdefense_sensor | | carbonblack_psc_account HAS cb_endpoint_protection | |  
cb_sensor_policy ENFORCES cb_endpoint_protection | | cbdefense_sensor ASSIGNED  
cb_sensor_policy |
```

### 32.1 Overview

JupiterOne provides a managed integration with GitHub. The integration connects directly to GitHub APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating a GitHub OAuth App in their account and providing the app credentials to JupiterOne.

### 32.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the customer's GitHub OAuth App `clientId` and `clientSecret` to authenticate requests to the GitHub REST APIs. [Detailed instructions for creating the OAuth App](#) are provided by GitHub.

### 32.3 Permissions

The integration is using GitHub Apps authentication, which requests permissions from the org/account installing the app.

Beside the Metadata Permissions always granted, our app is only requesting Read Only for Repository Metadata and Organization Members at this time.

Github References:

- <https://developer.github.com/apps/building-github-apps/setting-permissions-for-github-apps/>
- <https://developer.github.com/v3/apps/permissions/#metadata-permissions>
- <https://developer.github.com/v3/apps/permissions/#permission-on-contents>

## 32.4 Entities

The following entity resources are ingested when the integration runs:

## 32.5 Relationships

The following relationships are created/mapped:

### 32.5.1 Basic relationships within the integration instance account/resources

```
||-|github_account OWNS github_repo|github_account HAS github_user
```

### 33.1 Overview

JupiterOne provides a managed integration with Google. The integration connects directly to the G Suite Admin API to obtain account metadata and analyze resource relationships. Customers authorize read-only to access to a JupiterOne Service Account.

### 33.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the Organization Account ID and an administrator email. The JupiterOne Service Account must be added as an authorized API client with required permission scopes.

#### 33.2.1 Getting Organization Account ID

From your Google Admin console:

1. Click Security, then expand Setup single sign-on (SSO)
2. Copy the `idpid` property value from the SSO URL. For example, `https://accounts.google.com/o/saml2/idp?idpid=C1111abcd` provides the ID `C1111abcd`
3. Enter the value into the Account ID field in the JupiterOne integration configuration.

#### 33.2.2 Admin API Enablement

The Admin API is not accessible to the JupiterOne Service Account until the API is enabled in your G Suite organization and permission is granted to the Service Account.

From your Google Admin console:

1. Click Security, then expand Advanced settings and click on Manage API client access
2. Enter the JupiterOne Service Account client ID 102174985137827290632 into Client Name
3. Add the following API scopes (comma separated):

```
https://www.googleapis.com/auth/admin.directory.domain.readonly, https://www.  
↪googleapis.com/auth/admin.directory.user.readonly, https://www.googleapis.com/auth/  
↪admin.directory.group.readonly
```

1. Click Authorize
2. Return to the Admin console, click Security, then API reference
3. Check Enable API access

### 33.3 Entities

The following entity resources are ingested when the integration runs:

### 33.4 Relationships

The following relationships are created/mapped:



### 34.1 Overview

JupiterOne provides a managed integration with HackerOne. The integration connects directly to HackerOne APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating an API token in their target HackerOne account and providing that credential to JupiterOne.

### 34.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

HackerOne provides [detailed instructions on creating an API token](#) within your HackerOne account.

### 34.3 Entities

The following entity resources are ingested when the integration runs:

### 34.4 Relationships

The following relationships are created/mapped:



## 35.1 Overview

JupiterOne provides a managed integration with jamf. The integration connects directly to jamf APIs to obtain account metadata and analyze resource relationships. Customers authorize access by Basic Authentication with their target jamf account and providing that credential to JupiterOne.

## 35.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

jamf provides [detailed instructions on creating credentials](#).

## 35.3 Entities

The following entity resources are ingested when the integration runs:

## 35.4 Relationships

The following relationships are created/mapped:



### 36.1 Overview

JupiterOne provides a managed integration with Jira. The integration connects directly to Jira APIs to obtain project information and issues.

### 36.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

Customers authorize access by creating a Jira user and providing the username and password to JupiterOne for HTTP Basic Auth as described in the [Jira Security for Other Integrations](#) documentation.

### 36.3 Entities

The following entity resources are ingested when the integration runs:

### 36.4 Relationships

The following relationships are created/mapped:



### 37.1 Overview

JupiterOne provides a managed integration with KnowBe4. The integration connects directly to KnowBe4 APIs to obtain account metadata and analyze resource relationships. You authorize access by providing that an API token.

### 37.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

### 37.3 Entities

The following entity resources are ingested when the integration runs:

*Note a training module from KnowBe4 can be either a “Store Purchase” or an “Uploaded Policy”.*

### 37.4 Relationships

The following relationships are created/mapped:





### 38.1 Overview

JupiterOne provides a managed integration with Okta. The integration connects directly to Okta APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating an API token in your target Okta account and providing that credential to JupiterOne.

### 38.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

Instructions on creating an API token within your Okta account can be found [here](#).

### 38.3 Entities

The following entity resources are ingested when the integration runs:

*Note: the Service entities can later be connected to security policy procedures as control providers. This mapping establishes evidence that your organization security policies, procedures and controls are fully implemented, monitored, and managed.*

### 38.4 Relationships

The following relationships are created/mapped:

## 38.5 Tips

All Okta users are automatically mapped to a `Person` entity as an employee. If you have service accounts or generic users in Okta, set their `userType` attribute to `generic` or `service` or `bot` in Okta user profile to skip this mapping.

This allows you to find non-interactive users with a query like

```
Find User that !is Person
```

### 39.1 Overview

JupiterOne provides a managed integration with OneLogin. The integration connects directly to OneLogin APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating an API token in your target OneLogin account and providing that credential to JupiterOne.

### 39.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

Instructions on creating an API token within your OneLogin account can be found [here](#).

### 39.3 Entities

The following entity resources are ingested when the integration runs:

### 39.4 Relationships

The following relationships are created/mapped:



### 40.1 Overview

JupiterOne provides a managed integration with Openshift. The integration connects directly to Openshift APIs to obtain cluster metadata and analyze resource relationships.

### 40.2 Integration Instance Configuration

Authentication is currently designed to use a Service Account.

Login as admin:

```
oc login -u system:admin
```

Create service account:

```
oc create sa jupiterone
oc adm policy add-cluster-role-to-user cluster-reader -z jupiterone
```

Get service account token:

```
oc serviceaccounts get-token jupiterone
```

The integration instance configuration requires the cluster address and service account token.

### 40.3 Entities

The following entity resources are ingested when the integration runs:

## 40.4 Relationships

The following relationships are created/mapped:

### 41.1 Overview

JupiterOne provides a managed integration with SentinelOne. The integration connects directly to SentinelOne APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating an API token in their target SentinelOne account and providing that credential to JupiterOne.

### 41.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

SentinelOne provides Every API call requires authentication. The recommended authentication is API Token. If SSO or Two-Factor Authentication is mandatory for your username, you must use a Token.

Generating an API Token from your account WebUI:

1. In your Management Console, click Settings > USERS.
2. Click your username.
3. Click the edit button.
4. In Edit User > API Token, click Generate. If you see Revoke and Regenerate, you already have a token. If you revoke or regenerate it, scripts that use that token will not work. There is no confirmation. Revoke removes the token authorization. Regenerate revokes the token and generates a new token. If you click Generate or Regenerate, a message shows the token string and the date that the token expires.
5. Click DOWNLOAD.

### 41.3 Entities

The following entity resources are ingested when the integration runs:

## 41.4 Relationships

The following relationships are created/mapped:



### 42.1 Overview

JupiterOne provides a managed integration with Snyk. The integration connects directly to Snyk APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating an API token in their target Snyk account and providing that credential to JupiterOne.

### 42.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configurations requires the following two parameters:

- **Snyk API Key** (`snykApiKey`) In Snyk: In the upper right hand corner mouse over your account name, where a drop down will appear. Click on `account settings` and your API token will appear in a hidden form in the middle of the page. Click `show` and copy your key.
- **Snyk Organisation ID** (`snykOrgId`) In Snyk: Go to the dashboard. Click on `manage organisation` on the far right of the screen across from `Dashboard`. Here, your organisation ID is displayed.

### 42.3 Entities

The following entity resources are ingested when the integration runs:

### 42.4 Relationships

The following relationships are created/mapped:



### 43.1 Overview

JupiterOne provides a managed integration with [Tenable.io](#), the Cloud Managed Tenable Platform. The integration connects directly to [Tenable Cloud APIs](#) to obtain account metadata, vulnerability information, and application scan results for ingestion into JupiterOne. Customers authorize access by providing API keys to JupiterOne.

### 43.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance, including the API access key and secret key provided by the user.

### 43.3 Entities

The following entity resources are ingested when the integration runs:

### 43.4 Relationships

The following relationships are created/mapped:



### 44.1 Overview

JupiterOne provides a managed integration with Threat Stack. The integration connects directly to Threat Stack APIs to obtain agents and vulnerability findings data. Customers authorize access by creating an API Key in their target Threat Stack account and providing that credential to JupiterOne.

### 44.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the following three parameters for API authentication:

Go to **Settings > Application Keys** from the web console of your Threat Stack account, then find the following three values under **REST API Key**, copy/paste each of them into your integration configuration screen in JupiterOne.

- **Organization Name** (`orgName`)
- **Organization ID** (`orgId`)
- **User ID** (`userId`)
- **API Key** (`apiKey`)

### 44.3 Entities

The following entity resources are ingested when the integration runs:

## 44.4 Relationships

The following relationships are created/mapped:

Relationships					
				threatstack_account	<b>HAS</b>
threatstack_agent		threatstack_agent	<b>PROTECTS</b>	aws_instance	
threatstack_agent	<b>PROTECTS</b>	server		threatstack_agent	<b>IDENTIFIED</b>
				cve	

### 45.1 Overview

JupiterOne provides a managed integration with Veracode. The integration connects directly to Veracode APIs to obtain Vulnerability and Finding metadata and analyze resource relationships. Customers authorize access by creating an API ID and secret in the their target Veracode account and providing those credentials to JupiterOne.

### 45.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

The integration instance configuration requires the customer's API ID and secret to authenticate requests to the Veracode REST APIs. Veracode provides [detailed instructions for obtaining these credentials](#).

### 45.3 Entities

The following entity resources are ingested when the integration runs:

### 45.4 Relationships

The following relationships are created/mapped:

#### 45.4.1 Intra-Instance

#### 45.4.2 Extra-Instance / Mapped





### 46.1 Overview

JupiterOne provides a managed integration with [Wazuh](#). The integration connects directly to Wazuh Manager APIs to obtain agent information. Customers authorize access to their self-hosted servers by providing the manager base URL and a username and password to JupiterOne.

### 46.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

### 46.3 Entities

The following entity resources are ingested when the integration runs:

### 46.4 Relationships

The following relationships are created/mapped:



### 47.1 Overview

JupiterOne provides a managed integration with Whitehat. The integration connects directly to Whitehat APIs to obtain account metadata and analyze resource relationships. Customers authorize access by creating an API key in their target Whitehat account and providing that credential to JupiterOne.

### 47.2 Integration Instance Configuration

The integration is triggered by an event containing the information for a specific integration instance.

To obtain the API token for a Whitehat account, sign in to Sentinel. Click the “My Profile” button in the top right and then “API Key”. Enter the account password and copy the displayed API Key.

### 47.3 Entities

The following entity resources are ingested when the integration runs:

### 47.4 Relationships

The following relationships are created/mapped:

#### 47.4.1 Intra-Instance

#### 47.4.2 Extra-Instance / Mapped



## CHAPTER 48

---

### AccessKey

---

A key used to grant access, such as ssh-key, access-key, api-key/token, mfa-token/device, etc.

Includes properties from:

- [Key](#)
- [Entity](#)
- [Metadata](#)



A policy for access control assigned to a Host, Role, User, UserGroup, or Service.

Includes properties from:

- Entity
- Metadata

#### 49.1 `admin` (boolean) - Optional

Indicates if the policy grants administrative privilege.

#### 49.2 `rules` (array of string) - Optional

Rules of this policy. Each rule is written ‘as-code’ that can be operationalized with a control provider or within JupiterOne’s rules engine.

#### 49.3 `content` (string) - Optional

Content of a policy contains the raw policy rules, if applicable. For example, the JSON text of an AWS IAM Policy. This is stored in raw data.





---

## AccessRelationship

---

A Relationship that represents permission settings/rules between two entities.

Includes properties from:

- Relationship
- Metadata

### 50.1 `_class` (string) - Optional

Contains an enumeration of defined Relationship classes.

#### Options

- `ALLOWS`
- `CAN_ACCESS`
- `DENIES`
- `PERMITS`
- `REJECTS`

### 50.2 `permissions` (array of string) - Optional

Defines permissions of a Relationship. For example, `ses:Get*`, `s3:GetObjects` for access policy; or `<protocol>.<port-range>` for firewall rules.

## 50.3 accessLevel1 (array) - Optional

Defines the CRUD level of access - CREATE, READ, UPDATE, DELETE - and additionally, ADMIN. For CAN\_ACCESS Relationship.

### Options

- CREATE
- READ
- UPDATE
- DELETE
- ADMIN

## 50.4 protocol (string) - Optional

Network traffic protocol (e.g. TCP, UDP, ICMP)

### Options

- TCP
- UDP
- ICMP
- ALL

## 50.5 portRange (string) - Optional

Network traffic port range. This can be a single port (e.g. 80), or a range (e.g. 8080-8082), or any/all (represented by the string 'any' or '0-65535').

## 50.6 type (string) - Optional

Named type of access. For example: 'SSH', 'HTTPS', or 'S3 Read Access'.

---

### AccessRole

---

An access control role mapped to a Principal (e.g. user, group, or service).

Includes properties from:

- [Entity](#)
- [Metadata](#)



---

### Account

---

An organizational account for a service or a set of services (e.g. AWS, Okta, Bitbucket Team, Google G-Suite account, Apple Developer Account). Each Account should be connected to a Service.

Includes properties from:

- Entity
- Metadata

#### 52.1 `production` (boolean) - Required

Indicates if this is a production account, defaults to false.

#### 52.2 `accessURL` (string) - Optional

The main URL to access this account, e.g. <https://lifeomic.okta.com>

Format: uri

#### 52.3 `mfaEnabled` (boolean) - Optional

Specifies whether multi-factor authentication (MFA) is enabled/required for users of this account.



A software product or application.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 53.1 cots (boolean) - Optional

Indicates if this is a Commercial Off-The-Shelf software application. Custom in-house developed application should have this set to false.

### 53.2 foss (boolean) - Optional

Indicates if this is a Free or Open-Source software application or library. Custom in-house developed application should have this set to false.

### 53.3 saas (boolean) - Optional

Indicates if this is a Software-as-a-Service product.

### 53.4 external (boolean) - Optional

Indicates if this is an externally acquired software application. Custom in-house developed application should have this set to false.

## 53.5 `mobile` (boolean) - Optional

Indicates if this is a mobile app.

## 53.6 `license` (string) - Optional

Stores the type of license

### Example Values

- BSD
- CC-BY-3.0
- CC-BY-4.0
- GPL-2.0
- GPL-3.0
- LGPL-2.0
- LGPL-2.1
- LGPL-3.0
- MIT
- EULA
- Proprietary
- UNLICENSED
- other

## 53.7 `licenseURL` (string) - Optional

The URL to the full license

Format: uri

## 53.8 `productionURL` (string) - Optional

The Production URL

Format: uri

## 53.9 `stagingURL` (string) - Optional

The Non-Production / Staging URL

Format: uri



## 53.10 `devURL` (string) - Optional

The Development URL

Format: uri

## 53.11 `testURL` (string) - Optional

The Test URL

Format: uri

## 53.12 `alternateURLs` (array of string) - Optional

The additional URLs related to this application.



An object to represent an assessment, including both compliance assessment such as a HIPAA Risk Assessment or a technical assessment such as a Penetration Testing. Each assessment should have findings (e.g. Vulnerability or Risk) associated.

Includes properties from:

- Entity
- Metadata

### 54.1 category (string) - Required

The category of the Assessment.

#### Options

- Risk Assessment
- Readiness Assessment
- Gap Assessment
- Validation Assessment
- Compliance Assessment
- Self Assessment
- Certification
- Audit
- Technical Review
- Operational Review
- Penetration Testing

- Vulnerability Scan
- Other

## 54.2 `summary` (string) - Required

The summary description of the Assessment.

## 54.3 `internal` (boolean) - Required

Indicates if this is an internal or external assessment/audit. Defaults to true.

## 54.4 `startedOn` (number) - Optional

The timestamp (in milliseconds since epoch) when the Assessment was started.

Format: date-time

## 54.5 `completedOn` (number) - Optional

The timestamp (in milliseconds since epoch) when the Assessment was completed.

Format: date-time

## 54.6 `reportURL` (string) - Optional

Link to the assessment report, if available.

Format: uri

## 54.7 `assessor` (string) - Optional

Email or name or ID of the assessor

## 54.8 `assessors` (array of string) - Optional

List of email or name or ID of the assessors

---

### Attacker

---

An attacker or threat actor.

Includes properties from:

- [Entity](#)
- [Metadata](#)



## CHAPTER 56

---

### Certificate

---

A digital Certificate such as an SSL or S/MIME certificate.

Includes properties from:

- [Entity](#)
- [Metadata](#)





---

### Cluster

---

A cluster of compute or database resources/workloads.

Includes properties from:

- [Entity](#)
- [Metadata](#)



A code commit to a repo. The commit id is captured in the `_id` property of the Entity.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 58.1 `branch (string)` - Required

The branch the code was committed to.

### 58.2 `message (string)` - Required

The commit message.

### 58.3 `merge (boolean)` - Required

Indicates if this commit is a merge, defaults to false.

### 58.4 `versionBump (boolean)` - Required

Indicates if this commit is a versionBump, defaults to false.



A code deploy job.

Includes properties from:

- `RecordEntity`
- `Metadata`

#### 59.1 `jobName` (string) - Optional

Build/deploy job name.

#### 59.2 `jobNumber` (integer) - Optional

Build/deploy job number.

#### 59.3 `summary` (string) - Optional

Descriptive text of the job.

#### 59.4 `action` (string) - Optional

Deploy action (e.g. plan, apply, destroy, rollback).

## 59.5 `target` (string) - Optional

Name of the target system or environment.

## 59.6 `production` (boolean) - Optional

Indicates if this is a production deploy, defaults to true.

An application code module/library. Such as an npm-module or java-library.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 60.1 `public` (boolean) - Optional

Indicates if this is a public module.





A source code repository. A CodeRepo is also a DataRepository therefore should carry all the required properties of DataRepository.

Includes properties from:

- Entity
- Metadata

### 61.1 `application` (string) - Optional

The application that this repo is part of.

### 61.2 `project` (string) - Optional

The project that this repo belongs to.

### 61.3 `public` (boolean) - Optional

Indicates if this is a public repo.



A code review record.

Includes properties from:

- `RecordEntity`
- `Metadata`

### 62.1 `title (string)` - Required

The title text of the review.

### 62.2 `summary (string)` - Optional

The summary text of the review.

### 62.3 `state (string)` - Optional

The state of the review.



---

### Configuration

---

A Configuration contains definitions that describe a resource such as a Task, Deployment or Workload. For example, an *aws\_ecs\_task\_definition* is a *Configuration*.

Includes properties from:

- Entity
- Metadata



---

### Control

---

A security or IT Control. This is most likely an additional Class applied to a Service (e.g. Okta SSO), a Device (e.g. a physical firewall), or a HostAgent (e.g. Carbon Black CbDefense Agent).

Includes properties from:

- [Entity](#)
- [Metadata](#)





An operational or configuration compliance policy with technical specifications / rules that governs (i.e. enforces, evaluates, or monitors) a security control or IT system.

Includes properties from:

- Entity
- Metadata

### 65.1 category (string) - Optional

The category of control policy.

#### Options

- compliance
- config
- password
- other

### 65.2 rules (array of string) - Optional

Rules of this policy. Each rule is written 'as-code' that can be operationalized with a control provider or within JupiterOne's rules engine.

## 65.3 content (string) - Optional

Content of an AccessPolicy or ControlPolicy contains the raw policy rules, if applicable. For example, the JSON text of an AWS IAM Policy.

## CHAPTER 66

---

### CryptoKey

---

A key used to perform cryptographic functions, such as an encryption key.

Includes properties from:

- [Key](#)
- [Entity](#)
- [Metadata](#)



An individual data object, such as an aws-s3-object, sharepoint-document, source-code, or a file (on disk). The exact data type is described in the `_type` property of the Entity.

Includes properties from:

- Entity
- Metadata

### 67.1 `category` (string) - Optional

A user-provided category of the data, such as 'Source Code', 'Report', 'Patent Application', 'Business Plan', 'Customer Record', 'Genetic Data', etc.

### 67.2 `format` (string) - Optional

The format of the data, such as 'document', 'raw', 'plaintext', 'binary', etc.

### 67.3 `classification` (string) - Required

The sensitivity of the data; should match company data classification

#### Example Values

- critical
- confidential
- internal
- public

## 67.4 `location` (string) - Optional

URI to the data, e.g. file path

## 67.5 `PII` (boolean) - Optional

Indicates if this data object is or contains Personally Identifiable Information

## 67.6 `PHI` (boolean) - Optional

Indicates if this data object is or contains Protected Health Information

## 67.7 `PCI` (boolean) - Optional

Indicates if this data object is or contains Payment Card Information

## 67.8 `encryptionRequired` (boolean) - Optional

If the data needs to be encrypted

## 67.9 `encrypted` (boolean) - Optional

If the data is encrypted

## 67.10 `public` (boolean) - Optional

Indicates if the data object is open to public access

---

### DataStore

---

A virtual repository where data is stored, such as aws-s3-bucket, aws-rds-cluster, aws-dynamodb-table, bitbucket-repo, sharepoint-site, docker-registry. The exact type is described in the `_type` property of the Entity.

Includes properties from:

- Entity
- Metadata

#### 68.1 `location` (string) - Optional

URI to the data store, e.g. <https://docker-registry.lifeomic.com> or <https://lifeomic.sharepoint.com>. Or a description to the physical location.

#### 68.2 `encryptionRequired` (boolean) - Optional

If the data needs to be encrypted

#### 68.3 `encryptionAlgorithm` (string) - Optional

Encryption algorithm used to encrypt the data store

#### 68.4 `encryptionKeyRef` (string) - Optional

Reference to the encryption key used to encrypt the data store

## 68.5 `encrypted` (boolean) - Optional

If the data store is encrypted

## 68.6 `public` (boolean) - Optional

Indicates if the data store is open to public access

## 68.7 `hasBackup` (boolean) - Optional

Indicates if the data store is data backup has been configured/enabled.



A database cluster/instance.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 69.1 `location (string)` - Optional

URI to access the database.

### 69.2 `encryptionRequired (boolean)` - Optional

If the data needs to be encrypted

### 69.3 `encrypted (boolean)` - Optional

If the repository is encrypted

### 69.4 `classification (string)` - Required

The sensitivity of the data; should match company data classification scheme

**Example Values**

- critical

- confidential
- internal
- public

A deployment of code, application, infrastructure or service. For example, a Kubernetes deployment. An auto scaling group is also considered a deployment.

Includes properties from:

- Entity
- Metadata

### 70.1 `desiredSize` (number) - Optional

Desired size (i.e. number of instances) associated with this deployment.

### 70.2 `currentSize` (number) - Optional

Current size (i.e. number of instances) active with this deployment.

### 70.3 `maxSize` (number) - Optional

Maximum size (i.e. number of instances) limited by this deployment.



A physical device or media, such as a server, laptop, workstation, smartphone, tablet, router, firewall, switch, wifi-access-point, usb-drive, etc. The exact data type is described in the `_type` property of the Entity.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 71.1 `category` (string) - Required

The device category

#### Example Values

- server
- endpoint
- storage-media
- mobile
- other

### 71.2 `hardwareVendor` (string) - Required

The manufacturer or vendor of the device, e.g. Apple Inc., Generic

### 71.3 `hardwareModel` (string) - Required

The device hardware model, e.g. MacBookPro13,3

## 71.4 hardwareVersion (string) - Optional

The device hardware version

## 71.5 hardwareSerial (string) - Required

The device serial number

## 71.6 assetTag (string) - Optional

The asset tag number/label that matches the identifier in asset tracking system, for company owned physical devices

## 71.7 platform (string) - Optional

Operating System Platform

### Options

- darwin
- linux
- unix
- windows
- android
- ios
- embedded
- other

## 71.8 osDetails (string) - Optional

Operating System Full Details (e.g. macOS High Sierra version 10.13.6)

## 71.9 osName (string) - Optional

Operating System Name (e.g. macOS)

## 71.10 osVersion (string) - Optional

Operating System Version (e.g. 10.13.6)

## 71.11 `userEmails` (array of string) - Optional

The email addresses of the users this device is assigned to. Used if the device is shared by more than one user. Otherwise the 'owner' is the sole user. Leave empty/undefined if the device is unassigned.

## 71.12 `location` (string) - Optional

Site where this device is located.

## 71.13 `cost` (number) - Optional

The purchase cost of the device.

## 71.14 `value` (number) - Optional

The estimated business value of the device. The value is typically calculated as the monetary cost of the device + the value of data on the device.

## 71.15 `BYOD` (boolean) - Required

Indicates if this is a BYOD device – an employee-provided device that has access to company systems/resources.

## 71.16 `status` (string) - Optional

Status label of this device

### Options

- assigned
- archived
- decommissioned
- defective
- deployed
- disposed
- locked
- lost/stolen
- pending
- ready
- unknown
- other





## CHAPTER 72

---

### Document

---

A document or data object.

Includes properties from:

- [Entity](#)
- [Metadata](#)



## CHAPTER 73

---

### Domain

---

An internet domain.

Includes properties from:

- [Entity](#)
- [Metadata](#)



A node in the graph database that represents an Entity. This reference schema defines common shared properties among most Entities.

Includes properties from:

- [Metadata](#)

### 74.1 `name (string)` - Required

Name of this entity

### 74.2 `displayName (string)` - Required

Display name, e.g. a person's preferred name or an AWS account alias

### 74.3 `summary (string)` - Optional

A summary / short description of this entity.

### 74.4 `description (string)` - Optional

An extended description of this entity.

## 74.5 `classification` (string) - Optional

The sensitivity of the data; should match company data classification scheme

### Example Values

- critical
- confidential
- internal
- public

## 74.6 `criticality` (integer) - Optional

A number that represents the value or criticality of this entity, on a scale between 1-10.

## 74.7 `risk` (integer) - Optional

The risk level of this entity, on a scale between 1-10.

## 74.8 `trust` (integer) - Optional

The trust level of this entity, on a scale between 1-10.

## 74.9 `complianceStatus` (number) - Optional

The compliance status of the entity, as a percentage of compliancy.

## 74.10 `status` (string) - Optional

Status of this entity set by the external source system or by a user, e.g. Active, Inactive, Decommissioned

### Options

- active
- inactive
- suspended
- terminated
- open
- closed
- pending
- unknown
- other

## 74.11 `active` (boolean) - Optional

Indicates if this entity is currently active.

## 74.12 `public` (boolean) - Optional

Indicates if this is a public-facing resource (e.g. a public IP or public DNS record) or if the entity is publicly accessible. Default is false.

## 74.13 `validated` (boolean) - Optional

Indicates if this node has been validated as a known/valid Entity.

## 74.14 `temporary` (boolean) - Optional

Indicates if this node is a temporary resource, such as a lambda instance or an EC2 instance started by ECS.

## 74.15 `createdOn` (number) - Optional

The timestamp (in milliseconds since epoch) when the entity was created at the source. This is different than `_createdOn` which is the timestamp the entity was first ingested into JupiterOne.

Format: date-time

## 74.16 `updatedOn` (number) - Optional

The timestamp (in milliseconds since epoch) when the entity was last updated at the source.

Format: date-time

## 74.17 `expiresOn` (number) - Optional

If the entity is a temporary resource, optionally set the expiration date. For example, the expiration date of an SSL cert.

Format: date-time

## 74.18 `webLink` (string) - Optional

Web link to the source. For example: <https://console.aws.amazon.com/iam/home#/roles/Administrator>. This property is used by the UI to add a hyperlink to the entity.

Format: uri

## 74.19 `owner` (string) - Optional

The owner of this entity. This could reference the name of the owner, or as reference ID/key to another entity in the graph as the owner.

## 74.20 `tag.*` (string) - Optional

Named tags assigned to the entity (i.e., 'tag.Name', 'tag.OtherName')

## 74.21 `tags` (array of string) - Optional

An array of unnamed tags

## 74.22 `notes` (array of string) - Optional

User provided notes about this entity



---

## Finding

---

A security finding, which may be a vulnerability or just an informative issue. A single finding may impact one or more resources. The *IMPACTS* relationship between the Vulnerability and the resource entity that was impacted serves as the record of the finding. The *IMPACTS* relationship carries properties such as 'identifiedOn', 'remediatedOn', 'remediationDueOn', 'issueLink', etc.

Includes properties from:

- [RecordEntity](#)
- [Metadata](#)

### 75.1 **assessment (string) - Optional**

The name/id of the assessment that produced this finding.

### 75.2 **status (string) - Optional**

Status of the vulnerability

### 75.3 **severity (string) - Required**

Severity rating based on impact and exploitability. Can be a string such as 'critical', 'high', 'medium', 'low', 'info'. Or an integer usually between 0-5.

## 75.4 `priority` (string) - Optional

Priority level mapping to Severity rating. Can be a string such as 'critical', 'high', 'medium', 'low', 'info'. Or an integer usually between 0-5.

## 75.5 `score` (number) - Optional

The overall vulnerability score, e.g. CVSSv3.

## 75.6 `impact` (string) - Optional

The impact description or rating.

## 75.7 `exploitability` (number) - Optional

The exploitability score/rating.

## 75.8 `vector` (string) - Optional

The vulnerability attack vector. (e.g. a CVSSv3 vector looks like this - 'AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N')

## 75.9 `stepsToReproduce` (array of string) - Optional

Steps to reproduce this finding.

## 75.10 `recommendation` (string) - Optional

Recommendation on how to remediate/fix this finding.

## 75.11 `targets` (array of string) - Optional

The target listing of projects, applications, repos or systems this vulnerability impacts. Specifying either the project/repo name or the application URL here will auto-map this Vulnerability to the corresponding Project/CodeRepo/Application entity if a match is found.

## 75.12 `targetDetails` (array of string) - Optional

Additional details about the targets. Can be a string or an array.

## 75.13 `remediationSLA` (integer) - Optional

The number of days that the Vulnerability must be remediated within, based on SLA set by the organization's internal vulnerability management program policy. The actually due date is set by 'remediationDueOn' property on the *IMPACTS* relationship between the Vulnerability and its impacted resource entity.

## 75.14 `blocksProduction` (boolean) - Optional

Indicates whether this vulnerability finding is a blocking issue. If true, it should block a production deploy. Defaults to false.

## 75.15 `open` (boolean) - Required

Indicates if this is an open vulnerability.

## 75.16 `production` (boolean) - Required

Indicates if this vulnerability is in production.

## 75.17 `public` (boolean) - Required

Indicates if this is a publicly disclosed vulnerability. If yes, this is usually a CVE and the 'webLink' should be set to '[https://nvd.nist.gov/vuln/detail/\\${CVE-Number}](https://nvd.nist.gov/vuln/detail/${CVE-Number})' or to a vendor URL. If not, it is most likely a custom application vulnerability.

## 75.18 `validated` (boolean) - Optional

Indicates if this Vulnerability finding has been validated by the security team.

## 75.19 `references` (array of string) - Optional

The array of links to references.



A piece of hardware or software that protects a network/host/application.

Includes properties from:

- Entity
- Metadata

### 76.1 category (array of string) - Required

The category of the Firewall. Indicates the scope that the Firewall applies to – i.e. Network, Host, Application.

#### Options

- network
- host
- application
- other

### 76.2 isStateful (boolean) - Optional

Indicates if the rules in the firewall is stateful.



An object to represent a standard compliance or technical security framework.

Includes properties from:

- [Metadata](#)

#### **77.1 name (string) - Required**

Name of this entity

#### **77.2 displayName (string) - Required**

Display name

#### **77.3 summary (string) - Optional**

A summary / short description of this entity.

#### **77.4 description (string) - Optional**

An extended description of this entity.

## 77.5 `standard (string)` - Required

The name of the framework standard.

### Options

- HIPAA
- HITRUST CSF
- CSA STAR
- PCI DSS
- NIST CSF
- FedRAMP
- ISO 27001
- SOC
- OWASP
- Other

## 77.6 `version (string)` - Required

The version of the framework. For example, HITRUST CSF may have version 8.1, 9.0, 9.1, etc.; OWASP may have version 2010, 2013, 2017.



A virtual application function. For example, an `aws_lambda_function`, `azure_function`, or `google_cloud_function`

Includes properties from:

- `Entity`
- `Metadata`

### 78.1 `image (string)` - Optional

The image of this function, typically refers to a zip package.

### 78.2 `version (string)` - Optional

The version of this function.

### 78.3 `runtime (string)` - Optional

The runtime of this function. For example: `'nodejs6.10'`, `'nodejs8.10'`, or `'python2.7'`.

### 78.4 `memorySize (string)` - Optional

The allocated memory of this function to execute.

## 78.5 `codeSize` (string) - Optional

The size of code of this function.

## 78.6 `codeHash` (string) - Optional

The hash of code of this function.

## 78.7 `trigger` (string) - Optional

What triggers this function to execute.

## 78.8 `handler` (string) - Optional

The handler of this function

---

## Gateway

---

A gateway/proxy that can be a system/appliance or software service, such as a network router or application gateway.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 79.1 category (array of string) - Required

The category of the Gateway (corresponds to which OSI layer does the Proxy operates at).

#### Options

- network
- application
- data
- other

### 79.2 function (array of string) - Required

The function of the Gateway

#### Options

- routing
- nat
- api-gateway
- content-filtering

- content-distribution
- load-balancing
- firewall
- ssl-termination
- reverse-proxy
- remote-access-gateway
- application-protection
- intrusion-detection
- intrusion-prevention
- mail-filtering
- malware-protection
- other

### 79.3 `public` (boolean) - Required

Indicates if the Gateway is open to public access

## CHAPTER 80

---

### Group

---

A defined, generic group of Entities. This could represent a group of Resources, Users, Workloads, DataRepositories, etc.

Includes properties from:

- [Entity](#)
- [Metadata](#)



A compute instance that itself owns a whole network stack and serves as an environment for workloads. Typically it runs an operating system. The exact host type is described in the `_type` property of the Entity. The UUID of the host should be captured in the `_id` property of the Entity

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 81.1 `hostname (string)` - Required

The primary/local hostname

### 81.2 `ipAddress (string)` - Optional

The main IP address. This property is usually used to store the primary IP address of a Host.

Format: ip

### 81.3 `publicDnsName (string)` - Optional

The public DNS name

Format: hostname

## 81.4 `privateDnsName` (string) - Optional

The private DNS name

Format: hostname

## 81.5 `publicIpAddress` (string) - Optional

The public IP address

Format: ipv4

## 81.6 `privateIpAddress` (string) - Optional

The private IP address

Format: ipv4

## 81.7 `ipAddresses` (array of string) - Optional

A listing of all IPv4 addresses associated with this Host

## 81.8 `ipv6Addresses` (array of string) - Optional

A listing of all IPv6 addresses associated with this Host

## 81.9 `macAddress` (string) - Optional

Primary MAC address

## 81.10 `platform` (string) - Optional

Operating System Platform

### Options

- darwin
- linux
- unix
- windows
- android
- ios
- embedded



- other

### **81.11 osDetails (string) - Optional**

Operating System Full Details (e.g. macOS High Sierra version 10.13.6)

### **81.12 osName (string) - Optional**

Operating System Name (e.g. macOS)

### **81.13 osVersion (string) - Optional**

Operating System Version (e.g. 10.13.6)

### **81.14 macAddresses (array of string) - Optional**

A listing of all MAC addresses associated with this Host

### **81.15 isPhysical (boolean) - Optional**

Indicates if this is a physical host, such as a physical server.



A software agent or sensor that runs on a host/endpoint

Includes properties from:

- Entity
- Metadata

### 82.1 function (array of string) - Required

The function of sensor/agent

#### Options

- endpoint-compliance
- endpoint-configuration
- endpoint-protection
- anti-malware
- DLP
- FIM
- host-firewall
- HIDS
- log-monitor
- activity-monitor
- vulnerability-detection
- container-security
- other



---

### Image

---

A system image. For example, an AWS AMI (Amazon Machine Image).

Includes properties from:

- [Entity](#)
- [Metadata](#)



An operational or security incident.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 84.1 `category` (string) - Required

The category of the incident

#### Options

- 1. General Incident
- 2. Attack on Internal Facing Assets
- 3. Attack on External Facing Assets
- 4. Malware
- 5. Social Engineering
- 6. Data Breach
- 7. Physical or Environmental

### 84.2 `severity` (string) - Required

Severity rating based on impact. Can be a string such as 'critical', 'major', 'minor', or an integer usually between 1-3.

### **84.3 `impacts` (array of string) - Optional**

The target listing of [IDs/keys to] systems and resources this incident impacts.

### **84.4 `reportable` (boolean) - Required**

Indicates if this is a reportable incident per applicable regulations, such as HIPAA, PCI, or GDPR.

### **84.5 `reporter` (string) - Optional**

The person/entity who reported this incident.

### **84.6 `postmortem` (string) - Optional**

Summary and/or a link to the documented lesson learned.



---

### Internet

---

The Internet node in the graph. There should be only one Internet node.

Includes properties from:

- [Metadata](#)

#### 85.1 `displayName` (string) - Optional

Display name

#### 85.2 `CIDR` (string) - Optional

The IPv4 network CIDR block

#### 85.3 `CIDRv6` (string) - Optional

The IPv6 network CIDR block

#### 85.4 `public` (boolean) - Optional

Indicates if the network is open to public access



---

### IpAddress

---

An re-assignable `IpAddress` resource entity. Do not create an entity for an IP Address `_configured_` on a Host. Use this only if the IP Address is a reusable resource, such as an Elastic IP Address object in AWS.

Includes properties from:

- `Entity`
- `Metadata`

#### 86.1 `dnsName (string)` - Optional

The assigned DNS name

Format: hostname

#### 86.2 `publicIpAddress (string)` - Optional

The assigned public IP address

Format: ip

#### 86.3 `privateIpAddress (string)` - Optional

The assigned private IP address

Format: ip

## 86.4 `ipVersion` (integer) - Optional

Indicates IP version 4 or 6

### Options

- 4
- 6

An ssh-key, access-key, api-key/token, pgp-key, etc.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 87.1 `fingerprint (string)` - Optional

The fingerprint that identifies the key

### 87.2 `material (string)` - Optional

The key material

### 87.3 `usage (string)` - Optional

The key usage - for example: ssh access or data encryption



The standard metadata properties of a given entity/relationship. These properties are system generated (e.g. set by an integration). They are viewable in the UI but not directly editable by a user.

#### 88.1 `__accountId` (string) - Required

The account / tenant identifier

#### 88.2 `__id` (string) - Required

An auto-generated and globally unique ID

#### 88.3 `__key` (string) - Required

A unique identifier of an entity/relationship within the scope of a single integration instance. For example, for a Bitbucket repo, this `__id` will be the GUID of the repo as assigned by Bitbucket. For an IAM Role, the `__id` will be the ARN of the role.

#### 88.4 `__iconPath` (string) - Optional

Path to the icon used in the web app UI

## 88.5 `_class` (string) - Required

Used to create an abstract security data model. For example, a EC2 instance will have `'_class': 'Host'`. An integration can supply one or more classes which can be used to indicate if a particular entity/relationship conforms to one or more standard classifications. This property is similar to `_type` except that `_class` refers to a type that has been standardized while `_type` is an entity type that only has to be unique in the scope of the provider. It is possible that an entity/relationship has a `_type` value but no `_class` value in cases where there is no standard classification for a given entity/relationship.

## 88.6 `_type` (string) - Required

Describes the type of entity/relationship as identified by the data source (often the integration or sometimes manual user input). The `_class` property is similar to `_type` but `_class` refers to a categorization that has been standardized and it is not unique to a single data integration.

## 88.7 `_integrationName` (string) - Optional

Name of the integration that created this entity.

## 88.8 `_integrationDefinitionId` (string) - Optional

The unique ID of the integration definition that created this entity.

## 88.9 `_integrationInstanceId` (string) - Optional

The unique ID of the integration instance that created this entity.

## 88.10 `_createdOn` (number) - Required

The timestamp (in milliseconds since epoch) when this node was created - the earliest timestamp for this entity as known by the security platform (might be different from when entity was actually created in external system)

Format: date-time

## 88.11 `_createdBy` (string) - Optional

The `entityId` of the user who created this node, if it is created manually and not by an integration.

## 88.12 `_beginOn` (number) - Required

The timestamp (in milliseconds since epoch) when this node was updated

Format: date-time



### 88.13 `_endOn` (number) - Optional

The timestamp (in milliseconds since epoch) when a new version of this node was created

Format: date-time

### 88.14 `_updatedBy` (string) - Optional

The entityId of the user who last updated this node, if it is created manually and not by an integration.

### 88.15 `_lastSeenOn` (number) - Required

The timestamp (in milliseconds since epoch) when this node was last seen in events/logs or other ingested data sources

Format: date-time

### 88.16 `_version` (integer) - Required

Numerical auto-incrementing value that represents the version number of this node. Increments every time the node is updated.

### 88.17 `_latest` (boolean) - Optional

Indicates if this node is the latest version of the Entity.

### 88.18 `_deleted` (boolean) - Optional

Indicates if this node is soft-deleted.

### 88.19 `vendorManaged` (boolean) - Optional

Indicates if this entity/relationship is managed by the vendor.

### 88.20 `inUse` (boolean) - Optional

Indicates if this entity/relationship is in use.

### 88.21 `ignore` (boolean) - Optional

Instructs the query to ignore this entity/relationship by default.



A software or hardware module. Such as an npm-module or java-library.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 89.1 `public` (boolean) - Optional

Indicates if this is a public module.



A network, such as an aws-vpc, aws-subnet, cisco-meraki-vlan.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 90.1 environment (string) - Required

The environment of network

#### Options

- development
- test
- staging
- production
- private
- wireless
- guest
- remote-access
- administrative
- other

## 90.2 CIDR (string) - Required

The IPv4 network CIDR block (e.g. 0.0.0.0/0)

Format: ipv4

## 90.3 CIDRv6 (string) - Optional

The IPv6 network CIDR block (e.g. ::/0)

Format: ipv6

## 90.4 public (boolean) - Required

Indicates if the network is publicly accessible.

## 90.5 internal (boolean) - Required

Indicates if this is an internal/private network.

## 90.6 wireless (boolean) - Optional

Indicates if this is a wireless network.

---

## NetworkInterface

---

An re-assignable software defined network interface resource entity. Do not create an entity for a network interface `_configured_` on a Host. Use this only if the network interface is a reusable resource, such as an Elastic Network Interface object in AWS.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 91.1 `macAddress` (string) - Optional

The assigned MAC address

Format: hostname

### 91.2 `dnsName` (string) - Optional

The assigned DNS name

Format: hostname

### 91.3 `publicIpAddress` (string) - Optional

The assigned public IP address

Format: ip

## 91.4 `privateIpAddress` (string) - Optional

The assigned private IP address

Format: ip

## 91.5 `ipVersion` (integer) - Optional

Indicates IP version 4 or 6

### Options

- 4
- 6



---

### Organization

---

An organization, such as a company (e.g. LifeOmic) or a business unit (e.g. HR). An organization can be internal or external. Note that there is a more specific Vendor class.

Includes properties from:

- Entity
- Metadata

#### 92.1 `_type` (string) - Optional

The type of organization (within the context of the primary organization).

##### Options

- company
- department
- business-unit
- subsidiary
- government-agency
- partner
- other

#### 92.2 `website` (string) - Optional

The organization's main website URL.

Format: uri

## 92.3 emailDomain (string) - Optional

The domain name for internal organization email addresses.

## 92.4 external (boolean) - Optional

Indicates if this organization is external

A pull request.

Includes properties from:

- `RecordEntity`
- `Metadata`

### 93.1 `title (string)` - Required

The title text of the PR.

### 93.2 `summary (string)` - Optional

The summary text of the PR.

### 93.3 `state (string)` - Required

The state of the PR.

#### Options

- `open`
- `merged`
- `declined`
- `superseded`

### **93.4 `source` (string) - Required**

The source branch.

### **93.5 `target` (string) - Required**

The target/destination branch.

### **93.6 `repository` (string) - Required**

The name of the CodeRepo this PR belongs to.

### **93.7 `approved` (boolean) - Optional**

Indicates if every commit associated with this PR has been approved by a reviewer other than the code author.

### **93.8 `validated` (boolean) - Optional**

Indicates if every commit associated with this PR was submitted by a validated author known to the organization.

A password policy is a specific *Ruleset*. It is separately defined because of its pervasive usage across digital environments and the well known properties (such as length and complexity) unique to a password policy.

Includes properties from:

- Entity
- Metadata

#### 94.1 minLength (integer) - Optional

Minimum password length

#### 94.2 requireSymbols (boolean) - Optional

Indicates if a password must contain at least one symbol

#### 94.3 requireNumbers (boolean) - Optional

Indicates if a password must contain at least one number

#### 94.4 requireUppercase (boolean) - Optional

Indicates if a password must contain at least one uppercase character

## 94.5 `requireLowercase` (boolean) - Optional

Indicates if a password must contain at least one lowercase character

## 94.6 `maxAgeDays` (integer) - Optional

Specifies how long (in days) a password remains valid before it expires (0 indicates no limit - passwords do not expire)

## 94.7 `minAgeMins` (integer) - Optional

Specifies the minimum time interval (in minutes) between password changes (0 indicates no limit)

## 94.8 `historyCount` (integer) - Optional

Specifies the number of previous passwords that users are prevented from reusing (0 indicates none)

## 94.9 `preventReset` (boolean) - Optional

Indicates if the user is allowed/prevented to change their own password

## 94.10 `expiryWarningDays` (integer) - Optional

Specifies the number of days prior to password expiration when a user will be warned to reset their password (0 indicates no warning)

## 94.11 `hardExpiry` (boolean) - Optional

Specifies whether users are prevented from setting a new password after their password has expired

## 94.12 `excludeUsername` (boolean) - Optional

Indicates if the username must be excluded from the password

## 94.13 `excludeAttributes` (array of string) - Optional

The user profile attributes whose values must be excluded from the password

## 94.14 `excludeCommonPasswords` (boolean) - Optional

Indicates whether to check passwords against a common/weak password dictionary

## 94.15 `lockoutAttempts` (integer) - Optional

Specifies the number of times users can attempt to log in to their accounts with an invalid password before their accounts are locked (0 indicates no limit)

## 94.16 `autoUnlockMins` (integer) - Optional

Specifies the time interval (in minutes) a locked account remains locked before it is automatically unlocked (0 indicates no limit)

## 94.17 `requireMFA` (boolean) - Optional

Specifies whether multi-factor authentication (MFA) is required





---

### Person

---

An entity that represents an actual person, such as an employee of an organization.

Includes properties from:

- [Entity](#)
- [Metadata](#)

#### 95.1 `firstName` (string) - Required

The person's official first name in the system (such as HR database)

#### 95.2 `lastName` (string) - Required

The person's official last name in the system (such as HR database)

#### 95.3 `middleName` (string) - Optional

The person's official middle name in the system (such as HR database)

#### 95.4 `email` (array of string) - Required

The email addresses of the person; the first one in the array is the primary email.

## 95.5 title (string) - Optional

The person's role or title within an organization

## 95.6 phone (array of string) - Optional

The person's phone numbers; the first one in the array is the primary contact number.

## 95.7 address (string) - Optional

The person's physical contact address

## 95.8 employeeId (string) - Optional

The person's employee ID/number within an organization

## 95.9 employeeType (string) - Optional

The type of employment

### Options

- employee
- contractor
- intern
- vendor
- advisor
- other

## 95.10 userIds (array of string) - Optional

One or more user IDs associated with this person

## 95.11 manager (string) - Optional

Name of the person's manager

## 95.12 managerId (string) - Optional

Employee ID of the person's manager

## 95.13 `managerEmail` (string) - Optional

Email of the person's manager

Format: email



A written policy documentation.

Includes properties from:

- `RecordEntity`
- `Metadata`

### 96.1 `title (string)` - Required

Title of the policy

### 96.2 `summary (string)` - Required

Summary or overview the describes the policy. Summary text is intended as guidance to the author and not included in the published version.

### 96.3 `author (string)` - Optional

Author of the record

### 96.4 `content (string)` - Required

Text content of the policy. For policies/procedures used by the Policy Builder app, this will contain the template text in markdown format. Stored in raw data.

## 96.5 `applicable` (boolean) - Optional

Indicates if policy or procedure is applicable based on the organization's current risk and compliance needs. A Policy that is not applicable may become applicable later as the organization's requirements and maturity change.

## 96.6 `adopted` (boolean) - Optional

Indicates if policy or procedure has been adopted. Only adopted policies and procedures are included in the published view of the Policy Builder app.

A written procedure and control documentation. A Procedure typically *IMPLEMENTS* a parent Policy. An actual Control further *IMPLEMENTS* a Procedure.

Includes properties from:

- [RecordEntity](#)
- [Metadata](#)

### 97.1 `title` (string) - Required

Title of the procedure

### 97.2 `summary` (string) - Required

Summary or overview the describes the procedure. Summary text is intended as guidance to the author and not included in the published version.

### 97.3 `author` (string) - Optional

Author of the record

### 97.4 `content` (string) - Required

Text content of the policy. For policies/procedures used by the Policy Builder app, this will contain the template text in markdown format. Stored in raw data.

## 97.5 `control` (string) - Optional

The type of control specified by this procedure.

### Options

- administrative
- technical
- physical
- operational
- other

## 97.6 `applicable` (boolean) - Optional

Indicates if procedure is applicable based on the organization's current risk and compliance needs. A Policy that is not applicable may become applicable later as the organization's requirements and maturity change.

## 97.7 `adopted` (boolean) - Optional

Indicates if procedure has been adopted. Only adopted policies and procedures are included in the published view of the Policy Builder app.



A compute process – i.e. an instance of a computer program / software application that is being executed by one or many threads. This is NOT a program level operational process (i.e. a Procedure).

Includes properties from:

- Entity
- Metadata

### 98.1 state (string) - Optional

Indicates the state of the process.



A software development project. Can be used for other generic projects as well but the defined properties are geared towards software development projects.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 99.1 `key` (string) - Optional

A defined project key. It is ideal for a code project to have a consistent key that matches that of issue tracking project. For example, the key for a Bitbucket project should match the key of its corresponding Jira project.

### 99.2 `productionURL` (string) - Optional

The Production URL

Format: uri

### 99.3 `stagingURL` (string) - Optional

The Non-Production / Staging URL

Format: uri

## 99.4 `devURL` (string) - Optional

The Development URL

Format: uri

## 99.5 `testURL` (string) - Optional

The Test URL

Format: uri

## 99.6 `alternateURLs` (array of string) - Optional

The additional URLs related to this application.

## CHAPTER 100

---

### Record

---

A DNS record; or an official record (e.g. Risk); or a written document (e.g. Policy/Procedure); or a reference (e.g. Vulnerability/Weakness). The exact record type is captured in the `_type` property of the Entity.

Includes properties from:

- `RecordEntity`
- `Metadata`



# CHAPTER 101

---

## RecordEntity

---

A node in the graph database that represents a Record Entity, with a set of different defined common properties than standard (resource) entities.

Includes properties from:

- [Metadata](#)

### 101.1 `name (string)` - Required

Name of this entity

### 101.2 `displayName (string)` - Required

Display name, e.g. a person's preferred name or an AWS account alias

### 101.3 `summary (string)` - Optional

A summary / short description of this entity.

### 101.4 `description (string)` - Optional

An extended description of this entity.

## 101.5 classification (string) - Optional

The sensitivity of the data; should match company data classification scheme. For example: critical - confidential - internal - public.

### Example Values

- critical
- confidential
- internal
- public

## 101.6 category (string) - Optional

The category of the official record

### Options

- exception
- finding
- hr
- incident
- issue
- job
- legal
- request
- policy
- procedure
- problem
- review
- risk
- other

## 101.7 webLink (string) - Optional

Hyperlink to the location of this record, e.g. URL to a Jira issue

Format: uri

## 101.8 content (string) - Optional

Text content of the record/documentation



## 101.9 `open` (boolean) - Optional

Indicates if this record is currently open. For example, an open Vulnerability finding (Vulnerability extends Record).

## 101.10 `public` (boolean) - Optional

If this is a public record. Defaults to false.

## 101.11 `production` (boolean) - Optional

If this is a production record. For example, a production change management ticket would have this set to *true*, and have a *category = change* property. Another example would be a Vulnerability finding in production.

## 101.12 `approved` (boolean) - Optional

If this is record has been reviewed and approved.

## 101.13 `approvedOn` (number) - Optional

The timestamp (in milliseconds since epoch) when this record was approved.

Format: date-time

## 101.14 `approvers` (array of string) - Optional

The list of approvers on the record.

## 101.15 `reporter` (string) - Optional

The person or system that reported or created this record.

## 101.16 `reportedOn` (number) - Optional

The timestamp (in milliseconds since epoch) when this record was reported/opened. In most cases, this would be the same as *createdOn* but occasionally a record can be created at a different time than when it was first reported.

Format: date-time

## 101.17 `createdOn` (number) - Optional

The timestamp (in milliseconds since epoch) when the entity was created at the source. This is different than `_createdOn` which is the timestamp the entity was first ingested into JupiterOne.

Format: date-time

## 101.18 `updatedOn` (number) - Optional

The timestamp (in milliseconds since epoch) when the entity was last updated at the source.

Format: date-time

A Relationship is the edge between two Entity nodes in the graph. The *\_class* of the relationship should be, in most cases, a generic descriptive verb, such as ‘HAS’ or ‘IMPLEMENTS’.

Includes properties from:

- [Metadata](#)

### 102.1 *\_class* (string) - Optional

Contains an enumeration of defined Relationship classes.

#### Options

- ALLOWS
- ASSIGNED
- CONNECTS
- CONTAINS
- CONTRIBUTES\_TO
- DENIES
- DEPLOYED\_TO
- EVALUATES
- EXPLOITS
- EXTENDS
- HAD
- HAS
- IS

- IDENTIFIED
- IMPACTS
- IMPLEMENTS
- MANAGES
- MITIGATES
- MONITORS
- PERFORMED
- PERMITS
- PROTECTS
- PROVIDES
- REJECTS
- OWNS
- TRIGGERS
- TRUSTS
- USES

## 102.2 displayName (string) - Optional

Display name of this relationship. By default, or if this property is not set, the Relationship should display the value of its `_class`, such as 'HAS' or 'IMPLEMENTS'.

## 102.3 webLink (string) - Optional

Web link to the source. For example, with a relationship like *CodeRepo -HAS-> Vulnerability*, there could be a `webLink` on the *HAS* relationship that points to a Jira issue to track that particular finding instance.

Format: uri

## 102.4 isValidated (boolean) - Optional

Indicates if this relationship has been validated.

## 102.5 isTemporary (boolean) - Optional

Indicates if this is a temporary relationship.

## 102.6 `isGroupLayout` (boolean) - Optional

Indicates if relationship represent a group. If true, visually this should be implemented with group styling such that the child nodes are shown contained within their parent boundary, instead of shown as lines connecting the nodes.

## 102.7 `tag.*` (string) - Optional

Named tags assigned to the entity (i.e., 'tag.Name', 'tag.OtherName')

## 102.8 `tags` (array of string) - Optional

An array of unnamed tags



An individual requirement for security, compliance, regulation or design.

Includes properties from:

- `RecordEntity`
- `Metadata`

### **103.1 `title (string)` - Required**

The title text of the requirement.

### **103.2 `summary (string)` - Optional**

The summary text of the requirement.

### **103.3 `state (string)` - Optional**

The state of the requirement (e.g. 'implemented').





## CHAPTER 104

---

### Resource

---

A generic assignable resource. A resource is typically non-functional by itself unless used by or attached to a host or workload.

Includes properties from:

- [Entity](#)
- [Metadata](#)



A review record.

Includes properties from:

- `RecordEntity`
- `Metadata`

### 105.1 `title (string)` - Required

The title text of the review.

### 105.2 `summary (string)` - Optional

The summary text of the review.

### 105.3 `state (string)` - Optional

The state of the review.



An object that represents an identified Risk as the result of an Assessment. The collection of Risk objects in JupiterOne make up the Risk Register. A Control may have a *MITIGATES* relationship to a Risk.

Includes properties from:

- [RecordEntity](#)
- [Metadata](#)

### 106.1 **assessment (string) - Optional**

The name/id of the assessment that produced this risk.

### 106.2 **category (string) - Optional**

The category (or area) of the risk. For example, 'process maturity' or 'natural disaster'.

### 106.3 **probability (integer) - Required**

Probability rating of the risk: '3: high/certain', '2: medium/likely', '1: low/unlikely', '0: none/negligible'.

#### Options

- 0
- 1
- 2
- 3

## 106.4 `impact` (integer) - Required

Impact rating. '3: high/severe', '2: medium/moderate', '1: low/minor', '0: none/insignificant'.

### Options

- 0
- 1
- 2
- 3

## 106.5 `score` (integer) - Required

Overall Risk Score = Probability x Impact

## 106.6 `details` (string) - Optional

Additional details to describe the risk.

## 106.7 `mitigation` (string) - Optional

Description of the mitigation, either planned or implemented, if applicable.

## 106.8 `status` (string) - Required

Current status of this documented risk. Default status is *open*.

### Options

- reported
- acknowledged
- accepted
- mitigated
- prioritized
- transferred
- pending
- open

## CHAPTER 107

---

Root

---

The root node in the graph. There should be only one Root node per organization account.

Includes properties from:

- [Metadata](#)

### 107.1 `displayName` (string) - Optional

Display name





An operational or configuration compliance rule, often part of a Ruleset.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 108.1 category (string) - Optional

The category of ruleset.

#### Options

- compliance
- config
- password
- other

### 108.2 content (string) - Optional

Contents of the rule, if applicable.



An operational or configuration compliance ruleset with rules that governs (or enforces, evaluates, monitors) a security control or IT system.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 109.1 `category` (string) - Optional

The category of ruleset.

#### Options

- compliance
- config
- password
- other

### 109.2 `rules` (array of string) - Optional

Rules of ruleset. Each rule is written 'as-code' that can be operationalized with a control provider or within JupiterOne's rules engine.

### 109.3 `content` (string) - Optional

Contents of the raw rules, if applicable.



# CHAPTER 110

---

## Scanner

---

A system vulnerability, application code or network infrastructure scanner.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 110.1 category (array of string) - Required

The category of scanner

#### Options

- system
- network
- application
- other



A service provided by a vendor.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 111.1 category (array of string) - Required

The category of service, e.g. software, platform, infrastructure, other

#### Options

- software
- platform
- infrastructure
- other

### 111.2 endpoints (array of string) - Required

Array of service endpoints, e.g. ec2.amazonaws.com





The physical location of an organization. A Person (i.e. employee) would typically has a relationship to a Site (i.e. located\_at or work\_at). Also used as the abstract reference to AWS Regions.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 112.1 category (array of string) - Optional

Type of site

#### Options

- headquarters
- branch
- campus
- office
- aws-region
- data-center
- lab
- other

### 112.2 location (string) - Optional

The address/location of the site. Or an AWS Region (e.g. us-east-2).

### **112.3 hours (string) - Optional**

Hours of operation. e.g. M-F 9am-6pm

### **112.4 secured (boolean) - Optional**

Indicates the site is secured with physical controls such as key card access and surveillance cameras.

### **112.5 restricted (boolean) - Optional**

Indicates that access to the site is restricted (a level above secured access).

## CHAPTER 113

---

### Task

---

A computational task. Examples include AWS Batch Job, ECS Task, etc.

Includes properties from:

- [Entity](#)
- [Metadata](#)



---

### Team

---

A team consists of multiple member Person entities. For example, the Development team or the Security team.

Includes properties from:

- [Entity](#)
- [Metadata](#)

#### 114.1 `email` (string) - Optional

The team email address

Format: email



## CHAPTER 115

---

### Training

---

A training module, such as a security awareness training or secure development training.

Includes properties from:

- [RecordEntity](#)
- [Metadata](#)





---

## User

---

A user account/login to access certain systems and/or services. Examples include okta-user, aws-iam-user, ssh-user, local-user (on a host), etc.

Includes properties from:

- Entity
- Metadata

### 116.1 username (string) - Required

Username

### 116.2 email (string) - Optional

The email address associated with the user account

Format: email

### 116.3 shortLoginId (string) - Optional

The shortened login Id. For example, if the username is the full email address (`first.last@company.com`), the shortLoginId would be the part before `@` (`first.last`).

### 116.4 mfaEnabled (boolean) - Optional

Specifies whether multi-factor authentication (MFA) is enabled for this user.



# CHAPTER 117

---

## UserGroup

---

A user group, typically associated with some type of access control, such as a group in Okta or in Office365. If a UserGroup has an access policy attached, and all member Users of the UserGroup would inherit the policy.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 117.1 email (string) - Optional

The group email address

Format: email



An external organization that is a vendor or service provider.

Includes properties from:

- [Entity](#)
- [Metadata](#)

### 118.1 category (string) - Required

The category of vendor.

#### Options

- business-operations
- cloud
- facilities
- finance
- infrastructure
- legal
- purchasing
- security
- software
- platform-development
- platform-social-media
- professional-services-staffing
- professional-services-recruiting

- professional-services-consulting
- generic-service-provider
- generic-subscription
- CSP
- ISP
- MSP
- MSSP
- IdP
- other

## 118.2 `website` (string) - Optional

The vendor's main website URL.

Format: uri

## 118.3 `departments` (array of string) - Optional

List of business departments the vendor provides service for (e.g. IT, HR, Finance, Marketing, Development/Engineering, Security).

## 118.4 `emailDomain` (string) - Optional

The email domain for the vendor (e.g. @jupiterone.io).

## 118.5 `mainContactName` (string) - Optional

The vendor's point of contact person.

## 118.6 `mainContactEmail` (string) - Optional

Email of the vendor's point of contact person.

Format: email

## 118.7 `mainContactPhone` (string) - Optional

Phone number of the vendor's point of contact person.

## 118.8 `mainContactAddress` (string) - Optional

Main physical/mailing address of the vendor.

## 118.9 `admins` (array of string) - Optional

List of admin users to the vendor account, if applicable. If this vendor account is integrated directly to JupiterOne and its data is ingested, the admin users should be already mapped as User entities.

## 118.10 `breachResponseDays` (integer) - Optional

The number of days the vendor agrees to report an identified data breach, per vendor agreement and/or SLA. This is typically 3 to 30 days. Note that GDPR requires breach notification within 3 days / 72 hours.

## 118.11 `linkToNDA` (string) - Optional

Link to Non-Disclosure Agreement (NDA) document.

Format: uri

## 118.12 `linkToMSA` (string) - Optional

Link to Master Service Agreement (MSA) document.

Format: uri

## 118.13 `linkToSLA` (string) - Optional

Link to Service Level Agreement (SLA) document.

Format: uri

## 118.14 `linkToBAA` (string) - Optional

Link to Business Associate Agreement (BAA) document - for HIPAA only.

Format: uri

## 118.15 `linkToDPA` (string) - Optional

Link to GDPR Data Processing Addendum (DPA) document - for GDPR only.

Format: uri

## 118.16 `linkToVTR` (string) - Optional

Link to the external vendor technology risk (VTR) report.

Format: uri

## 118.17 `linkToISA` (string) - Optional

Link to the external information security assessment (ISA) report.

Format: uri

## 118.18 `statusPage` (string) - Optional

Link to the vendor's service status page (e.g. <https://status.aws.amazon.com/>).

Format: uri



A security vulnerability (application or system or infrastructure). A single vulnerability may relate to multiple findings and impact multiple resources. The *IMPACTS* relationship between the Vulnerability and the resource entity that was impacted serves as the record of the finding. The *IMPACTS* relationship carries properties such as 'identifiedOn', 'remediatedOn', 'remediationDueOn', 'issueLink', etc.

Includes properties from:

- [RecordEntity](#)
- [Metadata](#)

### 119.1 category (string) - Required

The category of the vulnerability finding

#### Options

- application
- system
- infrastructure
- other

### 119.2 status (string) - Optional

Status of the vulnerability

### 119.3 severity (string) - Required

Severity rating based on impact and exploitability. Can be a string such as 'critical', 'high', 'medium', 'low', 'info'. Or an integer usually between 0-5.

### 119.4 priority (string) - Optional

Priority level mapping to Severity rating. Can be a string such as 'critical', 'high', 'medium', 'low', 'info'. Or an integer usually between 0-5.

### 119.5 score (number) - Optional

The overall vulnerability score, e.g. CVSSv3.

### 119.6 impact (number) - Optional

The impact score/rating.

### 119.7 exploitability (number) - Optional

The exploitability score/rating.

### 119.8 vector (string) - Optional

The vulnerability attack vector. (e.g. a CVSSv3 vector looks like this - 'AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N')

### 119.9 impacts (array of string) - Optional

The target listing of projects, applications, repos or systems this vulnerability impacts. Specifying either the project/repo name or the application URL here will auto-map this Vulnerability to the corresponding Project/CodeRepo/Application entity if a match is found.

### 119.10 remediationSLA (integer) - Optional

The number of days that the Vulnerability must be remediated within, based on SLA set by the organization's internal vulnerability management program policy. The actually due date is set by 'remediationDueOn' property on the *IMPACTS* relationship between the Vulnerability and its impacted resource entity.

## 119.11 blocking (boolean) - Required

Indicates whether this vulnerability finding is a blocking issue. If true, it should block a production deploy. Defaults to false.

## 119.12 open (boolean) - Required

Indicates if this is an open vulnerability.

## 119.13 production (boolean) - Required

Indicates if this vulnerability is in production.

## 119.14 public (boolean) - Required

Indicates if this is a publicly disclosed vulnerability. If yes, this is usually a CVE and the 'webLink' should be set to 'https://nvd.nist.gov/vuln/detail/\${CVE-Number}' or to a vendor URL. If not, it is most likely a custom application vulnerability.

## 119.15 validated (boolean) - Optional

Indicates if this Vulnerability finding has been validated by the security team.

## 119.16 references (array of string) - Optional

The array of links to references.



A security weakness.

Includes properties from:

- [RecordEntity](#)
- [Metadata](#)

### 120.1 category (string) - Optional

The category of the vulnerability finding

#### Options

- application
- system
- infrastructure
- other

### 120.2 exploitability (string) - Optional

Indicates the likelihood of exploit.

### 120.3 references (array of string) - Optional

The array of links to references.



A virtual compute instance, it could be an aws-ec2-instance, a docker-container, an aws-lambda-function, an application-process, or a vmware-instance. The exact workload type is described in the `_type` property of the Entity.

Includes properties from:

- Entity
- Metadata

### 121.1 `image (string)` - Optional

The image this workload is derived from, such as an AMI or docker image. At the abstract level, this usually maps to the `_id` of a Resource.

### 121.2 `fqdn (string)` - Optional

The fully qualified domain name of attached to the instance, if applicable





2018-09-21

### 122.1 New Features

- **Asset Inventory CSV Download** now available to generate report based on current selection of assets.

### 122.2 Improvements

- **AWS Integration:** Additional properties ingested for EC2 EBS Volume Entities, including encryption status, availability zone, size, IOPS, state, snapshot ID, and volume type.
- **Galaxy Graph Viewer:** Major performance improvements.
- **Asset Inventory:** Filtering is now using a more intuitive condition logic for property and tag values.
- **Asset Inventory:** Tags are now grouped together at the top of filtering pane.
- **Asset Inventory:** Added icons to better indicate the `true/false/empty_string/null` values of a property.

### 122.3 Bug Fixes

- Root / now redirects properly to the home/landing page.
- Integration links on landing page are now linked to page for viewing existing instances.
- Encryption status for S3 Bucket Entities are now correctly displayed in Asset Inventory app.



2018-10-04

### 123.1 New Features

- **AWS Integration Multi-Region** is now supported. JupiterOne will now ingest and analyze resources across all regions by default. This is automatically enabled and there is no change needed on your side. This will enable you to see any resource that is potentially misplaced or malicious in a region you did not intend to use.
- **In-app Help and Guides** is now available. It can be activated via the Help Menu at the top navigation bar.

### 123.2 Improvements

- Backend and infrastructure stability improvements
- Improved error handling
- Captured additional properties from AWS EC2 instances



2018-10-15

### 124.1 New Features

- **Search and Query** with the JupiterOne Query Language (J1QL) - J1QL seamlessly blends full-text search and querying of the entity-relationship graph. It is simple to construct which aspires to be as close to natural language as possible. It is available directly on the home page / Landing Zone app.
- **Automatic mapping of AWS IAM users to Person/employee entities** if the IAM username is the same as the email address belonging to the Person/employee.

### 124.2 Improvements

- Stability improvements of the graph database backend
- Bug fixes for indexing and full-text search services



2018-11-14

### 125.1 New Features

- **Packaged Questions** are now provided as part of the Query Library, accessible directly in the Landing Zone app, next to the Query/Search bar. This allows you to ask simple compliance and operational questions with our pre-built queries based on the data model and/or specific data integrations. The packaged questions are tagged with flags such as `compliance`, `DevOps`, `SecOps`, `aws`, `HIPAA`, etc. so that you can easily search/filter them.
- **Saved Queries/Questions and History:** From the Landing Zone, you can access history of your previously executed queries. You can save them to be readily accessible in the future without having to re-type the query. You can also clone a packaged question/query and customize it in your saved query library.
- **Okta Applications:** JupiterOne now ingest Application entities from Okta. You can see these in the Asset Inventory view or Graph Viewer or as part of your query results.
- **Github Integration:** The first iteration of Github integration is live! In this version, you will need to configure JupiterOne as an OAuth app directly in your Github application and use the OAuth Key/Secret to configure the integration within JupiterOne. A future version will enable JupiterOne as a published Github app.
- **In-app Support:** You can now easily search for knowledge base articles or submit a support ticket directly within the JupiterOne web interface.

### 125.2 Improvements

- Improved how full text search handles multiple search strings.
- Various tweak and improvements to J1QL.
- Improved relationship mapping rules and engine.





2018-12-18

### 126.1 New Features

- Lots of features added to JupiterOne Query Language (J1QL)
  - Query now supports **aggregations**  
*Example:* `FIND User as u return COUNT(u)` to get a count of all users
  - Query now supports **date comparison**  
*Example:* `FIND * with _beginOn > date.now-24hrs` to find all resources that changed in the last 24 hours.
  - Query now supports **simplified selection of multiple entities (OR)**  
*Example:* `Find (Host|Device) with ipAddress='10.50.2.17'` is equivalent to `Find * with _class='Host' or _class='Device' with ipAddress='10.50.2.17'`
- You can now manually **add entities** via the Asset Inventory app
- You can see **detailed properties in a side panel** by selecting an entity in the Asset Inventory app
- New **Users and Access, My Profile** and **Invitations** experience
- JupiterOne now ingests **AWS networking** data and maps out detailed relationships to enable deep analysis of network traffic access permissions. This allows security teams to gain accurate insight into which host(s) or network(s) are truly accessible from an external network or host (e.g. the Internet).
- JupiterOne now analysis **AWS assume role** policy to determine trust relationships between an IAM role and a service or another IAM principal either within the same account or from an external account.
- Users can now sign on to JupiterOne via **Google Sign On**.
- New packaged questions and queries added/updated by operational domain:

- [general] *Who are the new hires within the last 12 months?*
- [general] *What business applications are we using?*
- [general] *Which are my documented risks?*
- [general] *Who are my vendors? Do I have a BAA/DPA/NDA/MSA with them?*
- [general] *What changed in my environment in the last 24 hours?*
- [general] *What was added to my environment in the last 24 hours?*
- [access] *Are there external users with access to our systems?*
- [appdev] *What are the code repos for a particular application or project?*
- [data] *Which data stores do not have proper classification tags?*
- [data] *Which production data stores do not have proper classification tags?*
- [data] *Is there any known critical data outside of production?*
- [data] *Show me evidence of data-at-rest encryption for production servers.*
- [data] *Is my critical data in production encrypted?*
- [data] *Is my production or PHI/PII data stores encrypted?*
- [data] *Which production data stores do not have proper classification tags?*
- [data] *Is there any known critical data outside of production?*
- [data] *Is there unencrypted ePHI or PII?*
- [infra] *Is there proper segmentation/segregation of internal networks?*
- [infra] *Show listing of network layer firewall protection across all my environments.*
- [infra] *Show listing of active firewall protection across all my environments.*
- [infra] *Are there any active systems without host firewall protection?*
- [aws] *Which IAM roles are assigned which IAM policies?*
- [aws] *Who has access to my AWS accounts?*
- [aws] *Who has access to my production AWS accounts?*
- [aws] *Who has direct user access to my AWS accounts?*
- [aws] *Who has direct user access to my production AWS accounts?*
- [aws] *Who has access to my AWS accounts vis SSO?*
- [aws] *Who has access to my production AWS accounts via SSO?*
- [aws] *Who has access to my AWS accounts via SSO in a multi-account environment?*
- [aws] *Who can assume which role across my AWS environment?*
- [aws] *Are there assume role trusts to external entities?*
- [aws] *Are there any EBS volumes not in use?*
- [aws] *What Lambda functions are in my environment?*
- [aws] *How are my Lambda functions invoked?*
- [aws] *Which security group rules allow inbound traffic from a public network or host on the Internet?*
- [aws] *Which security group rules allow outbound traffic to a public network or host on the Internet?*

- [aws] *Which security group rules allow inbound traffic from the Internet?*
- [aws] *Which security group rules allow outbound traffic to the Internet?*

## 126.2 Improvements and Bug Fixes

- Improvements on relationship mapping:
  - Support mapping to multiple targets from an array of source property values
  - Support mapping from multiple source properties to a single target property
  - Allow an integration to create target mapped entity from a relationship
  - Trigger mapper to run when `mappings.json` configuration changes
- Added lots of documentation on data model, details of data ingestion from integrations, and data security.
- Fixed “encrypted” property showing up as “undefined” for some AWS entities
- Added AWS S3 region and improved encryption properties

e.g. *you can find data stores outside of EU with a query like* `FIND DataStore with region != 'eu-central-1'`
- A few other small bug fixes and UI improvements



2019-01-07

### 127.1 New Features

- **Graph** mode for the search/query results in Landing Zone is now available! You can switch to Graph mode for any search or query to get a focused visual of the entities and relationships from the results. The graph is interactive so that you can further expand for deeper analysis.
- Much improved **Search in Landing Zone** that allows all of the following modes in one place:
  1. **Keywords search** to ask saved/packaged questions
  2. **Full text search** across all entities based on their property values
  3. **JupiterOne query language (J1QL)** for precise querying of entities and relationships
  4. **Combining** full text search with J1QL
- New ingestion and analysis from **AWS**:
  - S3 Bucket ACL processing and access mapping
  - S3 Bucket public access block configuration
  - Account password policy
  - IAM User MFA devices and access keys
- Added **OR** operator support on relationship keywords in **J1QL**. For example: Find HostAgent that (PROTECTS|MANAGES|MONITORS) Host
- Condensed **quick filter by entity class icons** in Asset Inventory app.
- You can **edit or delete** an entity manually from the Asset Inventory app.
- **Web links** are added to most entities ingested, allowing you to directly open in a new tab to view the resource in the source web console.

- Added linking to **Geolocation lookup of IP Address and CIDR** of a Host or Network.
- New packaged questions and queries added:
  - [general] *What are my information assets?*
  - [general] *What are my production data stores and databases?*
  - [general] *What are my production resources?*
  - [general] *What are my production applications?*
  - [general] *Which devices have been disposed in the last 12 months?*
  - [access] *Who has been assigned permissions with 'Admin' access?*
  - [access] *Who owns which user accounts?*
  - [access] *What are the shared/generic/service accounts? (user accounts that are not individually owned)*
  - [access] *Show me the current password policy and compliance status.*
  - [access] *Find anything that allows public access to everyone.*
  - [appdev] *Were there any Code Repos added in the last 24 hours?*
  - [data] *Is my production or PHI/PII data stores encrypted?*
  - [data] *Are there any non-public data stores incorrectly configured with public access to everyone?*
  - [endpoint] *What is the configuration and compliance status of my endpoint devices?*
  - [endpoint] *Whose endpoint is out of compliance?*
  - [endpoint] *Is there malware protection for all endpoints?*
  - [endpoint] *Are there security agents monitoring and protecting my endpoint hosts/devices?*
  - [endpoint] *Are my servers and systems protected by hosted-based firewall?*
  - [infra] *Are there potential IP collisions among the networks/subnets in my environment?*
  - [infra] *What are directly connected to the Internet?*
  - [infra] *What network traffic is allowed between internal and external networks?*
  - [infra] *Is there proper segmentation/segregation of internal networks?*
  - [infra] *Are wireless networks segmented and protected by firewalls?*
  - [infra] *Are there VPN configured for remote access?*
  - [infra] *Show all inbound SSH firewall rules across my network environments.*
  - [infra] *Is inbound SSH allowed directly from an external host or network?*
  - [aws] *Is MFA enabled for the Account Root User for all my AWS accounts?*
  - [aws] *Are there root user access keys in use for any of my AWS accounts?*
  - [aws] *Is public access block configured for non-public S3 Buckets?*
  - [aws] *Is public read access enabled for any S3 Bucket?*
  - [aws] *Is public write access enabled for any S3 Bucket?*
  - [aws] *Is S3 bucket access granted to anybody outside of the account?*
  - [aws] *Is there any S3 bucket that grants full control access to anybody other than the owner?*

- [aws] *What are the service roles in my AWS accounts (i.e. an IAM Role that has a trust policy to an AWS Service)?*
- [aws] *Are all EBS volumes encrypted?*
- [aws] *Is default server side encryption enabled for all S3 Buckets?*
- [aws] *Who has been assigned full Administrator access?*
- [aws] *Are there assume role trusts to external entities?*
- [aws] *Are all the AWS Config rules compliant? (if AWS Config service is enabled)*
- [aws] *Are there any noncompliant production resources in AWS per Config evaluation? (if AWS Config is enabled)*
- [aws] *Are there EC2 instances exposed to the Internet?*
- [aws] *Which EC2 instances may have external network connections?*

## 127.2 Improvements and Bug Fixes

- Improved username display next to the user avatar.
- UI/UX improvements on Landing Zone search, with **Clear**, **Save**, and **Clear All** action buttons for query results.
- Improved accuracy of full-text search.
- Fixed missing column in some query/search results.
- Fixed account name tagging not enabled by default in certain integration configurations.
- Several stability and robustness improvements on backend services.
- New icons for several entity classes.





2019-02-21

### 128.1 New Features

- New ingestion and analysis from **AWS**:
  - **RDS** clusters and instances  
`TryFind aws_rds_cluster that CONTAINS aws_db_instance return tree`
  - **DynamoDB** tables  
`TryFind aws_dynamodb_table that relates to * return tree`
  - **S3** bucket public access settings  
`TryFind aws_s3_bucket with BlockPublicAcls != true`
  - **AMI** images - note that only custom AMI images are currently ingested, not public or marketplace AMIs.  
`TryFind aws_ami that relates to * return tree`
- **SAML Single Sign On (SSO)** now generally available to enterprise customers
- **Endpoint Compliance Agent** powered by Stethoscope app released for macOS devices. Access it from the “Power Ups” menu, and send invite to your users by email. The agent checks the following endpoint configuration with the default policy:
  - OS version
  - Patching/update status
  - Host firewall status
  - Disk encryption status
  - Screensaver / screen lock protection

- Remote login status
- **Veracode Integration** first iteration - supports ingestion of Vulnerability findings.
- **Google Integration** first iteration - supports ingestion of Users and User Groups.
- **Sharing URL** is added to query results from Landing Zone.
- New packaged questions and queries added:
  - [aws] *Find all the IAM user access keys in production AWS accounts.*
  - [aws] *Find all the SSH key pairs in production AWS accounts.*
  - [aws] *Are there SSH keys not in use?*
  - [aws] *Is there anything that connects to an external AWS account that is not part of my organization?*
  - [access] *Did we remove all access from employees who left?*
  - [access] *Which user accounts do not have multi-factor authentication enabled?*
  - [appdev] *Who are the most recent contributors to this repo?*
  - [appdev] *Which PRs did this developer open in the last 5 days?*
  - [data] *What is the inventory of my sensitive data stores?*
  - [endpoint] *Is operating system patching and auto update enabled on endpoint hosts?*
  - [endpoint] *Is application patching and auto update enabled on endpoint hosts?*
  - [endpoint] *What are the approved server/system images?*
  - [endpoint] *Are all system images updated in the past six months?*
  - [endpoint] *Which hosts are (or are not) using approved standard images?*
  - [infra] *What production resources are directly connected/exposed to the Internet/everyone?*
  - [general] *What applications and operating systems are in use?*
  - [general] *Who are my software vendors? Do I have proper vendor support for my software applications?*

## 128.2 Improvements and Bug Fixes

- Added column sorting of query results from Landing Zone.
- Continued improvements of backend services.
- Several bug fixes of indexer, mapper, persister and integrations.
- Fixed action and display bugs associated with adding/editing an entity in Asset Inventory.
- Lots of improvements made to the managed SDK to support open source integration development.
- Updated timestamp properties for AWS integration to number instead of string so that queries can use them for date/time comparison.
- Renamed AWS entity `_type` definitions to be more consistent with the Terraform type naming convention.

2019-03-05

### 129.1 New Features

- New ingestion and analysis from **AWS**:
  - **ELB** entities - application load balancers and network load balancers  
`TryFind (aws_alb|aws_nlb)`
  - **ACM** certificates and relationships to resources that use each certificate  
`Try Find aws_acm_certificate that relates to * return tree and Find aws_acm_certificate that !USES *`
  - **CloudFront** distributions and their origins  
`TryFind aws_cloudfront_distribution that CONNECTS * return tree`
  - **WAF** and the resources they protect  
`TryFind aws_waf_web_acl that PROTECTS * return tree`
- **Vendor** are now automatically inferred and mapped from single sign on applications (e.g. a SAML application in Okta). Additional properties of each vendor can be added via the Asset Inventory app. This gives you a jumpstart in cataloging and managing vendors.

### 129.2 Improvements and Bug Fixes

- Fixed missing tags for DynamoDB resources
- Raw data storage to support versioning and document store (e.g. policies)
- Various improvements to the managed integrations SDK

- Bug fixes and improvements to the mapper

2019-03-18

### 130.1 New Features

- Major UI update to provide a true single page app experience.
- Brand new experience for integrations configuration.  
new-integrations-ux
- New ingestion and analysis from **AWS**:
  - **AWS Transfer for SFTP** servers and users. Try this query:

```
Find aws_account
  that HAS aws_transfer
  that HAS Host
  that HAS User
  that RELATES TO *
return tree
```

- **Carbon Black PSC and Defense Sensor integration** released - ingests sensor agents, policies, and relationships indicating which sensor is assigned which policy. Also maps the sensors to end-user devices and the device owner. See [docs](#) for more details.

Try these queries:

```
Find cbdefense_sensor that PROTECTS Device

Find Person
  that OWNS Device
  that PROTECTS cbdefense_sensor
```

- **OneLogin integration** released - ingests users, user groups, applications and their relationships. Similar to the Okta integration, this allows you to easily query and visualize *who has access to what* within your identity and access configuration and the connections to the rest of your digital infrastructure. See [docs](#) for more details.

## 130.2 Improvements and Bug Fixes

- Improved entity relationships for Veracode integrations. Findings are mapped to vulnerabilities and weaknesses when applicable.
- Lots of documentation updates on <https://docs.jupiterone.io>, including:
  - Updated docs for each integration
  - [JupiterOne query tutorial](#)
- Improvements and bug fixes with AWS integration:
  - Lambda functions and Redshift clusters are now properly connected to their corresponding VPCs
  - Fixed an issue where duplicate Bucket ACL grants are processed
  - Added mapping to external/public AMIs not owned by the AWS account instance
  - Fixed a couple of Type Errors associated with property ingestion
- Fixed an authentication parameter issue with Google integration.
- Fixed an issue where deleted entities still appear in the Asset Inventory.

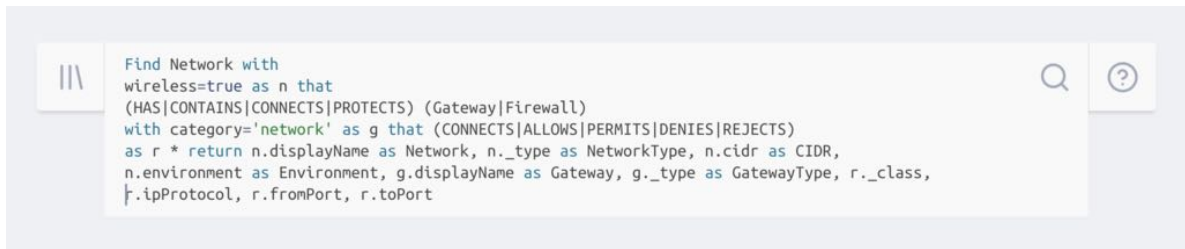
2019-04-01

### 131.1 New Features

*We are excited to announce a major release with a ton of new features:*

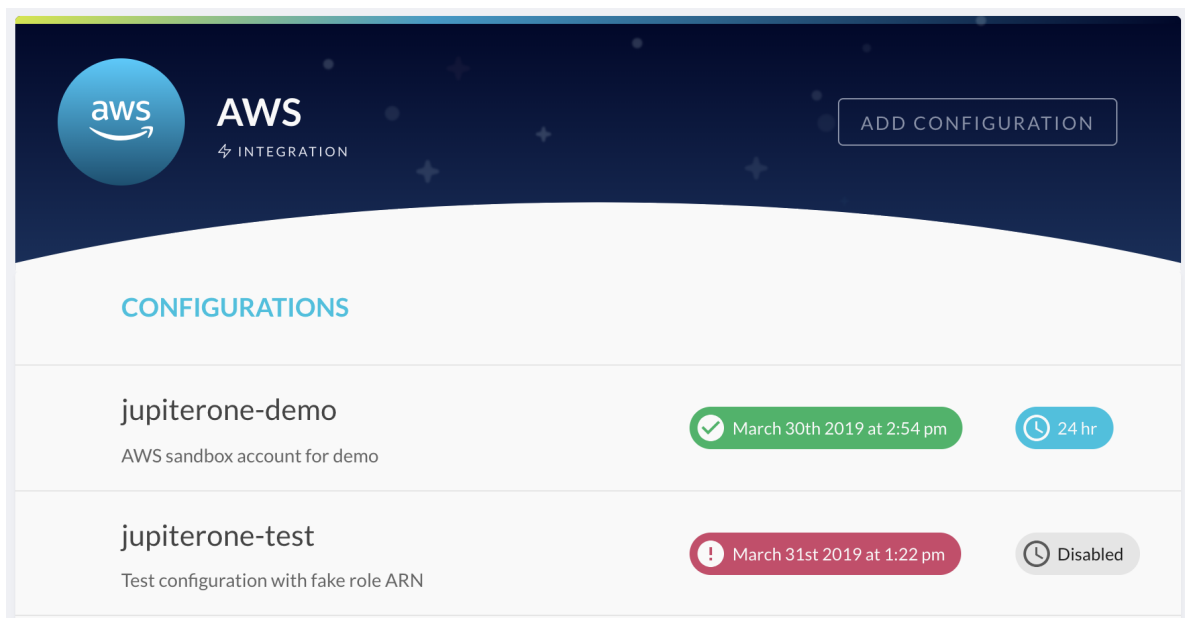
- Major update to the **Policies** app, including
  - the ability to build (from a library of **over 150 document templates**), edit and continuously manage your organization's security policies and procedures directly online, using the web app;
  - a hierarchical graph view of the policies and the procedures that implements each policy, which serves as a visual proof to compliance auditors that the policies are fully implemented;
  - ability to allow general staff (all employees) to log in to JupiterOne to view published version of the policies and procedures, and capture their acknowledgement and acceptance.
- Updated **Policy Builder CLI** with `publish` command to promote local copies of the policies and procedures to your JupiterOne account online, to easily take advantage of the new features in the web app.
- Brand new **Compliance** app (early access release) that
  - automatically connects policies, procedures to each individual requirements of supported compliance frameworks; and
  - maps stored queries to compliance requirements to generate evidences for audits and assessments, in a data-centric and fully automated manner.
  - provides support for compliance frameworks applicable to your organization, such as NIST and HIPAA.
- Brand new **Onboarding** experience when you first sign up for a JupiterOne account, with guided tutorials for the major features.
- Early access release of the **Alerts** app – alert rules must be provisioned via the API in this pre-release. Rules are configured using JIQL queries and alerted in the app. Future release will include email and Slack notifications.

- **WhiteHat Security integration** released - ingests vulnerability findings and maps them to applications or code repos as well as associated CVEs and CWEs as applicable. See [docs](#) for more details.
- **Wazuh OSSEC agent integration** released - ingests Wazuh OSSEC agents from your internally managed Wazuh deployment, and maps the agents to the hosts they protect. See [docs](#) for more details.
- New **App Switcher** design  
app-switcher
- **Multi-line query input** with syntax highlighting in the Landing app



line-query

- New **status indicators** to easily spot data ingestion or authentication errors for each integration configuration.



indicators

## 131.2 Improvements and Bug Fixes

- Fixed an issue with Okta integration where mapped relationship between an Okta user to the AWS IAM role assigned to the user was not properly removed when the assignment is revoked.
- Fixed the component height issue with the query result graph.
- Fixed an issue Asset Inventory app not properly filtering boolean properties.
- Improved performance and user experience when creating/updating/deleting an entity from the Asset Inventory web UI.



- Addressed the 30-second timeout limitation of Amazon API Gateway, to allow more complex queries to continue to execute in the background instead of returning an error.



2019-04-15

### 132.1 New Features

- Updates to early access **Alerts** app:
  - View alert details and dismiss alerts
  - Create and edit alert rules in the webapp (previously only via the API)
  - First alert rule pack released - rules for AWS configuration auditing: <https://github.com/JupiterOne/jupiterone-alert-rules/blob/master/rule-packs/aws-config.json>

Also see: <https://docs.jupiterone.io/en/latest/guides/j1-queries-for-aws-config.html>

- New **JupiterOne CLI** for querying and entity/relationship/alert operations via the command line. A **JupiterOne NodeJS Client** is also available to help with your own automation. <https://github.com/JupiterOne/jupiterone-client-nodejs>
- Ability to **enable API Key access** for one or more user groups to allow the users to generate API keys used for the external client or CLI.
- **Jira integration** initial release - ingests Jira issues and store them as Record entities from specified project(s). Maps the Jira users to employees and to the issues they created or reported.

This is especially useful if you track incidents and risks in Jira and would like them to be consolidated and mapped to the rest of your resources.

*The ability to create a Jira issue from a query or an alert is coming soon.*

- **SentinelOne integration** initial release - ingests SentinelOne endpoint agents and connects them to the devices and their owners. You can leverage the agent status as a contextual data point in security analysis.

For example, the following query gives you a visual graph of the employee that has an inactive SentinelOne agent, that person's device, and the user accounts that person has access to:

```
Find sentinelone_agent with isActive!=true as agent
  that protects Device as d
  that relates to Person as p
  that is User as u
return tree
```

QUERY Find sentinelone\_agent as agent that protects Device as d that relates to Person as p that is User as u return tree

Erkangs-LifeOmic-MacBook  
sentinelone\_agent (HostAgent)

Properties Tags Metadata

isUpToDate true

osStartTime 2019-04-08T06:12:52Z

lastLoggedInUserName erkang.zheng

registeredAt 2019-04-06T17:52:37.031854Z

osName OSX

coreCount 4

modelName MacBookPro13,3

sentinelone-inactive-user

- **AWS Inspector and GuardDuty integration** - You can now query for Inspector and GuardDuty findings in JupiterOne, and see a graph visualization of how the findings relate to CVEs and the resources they impact.

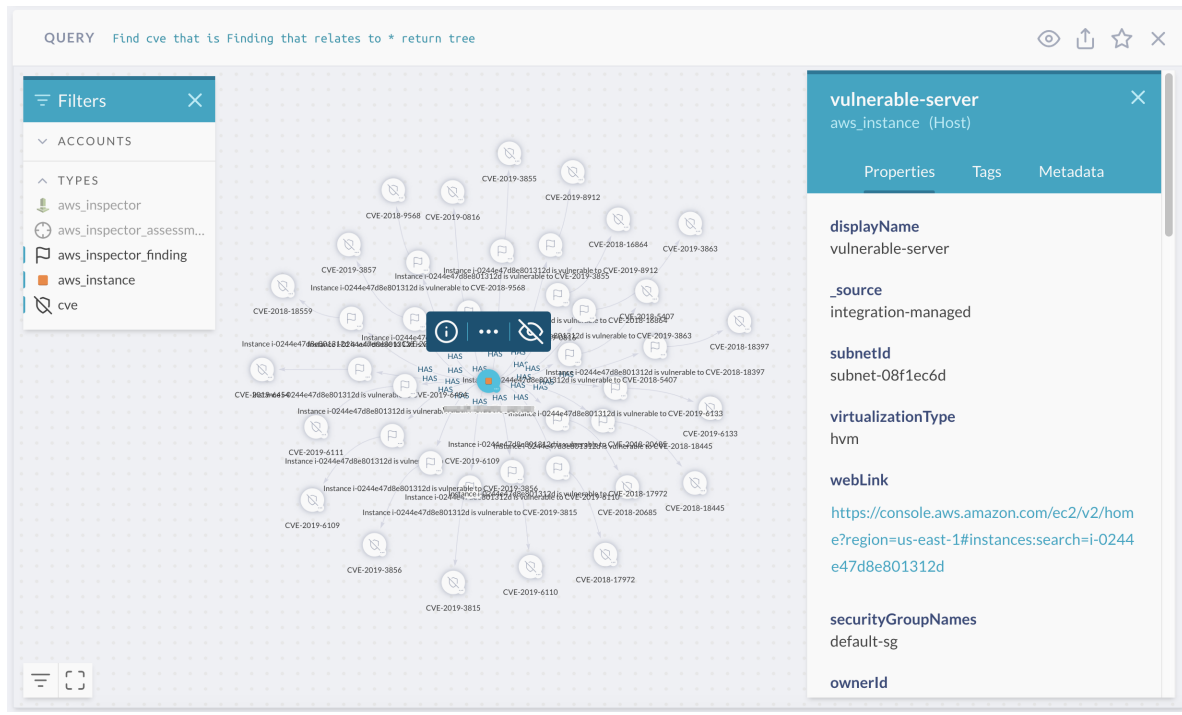
FILTER BY CLASS

FILTER BY TYPE

DISPLAY NAME	CLASS	TYPE	ACCOUNTNAME	PRODUCTION	NUMERIC SEVERITY	WEB LINK	DESCRI
Instance i-0244e47d8e801312d is vulnerable to CVE-2019-3855	Finding	aws_inspector_finding		—	9	<a href="#">https://co-1/finding</a>	An integ
Unusual user permission reconnaissance activity by Administrator.	Finding	aws_guardduty_finding		—	5	<a href="#">https://co-6f799b0d62</a>	APIs cor
Instance i-0244e47d8e801312d is not compliant with rule 5.2...	Finding	aws_inspector_finding		—	9	<a href="#">https://co-1/finding</a>	Descrip
Outbound portscan from EC2 instance i-0556e44e5478c9874.	Finding	aws_guardduty_finding		—	5	<a href="#">https://co-37fb08e697e</a>	EC2 insi
Aggregate network exposure: On instance i-09a3f0aa8dad85a4, por...	Finding	aws_inspector_finding		—	0	<a href="#">https://co-1/finding</a>	On inst
Instance i-0244e47d8e801312d is vulnerable to CVE-2019-3857	Finding	aws_inspector_finding		—	9	<a href="#">https://co-1/finding</a>	An integ
Unprotected port on EC2 instance i-06b850be795cea3cb is being probed.	Finding	aws_guardduty_finding		—	2	<a href="#">https://co-3852946892</a>	EC2 insi
Aggregate network exposure: On instance i-0674c2acd622e7ee, por...	Finding	aws_inspector_finding		✗	0	<a href="#">https://co-1/finding</a>	On inst
Instance i-0244e47d8e801312d is not compliant with rule 3.2...	Finding	aws_inspector_finding		—	9	<a href="#">https://co-1/finding</a>	Descrip
Unusual user permission reconnaissance activity by Administrator.	Finding	aws_guardduty_finding		—	5	<a href="#">https://co-afad7628ba</a>	APIs cor
Outbound portscan from EC2 instance i-0ed6606152e2c53b8.	Finding	aws_guardduty_finding		—	5	<a href="#">https://co-7dfb0adf96</a>	EC2 insi
On instance i-0244e47d8e801312d, process 'sshd' is listening on ...	Finding	aws_inspector_finding		—	6	<a href="#">https://co-1/finding</a>	On this

Rows per page: 50 1-50 of 238

findings-table



graph

- Inspector findings from multiple assessment runs are de-duplicated which significantly cuts down the noise.
- You can also configure alerts based on the configuration and contextual relationships of the impacted resources. For example, an alert with the following query:

```
Find (Host|DataStore) with classification='critical'
  that has Finding with numericSeverity > 7
```

- **Backup configuration** is captured for AWS S3, RDS, and DynamoDB data stores and databases. You can simply run the following query to find anything that has backup enabled (switch to `false` to find those with no backup):

```
Find DataStore with hasBackup=true
```

## 132.2 Improvements and Bug Fixes

- Improved typography and added app icon to the navigation bar.
- Improved new user onboarding UI/UX.
- Fixed an issue that prevents email address from correctly saving on a Person entity in the Asset Inventory app.
- Fixed a bug where mapper failed to map a trust relationship in an edge case.
- Several other UI fixes and adjustments.



2019-04-30

### 133.1 New Features

- **Alerts** app updated and released with the following capabilities:
  - New UX that combines alerts and findings management into one app
  - Updated UI for creating/editing alert rules
  - Ability to import alert rule packs. See available rule packs at: <https://github.com/JupiterOne/jupiterone-alert-rules/>
  - Receive daily email notifications of active/new alerts
- Updated **JupiterOne Client and CLI** to support managing custom questions.
  - Custom questions will show up in the Question/Query Library.
  - They can be access via keywords search in the Landing app.
  - They will also be mapped to compliance requirements in the Compliance app, if the question is configured with a corresponding mapping.
- Ability to **enable API Key access** for one or more user groups to allow the users to generate API keys used for the external client or CLI.
- Simplified and improved **full text search**:
  - You no longer have to wrap keywords in quotes to perform a full text search
  - Partial keywords search is supported – property value index is updated to tokenize on capital letters as well as common non-space non-alphanumeric characters
  - Cross-field matching in supported – search will return results that match keywords across any property of a particular entity. For example:

- \* searching administrator policy will match an entity with `_class: Policy` and `name: AdministratorAccess` in two different properties
- \* searching prod instance will match entities with `_type: aws_instance` and `tag.AccountName: jupiterone-prod-us` properties

- **J1QL shorthand comparison for property filters.** For example, you can type

```
Find DataStore with classification=('confidential' or 'restricted')
```

instead of

```
Find DataStore with classification='confidential' or classification='restricted'
```

- **Jamf** integration initial release. See details at <https://docs.jupiterone.io/en/latest/docs/integrations/jamf/jupiter-integration-jamf.html>
- **Tenable Cloud** integration initial release. See details at <https://docs.jupiterone.io/en/latest/docs/integrations/tenable-cloud/jupiter-integration-tenable-cloud.html>
- **OpenShift** integration initial release. See details at <https://docs.jupiterone.io/en/latest/docs/integrations/openshift/jupiter-integration-openshift.html>
- **AWS EC2 Auto Scaling** supported added to the AWS integration. Here are two example queries that will allow you to find instances that are / are not part of an auto-scaling group.

```
Find aws_instance that has aws_autoscaling_group
```

```
Find aws_instance that !has aws_autoscaling_group
```

## 133.2 Improvements and Bug Fixes

- Improved notification email design
- Updated Bitbucket integration due to Bitbucket API v2.0 change which removes the reference to `username`
- Non-admin users can capture their *review and acceptance* of security policies in the **Policy** app
- Added direct linking support to specific a policy/procedure document in the **Policy** app and fixed broken links referenced in the documents.
- Fixed a couple of bugs related to updating the entity properties as part of a relationship mapping in `jupiter-mapper`
- Properly handling relationship deletions in `jupiter-mapper`
- Updated error messages during onboarding to be more descriptive of the issue
- Several other UI/UX improvements and minor bug fixes
- Updated *API docs* and *SSO integration guide*.
- Added *guide* to describe how to use JupiterOne together with AWS GuardDuty and Inspector for proactive threat monitoring in AWS.



2019-05-14

### 134.1 New Features

- Added support for `UNIQUE` keyword in **J1QL** to return de-duplicated values, or value combinations, in the query return results.
- **Alerts** app updates:
  - Added **Sorting** for alerts and findings table; **Filtering** for findings.
  - Updated daily email format and included a count of resources added in past 24 hours as part of the “daily digest”.
  - Several other UI tweaks for the **Alerts** app.
  - Added **Evaluate Now** action button to run an alert rule on-demand.
  - *More exciting features and updates to come!*
- Added the ability to select/tweak your own vanity URL as part of account creation / onboarding.
- **HackerOne** integration initial release. See details at <https://support.jupiterone.io/hc/en-us/articles/360022902553-HackerOne>

### 134.2 Improvements and Bug Fixes

- Added check to prevent the default *Administrators* group from being accidentally deleted.
- Made `description` field optional when creating/updating a question in the library.
- Fixed an issue in the mapper when an assessor’s email (from an `Assessment` entity) is incorrectly mapped to a `Person`.

- Minor updates and fixes across Veracode, WhiteHat, Bitbucket and Github integrations.

### 134.3 Additional Notes

Please note that we are migrating our documentation site to a consolidated **docs + support + community** site at <https://support.jupiterone.io>

- Check out the articles in **Getting Started Steps**, **User Guides**, and **Developer Docs**: <https://support.jupiterone.io/hc/en-us>
- Submit and vote on **feature requests**: <https://support.jupiterone.io/hc/en-us/community/topics/360000688873-Feature-Requests>

2019-05-28

### 135.1 New Features

- Added **table of contents sidebar** with scroll spy on Landing, so that it is easy to jump to specific query results on the page.
- Policies and procedures related to a **Compliance** requirement now has direct web links to them that open up in the **Policies** app.
  - Also, details of each compliance requirement now display in full screen view.
- **Github** integration now supports suspicious PR analysis. See details at: <https://support.jupiterone.io/hc/en-us/articles/360022721934-Detect-Suspicious-Code-Commits>
- **Threat Stack** integration initial release - captures TS agents. See details at <https://support.jupiterone.io/hc/en-us/articles/360023924853-Threat-Stack>
- **KnowBe4** integration initial release - captures users, user groups, training campaigns, training modules and associated relationships among those entities. See details at <https://support.jupiterone.io/hc/en-us/articles/360023741834-KnowBe4>

### 135.2 Improvements and Bug Fixes

- UI improvements of the **Policies** app.
- Fixed an issue where integration logs were not displayed beyond 50 lines.
- Fixed an issue with sharing query links that has single quotes ( ' ) in the query.
- Minor updates to the data model classes and properties. Introduced Deployment, Module, Process, Requirement, Rule, Ruleset and Scanner entities.

- Fixed UI issue where sorting arrow in **Asset Inventory** table showing the wrong direction.
- **Mapper** rules now allow transformation to be specified to normalize values when producing mappings
- Fixed a pagination issue with Google users and added additional user properties.
- Added bounty properties to HackerOne integration.
- Correctly process Jamf users, groups, admin users and computer device users.
- Major improvement to Okta integration to support accounts with thousands of users or more.

2019-06-11

### 136.1 New Features

- Query language now supports **group by aggregates** based on entity attributes in addition to relationships. For example, try:

```
// Count number of pull requests by repo and then state  
  
Find PR as pr  
  return pr.repository, pr.state, count(pr)
```

or

```
// Count unencrypted data stores by type  
  
Find DataStore with encrypted=false as ds  
  return ds._type, count(ds) as value
```

- Updated **AWS GuardDuty** integration to better support threat analysis by parsing the threat intel list and attack source details. Try:

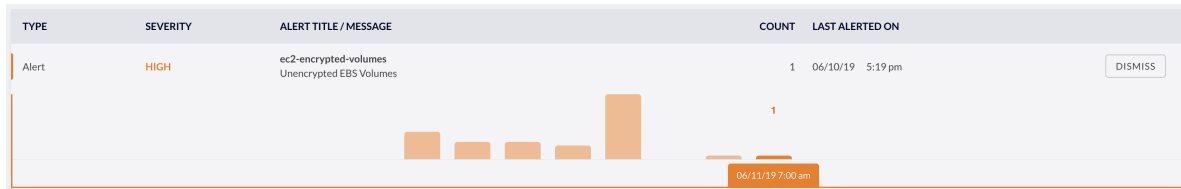
```
Find Finding as f return f.threatList, count(f)
```

or

```
Find Finding as f return f.attackSource, count(f)
```

- **Alerts Trend and History Data:**

Each alert now displays a **trend chart** to visualize changes over time. Selecting a data point on the trend chart will show you the results data captured by the alert at that point in time.



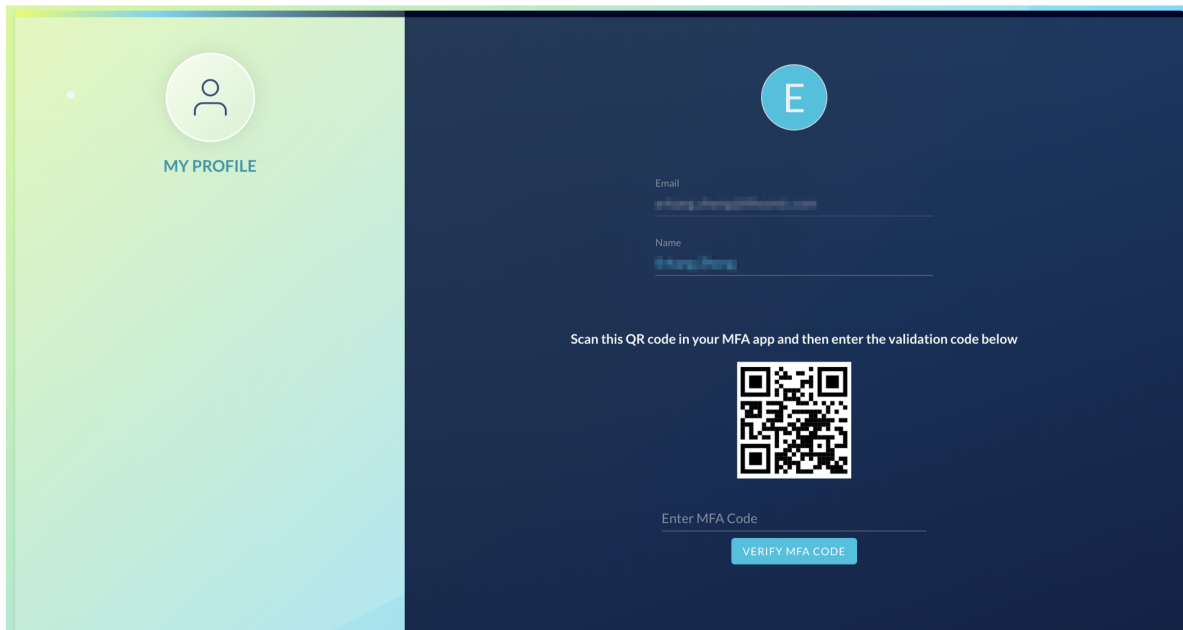
trend

- Added capabilities in **Compliance** app that
  - allows you to import/upload any compliance framework, including your own custom security controls and best practices;
  - performs automated **continuous gap analysis** based on matching queries;
  - shows improved indicators to show status of *policies*, *evidence collected*, and *gap analysis outcome*; and
  - maps policies, procedures and evidences to **controls** in addition to requirements.

FILTER BY COMPLIANCE STANDARD		HITRUST CSF Requirements		
CIS Controls	HIPAA	0 - Information Security Management Program <span>1 requirement</span>		
CSA CCM	HITRUST CSF	00.a Information Security Management Program An Information Security Management Program (ISMP) shall be defined in terms of the characteristics of the business and established and managed including monitoring, maintenance and improvement. <span>✓ Policies</span> <span>✓ Evidence</span> <span>✓ Compliant</span>		
PCI DSS	NIST CSF	1 - Access Control <span>25 requirements</span>		
MyStd2 9.2	HITRUST CSF 9.2 my controls	2 - Human Resources Security <span>9 requirements</span>		
HITRUST CSF 9.2		3 - Risk Management <span>4 requirements</span>		
		03.a Risk Management Program Development Organizations shall develop and maintain a risk management program to manage risk to an acceptable level. <span>✓ Policies</span> <span>✓ Evidence</span> <span>✓ Compliant</span>		
		03.b Performing Risk Assessments Risk Assessments shall be performed to identify and quantify risks. <span>✓ Policies</span> <span>✓ Evidence</span> <span>✓ Compliant</span>		
		03.c Risk Mitigation Risks shall be mitigated to an acceptable level. <span>✓ Policies</span> <span>✓ Evidence</span> <span>✓ Compliant</span>		
		03.d Risk Evaluation Risks shall be continually evaluated and assessed. The risk management program includes the requirement that risk assessments be re-evaluated at least annually, or when there are significant changes in the environment. <span>✓ Policies</span> <span>✓ Evidence</span> <span>✓ Compliant</span>		
		4 - Security Policy <span>2 requirements</span>		

view

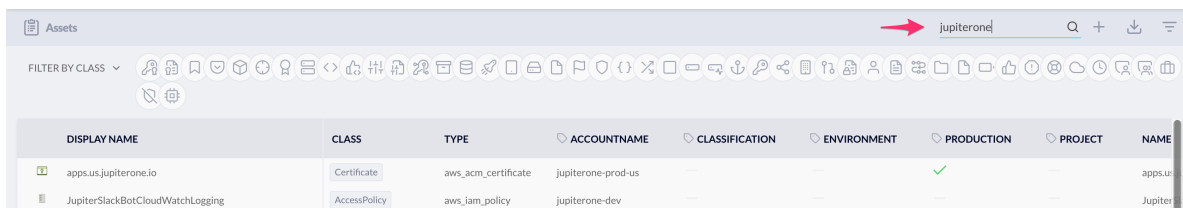
- Multi-factor authentication (MFA) can be enabled for JupiterOne user accounts (requires feature enablement per account)



user-

profile-mfa

- **Quick Search** in **Asset Inventory** allows you to type in simple full text search strings to quick filter results based on entity property values.



asset-

quick-search

- New packaged questions and queries added:
  - [aws] *Show me correlation of instances impacted by Inspector findings and GuardDuty findings*
  - [appdev] *Are code changes reviewed and approved?*
  - [appdev] *Are there code commits by an unknown developer in a PR?*
  - [grc] *Are security policies and procedures updated or reviewed within the past 12 months?*
  - [grc] *Is vendor SLA being monitored? Is there regular status reporting for my vendors?*
  - [general] *What vendor software applications are in use?*
  - [general] *What operating systems are in use?*
  - [general] *What applications are we developing?*

## 136.2 Improvements and Bug Fixes

- Significant query performance improvements
- Query result improvements:
  - common timestamp properties are parsed into human readable ISO datetime strings

- URL / web link properties are parsed and hyperlinked
  - IP address properties are linked out to geolocation details
- Bitbucket integration now allows you to selectively enable/disable ingestion of PR data for analysis. See details in [this article](#).
- Various small updates across all integrations.
- Many updates to package questions/queries to support compliance gap analysis.



2019-06-25

### 137.1 New Features

- Compliance app updates:
  - You can edit **policy/procedure-to-requirement/control mappings** directly in the webapp UI
  - You can add **links to external compliance evidence** to each requirement/control and optionally provide notes on the external evidence.
  - Improved compliance gap analysis logic - added warning/attention status in addition to compliant/fulfilled, gap, and indeterminate.
  - All mappings now work for controls of a standard framework in addition to requirements.
- Added an option to **Hide unrelated node** to filter out nodes in the graph that are not directly connected to the selected node.
- **Snyk** integration initial release - captures open source dependency vulnerability findings identified by Snyk scans and map them to code repos, CVEs, and CWEs. See details at <https://support.jupiterone.io/hc/en-us/articles/360024788554-Snyk>

### 137.2 Early Access / Beta Features

- **Insights** app with customizable dashboards and metric charts. (For Enterprise tier subscriptions only)
  - Supports query driven charts in *Number*, *Pie/Donut*, *Line*, *Table*, or *Matrix* format.
  - Supports customizable *Team* (shared) and *Personal* dashboards. Layout of each dashboard is individually customizable per user.
- **Trends** for saved/packaged questions. (For Enterprise tier subscriptions only)

- When enabled, question result will present a chart showing historic data trends.
- The timeframe of the trend chart can be switched to *WEEK*, *MONTH*, *QUARTER* or *YEAR*.
- You can also save/add the trend chart to a dashboard in the Insights app via the **Add to Dashboard** button.



trend

- Support `CREATE_JIRA_TICKET` and `SEND_EMAIL` as an alert action. This must be configured via the advanced rule editor, in the `operations.actions` portion of an alert rule's JSON configuration.

For example:

```
{
  ...
  "operations": [
    {
      ...
      "actions": [
        ...
        {
          "type": "CREATE_JIRA_TICKET",
          "summary": "Summary text of the Jira issue",
          "project": "11024",
          "issueType": "Task",
          "integrationInstanceId": "88ce9ad3-a49d-4995-aa9f-56d996f88b34",
          "entityClass": "Vulnerability"
        },
        {
          "type": "SEND_EMAIL",
          "recipients": [
            "user@company.com"
          ]
        }
      ]
    }
  ]
}
```

Each action can be configured independently on a rule.

Notes on Jira issue creation:

- Requires a Jira integration to have been configured, since the action references its `integrationInstanceId`. This is the UUID in the URL by going to your Jira integration configuration.
- `project` specifies the Jira Project ID (*not* Project Key) - the `pid` number in this URL: <https://yourjira.domain/secure/project/EditProject!default.jspa?pid=11024>
- UI improvements are coming soon to make configuration the above easier.

## 137.3 Improvements and Bug Fixes

- More query performance improvements
- Added pagination support for query results in the web UI (for more than 250 items) and when browsing questions in the library
- UI improvements for compliance requirement details modal
- UI improvements for Assets Inventory app, including the data grid and quick search bar
- Fixed an issue with sorting in the Alerts > Vulnerability Findings view
- Updated documentation for common questions and queries. See details at <https://support.jupiterone.io/hc/en-us/articles/360024909073-Common-Questions-and-Queries-Catalog>
- “Clear All” button to clear query results also clears the query in search bar