Cyber Threats' Impact on Mission (CyTIM): Systems Integration Results

John H. James,* *Member IEEE*, Joshua J. Kraunelis,* Jennifer J. Mathieu,*
Deborah E. Flagg, Tristan M. Lewis, and Richard D. Flournoy

Abstract— A hybrid approach of mission simulation modeling integrated with a virtual environment to measure a cyber attack's effect on mission is demonstrated. While existing testbeds aim to support malware and network infrastructure experimentation, this study extends these capabilities with a mission-modeling and control layer that allows the collection of data not only on malicious software's impact on network infrastructure, but also on the network's mission analysts and their ability to complete assigned tasks. The operations center use case employed demonstrates that a notional denial-of-service attack on a key router cripples the operators' mission and delays task processing for hours, specifically 2 to 8 hour delays when chat is used as a mode of communication versus 4 to 12 hour delays when email is used to communicate. The metric end-to-end mission thread response time is demonstrated to help quantify an attack's severity and could help quantify the success of defenses as well as support a training environment for analysts.

Index Terms-- Modeling and simulation, discrete event, Petri net, mission thread, malware, metric, measurement techniques, evaluation, and virtual testbed environment.

1 Introduction

A virtual testbed for determining the impacts of cyber threats on mission could help prioritize cyber threats for on-going missions, allow for experimentation to test alternative response procedures and defensive measures, determine best data-related policies (e.g., user authentication policies), and support the evaluation of new tools and their insertion into the analyst workflow, including information assurance products. Four existing and emerging testbeds were identified and reviewed.

The Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) is an application developed by the Massachusetts Institute of Technology's Lincoln Laboratory starting in 2002 [1]. LARIAT specializes in generating, detecting, and evaluating user and network activities. LARIAT specializes in generating background traffic according to patterns of use as defined by state machines. Large numbers of virtual users may be set up to send traffic via various network protocols and use Windows applications that are usually run on virtual machines. LARIAT allows configuration of virtual systems and scenario parameters.

The cyber-Defense Technology Research (DETER) testbed is an application developed and hosted by the University of Southern California Information Sciences Institute and the University of California at Berkeley in 2002 for distributed national cyber security research [2, 6]. The DETER architecture consists of roughly 300 heterogeneous nodes, grouped into two geographically separate clusters that communicate over an Internet Protocol Security (IPsec) link. To increase fidelity, DETER has a set of commercial off the shelf routers and switches that can be linked into an experiment. DETER is built on the University of Utah's Emulab software, which it uses for host management, experiment switching, and rapid reconfiguration. Emulab allows several experiments to run simultaneously and can switch out inactive experiments to free nodes for other researchers to use. Experiment design is streamlined in DETER, as a collection of tools exists to design network topology, generate traffic, and even launch attacks. Researchers can access their DETER testbed via a web interface. Care has been taken to ensure DETER experiment traffic, which may contain malware packets, has no path to the Internet. Researchers from the government, academic, and private sectors use the testbed.

The Laboratoire de securité des systèmes d'information (SecSI lab) is an application developed at the *Ecole Polytechnique de Montreal in 2005* for inhouse use [3]. The production cluster at the SecSI lab consists of 98 blade servers running Virtual Machine (VMware ESX) server software. Therefore, any host involved in an experiment must be virtualized, including routing software. The cluster at SecSI uses IBM's Extreme Cloud Administration Toolkit (xCAT) to automate disk image loading and network configuration. The testbed has been used to experiment with botnets and other live malware. The SecSI testbed does not currently offer remote access for experiment design or administration. Because the cluster resides on a completely isolated

First two authors contributed equally to this manuscript

- *J.H. James is with The MITRE Corporation, Bedford, MA 01730.*
- *E-mail: jhj@mitre.org.*
- *J.J. Kraunelis is with The MITRE Corporation, Bedford, MA 01730.*
- *E-mail: jkraunelis@mitre.org.*
- *J.J. Mathieu is with The MITRE Corporation, Bedford, MA 01730.*
- *E-mail: jmathieu@mitre.org.*
- *D.E. Flagg is with The MITRE Corporation, Bedford, MA 01730.*
- *E-mail: dflagg@mitre.org.*
- *T.M. Lewis is with The MITRE Corporation, Bedford, MA 01730.*
- *E-mail: tmlewis@mitre.org.*
- *R.D. Flournoy is with The MITRE Corporation, Bedford, MA 01730.*
- *E-mail: rflourno@mitre.org.*

**To whom correspondence should be addressed*

network, researchers must be physically present at the facility to interact with SecSI hosts.

The National Cyber Test Range (NCR) testbed started in 2008 [4] to "enable a revolution in the Nation's ability to conduct cyber operations by providing a persistent cyber range." The testbed will include automated "virtual people" who will drive real software, adding realism to network activity. These users will demonstrate complex interaction, play multiple roles, and react to changes in network environment. However, it is unclear at this phase of development whether these actors will be scriptable or completely autonomous. The National Cyber Test Range is being designed to replicate large-scale heterogeneous networks and enable realistic testing of a global network.

Here, mission is a specific task with which a system of systems is charged to accomplish. Mission modeling defines a model of the task so that its performance may be measured. In this context we want to construct a mission modeling environment so that the performance can be a function of a cyber attack and its defense.

The objective of the Cyber Threats' Impact on Mission study was to integrate a mission-modeling capability with a cyber testbed environment. Typically, performance metrics to quantify cyber impacts are done at the communication level, application level, or transaction level [7]. In 2009, a thorough study was done using DETER to "produce measures of service denial that agree with human's perception of service impairment"–the metrics were based on the percentage of failed transactions [7]. In this paper, a metric is presented to quantify the impact of a cyber attack on an end-to-end mission workflow.

2 METHODS

Determining the impact of cyber threats on mission performance required integration of a modeling and simulation engine with a system component, as shown in Fig 1. Popular commercial simulation tools such as iGrafx[1] and AnyLogic[2] were considered for the mission model, but ultimately MSim, a MITRE built tool, was chosen because of its ease of modification for integration purposes. The existence of an applicable mission model developed in MSim was instrumental in its selection. A denial of service attack on a mission critical router was used for illustration purposes.

As outlined, virtual testbeds to explore cyber threats exist or are under development (e.g., LARIAT, DETER, SecSI, and NCR)—LARIAT was selected for this proof-of-concept since it could be stood up locally. A system representing the network and other hardware portions of the system used in the mission model was implemented. To allow safe susceptibility to malware, the system was implemented using virtual machines. This choice allowed for modeling hundreds to thousands of users, if needed, while maintaining a manageable amount of physical systems to emulate real software and network delays. Once the network was completed, realistic network traffic was required to increase the authenticity of the virtual environment. LARIAT's traffic generation tool was setup on the network to generate the necessary background traffic [1].

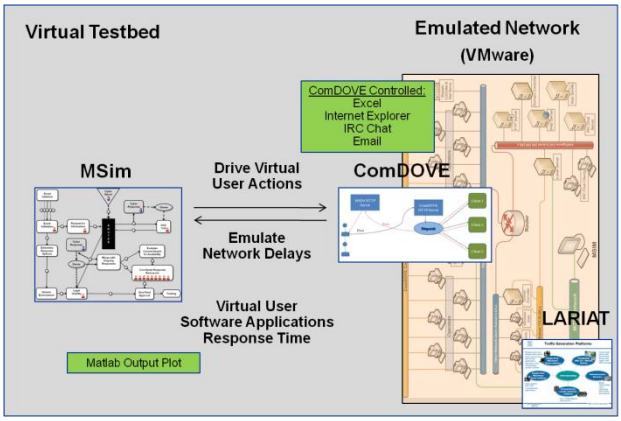

Fig. 1. Cyber Threats' Impact on Mission (CyTIM) virtual testbed with system components: MSim, ComDOVE, LARIAT, and a VM-ware emulated network.

A new capability was needed to integrate the mission-model simulation with the virtual environment—the Command Dispatch Outcome Verification Environment (ComDOVE) was built to allow simulated actions in the mission-model to be sent and executed on the virtual hosts. The success or failure of the actions is reported back to the simulation engine (i.e., MSim) from ComDOVE. The system-timing is measured by the synchronous response time of the messages. The mission model drives virtual user actions, and the emulated virtual network provides real software application response times and network delays. The mission-model incorporates any think-time by the simulated users or analysts. In addition, the setup provides for future human-in-the-loop testing by replacing a subset of the simulated analysts with cyber-focused or mission-focused analysts.

2.1 Use Case

The operations center process model use case is not intended to be an accurate representation of a real-world situation, and was purposely simplified to yield measurable, easy to interpret results. The use case includes an operations center, an intelligence cell, and other network users, as shown in Fig 2. The operations center subnet includes 16 analysts, a Windows domain controller running Microsoft Exchange Server, and an Internet Relay Chat (IRC) server. The intelligence cell and other network users contain 180 LARIAT simulated users and a Windows domain controller. At the edge of the base network is a router, connecting the operations center, intelligence cell, and other network users. The segment of the network most fundamental to the use case contains two analysts who represent intelligence analysts, a Windows domain controller, a Microsoft Exchange server, an IRC chat server, a File Transfer Protocol (FTP) server, and a web server.

[1] http://www.igrafx.com/

[2] http://www.xjtek.com/

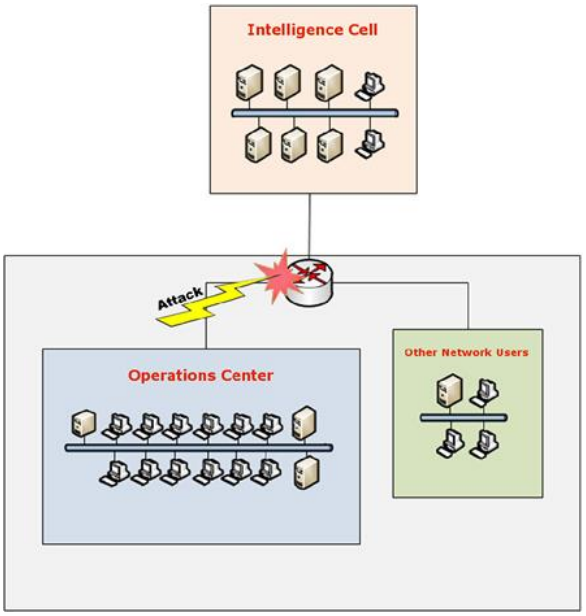

Fig. 2. Scenario models an operations center, an intelligence cell, and other network users.

The mission model processes time sensitive events over a 24-hour period. During that period, a few of the 16 analysts communicate with intelligence cell personnel when additional information is needed to process an event. In addition to that communication, the 180 users in the intelligence cell are frequently accessing documents from the network via HTTP and FTP. Roughly three hours into the day, an insider executes a denial of service attack on the router. When the operations center users are unable to complete their mission because communication has been interrupted, they alert their cyber information technology (IT) support staff. The IT personnel spend hours working to diagnose and repair the problem. Once the communication link has been restored, the events are reprocessed.

2.2 Network Traffic

LARIAT was used to generate realistic user-based background traffic. With the help of members of the LARIAT team at Lincoln Laboratory (see Acknowledgment), a simple configuration composed of an internal network communicating through a router to an external network was setup. Nodes on the internal network included a domain controller running Windows Server 2003 and three clients: one Windows XP system and two Linux systems running CentOS 4. The external network consisted of two Linux systems, one serving as a web server, the other as an FTP server. The traffic configuration selected was one user on the Windows XP system accessing web sites, and 180 simulated users on the two Linux clients sending and receiving files to the FTP server.

Tshark (www.wireshark.org), an open source packet capture program, was used to capture traffic sent across the router during the 24-hour period. To keep the captured traffic files manageable, tshark was configured to write a new, separate file for each hour's worth of traffic data. The traffic capture took place on the network interface connected to the intelligence cell (see Fig 2), ensuring that both the background traffic generated by LARIAT and the IRC/email traffic relevant to the experiment were captured.

Several open source traffic analysis and visualization tools were tried. Though these tools worked well for a relatively small amount of traffic data, none were able to handle the enormous amount of data we collected during the 24-hour experiment. A simple tool for traffic visualization was implemented, which focused on memory and computational efficiency. It uses tshark's filtering capability to narrow the dataset to only those network protocols specified by the experiment, decreasing the amount of data to be processed. Optionally, the resulting files can be merged together to create a continuous dataset. Using tshark and its companion software, mergecap, to manipulate the capture files, the tool was able to provide the necessary traffic output.

2.3 Mission-Model

MSim, a Macintosh based C++ Cocoa application, is a Petri net (PN) discrete event simulation tool that captures time delays in event processing, as shown in Fig. 3. For a given scenario, MSim produces performance metrics such as resource utilization, component throughput, and thread response time. These metrics can be used to 1) determine if the process modeled meets its operational performance requirements, 2) find the performance bottlenecks, and/or 3) evaluate the performance effectiveness of proposed process changes.

MSim is a resource constrained task flow model. It represents resource constraints of humans, i.e., there must be a human available to do a task, and the human can only work on one task at a time. The tasks may be prioritized. The current integration of the system components allows the tasks to be constrained with real system resources. The think-time for a given step is done in MSim, and the time of the system response is accomplished in the virtual machines.

The model used in MSim processes time sensitive tasks in an operations center. The model represents the tasking of 16 operations center analysts (person icons in Fig. 3), whose system actions were simulated or emulated in the virtual system. The model has six types of events to which it responds. All events were given the same priority except for one event type, which was given a higher priority. The effect of this was that the higher priority event preempted work on any other event type being processed in the same resource.

Five event or thread types (visualized by the five lines between the first two transitions, Event Initiation and Event Validation, in Fig. 3) are processed through *"Event Validation"* to *"Tasking"* in Fig. 3, and the end-to-end response time is measured. The Petri net in Fig. 3 shows *Transitions* (rectangular boxes) where activities are modeled that do work and produce outputs and *Places* (circles or ellipses) that represent the type of data that a *Transition* needs for input and the type of data that the *Transition* produces as output. Tokens represent the instances of data created and consumed by *Transitions*. Tokens are also associated with a thread type, and their movement is restricted along *Arcs* (edges or lines) that are associated with the same type. MSim supports a timed Petri net that allows discrete event

simulation, and in this case, *Transitions* usually have timing functions that introduce time delays in processing. The grey *Transitions/Places* and bold, dotted lines indicate the cyber attack and cyber response.

The operations center process model starts with *"Event Validation,"* when a first attempt at responding is made, and a decision is made whether additional information is necessary. If so, a *"Request for Information (RFI)"* is made to the intelligence cell to obtain data from available sources. Meanwhile, an initial effort is made to *"Assess the Environment."* Depending on the RFI results, this assessment may be redone at a high priority level.

The assessment results are then given to the *"Operations Center Legal Office"* to check guidelines for legal viability. Finally, the results of the efforts to *"Evaluate Current Assets" and "Merge with Ongoing Responses"* are brought before decision makers to *"Coordinate Response Resources."* The approved event is sent for execution *"Tasking."* To keep track of the response time of the event thread, the Petri net writes to a log whenever a token reaches the end of the thread. These logs are kept by type of thread and are processed using Matlab.

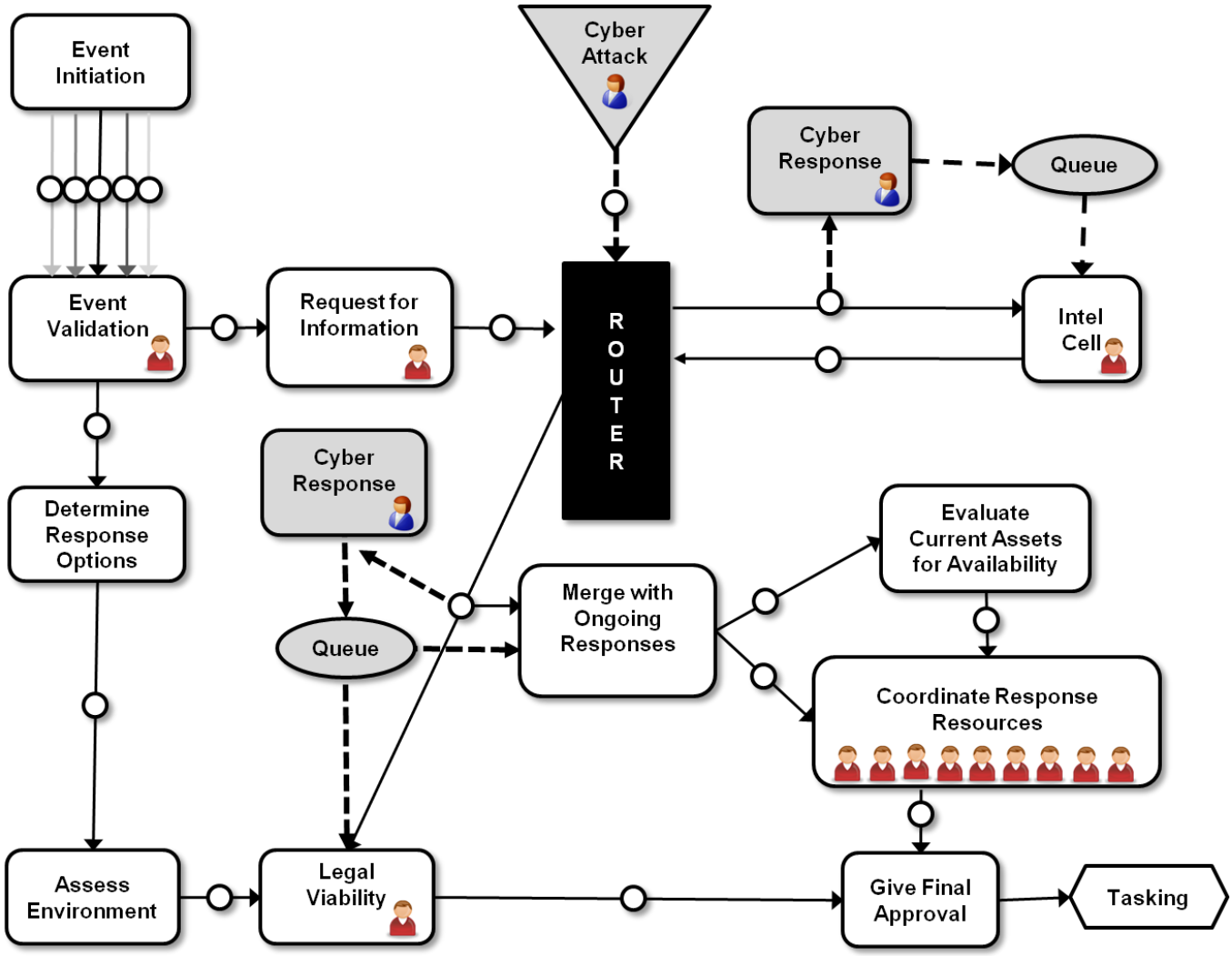

Fig. 3. The notional operations center process is a discrete event Petri net model with 16 virtual analysts shown using person icons. Five event types (shaded lines) are processed through *Event Validation* to *Tasking*, and this response time is measured. The schematic shows *Transitions* (rectangular boxes) where activities are modeled that do work and produce outputs and *Places* (circles or ellipses) that represent the type of data that a *Transition* needs for input and the type of data that the *Transition* produces as output. Tokens represent the instances of data created and consumed by *Transitions*. Tokens are also associated with a thread type (visualized by the five lines between the first two transitions), and their movement is restricted along *Arcs* (edges or lines) that are associated with the same type. MSim supports a timed Petri net that allows discrete event simulation, and in this case, *Transitions* usually have timing functions that introduce time delays in processing. The grey *Transitions/Places* and bold, dotted lines indicate the cyber attack and cyber response.

2.4 Environment and Mission Model Integration

The simulation model is integrated with the environment through the *Transitions* in the Petri net model. For example, if a task requires the use of email, chat, spreadsheet or web pages, the actions are emulated on the virtual system. When these Transitions fire in the simulation, they incorporate the "think-time" of the modeled task in the simulation. Before the Transition finishes, MSim sends an asynchronous message to ComDOVE that requests the required resource to occur in the virtual portion of the testbed. When the required service is finished, an asynchronous message is sent back to MSim, the Transition ends, and the tokens move forward in the Petri net model. Resource contention is determined in the PN with appropriate utilization in the virtual world.

ComDOVE, is a set of Python scripts for driving software applications over a network. It provides a simple web interface with multiple resources for interacting with and monitoring the status of its clients. ComDOVE consists of numerous clients and a single control point, called the ComDOVE interface. All communication to the clients happens via the interface. The interface can support several concurrent connections and is designed to control a large quantity of subordinate clients. If scalability becomes an issue, multiple interfaces can be linked together to distribute the workload. Both the client and interface software are platform independent.

The ComDOVE automation architecture can support a wide range of software. There are two main classifications of automation: infrastructure and application. The infrastructure group consists of software that supports tasks like maintenance and status reporting for the actual ComDOVE software environment. Examples of this include subversion repository updates, ComDOVE client uptime, and other control messages. The application group consists of software that will be used in the virtual environment, either by accessing ComDOVE programmatically by having MSim drive actions using the mission-model or having actual trainees interact with the environment. Examples include: Internet Relay Chat, Internet Explorer, Microsoft Word, Microsoft Outlook, and Microsoft Excel. When a client is issued a command via the interface, the software launches, performs the requested function, and returns its status via the interface.

Because performance is an important factor in calls made to ComDOVE, the standard HTTP protocol (RESTful) was chosen over higher overhead protocols such as XML-RPC and SOAP for network communication. Using HTTP POSTs required much less overhead than other protocols and allowed for a more accurate measurement of how long the issued command took to process. Parameters posted to the ComDOVE interface are interpreted and passed along to the appropriate client. When the client machine receives the POST, it launches the given command, waits for execution of that command to terminate, then replies back to the ComDOVE interface with a status code indicating whether the command was completed successfully. The interface will then POST a reply asynchronously back to MSim, using the address contained in the query string of the original ComDOVE post.

Several techniques and protocols have been employed within the ComDOVE client software to automate the given commands. Most low-level software can be driven using standard interprocess communication techniques. SSH and SVN are examples of this category of software. However, high-level software requires more sophisticated automation techniques. For the Microsoft products, the component object model (COM) applications programming interface is used. To interact with the mIRC chat application, two layers of functionality are required. First, the desired actions are scripted in mIRC's own scripting language, independent of ComDOVE. ComDOVE is then able to interact with the mIRC software using Dynamic Data Exchange (DDE). The broadest automation technique employed in ComDOVE uses the client operating system's API to control elements of the graphical user interface. This method is used by default when no other standard of interprocess communication exists.

Time coordination between distributed components is always an important consideration [5]. We can run the testbed in either a hybrid simulation time–real time mode, or in a real time mode. The virtual world must run in real time. However, it is possible to let the simulation run using a simulated future event list when no outstanding services are in progress. For our scenario and model, this reduces the time of a run from 24 hours to about 5 hours. If the LARIAT traffic generator needs to be coordinated by time of day, then the testbed needs to be run in real time mode.

2.5 Experimental Design

The model processes time sensitive tasks by 16 analysts in an operations center, whose system actions were emulated in the virtual system. As illustrated in Figure 1, the various analysts use Microsoft Excel, Internet Explorer, Internet Relay Chat, and Microsoft Email to complete their tasks, and the network delays and application response times are obtained. In the mission-model event thread, sometimes a second look from the intelligence cell is required. For the experiment, this task was completed using both Internet Rely Chat and Microsoft Email (three replicates each). This second look is done by a remote intelligence cell from the operations center (see Figure 2). This requires a message to be sent through a router that is notionally attacked by a denial-of-service. In our setup, this attack is performed by shutting down the router. A network interface was shut down on the router to degrade service instead of performing an actual denial-of-service attack to avoid affecting performance on the underlying virtual hosts. The Vyatta virtual routing appliance proved to be quite capable of handling massive amounts of traffic. It was clear that, to degrade service, the router would need to be under a much more severe load than the virtual environment could produce

To keep track of the response time of the event thread, the Petri net writes to a log whenever a token reaches the end of the thread. These logs are kept by type of thread and are processed using Matlab.

Commands originate in a *Transition* in the Petri net and communicated through ComDOVE. When this router is shut down, communication to and from the intelligence cell stops. Both the sender of a message and the receiver of the message have been commanded by a Petri net *Transition* to either send a specific message or look for that message. During the cyber attack, the receiver never sees the given message and times out. The Petri net model recognizes this timeout condition and sends the token on a new path in the model. In the experiment, the model sends a message to ComDOVE to fix the router. In the real world, this would correspond to an operations center analyst noticing that they have lost a chat connection with the intelligence cell and ask the cyber IT personnel to investigate the problem and apply the appropriate remedy.

3 Results and Discussion

After the start of the cyber attack (at time S in Fig. 4), the mission response time for the mission thread to complete increased by 4-12 hours for email results (Fig. 4a), and 2-8 hours for chat results (Fig. 4b) based on 3 replicated experiments for each. These numbers were compared to a response time of about 2 hours just before the cyber attack was present. Once the attack had been resolved (at time E in Figure 4), mission response times were delayed for an average of 5 more hours based on the 6 experiments, due to the accumulation of work that occurred during the downtime.

Using the operations center mission-model to drive actions using real software in a distributed environment led to several non-intuitive insights. For example, messages were sent across a router via email and chat using the same Petri Net logic—it was expected that the subtle replacement of email for chat would have little impact on the mission-model implementation. However, in the chat version of the experiment more Petri net logic was required than the email version due to the behavior of the real system software. When the router was shut down, the Exchange servers would delay email delivery until a connection was re-established, allowing the simulation to run without adjustment. However, when the router was shut down in the chat version of the experiment, all messages sent during that period were lost. This resulted in an update of the mission-model logic to resend the lost messages once the connection was reestablished.

The chat response time is about half that of the email response time during the cyber attack (Figures 4)—this is due to design differences between the IRC protocol and the Standard Mail Transfer Protocol (STMP). When the router is turned off, the effect of the loss of connectivity is noticeable almost immediately in the IRC environment. This fact would allow the analysts to notify the IT personnel sooner, leading to a remedy sooner. Moreover, the chat server's recovery time is much faster than the email server's recovery time. The effect of the cyber attack is not so apparent with email because the mail transfer agent used in the experiment, Microsoft Exchange, is designed to queue messages for retransmission later when connection can be established. Integrating mission-models with various communication modalities will require careful attention to each modality to ensure the mission-model accurately represents the respective system nuances.

Environment state has an obvious effect on automation control flow, but the designing methods to revert to a "clean" state for experiment repeatability was a subtler need. For the mission-model, unique numbers were inserted into the communication between clients in the virtual environment, ensuring that no message was confused with one previously received. At points throughout the 24-hour long experiments, the machine running the mission-model would lose its hostname and become unreachable. This was addressed by using the machine's Internet Protocol (IP) address instead of the hostname for all communication. Explicitly indicating the state of the environment during experimentation and being able to revert to previous states both from the software automation and mission-model simulation perspective is critical.

3 Network Traffic

During the experiments, the LARIAT traffic oscillated through high and low activity levels at roughly a 5-hour period throughout the day in Fig. 5a (no cyber attack) and in Fig 5b (under cyber attack). The effect of the attack on the LARIAT traffic was small. Notice at the point of lowest activity, the traffic never reaches 0 packets per minute without the attack, as shown in Fig. 5c, but it does stop completely when under attack. LARIAT's HTTP traffic is not affected by the disabled interface, but the IRC chat traffic is, as shown in Fig. 5d. The small amount of traffic from 4.0 to 4.2 hours is due to chat between the operations center and the intelligence cell (Fig. 2).

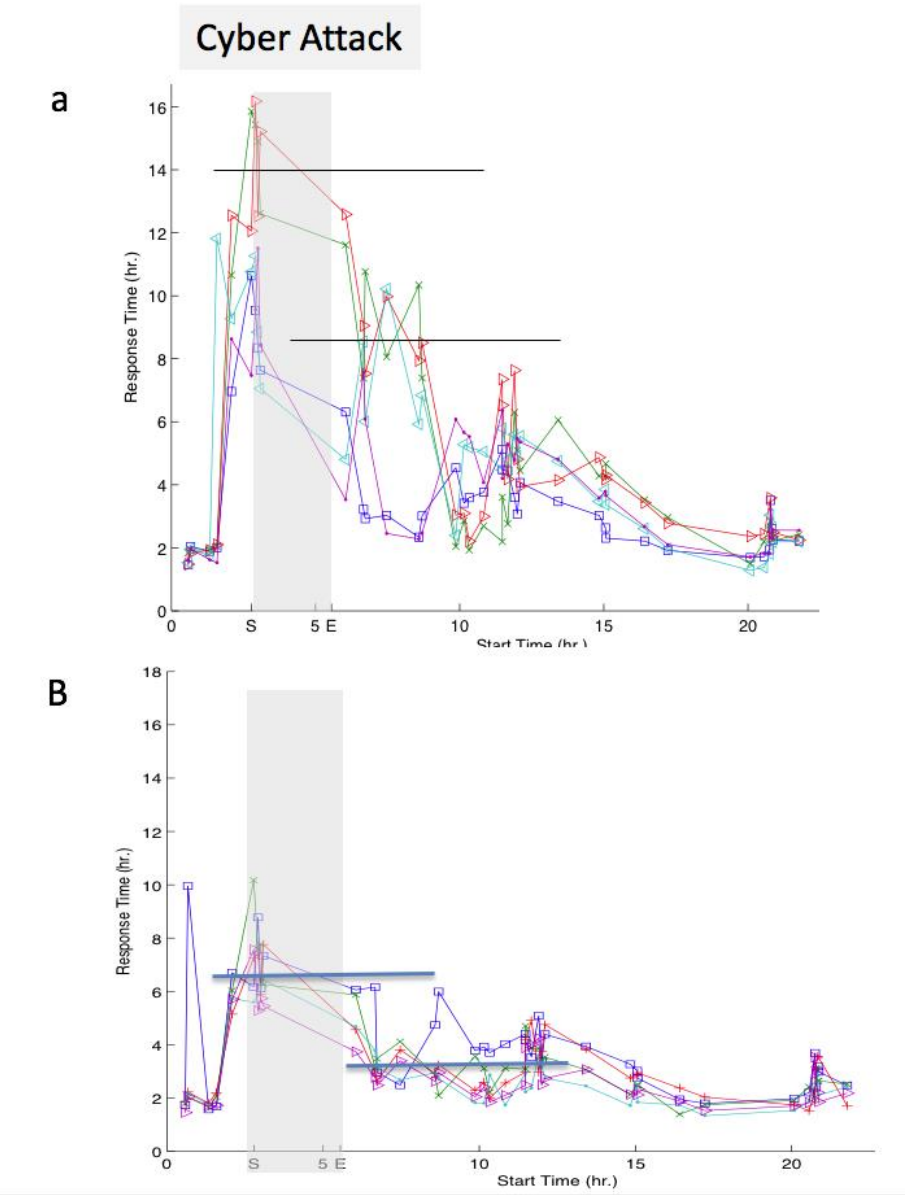

Fig. 4. Five runs with the mission model and virtual operations center operators' actions controlled by ComDOVE on virtual testbed without LARIAT. Operators communicate with the intelligence cell via email. Response time is time through operations center model vs. time critical event starts through operations center model vs. time critical event starts. a. Using Email, b. Using Chat.

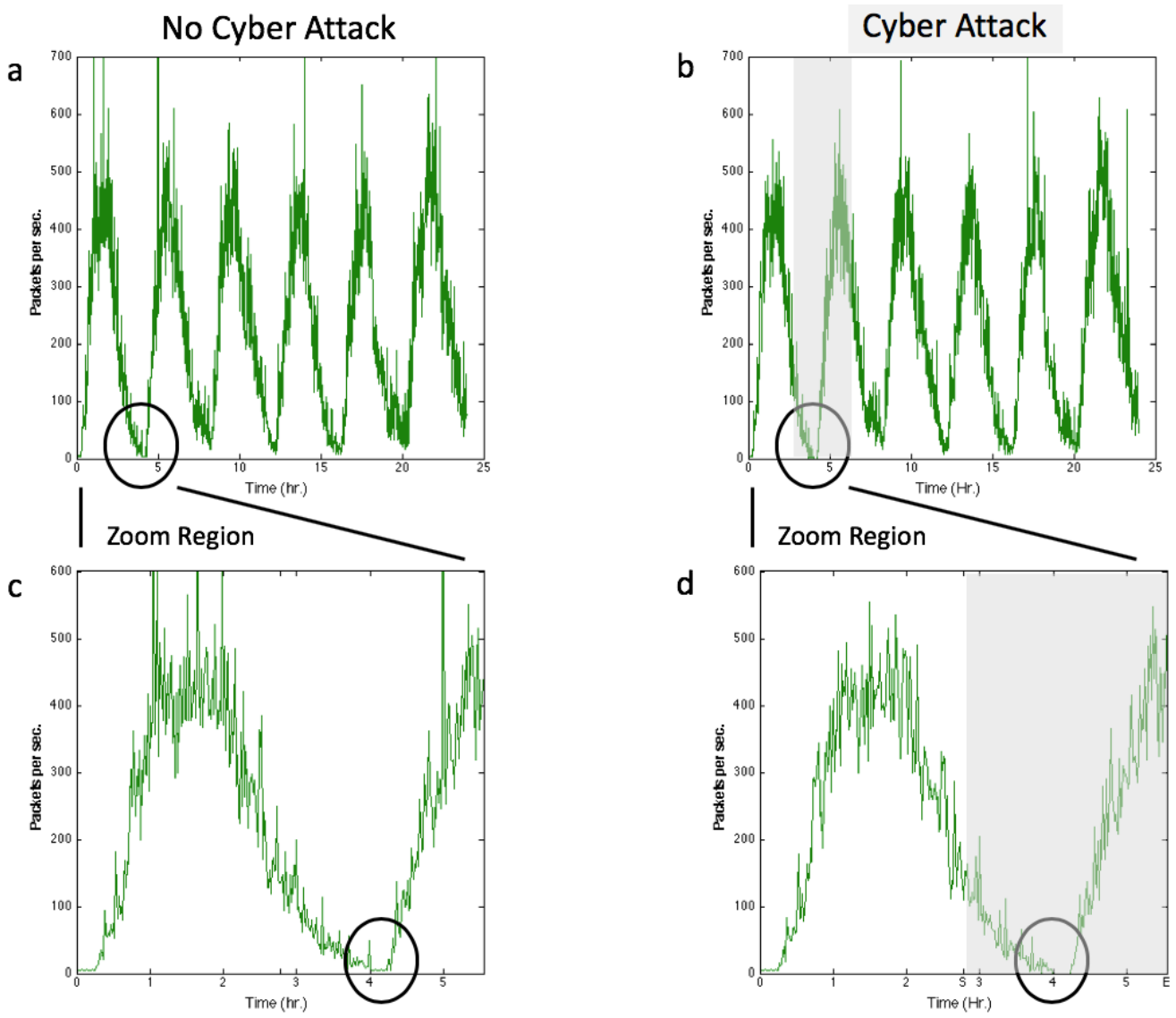

Fig. 5. Full day of LARIAT network traffic without and with the cyber attack (router shutdown).

When the cyber attack is made at 2.8 hours (shaded in Figure 6), the IRC traffic spikes up to 16 packets per minute. This behavior is due to the IRC servers' attempts to synchronize with one another after the link has been severed. There is no activity for the duration of the period the router is disabled. When the link is enabled, the gap between the end of the cyber attack and the restoration of IRC network traffic shows the amount of time taken for the IRC servers to reestablish their connection (marked delay in Fig 6.). During this period, the network is flooded with many types of handshake and other control packets, including IRC server control packets. The amount of time the IRC servers take to reconnect is dependent upon the IRC server implementation and is typically configurable. The traffic spikes again, indicating the chat-servers' successful exchange of synchronization packets.

All the open source network traffic visualization programs tested (wireshark, etherape, and Time-based Network Visualizer (TNV)) were unable to handle the amount of traffic data in this experiment. The tool used was built for this study and enabled a decrease in the amount of data, requiring less memory and making storage and processing more manageable.

3.2 Network Environment

Three VMware ESX servers hosted the experimental environment in a restricted environment. The added security of the restricted environment was desirable, but it was a disadvantage when physical access to a server was necessary. When a router or server was performing poorly, it sometimes took too long to contact an authorized party. In most cases, the solution was to reboot the malfunctioning hardware—a process that could have been completed sooner had the hardware been physically accessible. A similar disadvantage of using shared resources became apparent when disconnections would occur between the ESX

hosts and the VMware Virtual Center, resulting in loss of a virtual machine or loss of privileges. The reasons for these interruptions included regular software maintenance, patching, and administrative miscommunication.

The Windows XP machines in the environment were created using a corporate image with antivirus software and virus scan scheduling policies in place. Each client was scheduled to begin scanning its Network Storage Server (NFS) mounted virtual disk weekly at the same time. The 13 simultaneous scans proved too much for our environment network infrastructure to handle, effectively causing a denial-of-service throughout most of the scheduled scan day. Having no immediate role in experimentation, the antivirus software was disabled from the interface. Evidence of scanning activity was still found after the scheduled scan had been disabled, so the Symantec Endpoint Protection service was disabled via Windows XP Component Services. Testbed environments will have to manage these challenges effectively for maximum productivity.

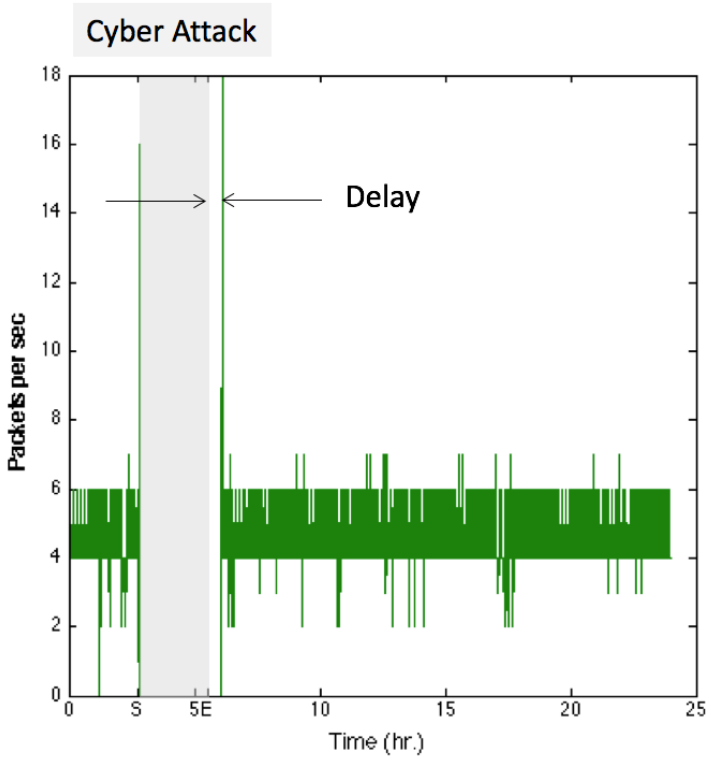

Fig 6. Full day of traffic with only IRC control messages plotted. Gap shows when the router is down.

The use of real software in this distributed system gave us several non-intuitive insights. For example, in one version of our operations center scenario, messages were sent across a router via email, and in another version via chat, using the same PN logic. We expected the subtle replacement of email with chat to have little impact on the simulation, but we found that it required more PN logic to model the behavior of the real system. When the router was down, the Exchange servers would delay email delivery until a connection was reestablished, allowing the simulation to run without adjustment. However, when the router went down in the chat version of the model, all messages sent during that period were lost, forcing us to alter the model to resend those messages once the connection was reestablished. In another example, the simulation was initially designed to model a web voting mechanism, which only took up a small fraction of the process. However, when we used the real software, we found that we would need to model the entire end-to-end event process to accurately emulate the short voting process. As a result, we modeled the software with a web interaction.

5 CONCLUSIONS

Mission-models tend not to model communications and data networks sufficiently to capture cyber effects. Conversely, network emulation environments tend not to provide adequate "hooks" for mapping network and computing components to the operational missions that depend on them. In this study, a hybrid mission-model simulation and virtual environment was integrated. The mission model captured time-sensitive tasks in an operations center and drove 16 virtual analyst actions, including analyst think-time, in the virtual environment. The virtual environment represented the network and other hardware components with realistic network background traffic. In

this study, both mission processes and network effects were represented at appropriate levels of detail so that the impact of a cyber attack, notional denial-of-service, on an operations center could be quantified.

Excessive response time of the network, applications, and humans will result in consequences for mission success. Transaction-based metrics [7] are the same as system-based metrics only in very simple systems. The end-to-end mission response time metric could be applicable to very complicated systems with many transaction types. During this effort, many technical lessons were learned that can be applied to enhance future efforts, including (1) different inherent behavior of chat and email, (2) nuances of measuring end-to-end workflows, (3) dynamics of simulation and real time nature of the experiment, and (4) pitfalls of distributed virtual environment and working with software images. In addition, the environment is designed to support human operators "in the loop" as well as virtual analysts. This proof-of-concept offers promise for mission-oriented cyber analysis and experimentation.

ACKNOWLEDGMENT

The authors wish to thank contributions from Warren Greiff, Sonny Nguyen, Paula Mahoney, Dan Potter, Gary Bogle, Meredith Keybl, Claudette Bishop from The MITRE Corporation; and Bob Cashman, Lee Rossey, Reed Porada, and Chris Connelly from Lincoln Laboratory. This work would not have been possible without the foresight of Jeff Higginson, Deirdre Doherty, and Jeffrey Schavland.

REFERENCES

[1] L.M. Rossey, R.K. Cunningham, D.J. Fried, J.C. Kabek, R.P. Lippmann, J.W. Haines, and M.A. Zissman, "LARIAT: Lincoln Adaptable Real-Time Information Assurance Testbed," *IEEE Aerospace Conference Proceedings*, vol. 6, 2002.

[2] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Design Deployment and Use of the DETER Testbed," *Proc. DETER Community Workshop on Cyber-Security and Test*, available at http://clifford.neuman.name/publications/2007/200708-usecdw-deter-design-deploy/, August 2007.

[3] J. Calvet, C.R. Davis, J.M. Fernandez, W. Guizani, and M. Kaczmarek, "Isolated Virtualized Clusters: Testbeds for High-Risk Security Experimentation and Training," *Proc. CSET '10 Workshop on Cyber Security Experimentation and Test,* 2010.

[4] "National Cyber Range," *DARPA BAA 08-43, Strategic Technology Office*, available at https://www.fbo.gov/index?tab=core&s=opportunity&mode=form&id=aa706e80db08db351ce0f1a321a2a490, 2008.

[5] A. Fercha, "Parallel and Distributed Simulation of Petri Nets," *Tutorial on Performance Tools '95*, September 20, 1995, Heidelberg, Germany.

[6] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Experiences with DETER: A Testbed for Security Research," *Proc. Second Int'l IEEE/Create-Net Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM '06)*, Mar. 2006.

[7] J. Mirkovic, A. Hussain, S. Fahmy, P. Reiher, R.K. Thomas, "Accurately Measuring Denial of Service in Simulation and Testbed Experiments," *IEEE Trans. Dependable and Secure Computing*, vol.6, no.2, pp. 81-95, April-June 2009, doi: 10.1109/TDSC.2008.73, available at http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4689468&isnumber=4912722.

John H. James has a B. Eng. (E.E) from Cornell University (1966), an M. Eng. (E.E.) from Cornell University (1967), and an M.S. (IEOR) from Berkeley University (1972); has worked for TRW Systems, University of California at Berkley, and The MITRE Corporation; has published over 20 papers including a book chapter; current research interests include modeling and simulation and performance of distributed systems; and is a member of the IEEE and the IEEE Computer Society.

Joshua J. Kraunelis received a Bachelor of Science in Computer Science from the University of Massachusetts Lowell in 2010; works for The MITRE Corporation; has research interests in the areas of cyber security, artificial intelligence, and machine learning.

Jennifer J. Mathieu received a BA in Chemistry, Math Minor from Boston University in 1993, MS in Biosystems Engineering from the University of Hawaii in 1996, and PhD in Biological and Environmental Engineering from Cornell University in 2004; has worked for University of Tokyo, NASA-Kennedy Space Center, Northeastern University, and The MITRE Corporation; has published 19 papers, mostly on modeling and simulation; has research interests in social behavioral science data and its use in modeling and simulation, hybrid modeling, and robust decision making; and is a member of American Society of Agricultural and Biological Engineers (ASABE), International Society for Horticultural Science (ISHS), and System Dynamics Society (SDS).

Deborah E. Flagg received a Bachelor of Arts in English from Stanford University in 1971, a Master of Arts in Teaching from Smith College in 1972, and a Certificate of Special Studies in Administration and Management from Harvard University Extension School in 1984.

Tristan M. Lewis received a Bachelor of Science in Computer Science from the University of Massachusetts Lowell in 2006; has worked for iRobot and The MITRE Corporation.

Richard D. Flournoy received a Bachelor of Science in Mechanical Engineering from The Pennsylvania State University in 1985, and a Master of Science in Operations Research from The George Washington University in 1991; has worked for Air Products and Chemicals, Inc., Analytic Services, Inc., and The MITRE Corporation.