NATO RFI Mission Modeling

William Heinbockel, Steven Noel, and James Curbo
The MITRE Corporation

The 18 May 2015 NATO Communications and Information Agency (NCIA) Request for Information (RFI) (CO-14068-MNCD2) [1] seeks a multi-nation cyber defence situational awareness (CDSA) capability. While the RFI was strong on cyber defence capabilities, the scenarios did not include enough mission orientation to give sufficient insight into the mission and operational dependency tracking and potential for Courses of Action[1]. MITRE, in conjunction with NCIA, worked to expand the RFI scenarios with additional operational context as well as an initial mission dependency model.

1. Process

The dependency model was generated using the Structured Cyber Resiliency Analysis Methodology (SCRAM) [2]. Illustrated in Figure 1, SCRAM defines the processes for performing varying levels of cyber resiliency analyses at different points in the lifecycle of a system, system-of-systems, or mission. It also details such resources as frameworks and models, value scales, and datasets that can be used to support such analyses.

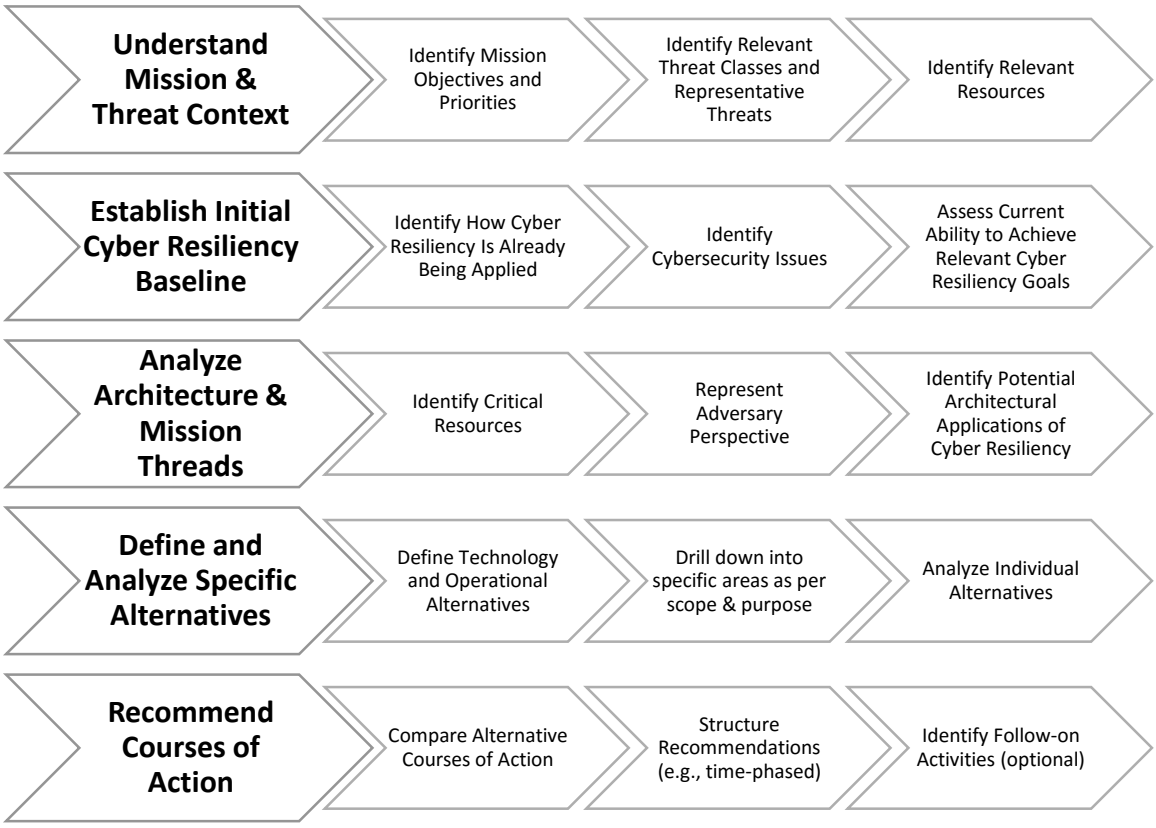

Figure 1: Overview of the SCRAM Process

[1] An operationally-complete scenario is required to flesh out the scope of CDSA Use cases UC06: View Asset Dependencies; UC08: Generate and select from Courses of Action; and UC19: Collect Dependencies.

In applying SCRAM to develop the model, we leverage a number of MITRE dependency analysis tools:

- *Crown Jewels Analysis* (CJA) [3] is a process and corresponding toolset for "identifying those cyber assets that are most critical to the accomplishment of an organization's mission"
- CyCS (Cyber Command System) [4] is MITRE's proof-of-concept cyber situational awareness tool for addressing "mission-assurance challenges for highly distributed enterprise systems of systems through vulnerability, threat, and consequence management"
- CyGraph [5] [6] is a tool for real-time cyber situational awareness that combines isolated data and events into an ongoing overall picture for decision support and situational awareness

SCRAM typically relies on dependency maps (Figure 2) to help understand what is most critical – beginning during system development and continuing through system deployment. The dependency map starts with identify missions and assign relative prioritization. From there, dependencies flow down through operational tasks and system function to cyber assets. These dependencies are expressed qualitatively in terms of impact on a parent node resulting from a failed or degraded child node, with provisions to minimize subjectivity. With a complete model, CJA can predict the impact of a cyber asset failure or degradation as the realization of each parent/child logical statement, tracing the potential impact upward to high-level mission tasks and objectives.

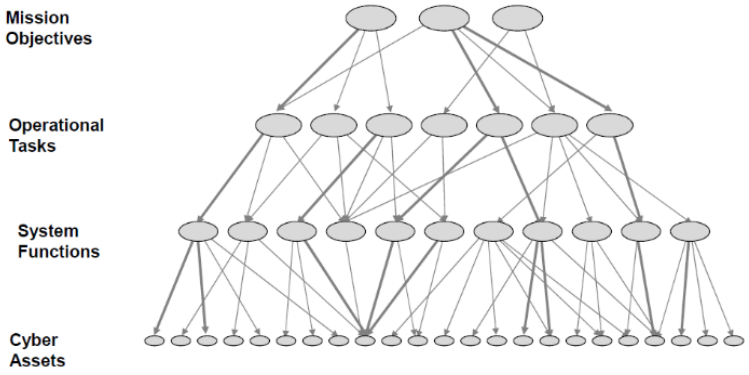

Figure 2: CJA Dependency Map

CJA provides a dependency map to associate missions, data flows, and cyber assets, along with the methodology to "roll up" cyber asset criticality based on higher-order associations, such as mission or operational priorities. The CJA model can also be inverted (Figure 3), allowing a CDSA to identify potential mission impacts of an incident to prioritize analyses.

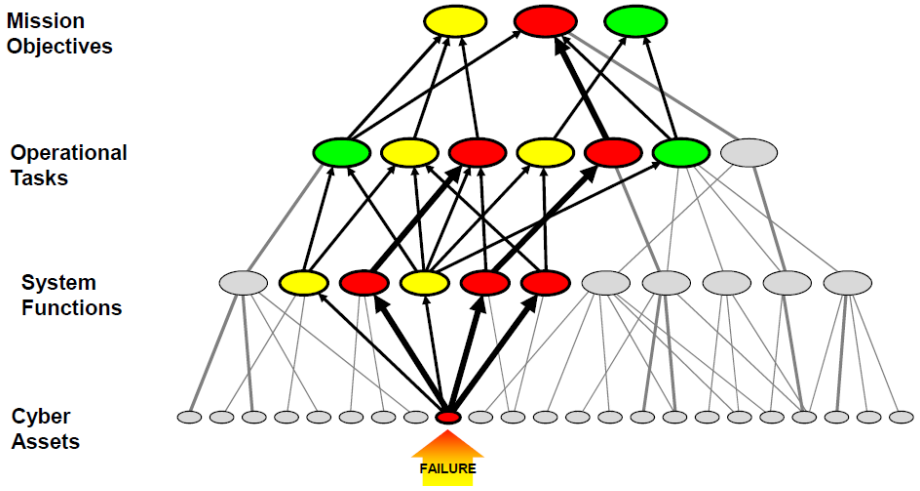

Figure 3: Assessing Failure Impact

2. Enhanced Scenario

Most of this effort was focused on enhancing the primary RFI scenario "Oranjeland APT."

As the NATO RFI is still in the beginning design and development stages, effort was focused on the first and third SCRAM steps: *Understanding Mission & Threat Context* and *Analyze Architecture & Mission Threads*. For the mission and threat context, the RFI documented the APT threat in the scenario, but gave limited insights into the supported operational mission of the Medevac.

An excerpt describing the original scenario and operational context is provided below [1]:

> A network administrator in the RATM Network Operations Centre (NOC) notices unusual network activity, which has not been detected by the antivirus, on a server at Regional Command North (RC-N), and suspects an Advanced Persistent Threat (APT).
>
> He contacts the NATO Cyber Security Operations (CSOps), who create an incident ticket. CSOps does a series of initial investigations using the CDSAS [cyber defence situational awareness system], and identifies this to be an Integrated Command and Control system (ICC) server. They collate all relevant information and options into a report and then they contact the Comprehensive Crisis and Operations Management Centre (CCOMC) Cybercell (CCC), recommending the course of action to disconnect the ICC server to disrupt the APT. CCC is aware of a planned mission in the area affected. Planned downtime is an important factor along with the negative impacts of the APT (e.g. exfiltration of data)…
>
> The [CDSAS] system can also then show the risk that if the server is turned off, CHAT would become unavailable as it is hosted on the same server. Loss of CHAT would appear with an impact of Medical Evacuations (Medevac) being hindered significantly as they are mainly done over CHAT. With a mission about to commence in the area, Medevac is likely to be an essential service. The Commander would need to judge whether the risk should be taken of running the mission without a CHAT capability, or whether it is essential to run the mission, and essential to have CHAT available, in which case the server will need to remain on. The system should allow him to compare the risks of keeping the server on, and whether that in itself will pose a threat to his mission.

2.1. Expanded Courses of Action

It is important that the CDSAS solution implementers understand the scope of the available courses of action. Sometimes the best course of action is obvious (e.g., use a redundant server), while other times the best suited course of action lies within a different command. A key capability of a CDSAS is to understand the mission dependencies and COP and help the commander make the best decisions, leveraging the maximum amount of information available.

MITRE recommends approaching the courses of action in order of resource intensity. Priority should be given to those courses of action that are fast, efficient, and require the least amount of coordination. As a guideline, we suggest that a CDSAS be able to provide courses of action within these three domains:

- Technical – redundant or spare cyber assets
- Service – redirect from other area or fallback on alternative functionality
- Operational – leverage CONOPS, call alternative commands for support

Within each domain, there are at least two categories of alternatives:

1. Technical
    - Replace: Can the cyber asset (e.g., system, network) be replaced with redundant components (e.g., spare servers, redundant network paths)?
    - Reconstitute: Can the cyber asset be reconstituted? For example, can the system replicate a server instance from a gold master virtual machine image, or dynamically reconfigure the network.
2. Service
    - Reposition: Are there identical services, potentially in neighboring geographic regions, that can be repositioned to cover the mission area?
    - Repurpose: Can the lost service functionality be (partially) replicated by repurposing other services? For example, email service may be used to provide some data transmission functionality similar to chat. Voice services (radio, VOIP) can be used as an alternative to digital communications (email, chat).
3. Operational
    - Reuse: Can the missing functionality be fulfilled by reusing a similar service offered by another entity or organization?
    - Retask: Can another entity or organization be retasked to complete or support the mission?

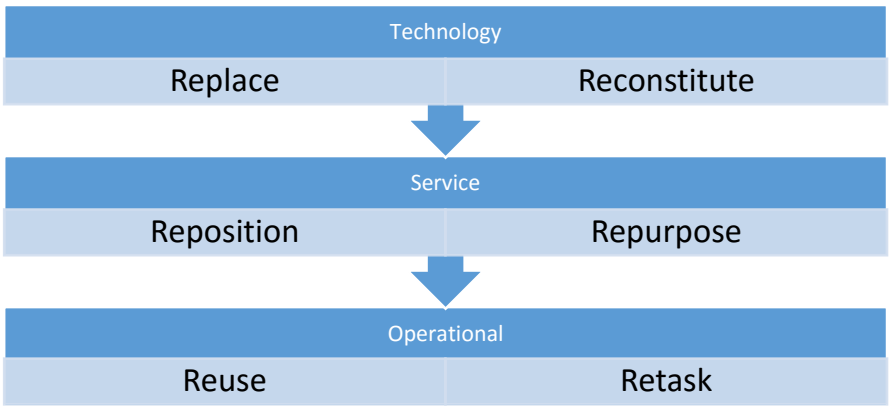

Figure 4: Courses of Action

To provide the CDSAS solutions with more introspective to battlefield decision support, MITRE proposes expanding the "Oranjeland APT" operational RFI scenario with more detailed alternatives. This expansion introduces three commands (AIR, LAND, SEA) with the Medevac being an AIR-led search and rescue mission:

The CDSAS system should evaluate the current medical evaluation mission dependencies and compare them against the available systems, services, and operational options available in within the COP and NSIS feeds.

First, the CDSAS system evaluates technical alternatives to identify any **replacement** (ICC) servers that be quickly brought online to provide the CHAT capability. Another course of action would be to determine whether the CHAT server can be **reconstituted** via backups or virtual machines.

Next, the CDSAS system might evaluate service oriented alternatives are usually more complex and take more resources to deploy. The easiest service course of action would be to **reposition** an existing chat service. For example, the NATO-led RATM coalition may be able to reposition the nearby Applestan CHAT service to cover the Medevac area. A second option is to evaluate whether the mission tasking can be altered by **repurposing** other, similar services to replicate the missing communications functionality provided by the CHAT service. For example, instead of using ICC CHAT, the Medevac may be able to use EMAIL or voice (radio, VOIP) to establish communication.

The final courses of action require operational-level coordination across commands. If the AIR-led Medevac is unable to perform the rescue mission, maybe the assets from another command (e.g., LAND or SEA) can assist. The CDSAS system should evaluate whether the SEA support forces have a CHAT service that can be **reuse** for AIR Medevac use. A last option is to **retask** the mission to another command. For example, maybe the LAND support forces have a nearby medic team that can be caravanned in, or the SEA support forces have a helicopter that can be used for a SEA-led Medevac.

2.2. Mission Model

We built a mission model that maps the dependencies from the high-level Rescue mission led by the NATO RATM, down to the system or asset level. Figure 5 is an overview of the model. This partial model focuses on expanding the Medevac scenario, and is incomplete (there are many nodes with missing dependencies, e.g., the top-level Medic, Rapid Deploy Medic, and SEA Medevac platforms).

Additionally, this overview does not include dependency criticality weighting or any type of dependency logic. Criticality is one way of denoting the impact on the parent if the dependency is lost or unavailable. To keep discussions focused, SCRAM uses a simple four level criticality model, which is also used across the United States Department of Defense [7] [8]. Ideally the model should express that either the primary or backup ICC Server must be available. These logical relations are ambiguous in the figure.

Table 1: SCRAM Criticality Levels

Criticality	Description
Level I	Total Mission Failure
Level II	Significant Degradation
Level III	Partial Capability Loss
Level IV	Negligible or No Loss

Figure 5: Medevac Rescue Model Overview

3. Data Requirements

In order to support the dependency complexities, geographic location, and other relationships, the CDSAS should minimally track the following properties for each component:

- Dependencies
- Strength of Dependency
- Location
- Area of Responsibility / Area of Impact
- Owning Unit

MITRE uses the GraphML XML format[2] to model the dependency graph. Each edge is a directed dependency from the source to the destination. Features such as the dependency criticality, descriptions, locations (latitude and longitude), and command are added through node and edge keys. Table 2 provides a list of GraphML node and edge keys used for the RFI model.

Table 2: NATO RFI Property Keys

Key Name	Edge or Node Key	Description
name	Node	The name of the node
description	Node	A description of the node
type	Node	The type of resource the node represents (Operational Mission, Platform, Objective, Task, System Function, or IT Asset)
latitude	Node	The node's geolocation latitude
longitude	Node	The node's geolocation longitude
radius	Node	The radius [in meters] of coverage or area of responsibility centered at [latitude, longitude]
location	Node	The location name where the node resource resides
command	Node	The owning command: LAND, SEA, or AIR
criticality	Edge	The dependency criticality on an ordinal scale from 0-100, where 100 represents a Level I critical dependency

To support dependency logic (AND and OR requirements), MITRE uses intermediate nodes labeled with the name `and` or `or`. So a dependency from `Resource A` on `Resource B` or `Resource C` would be represented as in Figure 6.

[2] http://graphml.graphdrawing.org/specification.html

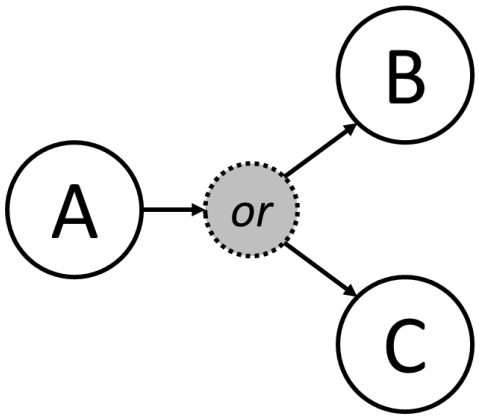

Figure 6: Intermediary AND/OR Nodes

An excerpt from the top-level of the NATO RFI Medevac model in GraphML would appear similar to the XML document on the following page.


```
<?xml version="1.0"?>
<graphml>
  <key attr.name="name" attr.value="string" id="name" for="node"/>
  <key attr.name="description" attr.value="string" id="description" for="node"/>
  <key attr.name="type" attr.value="string" id="type" for="node"/>
  <key attr.name="latitude" attr.value="double" id="latitude" for="node"/>
  <key attr.name="longitude" attr.value="double" id="longitude" for="node"/>
  <key attr.name="location" attr.value="string" id="location" for="node"/>
  <key attr.name="criticality" attr.value="double" id="criticality" for="edge"/>
  <graph id="RATM Rescue" edgedefault="directed">
    <node id="1">
      <data key="name">RATM Rescue</data>
      <data key="description">A NATO-led RATM search &amp;amp; rescue mission</data>
      <data key="type">Operational Mission</data>
    </node>
    <edge id="1-intermediate-dependency-4" source="1" target="intermediate-dependency-4">
      <data key="criticality">100</data>
    </edge>
    <node id="intermediate-dependency-4">
      <data key="name">or</data>
    </node>
    <edge id="intermediate-dependency-4-2" source="intermediate-dependency-4" target="2">
      <data key="criticality">100</data>
    </edge>
    <node id="2">
      <data key="name">Medevac [AIR]</data>
      <data key="description"/>
      <data key="type">Mission</data>
      <data key="latitude">50.875044</data>
      <data key="longitude">4.422221</data>
      <data key="location">NATO HQ</data>
    </node>
    <edge id="intermediate-dependency-4-3" source="intermediate-dependency-4" target="3">
      <data key="criticality">100</data>
    </edge>
    <node id="3">
      <data key="name">Medic [LAND]</data>
      <data key="description"/>
      <data key="type">Mission</data>
      <data key="latitude">50.939708</data>
      <data key="longitude">5.986826</data>
      <data key="location">NATO Joint Force HQ</data>
    </node>
    <edge id="intermediate-dependency-4-4" source="intermediate-dependency-4" target="4">
      <data key="criticality">100</data>
    </edge>
    <node id="4">
      <data key="name">Medevac [SEA]</data>
      <data key="description"/>
      <data key="type">Mission</data>
      <data key="latitude">35.50092</data>
      <data key="longitude">24.13147</data>
      <data key="location">NATO Maritime</data>
    </node>
    <edge id="intermediate-dependency-4-5" source="intermediate-dependency-4" target="5">
      <data key="criticality">100</data>
    </edge>
    <node id="5">
      <data key="name">Rapid Deploy Medic [LAND]</data>
      <data key="description"/>
      <data key="type">Mission</data>
      <data key="latitude">45.642871</data>
      <data key="longitude">8.860844</data>
      <data key="location">NATO Rapid Deployable Italian Corps</data>
    </node>
  </graph>
</graphml>
```


Figure 7 is a nested-table rendering of a portion of this GraphML document, i.e., the seven <key> elements and the first <node> and <edge> sub-elements of the <graph> element.

graphml

key (7)

	attr.name	attr.value	id	for
1	name	string	name	node
2	description	string	description	node
3	type	string	type	node
4	latitude	double	latitude	node
5	longitude	double	longitude	node
6	location	string	location	node
7	criticality	double	criticality	edge

graph

id	RATM Rescue
edgedefault	directed

node

id	1

data (3)

	key	Text
1	name	RATM Rescue
2	description	A NATO-led RATM search & rescue mission
3	type	Operational Mission

edge

id	1-intermediate-dependency-4
source	1
target	intermediate-dependency-4

data

key	criticality
Text	100

Figure 7: Partial GraphML for Top-Level of NATO RFI Medevac Model

4. Mission Dependency Analysis

This section describes analytic exploration of patterns of interdependent relationships within mission models. Given the large scale and high complexity typically expected for real missions, this is a crucially important capability.

For this analysis we apply *CyGraph* [5] [6], a MITRE a tool for capturing, analyzing, and visualizing knowledge about complex relationships in cyber environments. In CyGraph, a mission model (or any other set of graph relationships) is stored in a graph database, and the analyst formulates queries that match patterns in the stored graph.

CyGraph provides an easy and flexible way to capture, analyze, and visualize graph models. CyGraph query capabilities allow ad hoc exploration of graph models, for focusing on graph patterns of interest.

4.1. General Analytic Capabilities

In initial discussions on general requirements for mission dependency modeling, NATO provided MITRE with a spreadsheet containing a mock (fictional) set of mission dependencies. Figure 8 shows some of the rows of this spreadsheet.

	A	B	C	D	E	F
1	id	from_asset	to_asset	data_source	data_source_connector	source_creation_datetime
2	Dep_000	Asset_000	Asset_001	LIFO CMDB	LIFO CMDB DSC	2015-06-07 19:54:10+00:00
3	Dep_001	Asset_001	Host_00122	LIFO CMDB	LIFO CMDB DSC	2015-06-07 19:54:10+00:00
4	Dep_002	Asset_001	Host_20460	LIFO CMDB	LIFO CMDB DSC	2015-06-07 19:54:10+00:00
5	Dep_003	Asset_001	Host_18748	LIFO CMDB	LIFO CMDB DSC	2015-06-07 19:54:10+00:00

Figure 8: Spreadsheet Containing Mock Mission Dependencies

As highlighted in Figure 8, we import the 'from_asset' and 'to_asset' columns as dependencies (one dependency per row) into a CyGraph model. Figure 9 shows the corresponding CyGraph JSON input.

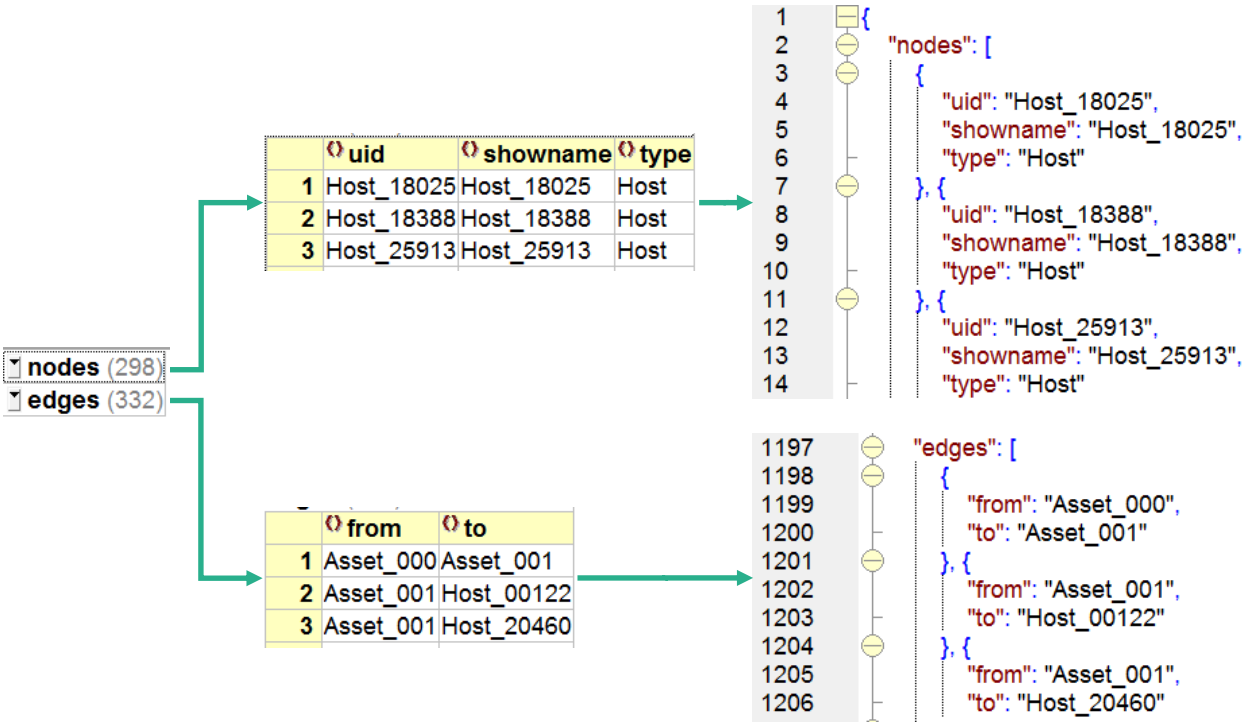

Figure 9: CyGraph Import JSON Format

Figure 10 shows one of the tools in the CyGraph suite for graph-based analytics and visualization. This tool allows an analyst to pose pattern-matching queries against a graph database (in this case, populated with the mock mission dependencies). It then renders the query result (matched sub-graph) through interactive graph visualization. It also has functions for graph statistics, styling, and temporal evolution.

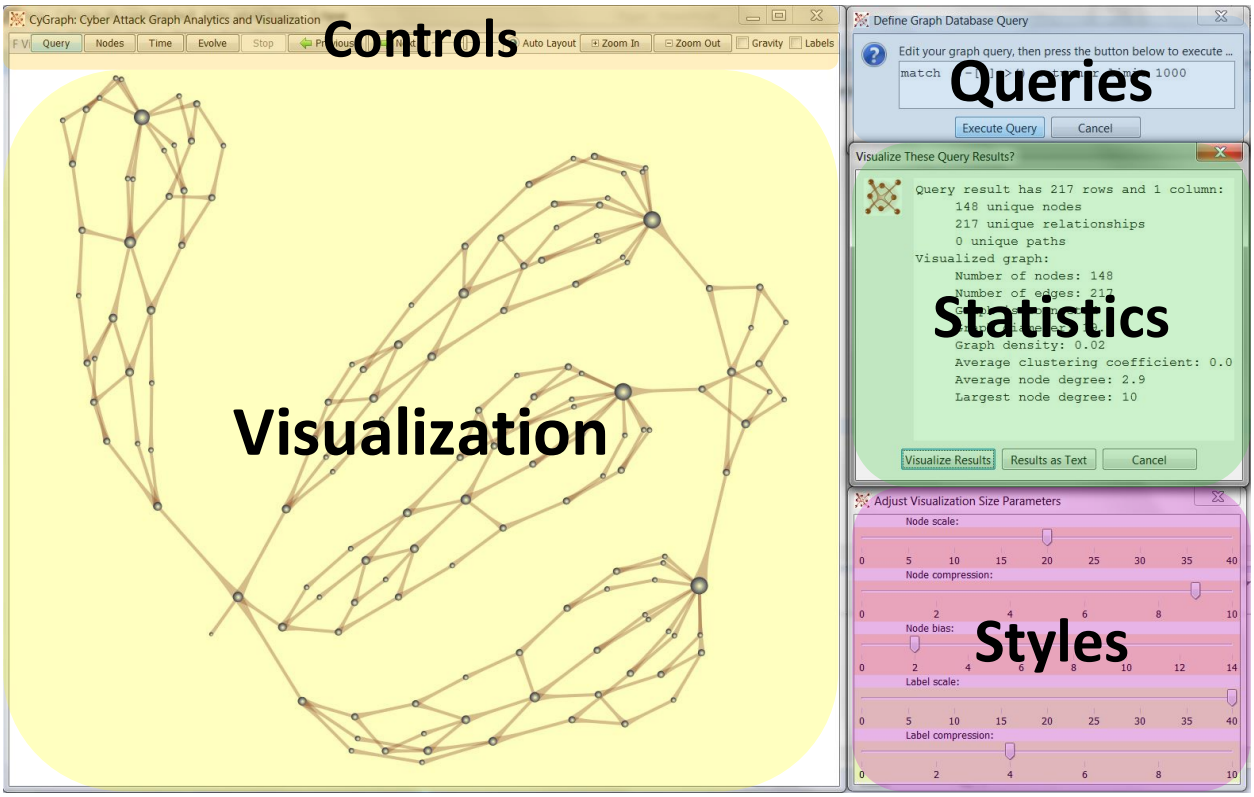

Figure 10: CyGraph Tool for Graph Analytics and Visualization

4.1.1. Analytic Capabilities

Figure 11 shows the full dependency graph imported into CyGraph. There are 298 nodes and 332 edges. In CyGraph, the query for this (expressed in the Neo4j [9] Cypher query language [10]) is

```
MATCH ()-[r]->() RETURN r
```

This query specifies that any edge relationship (denoted by r) between any pair of nodes is returned, i.e., all edges in the CyGraph database.

The graph is directed (directed from 'from_asset' to 'to_asset'). Directionality is portrayed with wide "arrow tails" and narrow "arrow heads." The graph has 2 separate (weakly) connected components, i.e., one large and one small.

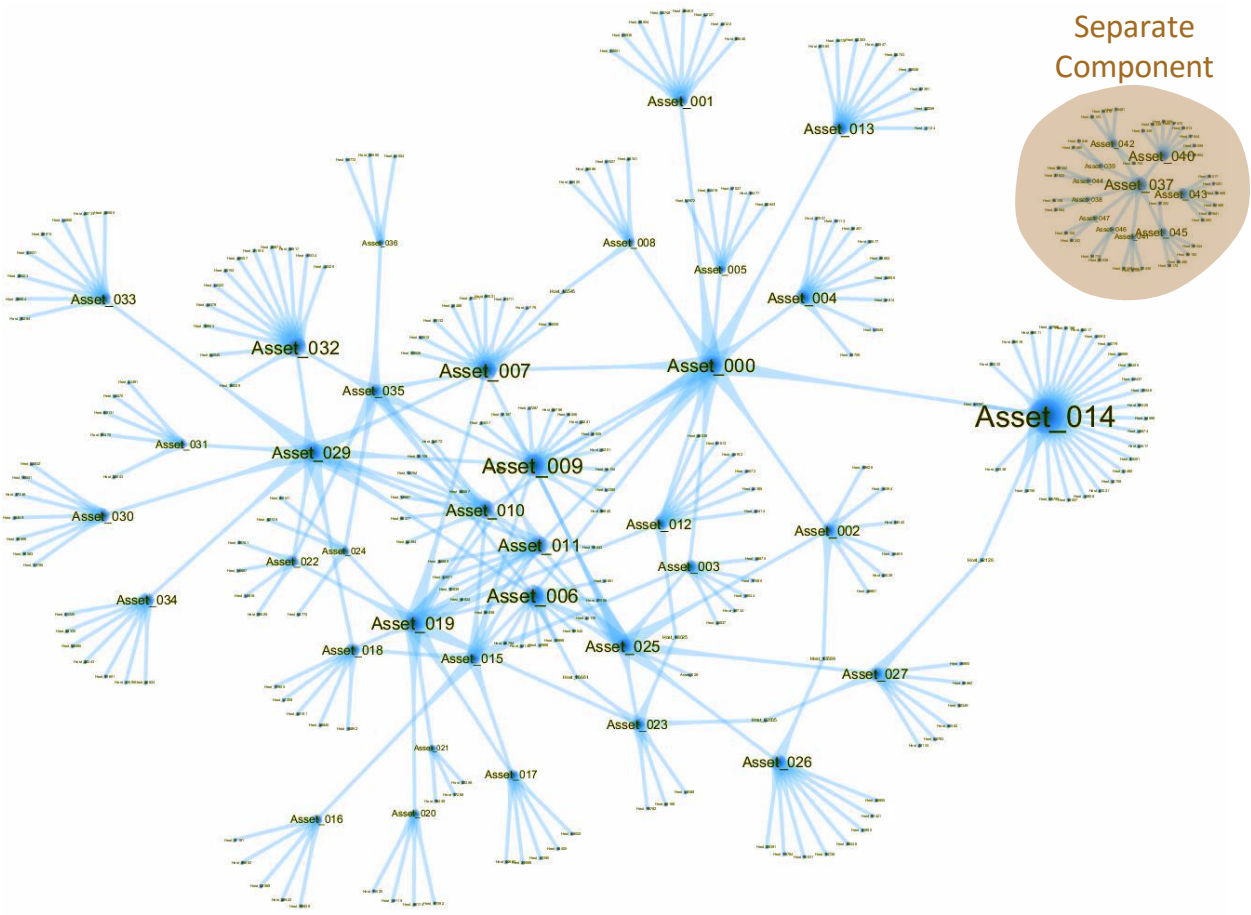

Figure 11: Full Dependency Graph for Mock Mission Model

4.1.2. Isolating Dependencies for a Particular Root Asset

In the full dependency graph (Figure 11) there are 7 nodes that are "root assets," i.e., they have no parent assets: *Asset_000*, *Asset_015*, *Asset_019*, *Asset_025*, *Asset_029*, *Asset_035*, and *Asset_037*. In this fictional model, these represent mission dependencies at the highest level of abstraction. Figure 12 shows the sub-graph of dependencies for root node *Asset_000*. In this way, we are able to focus the analysis on the lower-level assets and hosts that support a given high-level (root) asset.

The CyGraph (Cypher) query for Figure 12 is

```
MATCH p = ((root {CyGraphUid: 'Asset_000'})-[*]->()) RETURN p
```

This query matches the node *Asset_000* and all paths (arbitrarily deep) away from it (each edge indicating a "needs" dependency), i.e., everything upon which *Asset_000* depends.

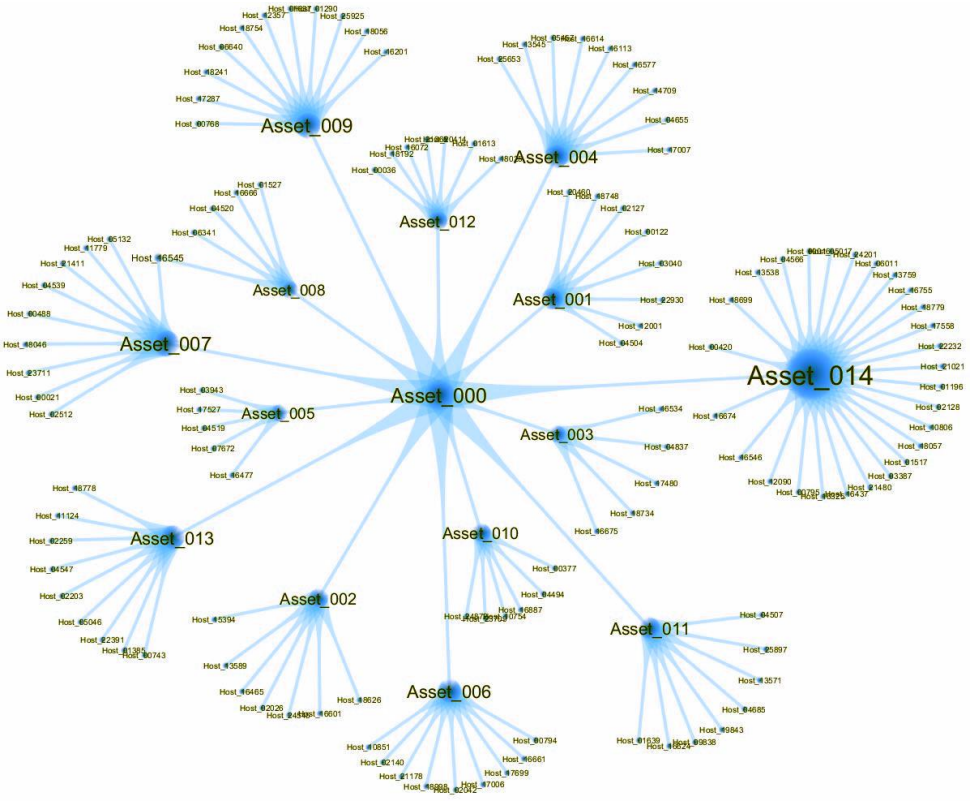

Figure 12: Dependencies for Root Node Asset_000

4.1.3. Isolating Dependencies for a Particular Root Asset

An interesting feature of Figure 12 is a shared dependency (more than one parent), i.e., a host (*Host_16545*) that is depended upon by two assets. This is highlighted in Figure 13.

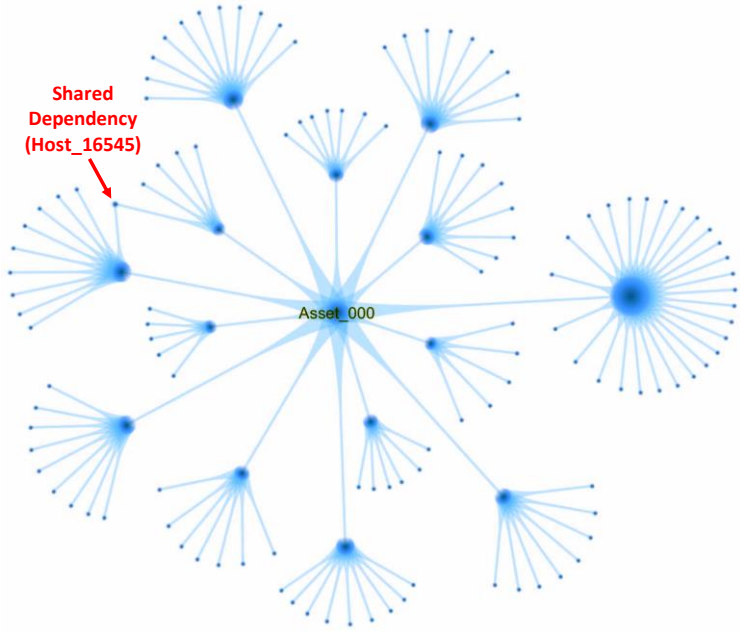

Figure 13: Shared Dependency for Asset_000 Sub-Graph

There might be use cases in which we need to isolate such shared dependencies within a mission dependency graph. This is done in Figure 14 for the shared dependency on *Host_16545* within the *Asset_000* sub-graph.

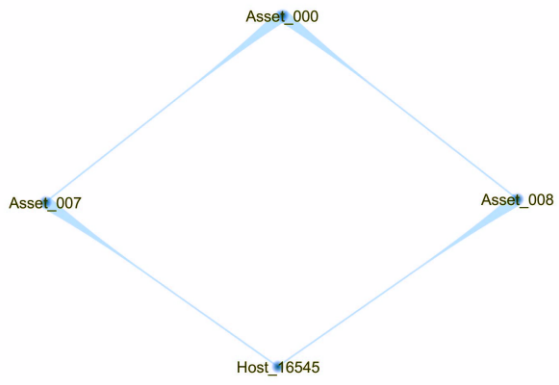

Figure 14: Shared Dependency on HOST_16545 within Asset_000 Sub-Graph

Here is the CyGraph query for Figure 14:

```
MATCH p = ((root {CyGraphUid: 'Asset_000'} )-[*]->
          (leaf {CyGraphUid: 'Host_16545'})) RETURN p
```

This query matches the *Asset_000* node, the *Host_16545* node, and all paths between them.

We might also need to understand all of the dependencies (transitively) upon a particular node. This is shown in Figure 15, for *Host_16545*. This shows that 4 other root nodes (*Asset_019*, *Asset_025*, *Asset_029*, and *Asset_35* also depend on *Host_16545*.

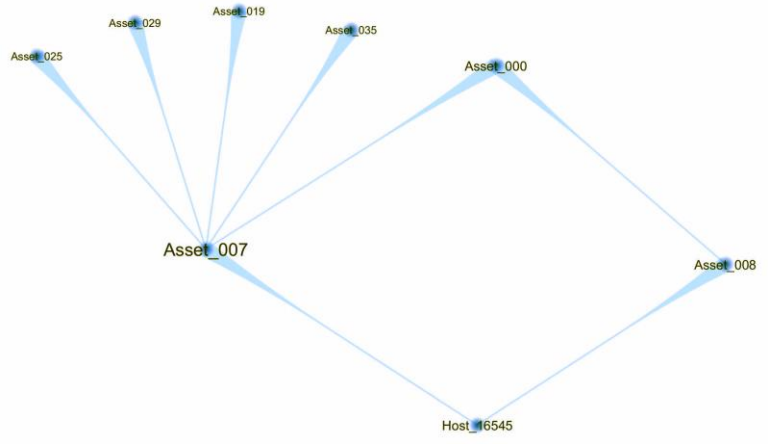

Figure 15: All Nodes that Depend on Host_16545

Here is the CyGraph query for Figure 15:

```
MATCH p = (()-[*]->(leaf {CyGraphUid: 'Host_16545'})) RETURN p
```

This query matches the *Host_16545* node and all paths leading to it, i.e., everything that depends on *Host_16545*.

4.1.4. Dependencies for Remaining Root Assets

Figure 16 shows the dependency sub-graphs for the remaining root (high-level asset) nodes, i.e., *Asset_015*, *Asset_019*, *Asset_025*, *Asset_029*, *Asset_035*, and *Asset_037*. Note that there is one shared-dependency host for the *Asset_019* sub-graph, and two shared-dependency hosts for the *Asset_025* sub-graph.

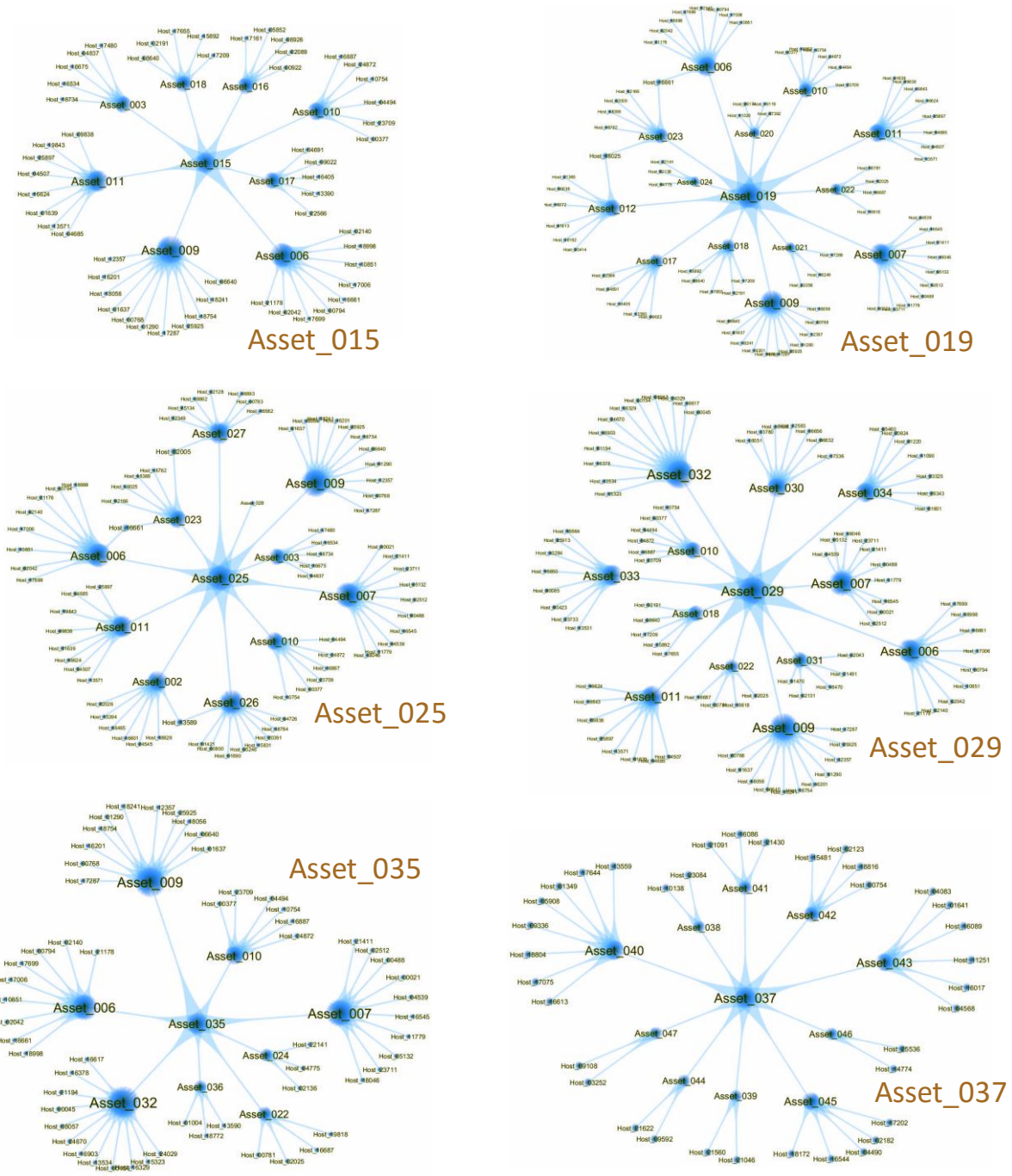

Figure 16: Dependency Sub-Graphs for Remaining Root Assets

Here are the CyGraph queries for Figure 16:

```
MATCH p = ((root {CyGraphUid: 'Asset_015'})-[*]->()) RETURN p
MATCH p = ((root {CyGraphUid: 'Asset_019'})-[*]->()) RETURN p
MATCH p = ((root {CyGraphUid: 'Asset_025'})-[*]->()) RETURN p
MATCH p = ((root {CyGraphUid: 'Asset_029'})-[*]->()) RETURN p
MATCH p = ((root {CyGraphUid: 'Asset_035'})-[*]->()) RETURN p
MATCH p = ((root {CyGraphUid: 'Asset_037'})-[*]->()) RETURN p
```

Each of these queries matches a given root node (high-level mission asset) and all paths leading from it, i.e., everything upon which that root node depends.

4.1.5. Multiple Shared Dependencies for a Root Asset

In Figure 16, the dependency graph for *Asset_025* has two occurrences of shared dependencies (hosts with two parents). Figure 17 isolates those shared dependencies, i.e., two hosts (*Host_16661* and *Host_13589*) that each have two parent assets that depend on them.

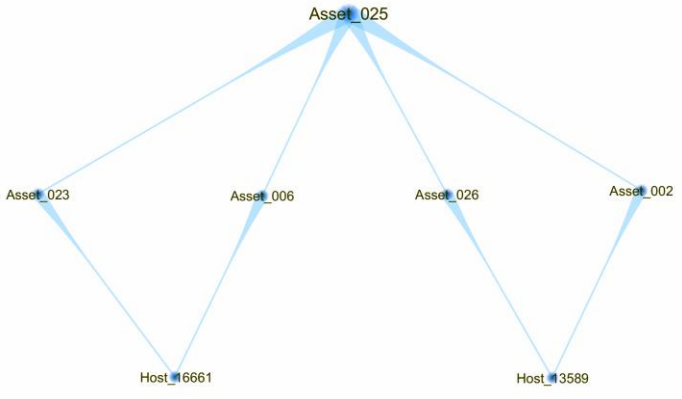

Figure 17: Two Shared Dependencies within Asset_025 Sub-Graph

Here is the CyGraph query for Figure 17:

```
MATCH p = ((root {CyGraphUid: 'Asset_025'})-[*]->(leaf))
WHERE leaf.CyGraphUid = 'Host_16661' OR
      leaf.CyGraphUid = 'Host_13589'
RETURN p
```

This query matches the *Asset_025* node and all paths leading to either *Host_16661* or *Host_13589*.

Figure 18 extends this to include all dependencies on *Host_16661* and *Host_13589*, with no constraint on root (top-level) asset. This shows all nodes that depend on *Host_16661* and *Host_13589* across the entire dependency graph.

Here is the CyGraph query for Figure 18:

```
MATCH p = (()-[*]->(leaf))
WHERE leaf.CyGraphUid = 'Host_16661' OR
      leaf.CyGraphUid = 'Host_13589'
RETURN p
```


This query matches all paths leading to either *Host_16661* or *Host_13589*.

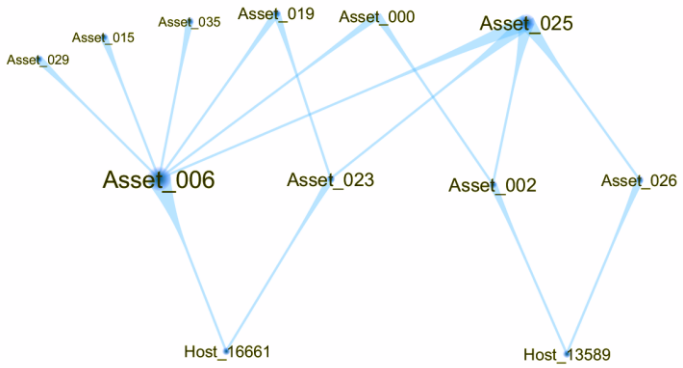

Figure 18: All Nodes that Depend on Host_16661 and Host_13589

4.2. Analyzing the RFI Mission Model

This section applies CyGraph for some analysis of the initial mission dependency model developed for NATO RFI scenarios. In particular, it is an instantiation of the mission model in **Error! Reference source not found.** expressed the JSON input format of Figure 9. Figure 19 shows the full model visualized in CyGraph. Here is the query:

```
MATCH ()-[r]->() RETURN r
```

This query simply matches and returns all relationships (edges) in the CyGraph database.

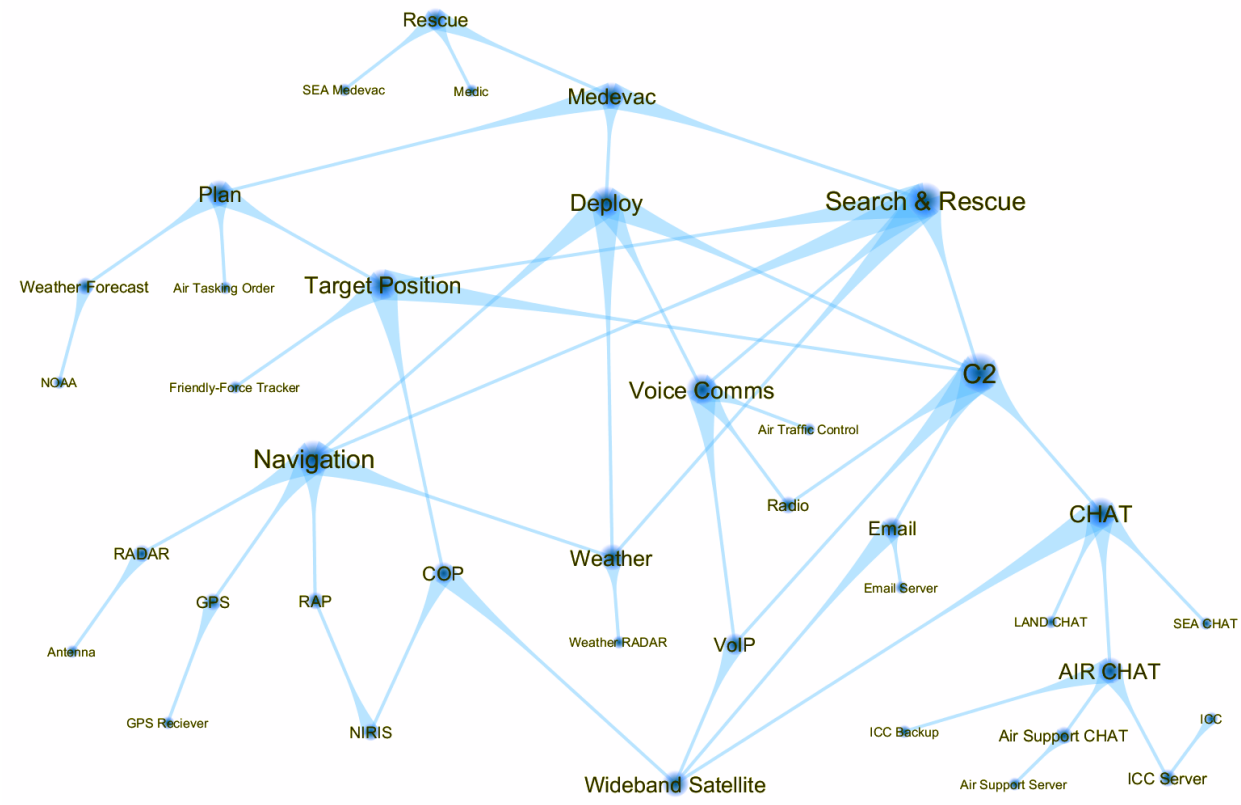

Figure 19: RFI Mission Model in CyGraph

One aspect of mission dependency modeling is that mission phases are not necessarily concurrent. For example, mission planning might be done well in advance of mission operations. Figure 20 illustrates this idea in the context of our RFI scenario mission model. For example, a commander might need to understand the components that support planning phase of the mission, independent of other dependencies.

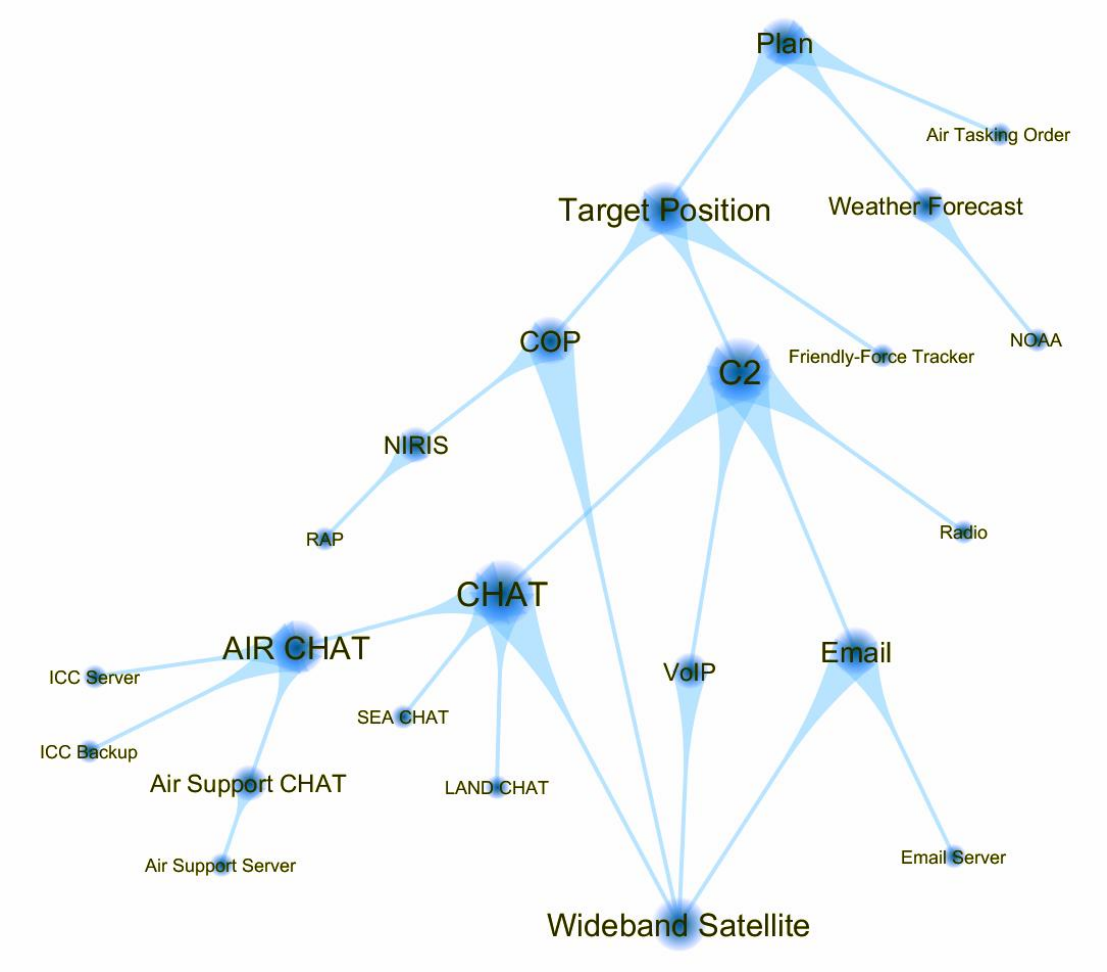

Figure 20: Components that Support Mission Planning

Here is the query for Figure 20:

```
MATCH p = ((root {CyGraphUid: 'Plan'})-[*]->()) RETURN p
```

This query matches the node *Plan* and all its dependency paths (arbitrarily deep), i.e., everything that supports the *Plan* node.

A core mission function is Command and Control (C2). This query shows all the dependencies for C2:

```
MATCH p = ((root {CyGraphUid: 'C2'})-[*]->()) RETURN p
```

Figure 21 is the query result. As an example of disjunctive (Boolean OR) relationships, consider the immediate dependencies of the *C2* node: *Radio*, *CHAT*, *VoIP*, and *Email*. These relationships capture the idea that these are 4 options for *C2*, although there are some shared dependencies for these at lower levels. On the other hand, communication via *Radio* is modeled as completely independent of the other *C2* communication options.

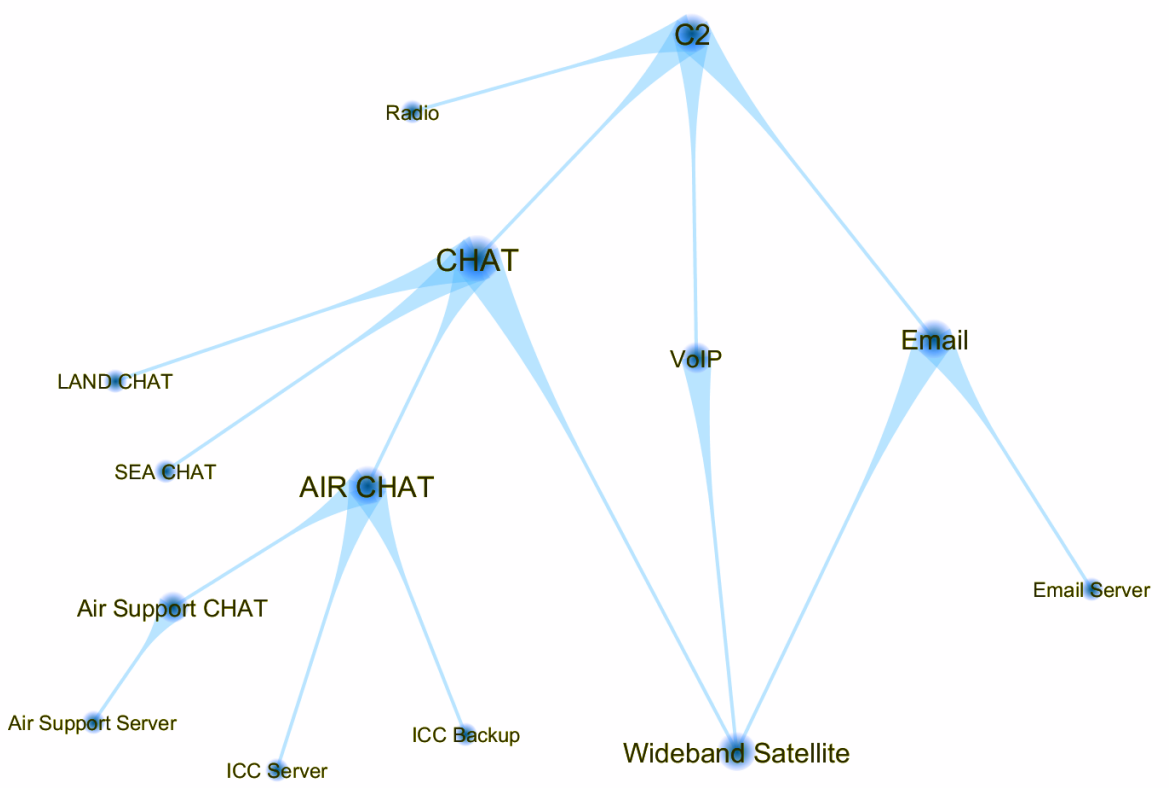

Figure 21: Components that Support Command and Control (C2)

A CyGraph query can also be oriented in the opposite direction, i.e., upward from lower-level nodes, up through higher-level nodes that depend on it. For example, in Figure 12 through Figure 14, the node *Wideband Satellite* is a key low-level mission capability. Here is the query that focuses on that:

```
MATCH p = ()-[*]->(leaf {CyGraphUid: 'Wideband Satellite'}))
RETURN p
```

In this query, the node constraint is on the "to" direction of the graph relationship. In particular, it matches any paths (transitive dependency relationships) that end (through any depth) at the *Wideband Satellite* node. Figure 22 shows the resulting matched subgraph.

For example, this could show a commander which mission elements are impacted by the loss of wideband satellite communication. In this case, all three high-level phases (*Plan*, *Deploy*, *Search & Rescue*) of the Medevac mission capability are impacted. On the other hand, according to this model the other high-level capabilities in Figure 19 (*Medic* and *SEA Medevac*) are not impacted by the loss of *Wideband Satellite*. This query result also explicitly determines that *Radio* (as an option for C2) is not impacted by the loss of *Wideband Satellite*.

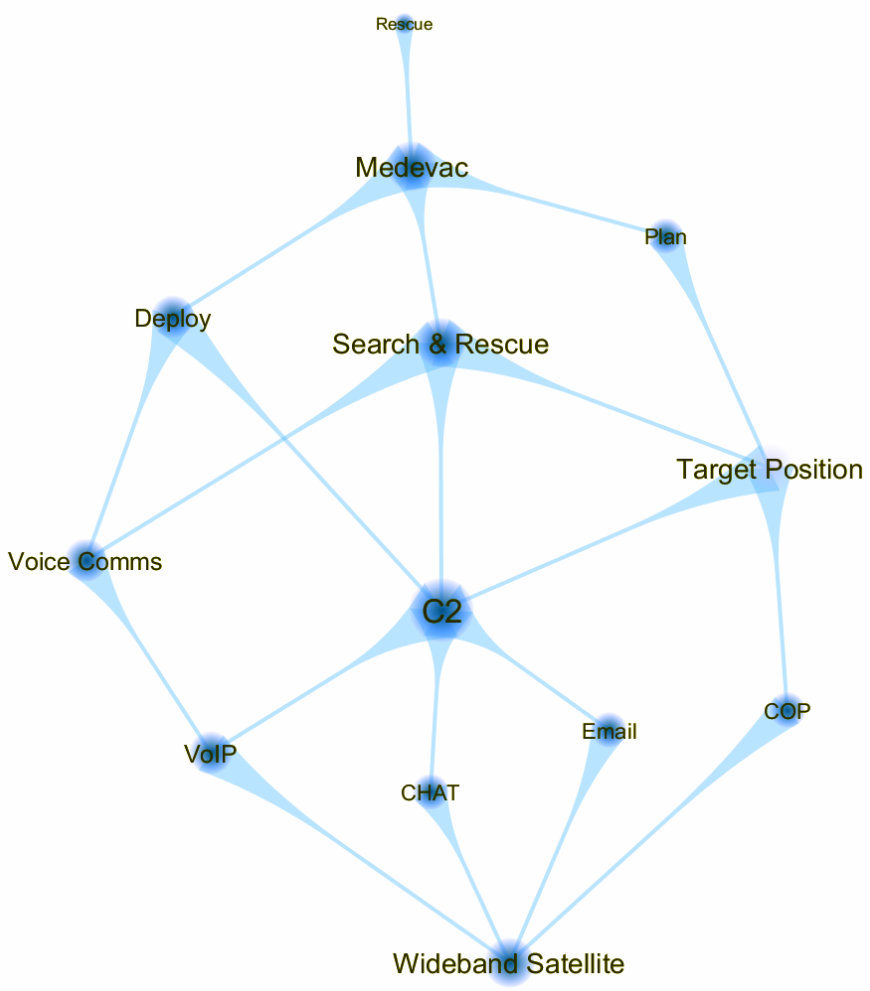

Figure 22: Mission Components that Depend on Wideband Satellite

4.3. Analyzing an Enhanced RFI Mission Model

This section enhances the CyGraph model for NATO RFI scenario mission dependencies with some useful semantic properties. In particular, we add nodes that represent disjunctive (Boolean OR) dependency relationships, as well as relationship (edge) weights that represent mission criticality. Figure 23 shows the upper five levels of the enhanced model. Here is the query:

```
MATCH p = ((root {CyGraphUid: 'RATM Rescue'})-[*1..4]->()) RETURN p
```

This query matches the model root node *RATM Rescue*, and all nodes that are within four child relationships from it. The immediate child of *RATM Rescue* is a Boolean OR node, indicating that the success of either of its children (*Medevac [SEA]*, *Medevac [AIR]*, *Rapid Deploy Medic [LAND]*, or *Medic [LAND]*) is sufficient for *RATM Rescue* to be successful. All other dependencies in the figure are conjunctive (Boolean AND), i.e., all children must succeed for their parent to succeed.

In this high-level overview, we clearly see that there are 4 nodes (*Obtain Target Position*, *Navigate the Aircraft*, *Establish Voice Comms*, and *Obtain Weather Information*) that are each required by two higher-level nodes (parents), thus having potentially greater mission impact if they are compromised.

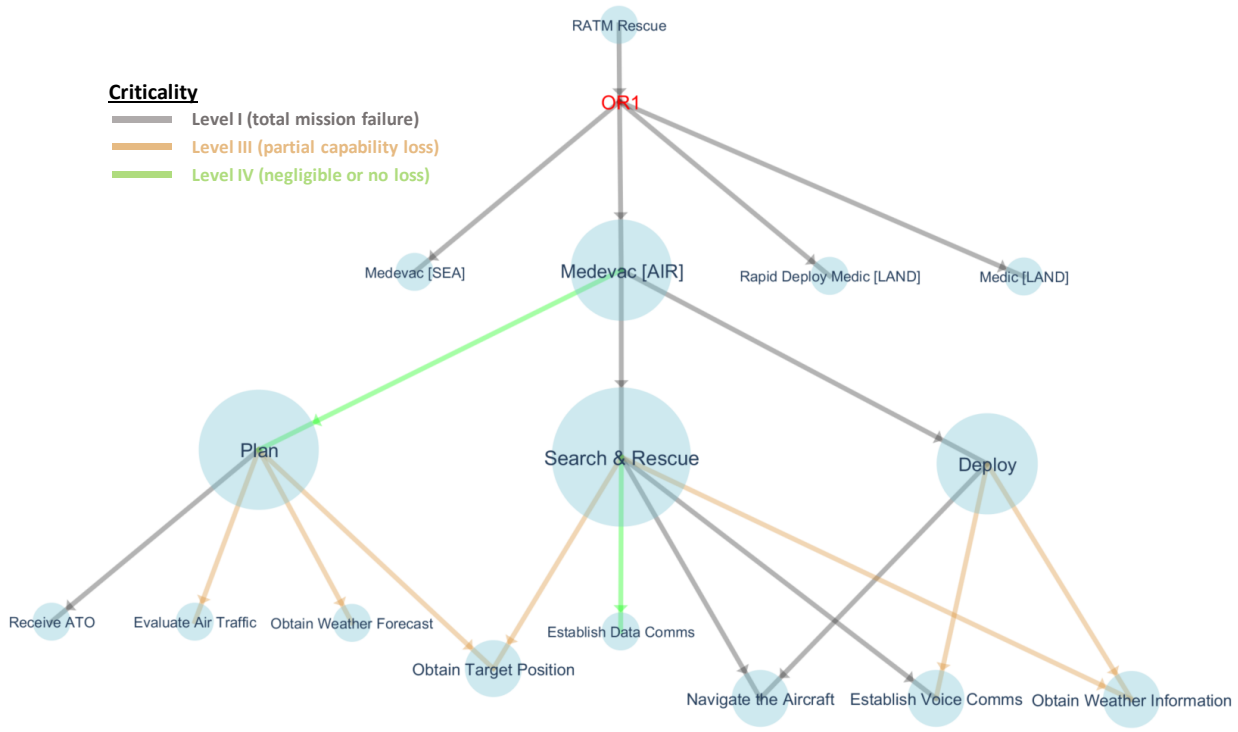

Figure 23: Top Five Levels of the Enhanced RFI Mission Model

Figure 23 also includes edge styling (colors) to indicate the level of mission criticality for a mission dependency relationship (graph edge), according the SCRAM Criticality levels defined in Table 1. In the figure, Level I criticality (total mission failure) is black, Level III criticality (partial capability loss) is orange, and Level IV criticality (negligible or no loss) is green. In this model, there are no dependency relationships with Level II criticality (significant degradation).

From Figure 23, there are three children of the *Medevac [AIR]* node: (1) *Plan*, (2) *Search & Rescue*, and (3) *Deploy*. Figure 24 shows the full sub-graph of dependencies for the *Plan* node. Here is the query:

```
MATCH p = ((root {CyGraphUid: 'Plan'})-[*]->()) RETURN p
```

This query begins on the *Plan* node and traverses all outgoing (dependency) edges. The query result (Figure 24) shows that the critical (Level I) dependency of *Plan* on *Receive ATO* (via *ATO Service*) has no redundancy, making it a single point of failure for mission planning of *Medevac [AIR]*.

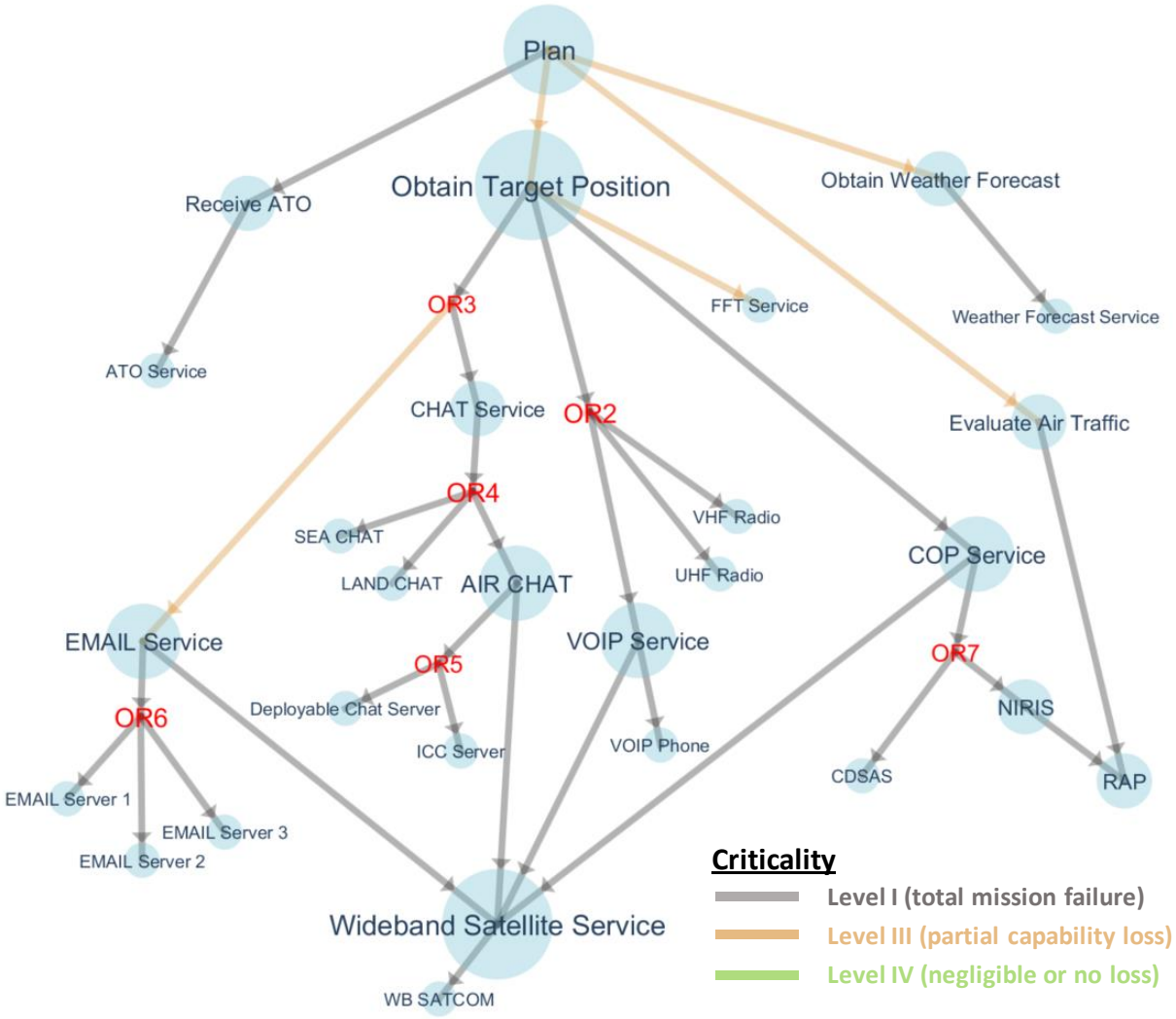

Figure 24: Dependencies for Medevac [Air] Mission Planning

Figure 25 shows the full sub-graph of dependencies for the *Search & Rescue* node. Here is the query:

```
MATCH p = ((root {CyGraphUid: 'Search & Rescue'})-[*]->()) RETURN p
```

This query follows the same pattern as the one for Figure 24, this time beginning at the *Search & Rescue* node. The query result in Figure 25 shows that there are two critical (Level I) dependencies for *Search & Rescue*, i.e., *Establish Voice Comms* and *Navigate the Aircraft*. Of those, *Establish Voice Comms* depends on redundant nodes and one with only partial loss of capability (Level III). The other (*Navigate the Aircraft*) does have a single point of failure of a critical (Level I) node, i.e., *Omni Radar Antenna*.

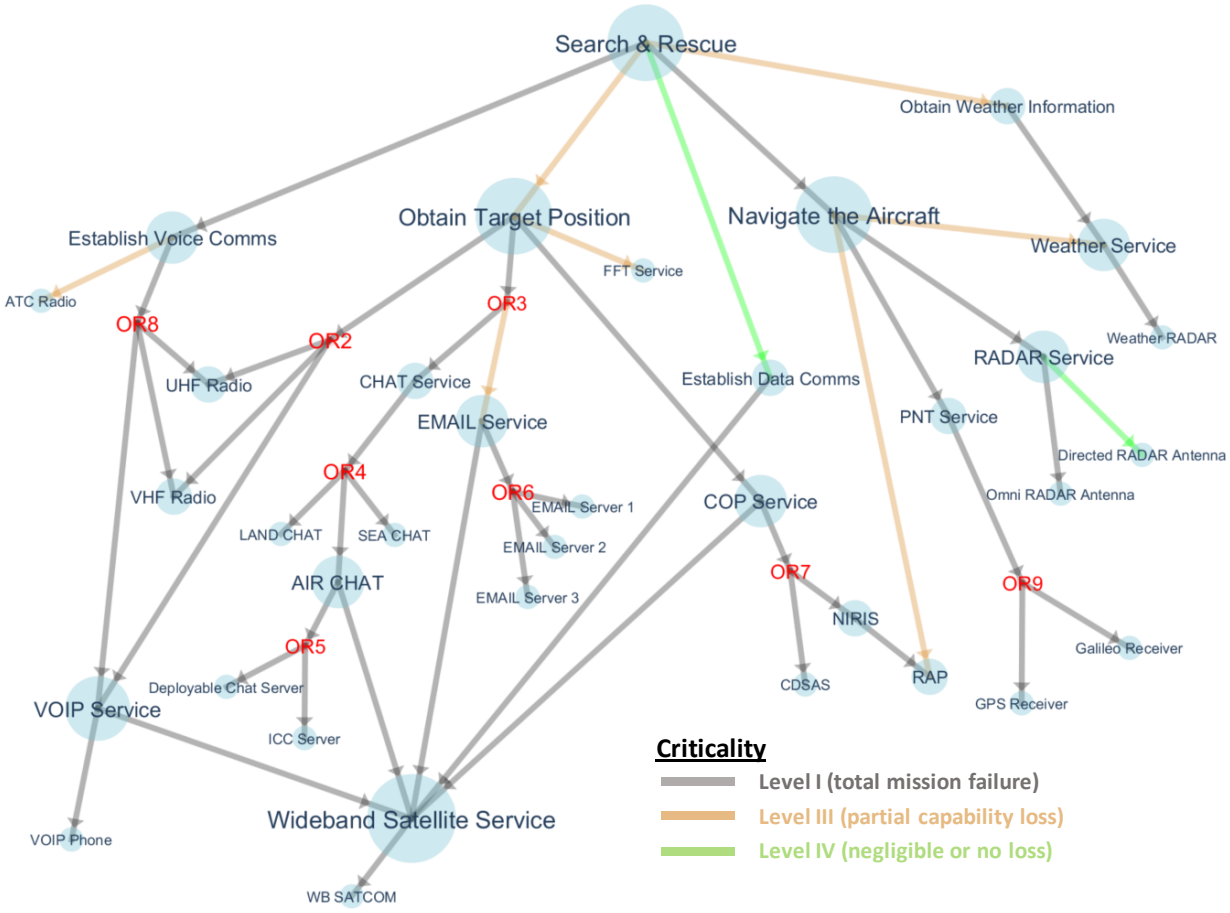

Figure 25: Dependencies for Medevac [Air] Search & Rescue

Figure 26 shows the full sub-graph of dependencies for the *Deploy* node. Here is the query, which follows the same pattern as for Figure 24 and Figure 25:

```
MATCH p = ((root {CyGraphUid: Deploy'})-[*]->()) RETURN p
```

This shows the same single point of failure of a critical node (*Omni Radar Antenna*), because of the dependency on *Navigate the Aircraft*.

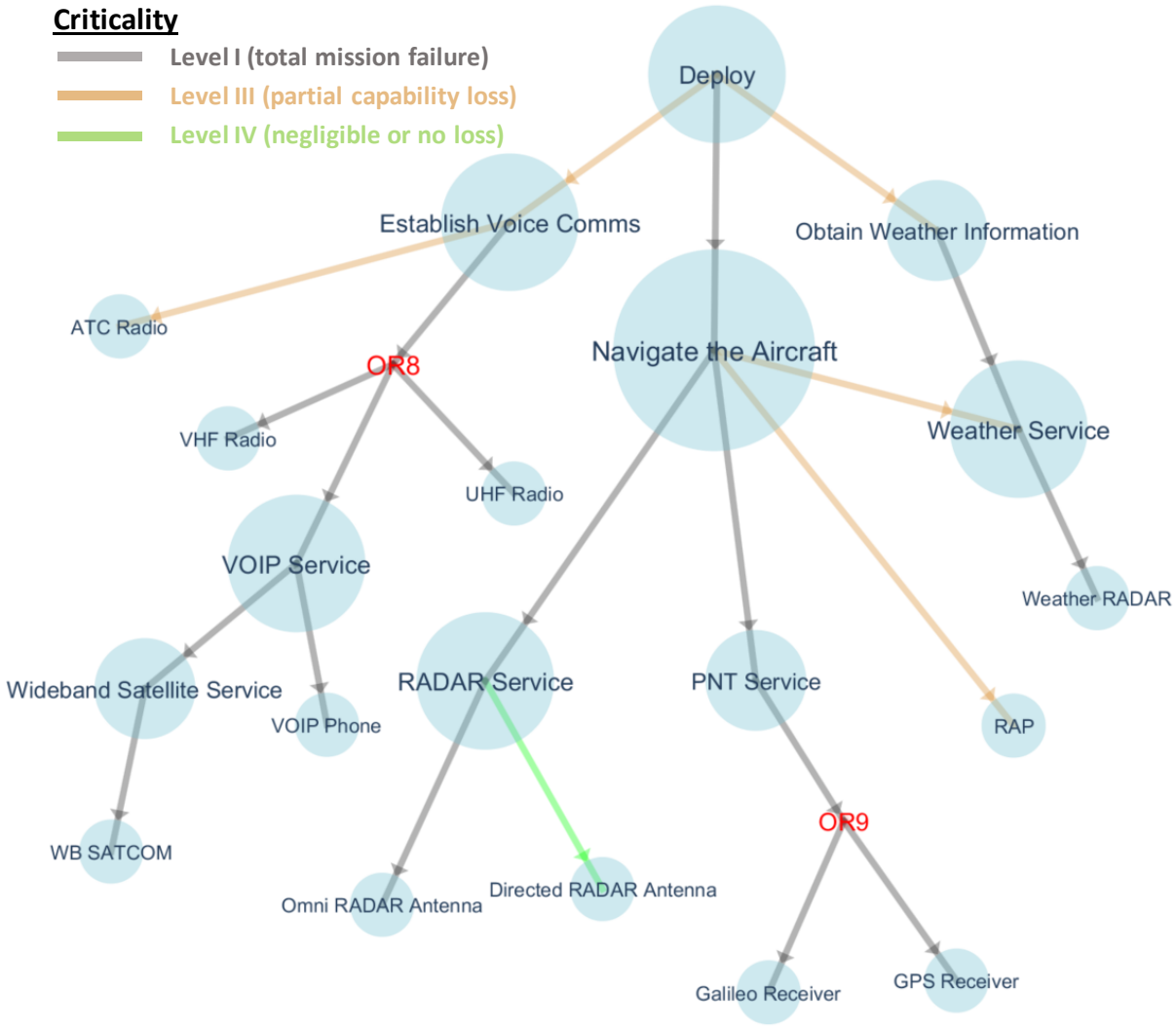

Figure 26: Dependencies for Medevac [Air] Deploy

In the previous section (Section 4.2), we apply a query that analyzes mission impact of the loss of wideband satellite. With our enhanced model, we can make a more precise assessment of mission impact. This query matches all dependencies (transitively) on the *WB SATCOM* node, i.e., everything that depends on it:

```
MATCH p = (()-[*]->(leaf {CyGraphUid: 'WB SATCOM'})) RETURN p
```

Figure 27 is the query result. This shows that the loss of *WB SATCOM* causes the loss of *Obtain Target Position* (at a minimum, via *COP Service*). This in turn causes partial loss of *Plan*, although that has negligible impact on *Medevac [AIR]*. It also causes at least partial loss of *Search & Rescue*, which also depends more critically on *Establish Voice Comms* (which potentially has redundancy protection from the loss of *WB SATCOM*), which in turn causes partial loss of *Deploy*. Overall, the *RATM Rescue* mission is only protected from the loss of *WB SATCOM* via its immediate child redundancies, i.e., *Medevac [SEA]*, *Rapid Deploy Medic [LAND]*, or *Medic [LAND]* as shown in Figure 23.

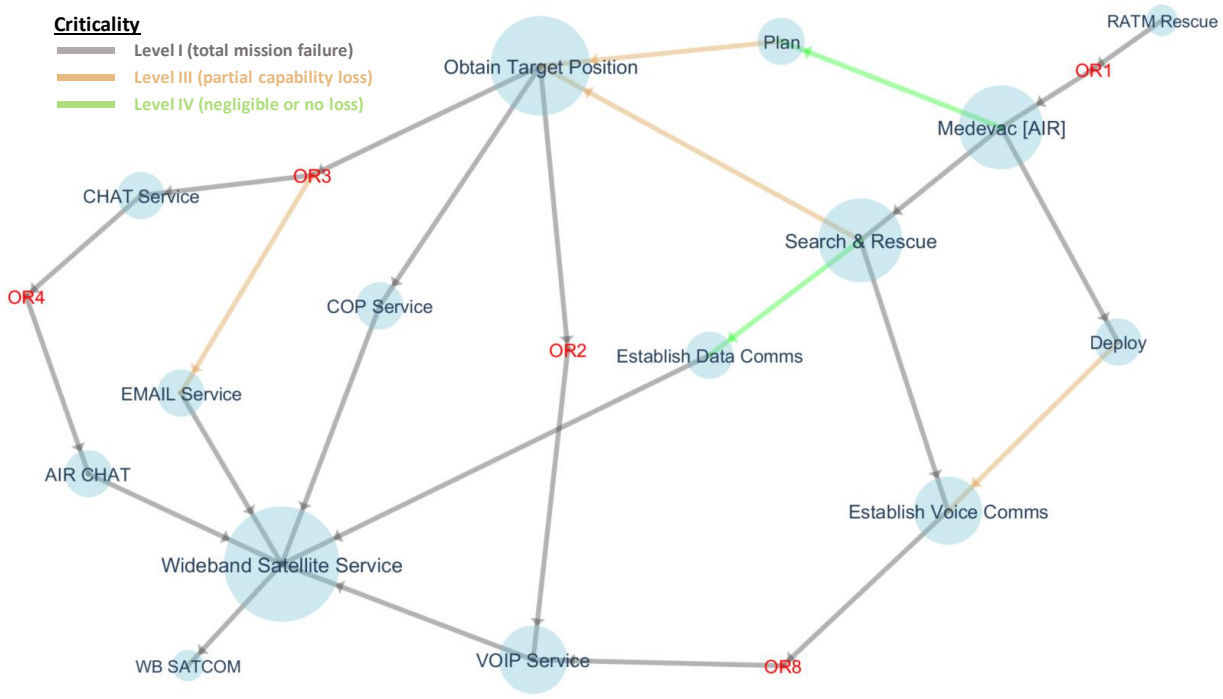

Figure 27: Mission Impact of the Loss of WB SATCOM

Figure 28 is the full set of dependencies for *Obtain Target Position*, which shows in more detail the impact from the loss of *WB SATCOM* and how particular additional redundancy could mitigate it. This uses the same query pattern as for Figure 24, Figure 25, and Figure 26, starting at *Obtain Target Position*.

```
MATCH p = (()-[*]->(leaf {CyGraphUid: 'Obtain Target Position'}))
RETURN p
```

We can ignore FFT Service, since it does not depend in any way on *WB SATCOM*. The redundancy via the *OR2* node provides alternatives to losing *WB SATCOM*. The *EMAIL Service* is lost because of losing *WB SATCOM* (while there are redundant email servers, there is still a required dependency on *WB SATCOM*), but the *CHAT Service* is still available because of redundancies at *OR4*.

There still remains the dependency on *COP Service*. While *OR7* provides one kind of redundancy, *COP Service* still depends directly on *WB SATCOM*. So to maintain the ability *Obtain Target Position* in the face of losing *WB SATCOM*, redundant alternatives for either *COP Service* itself or its dependence on *Wideband Satellite Service* is needed.

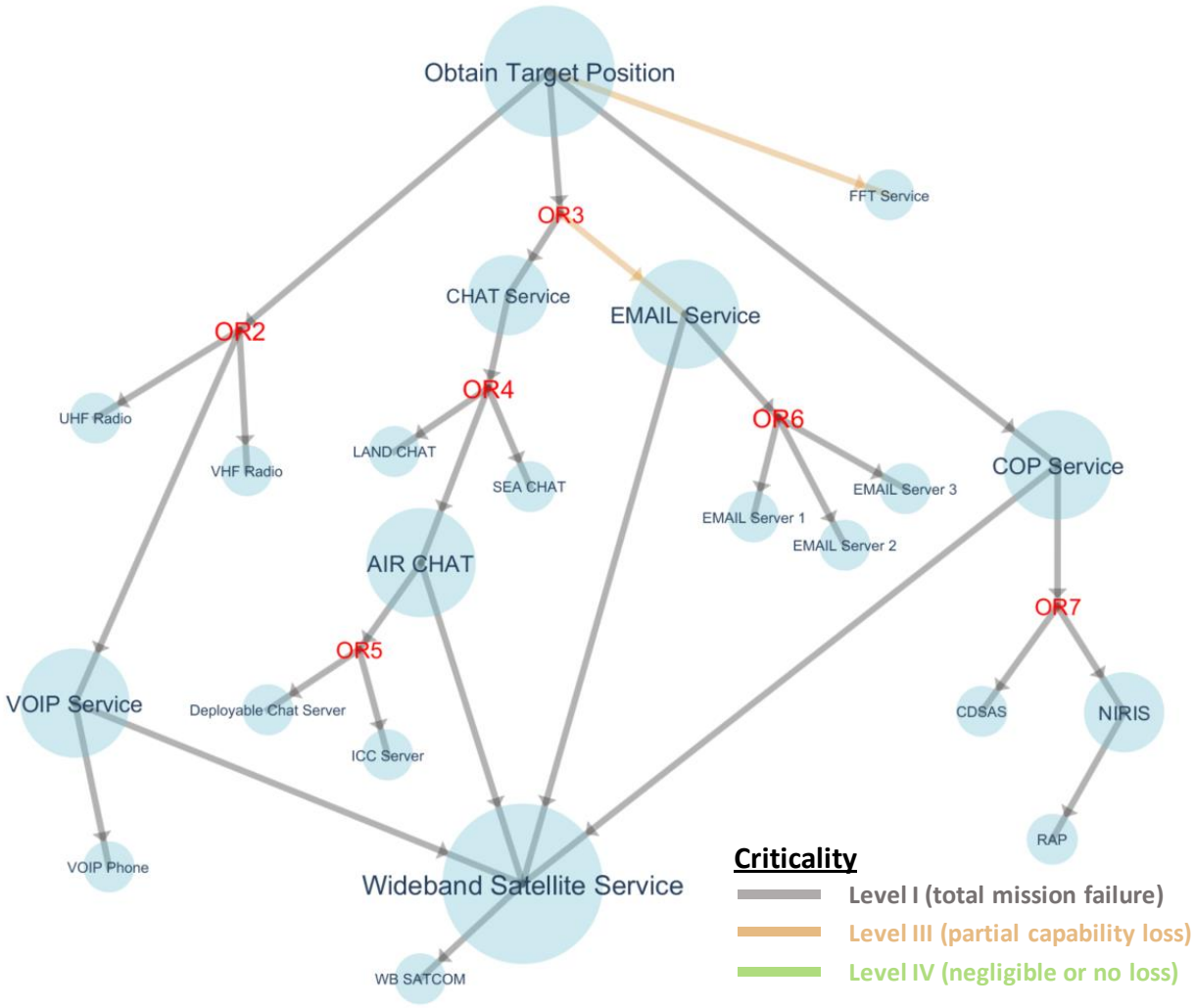

Figure 28: Dependencies for Medevac [Air] Obtain Target Position

The examples illustrate the additional situational awareness and decision support analytics provided by the enhanced mission model.

References

[1] T. Moye, R. Sawilla, R. Sullivan and P. Lagadec, "Cyber Defence Situational Awareness Demonstration/Request for Information (RFI) from Industry and Government (CO-14068-MNCD2)," NCI Agency Acquisition, 2015.

[2] D. Bodeau and R. Graubart, "Structured Cyber Resiliency Analysis Methodology (SCRAM)," 2016. [Online]. Available: https://www.mitre.org/sites/default/files/publications/pr-16-0777-structured-cyber-resiliency-analysis-methodology-overview.pdf.

[3] The MITRE Corporation, "Crown Jewels Analysis," September 2013. [Online]. Available: http://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-engineering-for-mission-assurance/crown-jewels-analysis.

[4] The MITRE Corporation, "Cyber Command System (CyCS)," [Online]. Available: http://www.mitre.org/research/technology-transfer/technology-licensing/cyber-command-system-cycs.

[5] S. Noel, E. Harley, K. H. Tam and G. Gyor, "Big-Data Architecture for Cyber Attack Graphs: Representing Security Relationships in NoSQL Graph Databases," in *IEEE Symposium on Technologies for Homeland Security*, Boston, Massachusetts, 2015.

[6] S. Noel, E. Harley, K. H. Tam, M. Limiero and M. Share, "CyGraph: Graph-Based Analytics and Visualization for Cybersecurity," in *Cognitive Computing: Theory and Applications (Volume 35 of Handbook of Statistics)*, Elsevier, 2016.

[7] Defense Acquisition University, "Defense Acquistions Guidebook (DAG)," [Online]. Available: https://dag.dau.mil/Pages/Default.aspx.

[8] Deputy Assistant Secretary of Defense for Systems Engineering (DASD(SE)) and Department of Defense Chief Information Officer (DoD CIO), "Trusted Systems and Networks (TSN) Analysis," June 2014. [Online]. Available: http://www.acq.osd.mil/se/docs/Trusted-Systems-and-Networks-TSN-Analysis.pdf.

[9] I. Robinson, J. Webber and E. Eifrem, Graph Databases (2nd edition), O'Reilly, 2015.

[10] O. Panzarino, Learning Cypher, Packt Publishing, 2014.

[11] Institute for Information Infrastructure Protection (I3P), "RiskMAP: Finding your corporate risks," February 2009. [Online]. Available: http://www.thei3p.org/docs/publications/riskmap-2009-02-09.pdf.

