# Analyzing Cyber Security Threats on Cyber-Physical Systems using Model-Based Systems Engineering

Aleksandr Kerzhner, Marc Pomerantz, Kymie Tan, Brian Campuzano, Kevin Dinkel,
Jeremy Pecharich, Viet Nguyen, Robert Steele, and Bryan Johnson
*Jet Propulsion Laboratory Califomia Institute of Technology*, *Pasadena, CA 91109*

## Nomenclature

*CPS* = Cyber-physical system
*MBSE* = Model-Based Systems Engineering
*SysML* = Object Management Group's Systems Modeling Language
*SME* = Subject Matter Expert
*CVSS* = Common Vulnerability Scoring System
*NVD* = National Vulnerability Database
SA = System Administrator

## Abstract

The spectre of cyber attacks on aerospace systems can can no longer be ignored given that many of the components and vulnerabilities that have been successfully exploited by the adversary on other infrastructures are the same as those deployed and used within the aerospace environment. An important consideration with respect to the mission/safety critical infrastructure supporting space operations is that an appropriate defensive response to an attack invariably involves the need for high precision and accuracy, because an incorrect response can trigger unacceptable losses involving lives and/or significant financial damage. A highly precise defensive response, considering the typical complexity of aerospace environments, requires a detailed and well-founded understanding of the underlying system where the goal of the defensive response is to preserve critical mission objectives in the presence of adversarial activity. In this paper, a structured approach for modeling aerospace systems is described. The approach includes physical elements, network topology, software applications, system functions, and usage scenarios. We leverage Model-Based Systems Engineering methodology by utilizing the Object Management Group's Systems Modeling Language to represent the system being analyzed and also utilize model transformations to change relevant aspects of the model into specialized analyses. A novel visualization approach is utilized to visualize the entire model as a three-dimensional graph, allowing easier interaction with subject matter experts. The model provides a unifying structure for analyzing the impact of a particular attack or a particular type of attack. Two different example analysis types are demonstrated in this paper: a graph-based propagation analysis based on edge labels, and a graph-based propagation analysis based on node labels.

## I. Introduction

The desire for systems and operations to be resilient during a cyber attack is gaining prominence in the cyber security industry. The increasing frequency, sophistication, and success of adversarial incursions has shown that traditional preventive approaches, e.g. perimeter defenses and firewalls, are extremely insufficient in reducing the impact of an attack. Consequently, the defense toolbox must include an approach that supports resilience. The defense must tolerate the presence of the adversary while minimizing their impact on critical mission operations; for example, mission critical operations could be protected by containing the threat to less critical systems. Resilience is particularly important during difficult attacks, such as the zero-day and supply chain incursions, where prevention by traditional means is nearly impossible.

At the core of the resilience paradigm is the ability for the defense to perform an impact analysis. By doing this, they can reason through the various consequences of adversarial activities so that an appropriate response can be composed and mission objectives can be preserved. In mission/safety critical environments such as those deployed to support space operations, the appropriate response invariably involves the need for high precision and accuracy, because an incorrect response can trigger unacceptable losses involving lives and/or significant financial damage.

Given the complexity of the systems that are typically deployed in mission/safety critical environments, impact analysis is often very difficult to accomplish for a number of reasons. Such systems tend to operate in "siloes" where the vertical focus is to obscure lateral dependencies on other siloes and underlying interconnectivities. The supporting infrastructure for an enterprise (its processes, IT components, communication patterns, and so forth) tend to evolve in pockets where the changes can be ineffectively documented and fragmented in time (e.g., upgrades that are phased in over time). These factors not only make it difficult to analyze the impact of adversarial activities, but they also make it problematic to quantify system resilience against attacks. The previously mentioned artifacts also inhibit effective security design and implementation decisions in context with other system attributes. One of the goals of the approach describes in this paper is to more easily visualize this information so enable enhanced stakeholder and subject matter expert engagement.

This paper describes a model-based approach aimed at enabling the defense analyze the various potential consequences of adversarial activities on mission objectives and compose an appropriate response. Although there have been a number of approaches for modeling the kinds of cyber physical systems typically deployed within space operations, these approaches either focus on a particular type of attack, utilize manual analysis to evaluate the system, or attempt to prove characteristics of oversimplified versions of the real system. A key deficiency with these past approaches is a lack of flexibility and adaptability to support evolving systems and emerging threats. Past approaches also lack quantifiable results that can be used to support decision making about the system.

This paper reports the results of an ongoing task that focuses on a structured approach for modeling the cyber physical system to enable impact assessment. The models encapsulate physical elements, network topology, software applications, system functions, and usage scenarios. We leverage Model-Based Systems Engineering (MBSE) methodology by utilizing the Object Management Group's Systems Modeling Language (SysML) to represent the system being analyzed and also by utilizing model transformations to change relevant aspects of the model into specialized analyses [SYSML].

This work can leverage other related MBSE initiatives to analyze additional aspects of the system, further supporting decision making about the system in context.

## II. Approach

The model provides a unifying structure for analyzing the impact of a particular attack or type of attack. By representing aspects of the system in an integrated model, connections between aspects can be explicitly captured. These connections facilitate analysis of the interdependence between elements in the system, which allows for coordination between elements to support system functions. The integrated analyses show the results of a cyber or physical attack. The unified model also allows changing the abstraction level at which the system is analyzed; the analysis of some cyber attacks may require high-level knowledge about the entire system while others may require detailed knowledge about specific elements (e.g. the version of particular software, the rules utilized by a particular firewall). Although some detailed information is not captured in the model, it provides an integrating framework between different domain-specific models and databases. The overall architecture of the modeling approach is shown in Figure 1. Information about the system is captured using No Magic's SysML editor MagicDraw (discussed durther in the next section). Model data is then exchanged between the "modeling tool", MagicDraw, and the visualization and analysis tools. The analyses are run on the appropriate subsets of the data, such as network connectivity information combined with identified vulnerabilities. The model approach is further described in Section III-A, the capture of cyber security information and analysis is described in Section III-B, and the visualization approach is described in section IV.

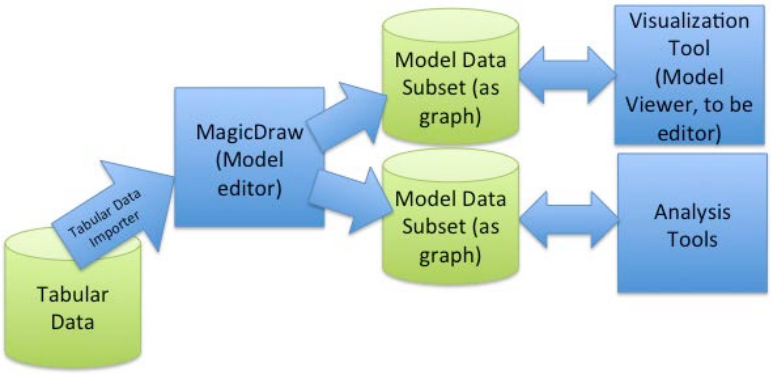

Figure 1: Architecture diagram of the modeling approach

**A. Modeling the System**

The modeling approach relies on the use of the SysML to capture the model information in NoMagic's MagicDraw modeling tool. This allows at least partial re-use of existing SysML models developed at JPL that capture the processes, information, and software utilized by system engineers responsible for the mission operations system. Instead of focusing on only the cyber-security related aspects of the system, we took a holistic approach where information is captured at varying abstraction levels. This approach was taken to identify risk and characterize potential impacts. The current model abstraction levels include:

- Objective
- Workflows
- Software
- Hardware

Modeling the interactions between abstraction layers permits tracing between high-level objectives and the underlying hardware and software, and also identifies which high-level workflows would be impacted if a piece of hardware or software was compromised.

The goal of the model is to communicate with SMEs familiar with the operational system and with a different set of SMEs more familiar with cyber-security details. This provides SMEs familiar with cyber-security details an avenue to understand the overall system. Currently, when cyber-security SMEs attempt to evaluate a system, they need to perform customer interviews and digest a number of disparate design documents to understand the system. This is time consuming and involves conflicting information which increases the cost of engagement with cyber security experts. An advantage of developing the model is it forces reconciliation of the various information sources. One difficulty in MBSE is constructing and maintaining the model so that the information captured remains a valid abstraction of the system being modeled.

Another difficulty with MBSE efforts is enabling aSME interaction with the model. Most SysML-based approaches rely on communicating the information captured in the model using static views (often referred to as Diagrams). More advanced methods rely on developing custom code or queries to present information about the system in tabular forms or even in interactive graphics [IMCE]. Usually, the custom code or queries are used to generate views for a specific purpose (traditionally described as views that respond to some stakeholder viewpoint). A difficulty encountered at JPL with this type of approach is that often "model" experts are needed to create the viewpoints and SMEs are not able to navigate the model effectively without the support of these intermediaries. Although this can be effective for projects where the time scale needed to retrieve information is sufficient, there is a need in the cyber domain to navigate between the different layers of the model more quickly.

Instead, as will be described in Section IV on visualization, we visualize the entire graph (as a directed labeled graph) and then integrate analysis and navigation capability directly into the tool. To simplify the visualization and analysis process the SysML model is exported to a simplified XML representation which has nodes, edges, and properties about these entities. Each node and edge is typed. Nodes are labeled with a type representation (Hardware, Software, Information, Workflow, Data Storage, etc.). The edges are also labeled to allow appropriate

traversal through the graph. This transformation is done by taking advantage of the underying nature of the SysML model and performing a model transformation to produce the directed graph.

A simplified version of the meta-model is illustrated in Figure 2. (A meta-model in this case describes the possible concepts and relationships that could be present in the current model). The classes in the diagram are possible types for the nodes and the associations are possible types for the edges. For instance, a Process may rely on a particular piece of Software through the depends on relationship illustrated between “WorkflowProcess” and “Software”. There are there cyber-related areas that are being considered for the model, protocols used for communication, access control rules, and the vulnerabilities associated with particular hardware and software elements. The vulnerabilities are taken from a common vulnerability database as described in section III-B.

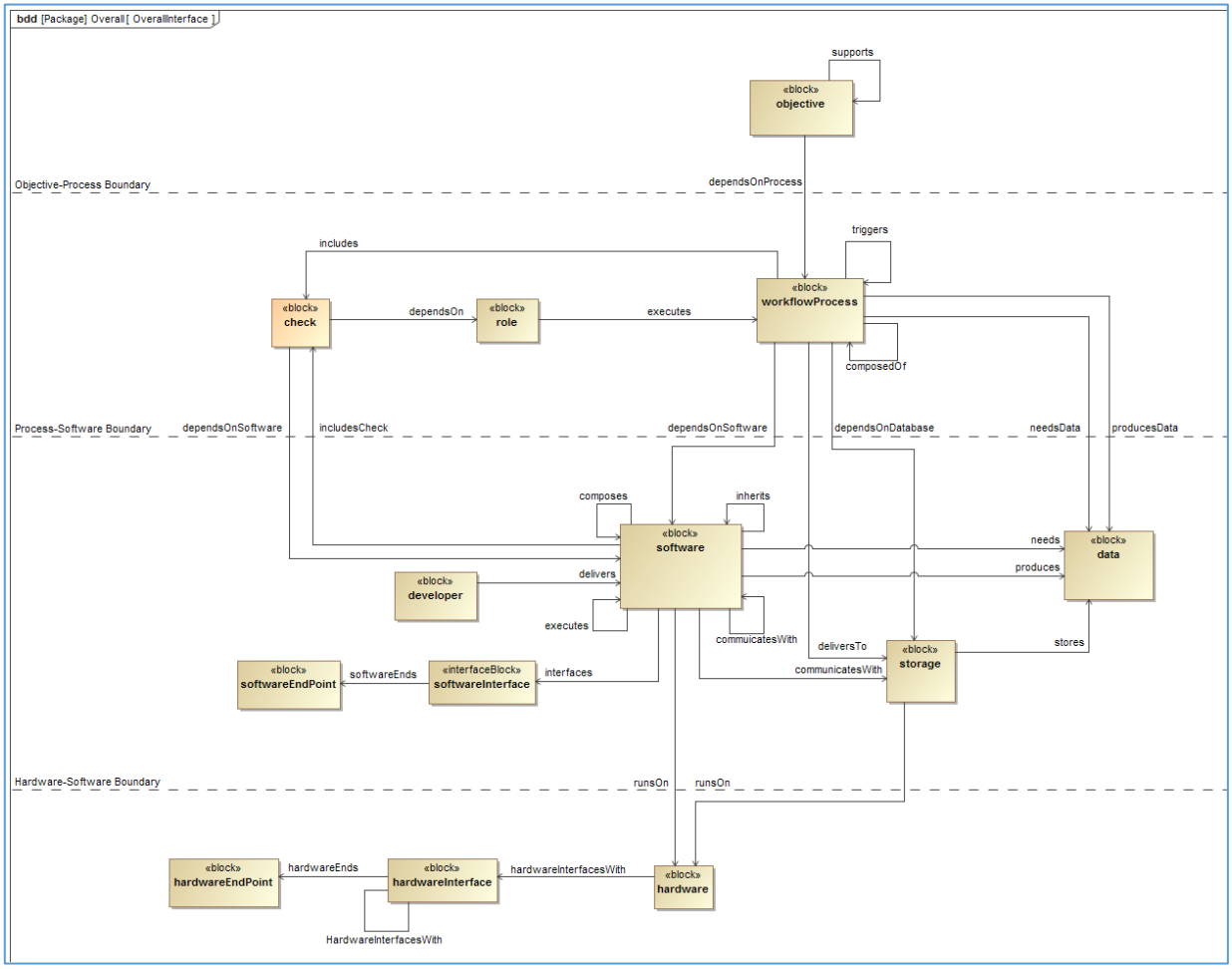


Figure 2: Screenshot of MagicDraw showing the meta-model structure

The information captured in the model evolves at varying scales. For instance, workflows may change but the vulnerability of the underlying hardware and software may change rapidly depending on security patches, details of the firewall, router, and software configuration. To address this issue, information in the model is tagged with information that characterizes the source of the information. Also, as will be discussed further in the future work section V, we are considering how to capture the time providence of the information. One avenue of possible future work is ingesting information about the underlying hardware and software configurations from COTs network analysis tools . As a simple example, to ensure that particular hardware is still available on the network, the currently developed tool can ping the modeled hardware to ensure it is still available. Simply pinging the hardware is a very simple approach, with several drawbacks, but it demonstrates the idea of incorporating real-time information with the analyses to provide evidence for the accuracy of the model and resulting analyses. Trying to gather information in real time about the system is difficult, but in some cases this information is already being gathered. Ingesting it into the model provides a context to analyze the information. Many other modeling approaches only a capture a part of the system; for example if a model only captures the underlying network connectivity between physical and/or virtual hardware, it can be difficult to understand how compromising a particular piece of hardware will impact the overall mission objectives. Therefore it is difficult to assess the criticality of that hardware or the impact of an attack on hardware or the risk that the overall function of the system is compromised.

### B. Capturing and Analyzing Cyber-Security Information

The cyber infrastructure of an aerospace mission network consists of many different components such as servers, firewalls, routers, and communication facilities, each with a different level of importance to the active mission. The

infrastructure is constantly changing to fit the goals of each stage of the mission. Thus the need for a dynamic cyber security analysis is highly desirable.

There has been much research in analyzing the vulnerabilities of an individual host and now standard techniques such as the Common Vulnerability Scoring System (CVSS) have been developed [CVSS] based on information from the National Vulnerability Database (NVD) [NVD]. The CVSS has now been incorporated into many different works to evaluate the security of a network [AY, FW, FWSJ, Jak, XLOLL]. Hence we will root our analysis using the NVD to make it transferable to additional networks. However, the individual vulnerabilities on a system may not seem critical, but an attacker can bring the network into an undesired state using a combination of unrelated vulnerabilities. From a defenders point of view it is important to foresee possible routes available to an attacker and identify critical routes. The vulnerabilities captures in these databases are correlated with the Hardware and Software present in the model. We then utilize an algorithm to traverse the model Below we outline a simple algorithm to identify a possible path that an attacker can take. We also propose a simple metric that can be used by a system administrator (SA) to focus their cyber security efforts.

As noted earlier the underlying infrastructure of the network captured in the model can be viewed as graph where the vertices can be servers, firewalls, routers, software, files, etc., and the edges represent the physical connectedness of the vertices or dependency in the network. Using this framework the analysis that we use is essentially a rule-based graph.

Since our analysis uses the NVD and CVSS in an essential way we explain a few more details about the CVSS. Underlying the CVSS is a vulnerability that has been associated to a piece of software and seven attributes that can be determined for a vulnerability:

1. Access Vector: Local, Adjacent Network, Network
2. Access Complexity: High, Medium, Low
3. Authentication: Multiple, Single, None
4. Confidentiality: None, Partial, Complete
5. Integrity: None, Partial, Complete
6. Availability: None, Partial, Complete
7. Gain Access: None, User, Admin.

The first three attributes deal with the exploitability of the vulnerability, while the last three attributes concern the impact on the data contained on the system. A score from 0 to 10 is assigned based on the first six attributes, commonly known as the *CVSS Base Score v2*. However, by definition of the Base Score it is independent of the Gain Access attribute. For example, the vulnerability CVE-2014-2653 against openssh version 6.4, where the seven attributes are (Network, Medium, None, Partial, Partial, None, None), has a Base Score of 5.8. See [CVSS] for a detailed explanation of the formula.

While the CVSS Base Score is a good indication of the vulnerability of a program, the main problem for network security is that it is data centric. For example, the vulnerability CVE-2011-4109 on openssl has a Base Score of 9.3, with Confidentiality, Integrity and Availability of data being Complete, but does not give the attacker escalated privileges. While CVE-2003-0131 has a Base Score of 7.5 but can escalate the privilege of attacker to *User* on the system. From the network viewpoint the latter vulnerability could be seen as more critical even though it has a lower Base Score. The reason this could be seen as more critical is that an attacker will seek to get a toehold into the network from which they can pivot to different parts of the network. Thus our algorithm does not consider the Base Score, but instead considers just the Access Vector, Access Complexity, and Gain Access.

The attack algorithm takes as inputs a starting node and a goal of the attacker, and outputs if the attacker will be successful along with a path and CVEs at each vertex that an attacker can deploy. We do not assume the attacker will deploy a particular attack, but instead an attack they can exploit from vulnerabilities on the system. The first step is to find all the servers that contain the goal, for example the attacker could be seeking a file contained on multiple servers. Since the graph could potentially be very large we exclude all servers that do not contain vulnerabilities where the attacker can escalate privilege from the Network. We then find a shortest path from the start to a server contained in the goal_server. If such a path exists we return the path with the CVEs the attacker uses

and the privilege gained at each step, or if such a path does not exist we return False. A short pseudo-code for the algorithm is the following:

```
procedure ATTACK(start,goal)
    for goal
        find goal_servers
    end for
    for each server in Network do
        if ∄ CVE with AV=Network, AC=Low, GA=User or Admin
            delete server
    end for
    if start is deleted
        return False
    else
        path=shortest_path(start to goal_servers)
        if path is not None
            return path, CVE_path
        else
            return False
```

The novelty in this approach is that it incorporates privilege escalation and the attack vector as opposed to relying on just the Base Score. We believe this method to be more accurate from an attacker's perspective for the reasons noted in the previous paragraph.

To incorporate privilege escalation into a metric that is easily interpretable we use the following score called the *Network Criticality*:

$$NC = \frac{3}{10}(Base\ Score) + 7\chi(Admin) + 4\chi(User) NC = \frac{3}{10}(Base\ Score) + 7\chi(Admin) + 4\chi(User)$$

where $\chi(Admin)\chi(Admin)$ is 1 if the vulnerability gives the attacker Admin level privileges and 0 otherwise, and a similar definition applies to $\chi(User)\chi(User)$. By definition the Network Criticality of vulnerability is again a score from 0 to 10, where 10 represents if the Base Score is 10 and admin level privilege is gained, e.g., CVE-2007-6529. A similar score has been proposed in [AY]. Notice that if a score is strictly greater than 3 then some sort of privilege has been gained. An SA could easily use this score by patching all software with a vulnerability such that

$$NC(Vuln) \geq \mu(NC) + 2\sigma(NC), NC(Vuln) \geq \mu(NC) + 2\sigma(NC)$$

i.e., the mean plus two standard deviations. There are many variants that one could use, but this would identify the software that is most vulnerable from the viewpoint of privilege escalation. Another simple metric for an SA could be to patch server where the sum of the *Network Criticality* of all the software is the greatest.

## III. Visualization System

The characterization of SME knowledge results in a model that is large and beyond the ability of a single user, SME or otherwise, to fit into working memory and explore naturally. It was understood early in the effort that being able to visualize, understand, and analyze the model in an interactive way would be an essential component of the on-going network modeling and analysis effort. The visualization system has been developed in conjunction with cyber security and MBSE experts and leverages the simulation and real time telemetry visualization [MIP1] and user interface design and development [MIP2] expertise in the JPL Robotics section.

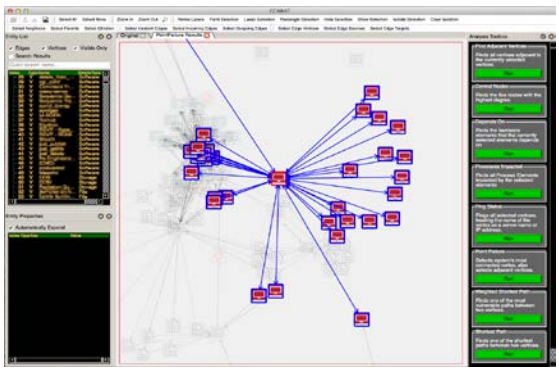

Figure 3: Visualization system showing the node and edge selection from an analysis.

Though still in beta, the visualization system is capable of:

- Visualizing the model as a directed graph
- Utilizing different graph layout algorithms from force directed to deterministic
- Extensibility for custom layout algorithms
- Displaying entity (node and/or edge) properties stored in the model
- Enabling various entity selection mechanisms based on graph topology or model semantics
- Querying entities based on model attributes
- Running custom analyses on the model
- Extensibility for custom analysis modules
- Visualization of analysis results directly on the graph

These features, developed with input from SME analysts and users, enable an exploratory workflow for users to examine attributes of the network model and perform analyses.. SME analysts can use the system to tackle the resiliency question by exploring paths of vulnerabilities; Identify hardware or software that impacts high level processes; Determine potential defensive options, etc. More importantly it allows users to explore the model in context and with local understanding. With these features an expert can start an exploit exploration from any point in the system, covering internal and external exposures. Step by step they can explore different paths to take based on the information stored in the model and the exploits tied to the model. They can also use the system as a launching point for automated vulnerability analysis and exploration.

### A. Navigation and Tools

At startup, the system visualizes the graph model in 3D space using a force directed layout algorithm. Visualizing in 3D increases the space available for separation of graph elements by the layout algorithm, making it easier to visualize a dense and complex network.

Navigation through the graph is supported by manipulation of a 3D camera using common mouse control paradigms such as dragging to spin a view, using a modifier and a mouse drag to zoom, using another modifier and a mouse drag to pan, etc. The camera can be moved to different entities with a simple double-click. Multiple views of a graph can be created and open to maintain different, potentially useful perspectives on the the model.

A list of entities and their properties are displayed conveniently to the left, following a standard and familiar interface paradigm used in many 3D and CAD software.

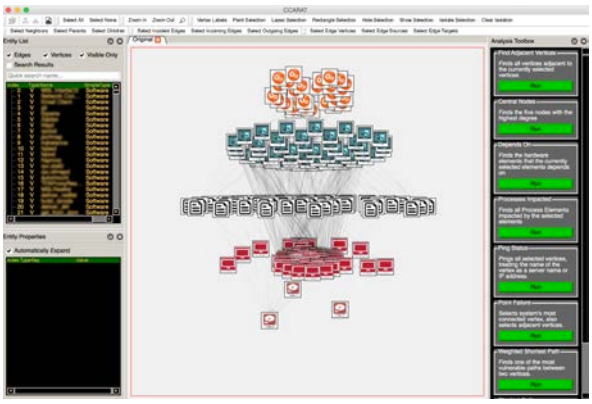

Figure 4: Visualization system showing a custom layered layout that utilizes model semantics

Layouts in 2D are also supported and utilized depending on context. More importantly, custom layouts are supported. Normally the layout algorithms available will respect graph topology but not graph semantics as encoded by the model. One of our goals is to develop layout algorithms that respect the modeling semantics to improve user understanding. Figure 4 shows a preliminary custom layout that places nodes on different abstraction layers depending on the node types encoded in the model. The ability to write these custom layouts and align semantic metrics to different spatial axes should greatly improve understanding.

### B. Technical Implementation

The visualization system is built entirely in Python 3.4 [PYTHON]. Python 3.4 is used for rapid development, cross platform capability, an excellent standard library, an expansive collection of third party libraries, automatic dependency management using Python setup tools, and for its ability to act as a glue amongst various software components.

Based off of existing visualization and interface expertise, Qt 4.8 is used as the interface framework with PyQt4 as the Python binding. We use the igraph [IGRAPH] library for graph represention, topology based graph traversal, basic network analysis algorithm, and graph layout algorithms. Components of the system were modularized into separate Python packages whose dependencies could be handled cleanly by Python setup tools, the Python package installer (pip), and Python virtual environments. This set up also simplifies deployment by allowing automated tools to handle resolving package dependencies.

Currently the visualization system deploys on Mac OS X 10.7 or higher and Ubuntu 15. Windows support is planned but not within the current scope. For each of the different platforms, the end user will only be required to install the software for their platform. Any of the required input files will be cross platform, requiring no special handling.

### C. Interfaces and Extensibility

A tight feedback loop between tool developers and tool customers meant an understanding that capabilities and requirements could rapidly change as development progressed. The novelty of the work meant a clear understanding of the requirements from the beginning was not feasible. Instead the requirements of the visualization system evolved with its capabilities. This development environment required the establishment of data and behavior interfaces which promote separation of concerns and future extensibility.

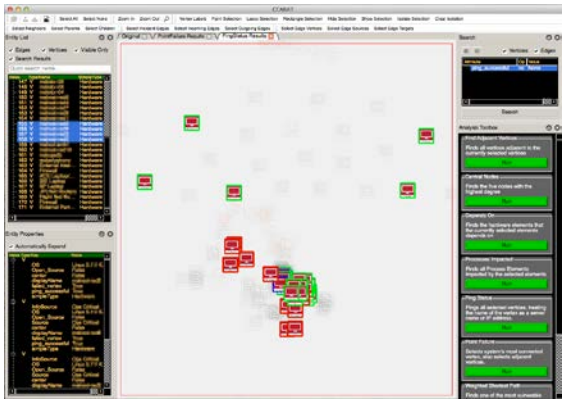

Figure 5: Visualization system showing results of an analysis being visualized on the graph.

For visualization system input, the modeling team transforms the MBSE model into an XML format that captures the models details. This custom XML format is parsed by the visualization tool into an internal data structure capable of algorithmic search. Information about security vulnerabilities is to be stored separately from the model itself in a secure format and loaded separately from the model as well. Relationships between the vulnerabilities and the model are tied together at run time.

MBSE and cyber security SME developers write analysis scripts to be performed on the model. To enable these developers to run analyses on the model interactively an interface between the system and analyses was developed allowing analysis writers to specify expected inputs, signal completion to the program, store results directly into the graph, and specify how it wants the results to be visualized. The system automatically populates an analysis toolbox based on scripts that implement the agreed upon interface. This allows analysis writers to not only develop the analyses separately from the tool but also utilize the tool's functionality, such as the properties viewer, to debug their work. It also enables the system to run the analyses concurrently in separate threads and even separate processes.

## IV. Future Work

There are three main areas of research in the future that we plan to incorporate into the model.

Automated attack tree exploration is needed. An attack tree is a diagram that breaks down the steps that an attacker could take to obtain a goal [Sch]. A Subject Matter Expert (SME) usually constructs these based on their knowledge of the field and historical attacks. Since attack trees are based on already seen attacks it would be beneficial to see

how resilient the infrastructure of a network would be against that attack. Attack trees can be very generic in relationship to a given network and the number of different attack trees could be immense with far too much information to be explored manually.

Many mission critical systems are time critical. For example, a cyber attack that takes Mission Control offline for even a short time during the entry, descent, and landing stage of Curiosity, aka *The Seven Minutes of Terror*, could be devastating to the mission, while a cyber attack of the same duration during the cruise stage may not have any impact. In light of this time criticality, the duration of a cyber attack needs to be incorporated into the model. Some approaches have already been proposed [Jak, MTTFP].

As important as the NVD is for the model, it is equally important to understand the vulnerabilities contained in the software developed in-house for a particular mission. We plan to generate reports using source code scans of mission critical software and incorporate them into the model in a similar way that was done using the NVD.

Beyond these research areas, development of the visualization system will continue in tight conjunction with the analysis team to ensure that the system evolves in a direction that is useful for network resilience exploration and testing. Improved visualization look, feel and performance, and layout methods for faster network graph understanding are key goals. Usability improvements and new semantically aware tools will make exploratory analysis faster and more efficient. Finally, we will investigate the possibility to use the visualization system in a real-time, telemetry display mode, where data from the actual, modeled network is displayed, possibly in combination with the predicted output from the analyses, giving SME and modeling engineers insight into model correctness and analysis performance.

## Acknowledgments

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors would like to thank Bob Vargo, Sami Saydjari, and Frank Kuykendall for support and discussion of the work.

## References

[AY] B. Argauer, and S. Yang: *VTAC: Virtual Terrain Assisted Impact Assessment for Cyber Attacks.* SPIE Defense and Security Symposium. International Society for Optics and Photonics, 2008.
[CVSS] Common Vulnerability Scoring System, https://www.first.org/cvss. August 2015.
[FW] M. Frigault, and L. Wang: *Measuring Network Security Using Bayesian Network-Based Attack Graphs*. IEEE, 2008.
[FWSJ] M. Frigault, L. Wang, A. Singhal, and S. Jajodia: *Measuring Network Security Using Dynamic Bayesian Network.* Proceedings of the 4th ACM workshop on Quality of protection. ACM, 2008.
[Jak] G. Jakobson: *Mission Cyber Security Situation Assessment Using Impact Dependency Graphs.* Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on. IEEE, 2011.
[MTTFP] S. Musman, A. Temin, M. Tanner, D. Fox, and B. Pridemore: *Evaluating the Impact of Cyber Attacks on Missions.* Proceedings of the 5th International Conference on Information Warefare and Security. 2010.
[NVD] National Vulnerability Database, http://nvd.nist.gov. August 2015.
[Sch] B. Schneier: *Attack Trees.* Dr. Dobb's journal 24.12 (1999): 21-29.
[XLOLL] P. Xie, J. Li, X. Ou, P. Liu, and R. Levy: *Using Bayesian Networks for Cyber Security Analysis*. Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International on. IEEE, 2010.
[PYTHON] Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org
[IGRAPH] Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. http://igraph.org
[SYSML] Friedenthal, S., Moore, A., and Steiner, R., *A Practical Guide to SysML: The Systems Modeling Language*, Morgan Kaufmann Publishers / OMG Press, 2008.
[IMCE] Bayer, T., Cooney, L., Delp, C., Dutenhoffer, C., Gostelow, R., Ingham, M., Jenkins, J.S., and Smith, B., "An Operations Concept for Integrated Model-Centric Engineering at JPL", *Proceedings of the IEEE Aerospace Conference 2010*, Big Sky, MT, March 2010, IEEEAC Paper #1120.