Advanced Tools for Cyber Ranges

Timothy M. Braje

In response to the growing number and variety of cyber threats, the government, military, and industry are widely employing network emulation environments for cyber capability testing and cyberwarfare training. These "cyber ranges" have been increasing in size and complexity to model the high-volume network traffic and sophisticated attacks seen on the Internet today. For cyber ranges to operate effectively and efficiently, organizations need tools to automate range operations, increase the fidelity of emulated network traffic, and visualize range activity. Lincoln Laboratory has developed a variety of such tools.

With the recent high-profile cyber attacks on government agencies, such as the U.S. Office of Personnel Management [1], and on companies, including Target, Home Depot, and Sony [2], the dangers of cyber attacks have gained national prominence. Cyber attacks threaten not only the security of personal data but also the national critical infrastructure, for example, power grids and transportation systems [3]. To mitigate the cyber threat, researchers are actively developing cyber defense tools. Before these tools can be deployed in corporate or military networks, they must be tested and validated in realistic environments. Simple tests conducted on developers' computers are insufficient because these tests do not have the required level of realism. Needed are high-fidelity, surrogate networks (i.e., cyber ranges) in which we can introduce attackers, defenders, and defensive and offensive capabilities, and measure the performance of these capabilities in the hands of skilled network defenders pitted against realistic adversaries. To help create and operate these cyber ranges, tools are needed to (1) automate the configuration and generation of complex network environments; (2) create high-fidelity emulated user traffic on these networks; and (3) effectively operate and visualize the rich traffic environment being executed on the range during an event, i.e., a scenario to test capabilities or train personnel.

Cyber Ranges (*heading level 1*)

At the crudest level, cyber ranges are racks of computer hardware. What makes them interesting, however, is their ability to be reconfigured into essentially endless complex network topologies and overlaid with different network traffic profiles. Because cyber ranges are typically disconnected from external networks (and thus have no access to the Internet or to any network resources) to prevent disruptions or damage to live networks during testing and training exercises, all network conditions and activity must be generated from scratch.

As a result, cyber ranges are a scarce, expensive resource. Teams of information technology (IT) staff are required to maintain the hardware and support the events that are executed on the range (e.g., assessing the effectiveness of a software or hardware system). Properly configuring the range for an event can be a daunting task: network machines need to be built, network

routing and defensive tools need to be installed, services to support an event need to be deployed, event-specific traffic generation and applications need to be set up, and, finally, this entire infrastructure needs to be configured. The range community has been scaling the size and capabilities of cyber ranges to more realistically depict the network environment (e.g., by increasing the number of network machines, generating more traffic, configuring additional applications), only complicating the aforementioned tasks. What we gain from this expense and complexity is the ability to perform assessments, experimentation, and training that would not be possible without cyber ranges. It is within this context that Lincoln Laboratory has developed a tool suite to help ease the workload burden on IT staff and to drive costs to a manageable level.

Range Tools (*heading level 1*)

As cyber ranges become larger and more complex and their use becomes more prevalent, the importance of automation and sophisticated tools increases; we need to be able to quickly and accurately build and configure networks and to describe the ranges and events we would like to execute. Once the networks are configured and operational, we need to overlay virtual users that automatically perform the activities of real users to generate simulated network traffic. Finally, we need analysis infrastructure so that we can monitor events as they execute and can examine in great detail the results of those events. Our tools extend automation capabilities, increase environment fidelity, and scale to cyber ranges of both high complexity and very large size. In this article, we discuss the tools we have developed, beginning with our efforts to develop a standard event-description language—an enabling technology for our entire tool suite.

Standardization (*heading level 2*)

In the cyber range business, the data used to describe how the range should be built and configured are typically separate from the data used to describe how the traffic generator should operate. These inconsistent descriptions result in traffic generators having an inaccurate understanding of the range's layout. Consequently, significant time and effort are wasted on reconciling discrepancies. . One straightforward way to avoid this inefficiency is to create a single description language that can be used by all of the tools that participate in a cyber range event. This description language needs to be precise, machine readable, portable, and comprehensive. Lincoln Laboratory has been developing ontologies (called the Common Cyber Event Representation) to describe the network (e.g., hosts, subnets, routing infrastructure, firewall rules, virtual local area networks). We feed data derived from these ontologies into all of our tools, from our Automatic Live Instantiation of a Virtual Environment (ALIVE) application for range build-out to our Lincoln Adaptable Real-time Information Assurance Testbed (LARIAT) application for traffic generation and range control.

Using a common cyber event data source offers more benefits than just a consistent view of the configuration data; it also allows us to perform integrity analysis on our cyber event data before we use any range time. Because cyber ranges are costly to operate and maintain and are relatively scarce, cyber range time is expensive, so catching data integrity issues before the

event begins is very important. We have developed many rules and validation checks that we perform on the event description while it is being developed, giving us a high degree of confidence that, when we deploy the described range, it will operate as expected.

Of course, a standard description language that is only used by the tools of the organization that developed it is not as useful as it could be. If shared by multiple organizations, the description language can enable tool interoperability and reuse. We are actively working with industry partners to develop a standard language that could be adopted by organizations in the cyber range business.

Range Automation (*heading level 2*)

A cyber range useful for a variety of purposes potentially needs to be configured differently for every event. While the hardware often remains the same for each scenario executed on the range, the network topologies, services, and traffic patterns layered on top of that hardware change. Typically, we use virtualization technologies like VMware to build out custom networks for every event. This network churn is a burden on cyber range administrators, who maintain the range hardware and set up custom environments for different events as they are scheduled. Automation tools are essential to relieve this burden. While each vendor (e.g., VMware, HP) has custom software solutions to help build virtual networks, these solutions are usually designed around a single use case with needs that significantly differ from those of a cyber range. As such, these solutions are optimized to repeatedly "stamp out" identical copies of the same network or virtual machine. Tools for rapid network design and reconfiguration are currently lacking.

Lincoln Laboratory has developed ALIVE to fill this gap. ALIVE ingests configuration files from the Common Cyber Event Representation and then automatically and reliably builds out the necessary virtual machines and networking infrastructure to make the network function. ALIVE can create virtualized networks within VMware Elastic Sky X (ESXi)[1], automating most of this network build-out, including the creation of end hosts (clients), routers, firewalls, and many of the servers needed to support interesting traffic generation (e.g., Microsoft Exchange Server, Active Directory). After the operating systems are installed and networking is configured, ALIVE can install on each host other software packages, from web browsers to office applications to email clients and other user software. ALIVE also creates the user accounts that are required for the traffic generators to operate. A typical enterprise network would have its own procedures for generating credentials for new users on the system, but for range events, the virtual users that will be operating on the environment are already known. User accounts are an essential component of the range enterprise environment, and ALIVE can create them in bulk (including Active Directory credentials and Microsoft Exchange mailboxes) as part of the range build-out and configuration.

[1] In the future, additional virtualization backends may be supported.

Emulation Environment (*heading level 2*)

Cyber ranges are disconnected from the Internet; however, most of what we do with computers requires Internet connectivity. Users connect to Facebook, Google Mail (Gmail), and corporate intranet sites, and send email to each other through webmail services or other email hosts (like Exchange). Without access to these services, we cannot make the range come to life with virtual users interacting with dynamic content, applications, and each other as real Internet users would.

To emulate the Internet, we leverage several techniques. We sample 10s of 1000s of sites very shallowly to scrape their content and efficiently and realistically rehost this scraped content by using our custom-written software. Through a similar process, we closely mirror sites so that the emulated users can browse deeply into the sites' content. This content is rehosted with Microsoft's Internet Information Services (IIS) or the Apache HTTP Server. Because the rehosted content is inherently very static, we periodically collect new content. Emulating rich web applications, which constitute the majority of the Internet traffic we see today, is not as straightforward as emulating content. Although we would like to emulate users' interactions with webmail servers like Gmail or Yahoo! Mail, Google and Yahoo are not going to give us their proprietary software and, without an Internet connection, we cannot access these servers directly. Instead, we must choose "surrogate" servers and then carefully model interactions with those surrogates. An open-source alternative, Zimbra Collaboration, allows us to build models for users that interact with a webmail server that we can call Gmail or Yahoo! Mail. While the modeled network traffic will not exactly match real network traffic, the interaction model will be very similar, and for most scenarios, the interactions are the important part of the traffic model. Lastly, we emulate the root Domain Name System structure of the Internet to provide the link between website names and their numeric addresses.

The Internet is not the only service users expect to have. Users access corporate email servers, directory services, websites, and file shares. Within the description of the environment we are building, we include all of these services. ALIVE is able to automatically build and configure many of them. The number and types of services that we deploy are constantly being expanded so that we can create environments of ever-increasing fidelity.

Given a high-fidelity emulation environment, we need to overlay virtual users onto the network so that the network appears as if it is being used by real people. On an actual network, users interact with applications, services, and each other, ultimately producing a rich network traffic environment. It is within this traffic environment that we need to test our tools and capabilities.

Background Traffic (*heading level 2*)

Background traffic is the term we use to describe the normal, random-looking traffic that you would see if you were to inspect the network. It is the by-product of everyday network activities: sending and receiving emails, interacting with content on the Internet, and chatting with friends and coworkers. This traffic affects the way tools work. For example, a network

intrusion-detection tool has a much more difficult time detecting malicious traffic within background traffic environments (normal traffic is commonly misidentified as malicious) than it does within "clean" environments in which only malicious traffic is present. To create high-fidelity testing environments for cyber range tools, we need to emulate the constant network activity that normal users produce. This background traffic also covers malicious traffic that is introduced onto a network, as oftentimes attackers hide their activity within the background.

There are several techniques for generating network traffic. Commercial solutions, such as Ixia's BreakingPoint, create realistic, packet-level traffic (i.e., streams of bits on the network) [4]. These techniques involve either replaying network packets or generating streams of bits on the network that emulate specific protocols. They are highly scalable, are relatively simple to add new traffic types to, and have sufficient fidelity for many scenarios, including those in which you want to push as many bits as possible across a link or through a piece of software. BreakingPoint is designed to efficiently generate this high-bit-rate traffic with a variety of network protocols, and we have found it useful for augmenting our background Internet traffic to increase traffic volume and protocol variety.

Instead of building a protocol emulator, Lincoln Laboratory is building a different kind of traffic generator—one that generates traffic that is tailored to real, specific user-application interactions. We hook into (i.e., programmatically control) existing installed applications on behalf of each virtual user in the emulated network, making them automatically perform their actions and, as a by-product, produce network traffic similar to that produced by a real user. This approach has several advantages over protocol emulation:

1. Each and every user interaction generates traffic in the same way a real user would, including second- and third-order effects (e.g., a Domain Name System lookup caused by a website visit).
2. Because our virtual users are interacting with real applications, they can click on malicious links, download compromised files, and carry out other actions that real users will inevitably perform on a network.
3. Unlike packet generators, traffic generators can provide real targets for malicious code propagation and endpoints for attackers to leverage for further attacks within the network.

This level of fidelity comes at the costs of increased complexity and smaller network sizes. For every traffic generator, the need for a fully configured operating system reduces the amount of traffic that can be produced for a given set of hardware. The events that we have designed LARIAT to support (e.g., red team [offense]/blue team [defense] exercises, evaluations of complex network tools) require this level of fidelity to allow for realistic attack propagation.

Blue Traffic (*heading level 3*)

A significant part of LARIAT is its actuation capability, which allows the system to realistically interact with applications that real users would have installed on their computers. For blue users, LARIAT contains actuators (i.e., application emulations) for standard user software, such

as office applications, mail clients, and web browsers. Using these kinds of software, virtual users can generate and edit documents, send emails to each other, and interact with web content and web applications. By finding programmatic hooks into user applications, LARIAT builds a model of the software and automatically executes the actions that a user would perform when interacting with the software. These same programmatic hooks that are used to control the applications' behavior also allow LARIAT to receive feedback from the software with which it interacts.

Many applications, however, are not controllable in this way. For those cases, we use image-processing techniques on the video output from the virtual user's machine to recognize available actions that can be performed on an application. Then, keyboard or mouse commands are sent to that application to make it perform its actions. For example, in order to browse to a website, we would use image-processing techniques to find the location of the URL bar, send mouse move commands to position the cursor at the correct place on the screen, send a mouse click command to bring the URL bar into focus, and then send keyboard click commands to type the URL. We have developed an actuator that works remotely by interacting with keyboard, video, and mouse (KVM) devices or through a virtual network computing connection. Using either of these connection types, this actuator (KVM-based 0 Artifact LARIAT Actuator, or K0ALA) interacts with applications in much the same way a real user would by recognizing relevant images from a video stream and then performing keyboard or mouse actions at those image locations. In many ways, this means of interacting with the application provides an even more realistic application interaction model than the one produced by typical LARIAT actuators.

Realizing we will be unable to build all actuators of interest to the cyber range community, we are also building a platform into which actuators can be plugged. Our actuation system in no way requires upfront knowledge of all the actuators that may be used within an event. We provide hooks for programmers to dynamically register their custom actuators to seamlessly work within our environment. In fact, we build our own actuators in this way so that we can refine our processes and application program interfaces. In particular, K0ALA provides a visual scripting language with which range developers who are interested in building interactions with applications can capture the necessary images and register the appropriate actions against those images; these actions can then be assembled into larger scripts that describe the application interaction model.

Red Traffic (*heading level 3*)

Many uses of cyber ranges involve testing offensive and defensive tools, or running red-on-blue exercises (Figure 1). Adversarial traffic is absolutely essential for creating a realistic environment for these events. This traffic is used not only as a cover for live red teams to help assess the stealth of their teams or their tools but also as a base level of attacks that the defensive tools must protect against. Malicious traffic has a different character from that of blue traffic. In many ways, it can look like normal system administrator traffic, with attackers scanning

computer ports, creating accounts, changing passwords, and installing software. Attackers also engage in more obviously malicious actions, such as creating botnets, performing network reconnaissance, and pivoting from host to host. Lincoln Laboratory has been developing an automated capability, the Lincoln Laboratory Attack Framework, to generate these kinds of malicious activities, including many of the exploits provided in Metasploit, a network-penetration testing software suite [5]. Generating coordinated attacks against blue networks, this framework provides a relatively large-scale, fairly sophisticated array of attacks that would be encountered in real environments.

FIGURE 1. During a red/blue exercise held at Lincoln Laboratory, members of the blue team look through data gathered by their defensive tools to tease out signatures of network attackers—both LARIAT virtual users and members of the live red team. The network defenders are from different Cyber Protection Teams, which are being created by the U.S. Cyber Command to help companies and government agencies defend their networks from cyber attacks.

User Modeling (*heading level 2*)

To emulate real network users, we need models for many kinds of users with different behaviors; at the same time, we need a modeling engine that is both simple and powerful so that general user behaviors can be described and easily encoded within the system. Fulfilling both of these requirements is particularly challenging because the behavior descriptions must be distributed across potentially 10s or 100s of 1000s of virtual users on a large network; thus, the description language must characterize many user behaviors in a succinct but precise

manner. Additionally, the execution of these models needs to be mostly self-contained and autonomous. We will be unable to scale a modeling architecture that requires a single master server to dole out actions to each virtual user; once a size threshold is met, the single master server cannot keep up with the workload. We must find other ways to build models in which the users coordinate actions to achieve a common goal.

User Modeling Basics (*heading level 3*)

LARIAT comes with a modeling engine that is provided separately from the actuators. The modeling engine is a language that allows us to aggregate our actuator actions into simple models, aggregate those simple models into larger models, and then build virtual users that are configured to use different aggregations of these interaction models. We decouple the modeling capability from our actuators, keeping us from mixing modeling and actuation logic and providing us with the ability to more easily integrate actuators written by others and to build single models that mix actions from different actuators. For example, we can combine actuator actions and build simple models of what it means to compose a Microsoft Word document or to randomly surf the Internet. We can take those models and aggregate them into more interesting models for surfing the Internet for some interesting facts on a particular topic and then feed those facts into the document we are creating. We could then vary how we combine these actions to make different models of what we could call an analyst, intelligence officer, or other type of user.

In addition to having these aggregation and composition capabilities, the modeling language can automatically interact with the environment, detecting and responding to failures. Consider the case of a corporate Microsoft Exchange Server going down: a user who had intended to use the server to send an email could use a webmail service instead. The modeling language also automatically handles the selection of specific applications needed to accomplish tasks (e.g., choosing Chrome, Firefox, or Internet Explorer when given a model of a web browser). Perhaps most importantly, the engine provides several developer conveniences, such as automatic handling of error propagation.

Mission Modeling (*heading level 3*)

Once we have established a modeling capability that supports random (but semi-intelligent) background traffic, the next level of interesting user behavior is mission modeling. Missions are coordinated actions among several virtual users that, in aggregate, achieve one large goal—for example, several agents at an air operations center are working to produce a portion of the daily air tasking order,[2] which needs to be sent to a commander for assembly into the final order [6]. We have just begun to model these kinds of missions and are researching ways to express coordinated actions within the modeling engine. We can already model simple coordinated tasks like the one above, but we are interested in expanding the fidelity and increasing the complexity of the models we can build.

[2] An air tasking order is a document created by an air operations center that has command and control of a particular theater. The document outlines how airpower will be used over a 24-hour period.

For missions that need to be very precisely controlled, we have prototyped a scripting capability that allows the author of a model to specify actions that should occur at a given time or within a certain time interval of another action. This scripting capability is currently fairly limited, but already we have used it to describe models of malicious actors working within an organization to sell the secrets of that organization.

Event Operations (*heading level 2*)

Given tools to precisely specify an event, automatically build out the cyber range based on the specification, and generate realistic network traffic, we still need to execute the event. LARIAT provides a graphical user interface (Figure 2) that helps with this task. This interface guides the range operator through the workflow of configuring the virtual users with the data needed to execute their behaviors, validating that the configuration is correct, and then starting and stopping traffic. While necessary, these functions are clearly not sufficient for comprehensive situational awareness of an event. Range operators running the event need to be able to build and maintain an accurate understanding of the current states of potentially many 1000s of machines, users, and traffic flows. An easy-to-understand visualization of the virtual user (or even of the host that the virtual user executes its actions on) states can help range operators understand their events to the level necessary. Additionally, event operators want to perform analyses of the event either during its execution or afterwards in order to measure the effectiveness of the event.

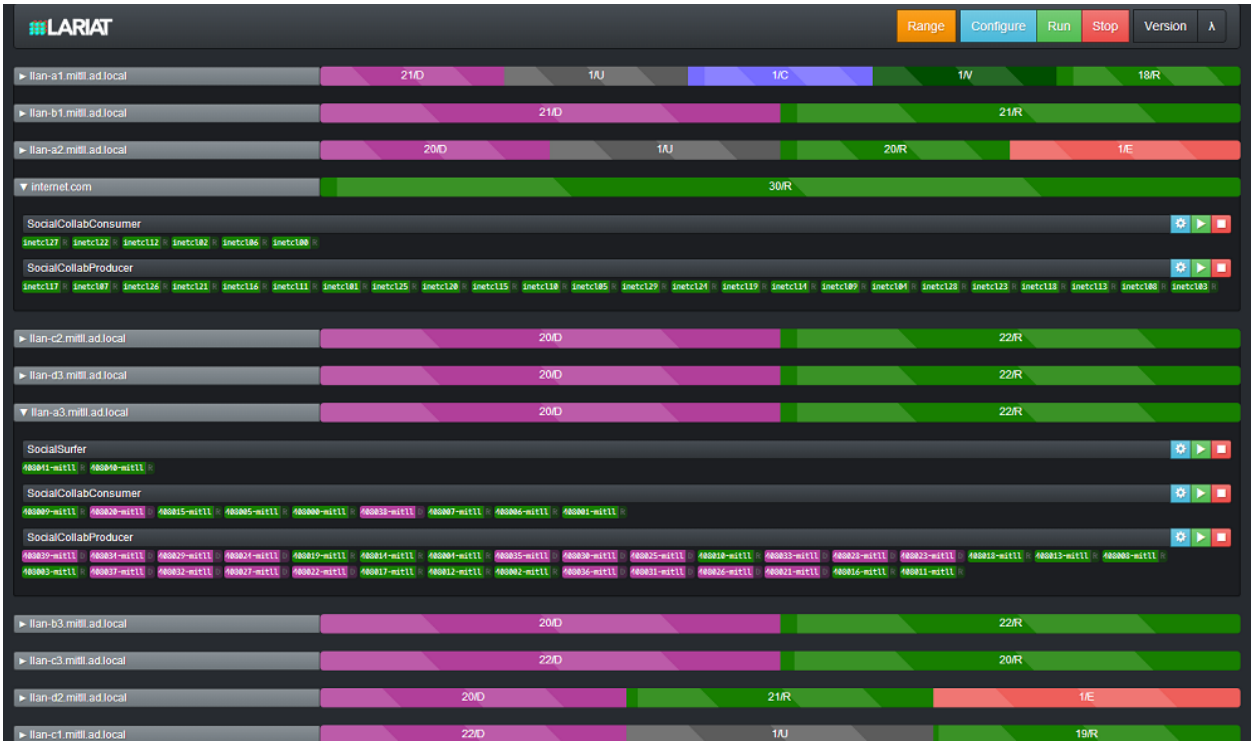

FIGURE 2. Each gray bar (most of which are collapsed) on LARIAT's graphical user interface represents a subnet (e.g., llan-c2.mitll.ad.local). Roll-up summaries show the statuses of the virtual users within that subnet; on the top row, the fuchsia bar (21/D) indicates that 21 users are currently unresponsive, the gray bar (1/U) specifies the one user that has never been heard from, the purple bar (1/C) represents one user in the configured state, the dark green bar

(1/V) shows one host that is ready to start running, and the light-green bar (15/R) represents 15 users that are running as expected. Expanding out a subnet view shows details at the user type or individual host level. For example, two user types are shown in the expanded view of the internet.com subnet: SocialCollabConsumer and SocialCollabProducer, with the individual users listed below them. The play, stop, and send configuration buttons allow the operator to control the operation of virtual users by sending them configuration data or commands to start or stop traffic.

Command and Control (*heading level 3*)

As ranges become larger, more intricate, and further distributed, we need a lightweight, scalable command-and-control (C2) system to operate the traffic generators. Simultaneously, we need to monitor in real time and with high accuracy how these traffic generators are performing and fix any errors that may arise. To avoid the latency introduced by the request-response cycles of synchronous C2 systems and to help us achieve the scalability requirements, we have built an asynchronous C2 system. However, because the asynchronous system does not provide immediate feedback from the virtual users under a range operator's control, the status monitoring problem is more difficult. We are able to achieve near-real-time reporting on the health and status of the traffic generator by using a messaging protocol, which analyzes messages as they periodically arrive from the virtual users. When we detect that a virtual user is unresponsive, we can take steps to fix the issue or, at the very least, notify the range operators that there is a problem.

Our C2 system works by pushing data to the virtual users when they need the information. The server "knows" what these users need for configuration and state changes (i.e., whether they should be running traffic or not). Virtual users continuously report to the server a signal that indicates whether they (a) have received the correct configuration and (b) are in the correct execution state. As the server detects inconsistencies, it may send out either updated configurations or other C2 messages to transition the virtual user into the appropriate state.

Because this C2 system is built around a loosely coupled, asynchronous messaging protocol, it is easy for organizations other than Lincoln Laboratory to augment LARIAT's capabilities by adding their own components (e.g., actuators) into LARIAT. A very near-term goal for the LARIAT development team is to break out the necessary components of this C2 system into a separate module that has very clear integration points for third parties. Then, a simple integration path could be created for traffic generators that are not built at Lincoln Laboratory.

Visualization and Analytics (*heading level 3*)

To help range operators build the necessary mental model of the entire range, we provide a visualization of the range state. The visualization shows the virtual user workflow states so that range operators understand if and when the virtual users are ready to start execution. These workflow states progress as follows:

1. There is no indication that the virtual user is available (i.e., before LARIAT installation).
2. The virtual user checks in at some point in time.
3. The virtual user is configured with a behavior model and ready to start executing.

Additionally, separate from the workflow state, virtual users are either responsive or unresponsive, determined by whether they have checked in recently. We give range operators a way to quickly determine how traffic is running and what, if any, parts of the range need to be fixed.

The fairly high-level status reporting and visualization described above is for a single virtual user. We have also built aggregate visualizations of large portions of the virtual users within the network so that the range operator can, for example, see where network traffic is flowing. The process for building visualizations begins with each actuator logging its actions as it performs them. These logs are then sent to a centralized server that stores them and makes them available for analysis. Using these data, we can create real-time graphs of, for example, the number of successful and unsuccessful website navigation attempts (Figure 3). Too many failed navigation attempts could indicate to the range operator that there is a problem with the web servers or the routers that allow traffic to flow through them. We provide a range of out-of-the-box queries and visualizations for actuator data but also allow users to write custom queries against the same data so they can monitor the activities that are most relevant to their events.

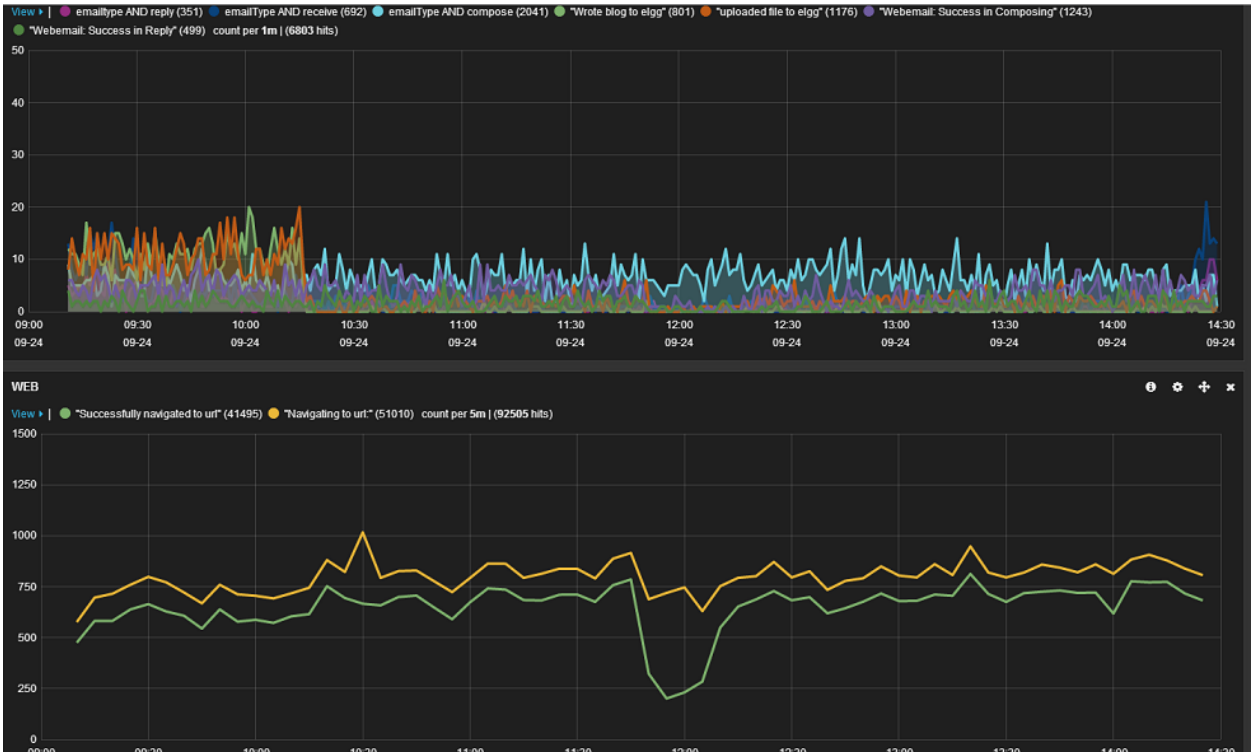

FIGURE 3. The LARIAT network traffic seen in the above visualization was produced during one day of a red/blue exercise hosted at the Laboratory. The top graph plots the counts of virtual users' actions as a function of time. For example, several users were uploading images to a social networking site (orange line) at the beginning of the exercise but this activity drops off drastically after an hour or so. Other actions include replying to an email (fuchsia), composing an email (light blue), and writing a blog post on the social networking site (green). Shown in the lower plot are counts over time of successful (green) and attempted (yellow) website navigation instances. About halfway through the plot, the number of successful navigations to the website plummets, perhaps because the web server became overloaded or a router was misconfigured.

Future Work (*heading level 1*)

We intend to continue driving toward increased range fidelity and to build more sophisticated tools for range operators to monitor the health and status of the range. Specifically, we will enhance our modeling engine with features that allow for more complex interactions with the environment, such as responding to dynamic stimuli (e.g., messaging windows popping up on the screen). Ultimately, we want to create mission activities that describe coordinated user actions and are woven into the normal background traffic. We will also be supporting additional actuator types so that we have more variation in our virtual users. Finally, we will augment our range introspection capabilities, provide better analytics, and develop more visualizations of the emulated-user log data to make the jobs of range operators and event analysts easier.

References

1. J. Sciutto, "OPM Government Data Breach Impacted 21.5 Million," CNN website, 10 Jul. 2015, available at http://www.cnn.com/2015/07/09/politics/office-of-personnel-management-data-breach-20-million/.
2. K. Granville, "9 Recent Cyberattacks Against Big Businesses," *The New York Times*, 5 Feb. 2015, available at http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?_r=0.
3. "Cyber Warfare: Sabotaging the System," CBS News website, 6 Nov. 2009, available at http://www.cbsnews.com/news/cyber-war-sabotaging-the-system-06-11-2009/.
4. "Network Testing with Simulated Traffic. Does Realism Matter?" An Ixia BreakingPoint Case Study, White Paper 915-3128-01, Rev. C., Aug. 2014.
5. Metasploit company website, available at http://www.metasploit.com/.
6. J. Mathieu, J. Melhuish, J. James, P. Mahoney, L. Boiney, and B. White, "Multi-scale Modeling of the Air Operations Center," The MITRE Corporation, Technical Papers, November 2007, available at http://www.mitre.org/sites/default/files/pdf/06_1497.pdf.

About the Author

Timothy M. Braje is a technical staff member in Lincoln Laboratory's Secure Resilient Systems and Technology Group, where he works on adaptive computing platforms and quantum computing algorithms. He also has interests in functional programming, formal methods for

verifying software systems, programming languages, and constructive theorem proving. Between 2009 and 2015, he led the effort to architect and build the Laboratory's next-generation advanced cyber tools platform, including tools for range control, visualization, and traffic generation. Prior to joining the Laboratory in 2009, he worked at MyVest, Solidware Technologies, and Coverity, building systems to help automate financial investment management and to help software developers analyze and improve the quality of their products. He holds a bachelor's degree in physics from the University of Florida and a doctoral degree in physics from Stanford University.