Chapter 13

USING AN EMULATION TESTBED FOR OPERATIONAL CYBER SECURITY EXERCISES

Christos Siaterlis, Andres Perez-Garcia and Marcelo Masera

Abstract The detection, coordination and response capabilities of critical infrastructure operators ultimately determine the economic and societal impact of infrastructure disruptions. Operational cyber security exercises are an important element of preparedness activities. Emulation testbeds are a promising approach for conducting multi-party operational cyber exercises. This paper demonstrates how an Emulab-based testbed can be adapted to meet the requirements of operational exercises and human-in-the-loop testing. Three key aspects are considered: (i) enabling secure and remote access by multiple participants; (ii) supporting voice communications during exercises by simulating a public switched telephone network; and (iii) providing exercise moderators with a feature-rich monitoring interface. An exercise scenario involving a man-in-the-middle attack on the Border Gateway Protocol (BGP) is presented to demonstrate the utility of the emulation testbed.

Keywords: Cyber security exercises, network security, emulation testbed

1. Introduction

The increasing dependence of critical infrastructures on information and communications technologies is a growing area of concern. Contingencies that involve abnormal events and disruptions – deliberate (e.g., cyber attacks) or unintentional (e.g., fiber cable cuts) – can result in dire consequences if critical infrastructure operators fail to react promptly, appropriately and effectively. Therefore, in the context of incident preparedness, it is important that the procedures performed during contingencies are carefully planned and tested in advance at the conceptual and technical levels. These activities can reveal vital details that could negatively affect incident detection, coordination and response capabilities.

The execution of cyber security exercises has been identified as a priority at the national [18] and international levels [2]. The U.S. Homeland Security Exercise and Evaluation Program (HSEEP) [3] identifies two main types of exercises: discussion-based exercises and operations-based exercises. Operations-based exercises provide valuable information about the behavior of operators during security incidents, including response times and levels of coordination. These exercises often engage the red team/blue team paradigm (i.e., the use of an attacking team and a defending team) to ascertain the security of a system or network [1, 12, 15]. Our approach diverges from this paradigm in that it focuses on cyber security exercises involving multiple stakeholders from different administrative domains with the primary objective of examining their coordination capabilities. Such exercises are of particular interest due to the distributed, global and privately-owned characteristics of the Internet infrastructure. In the case of multinational exercises, private infrastructure owners (e.g., network service providers (NSPs)) are the principal actors, but are typically competitors; moreover, the notion of a governing entity (e.g., public sector) is hard to define.

Conducting exercises using production systems raises concerns about the potential side-effects to mission-critical systems and services. Software-based simulation has been proposed as a solution [9, 10], but it is limited by the fact that operator behavior can be altered significantly if the exercise platform lacks realism. The third option, hardware-based emulation, is a good candidate because it combines realism and flexibility.

This paper discusses how an emulation testbed, specifically one based on the Emulab software [5], can serve as a platform for executing multi-party, operational cyber security exercises. An Emulab testbed can recreate a wide range of experimentation environments that support the development, debugging and evaluation of complex systems [4, 11]. In the context of cyber security exercises, the DETER testbed [13] has been used in the well-known Cyber Storm exercise to provide visual inputs to participants and help them understand the effects of attacks. We adopt a similar approach and investigate how an Emulab testbed can be adapted for multi-party cyber security exercises. In particular, we identify the missing elements and functionality needed to conduct robust cyber security exercises. The effectiveness of the approach is demonstrated using an exercise with multiple network operators involving a man-in-the-middle attack on the Border Gateway Protocol (BGP).

2. Operational Exercise Components

Cyber security exercises seek to raise the level of preparedness by confronting participants with artificial events and studying their reactions. Six main elements must be considered when designing an exercise:

- **Participants (Who):** Participants come from the government, private sector, media, etc. They have diverse roles such as players, observers, etc.
- **Location (Where):** Participants may be in the same physical location or may participate remotely.

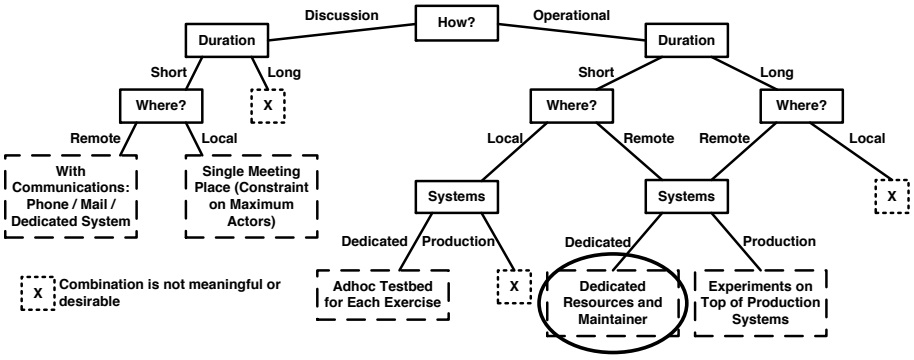

Figure 1. Design options for cyber security exercises.

- **Time and Duration (When):** Exercises may last from a few hours to several days or weeks.

- **Objectives (Why):** Exercise objectives can vary widely, e.g., testing recovery procedures and coordination capabilities.

- **Type (How):** Exercises can be grouped into two main categories: discussion-based exercises and operations-based exercises.

- **Scenario (What):** A scenario typically consists of a storyline of events, master event list and contextual information. A scenario usually incorporates the assets, vulnerabilities and asset topologies, hazards and potential adversaries, threats leading to attacks, attacks and their consequences (physical, psychological, etc.), and countermeasures and response actions.

Figure 1 presents the various exercise design options. Note that our focus is on operational exercises with remote participants. The exercise elements are not independent of each other. In Figure 1, local operational exercises, both long and short on production systems, are marked as not meaningful. The first option would require participants to be away from their offices for long periods of time. The second option is not feasible in most cases because production systems are located in specific facilities and are not easily accessible.

As mentioned above, this paper focuses on operational exercises that involve multiple remote participants. The objective is to involve key critical infrastructure stakeholders (mainly owners and operators of private infrastructures) at the operational and practical levels in order to assess the communication and coordination of operators during contingencies.

Four roughly sequential phases are involved in an exercise: design, setup, execution and analysis. Each phase presents its own challenges, which will be discussed individually after introducing the main concepts underlying an emulation testbed.

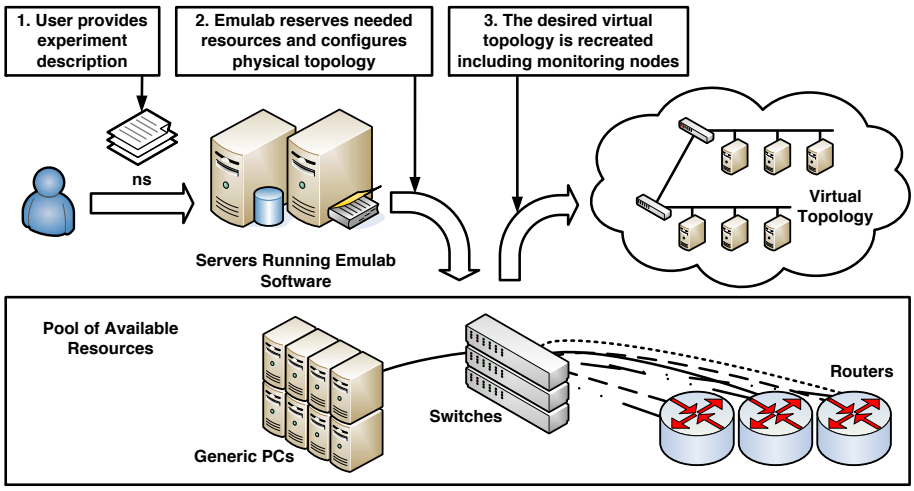

Figure 2. Recreating a virtual network configuration.

3. Emulab Overview

Using an emulation testbed is one of the most promising approaches for experimenting with large and complex systems. Pure software simulation is often too simplistic to recreate complex environments. Using an *ad hoc* testbed is not recommended because it can be time-consuming and error-prone to setup, maintain and modify. Consequently, emulation testbeds like Emulab [20] are becoming more popular. They are an attractive option for conducting cyber security exercises because they can support human-in-the-loop experimentation.

An Emulab testbed typically consists of two servers that run the Emulab software (named `boss` and `ops`) and a pool of physical resources (e.g., generic personal computers and network devices) used as experimental nodes. The Emulab software permits the automatic and dynamic mapping of physical components (e.g., servers and switches) to a virtual topology. In other words, it configures the physical components so that they emulate the virtual topology as transparently as possible. Thus, significant advantages are provided in terms of the repeatability, scalability and controllability of experiments.

Figure 2 shows the main steps involved in recreating a virtual network configuration in an Emulab testbed. The following steps are involved:

- A detailed description of the virtual network configuration is created using Emulab's experiment script, which is based on the TCL language with ns-2 and testbed-specific extensions.

- In the description, similar components are designated as different instances of the same component type. Consequently, templates of common components (e.g., Linux DNS server) can be easily reused and automatically deployed and configured.

- An experiment is instantiated using the Emulab software. The Emulab server automatically reserves and allocates the required physical resources from the pool of available components. This procedure is called "swap-in," in contrast with the termination of the experiment, which is called "swap-out."
- The software configures the network switches to recreate the virtual topology by connecting experimental nodes using multiple VLANs.
- In the final step before experimentation, the software configures packet captures at predefined links for monitoring purposes.

4. Configuring Emulab

An Emulab testbed is easily configured to support operational exercises. This section describes the steps involved in the design, setup, execution and analysis phases.

4.1 Design Phase

After determining the participants and developing a detailed scenario, the following tasks are performed:

- The network topology is described using an experiment script. The components (e.g., servers and routers) to be controlled by the participants are differentiated from the components that simulate the rest of the world (i.e., the context). For example, participants would not have direct access to nodes that generate background traffic (e.g., by replaying real traffic dumps [19]). All the components should be based on reusable templates, which reduces the costs involved in organizing exercises.
- The exercise scenario is described using the experiment script (e.g., scenario injects are represented as scheduled or dynamic events). For example, a fiber cable cut could be scheduled by introducing a "link-down" event. Emulab's event scheduling mechanism supports the execution of the scenario in real time instead of simulated time. However, it is important to consider the need to pause exercise execution because swapping-out the experiment can cause scheduled events to be replayed.
- The exercise monitoring infrastructure is described and the data collection mechanisms are configured.

4.2 Setup Phase

In the setup phase, the predefined systems are instantiated and configured as in any Emulab experiment. Exercise participants are given access to individual experimental nodes. Access control mechanisms are used to ensure that participants may only access the nodes "owned" by them in the scenario. However, exercise moderators are permitted to access all resources.

As an example, consider the case of a participant representing a network service provider. This participant would be given access to two logical nodes. The first is a router that implements the service provider's routing policy. The second node is a companion management host, on which the participant can install custom tools and scripts used in daily operations. Obviously, preparing the exercise platform in such a manner is important to conducting realistic operational exercises.

4.3 Execution Phase

During the exercise execution phase, the participants interact with the systems and among themselves. Their actions are monitored for further analysis after the end of the exercise (analysis phase). It is important that exercise moderators know how the exercise is evolving so that they can intervene (e.g., by injecting dynamic events) if necessary.

4.4 Analysis Phase

In the analysis phase, the emulation testbed is used to gather recorded data. The data is used to evaluate the response times, durations of actions, levels of coordination, etc. The data collected depends on the scope of the exercise.

5. Challenges

Based on the analysis above, three principal challenges must be addressed in order to use an Emulab testbed for operational exercises:

- **Multiple Remote Users:** The Emulab architecture supports multiple remote users, but does not provide secure access. To address this challenge, we propose the use of VPN connections for secure remote access.

- **Realistic Environment:** Using an emulation testbed addresses the need to recreate IP networks. However, it is necessary to integrate a simulated and monitored public switched telephone network (PSTN) that could be used by participants to communicate and coordinate their activities. This means that telephone calls must be supported as in the real world.

- **Flexible and Automated Monitoring:** Emulab offers limited functionality beyond link-tracing (i.e., packet capture of network traffic). Also, it lacks support for measuring individual node metrics (e.g., CPU utilization) and does not provide a user-friendly monitoring GUI. Therefore, we have integrated Zabbix, a powerful open-source network monitoring application, with the Emulab architecture, enabling the automated monitoring of experimental nodes.

These challenges are discussed in more detail in the following sections.

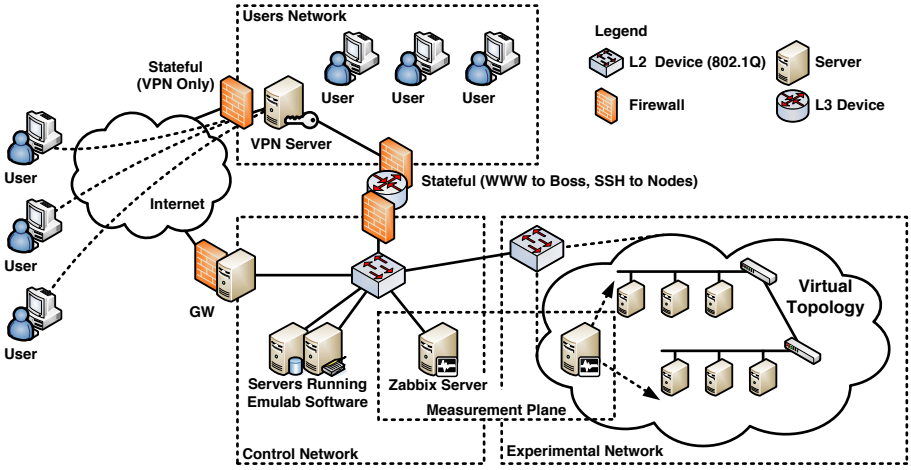

Figure 3. Emulab testbed with secure remote access.

5.1 Secure Remote Access Architecture

The organization of multi-party exercises is simplified by allowing geographically distributed participants to remotely access the exercise platform. Our secure remote access architecture essentially isolates the testbed by allowing remote access only through VPN connections (Figure 3). An OpenVPN server enables remote users to connect securely through a public network such as the Internet. The confidentiality and integrity of transmitted information is ensured by tunneling protocols and encryption algorithms. Non-interference between participants is implemented via the "no client-to-client" configuration of the OpenVPN server.

All remote users authenticate themselves with the VPN server, which is protected by a firewall. Once a user is connected to the "users network," an internal firewall guarantees that access is available only to the required resources (typically `www` to `boss` and `ssh` to nodes). Also, a user cannot reuse a new connection (IP within the remote users network) to reach the Internet. This architecture provides remote users with access to the platform, but not to the Internet. The only access to the Internet from the testbed is via an authenticated proxy (GW in Figure 3), which is restricted to platform administrators.

Having two layers of firewalls provides a high level of security and facilitates the specification of access policies from the remote user network to internal resources. Of course, this architecture is by no means optimal or unique, but it is presented as a reference implementation for secure remote access to the testbed. Of course, the architecture complements any security enhancements provided by the Emulab architecture [8].

5.2 Support for Voice and Data Networks

An operational exercise platform should provide a realistic environment without any artificial features that could alter participant behavior. Unlike a network simulator [10], an emulation testbed addresses this issue because participants can interact with a realistic IP network that could be based on real routers or on software routers like Quagga. Although software routers are not a replacement for real routers, they permit the testbed to scale to much larger topologies while maintaining a certain degree of fidelity. An important detail is that by having the exercise platform isolated from the Internet, real-world IP addresses, autonomous systems (ASs), etc. can be safely reused in the experimental network, which increases the realism perceived by the participants.

The exercise platform is extended to simulate a public switched telephone network as in the real world [17]. The extension uses a separate logical network consisting of a central VoIP server and several clients (e.g., soft phones), which are handed to the participants. Our testbed uses an Asterisk server with FreeBSD 8, which is automatically configured by launching scripts from Emulab. However, for the purposes of an exercise, this functionality is augmented with call recording so that communications between participants can be logged for subsequent analysis. A VoIP server like Asterisk also facilitates monitoring, because it supports the capture of two-party calls in separate files (after multiplexing both voice streams).

5.3 Improving Network Monitoring Support

Permitting exercise moderators to monitor the execution of a complex cyber security exercise in real time enables them to intervene when needed and to properly simulate non-participating entities. Although some work on extending Emulab's monitoring capabilities has been performed (e.g., SEER [16]), the integration of Emulab with general purpose network monitoring software can guarantee more frequent updates, more functionality and support by a wider community.

For these reasons, we have integrated Zabbix, which offers advanced monitoring, alerting and visualization features in a scalable and automated manner. First, we created a template operating system image of a Zabbix server. The experimental nodes can either have a pre-installed Zabbix agent or have the agent installed at runtime, e.g., using the `tb-set-node-tarfiles` command. The Zabbix server runs on a separate node and communicates with the agents using the control network to avoid interference with the experiment and to ensure communications with agents despite potential disruptions in the experimental plane (part of the exercise scenario).

The challenge is to automatically configure the server to monitor all the agents. This is because the IP addresses of the nodes allocated to an experiment are not known *a priori*. We address this issue by including custom `emulab_mon` code in Emulab's experimental script, which specifies the nodes

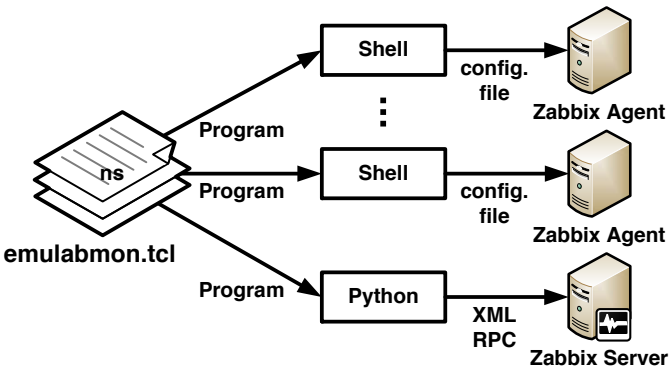

Figure 4. Zabbix auto-configuration processes.

that are to be monitored and the Zabbix templates to which they should be attached (Figure 4). At swapping-in time, the code calls shell scripts on all the monitored nodes to configure the agents, and invokes a Python script that configures the Zabbix server using its built-in XML-RPC API. This API is in its infancy and, therefore, Zabbix server version 1.8 is the minimum version that can be used.

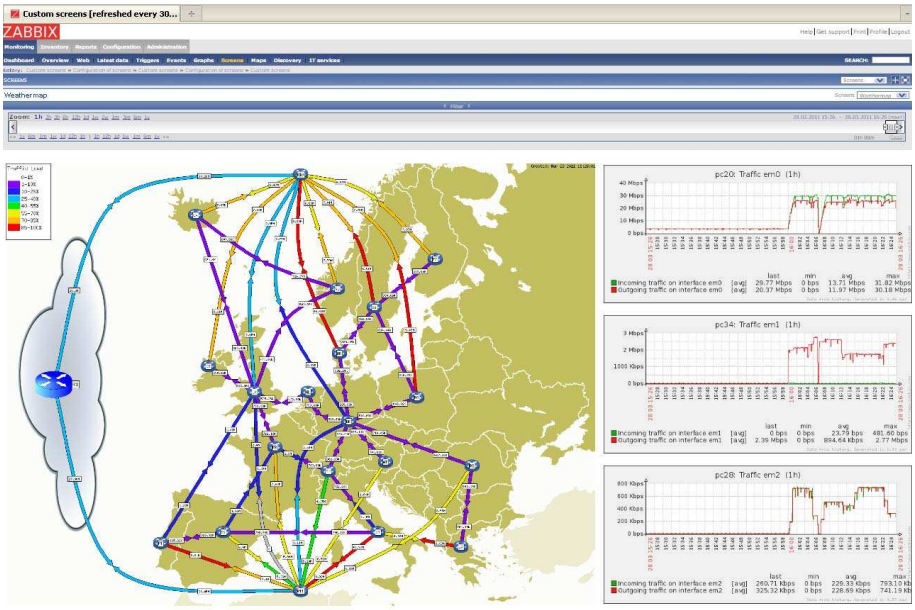

Figure 5. Zabbix web interface for experiment monitoring.

This process automatically configures the powerful, user-friendly web interface presented in Figure 5. The interface can be used for general purpose

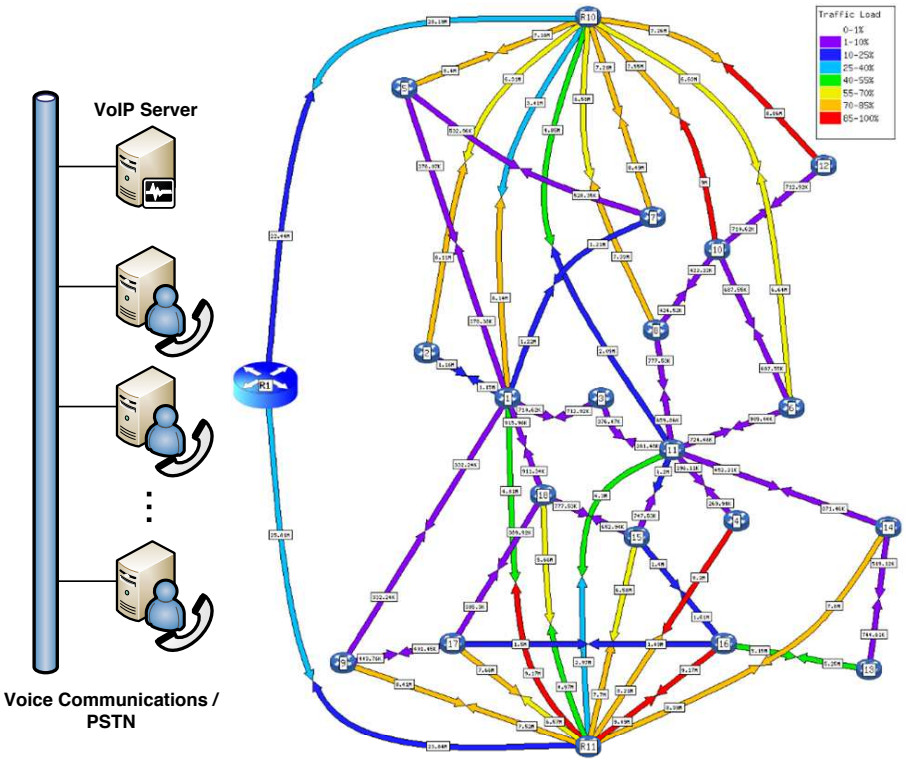

Figure 6. Exercise topology.

experiment monitoring such as presenting graphs of traffic loads, CPU and memory usage of individual nodes, as well as a network weathermap [7]. Exercise moderators can also use the interface as a central exercise monitoring screen.

6. BGP Attack Response Scenario

This section describes an operational exercise intended to assess the communications and coordination of network service providers during a BGP man-in-the middle attack as used in the infamous YouTube hijacking incident [14]. The use case demonstrates how participants could use an emulation testbed, how a scenario could be played and how information for studying response strategies and communication patterns can be captured.

The exercise involved 21 participants: eighteen national network service providers and three global network service providers (R1, R10 and R11) that simulate the Internet core (Figure 6). In this simplified Internet model, each network service provider communicated directly with its neighbors (if they shared a link); otherwise they communicated through the Internet core. An

eBGP session between network service providers that shared a link was used to exchange their prefixes.

The exercise scenario assumed that a network service provider was compromised by an internal attack that hijacked the IP address spaces of two network service providers, NSP12 and NSP16. In addition to hijacking the IP address space, the attack also performed a man-in-the-middle exploit by forwarding traffic to the destination. This was accomplished by announcing more specific prefixes of the victims and applying "AS-path prepend" of the intended network service providers in the path [6]. Thus, the attack was able to copy and manipulate traffic between NSP12 and NSP16, thereby compromising the integrity and confidentiality of communications. The scenario assumed that the operators of the compromised network service providers were not reachable and were unable to mitigate the internal attacks promptly.

This scenario was recreated in our Emulab-based testbed using 21 router-nodes running Quagga software with the appropriate BGP configuration. Links between network service providers had 10 Mbps of bandwidth and 10 ms of delay, while links in the core had 100 Mbps of bandwidth and 0 ms of delay. Each network service provider announced a `/16` prefix that was configured on a loopback interface. An isolated node with Zabbix software connected to the routers via the control network was used to collect traffic statistics. Another experiment with Asterisk software was used to simulate the public switched telephone network and support voice communications between the exercise participants.

Every process was automated during experiment start-up using a different script. The scripts enabled Quagga to load the right configuration file on each router. Also, they enabled the Zabbix server to configure the agents and itself with the required hosts and graphs, and the network weathermap to visualize traffic load. Such automation is very important from the point of view of scalability.

After the experiments were instantiated, the events corresponding to the attack were launched. Initially, sensitive traffic between NSP12 and NSP16 followed the path NSP12-R10-R1-R11-NSP16 and vice versa (left-hand side of Figure 7). However, after the attack changed the BGP configuration to hijack the IP address spaces of NSP12 and NSP16 by announcing more specific prefixes, sensitive traffic followed the path NSP12-R10-NSP1-R11-NSP16 and vice versa (right-hand side of Figure 7). Note that Figure 7 presents the visualization of sensitive traffic as seen from the monitoring server.

Since the attack forwarded traffic to the intended destination, the source and destination network service providers were unaware of the route modification and the adversary was able to capture and eventually modify the packets. Although the attack was transparent from the point of view of communications, the users experienced higher delays and the operators could see that traffic was diverted to other interfaces in the routers.

Additional details can be obtained by examining the BGP tables before and after the attack. Before the attack, the path between NSP12 and NSP16 (in

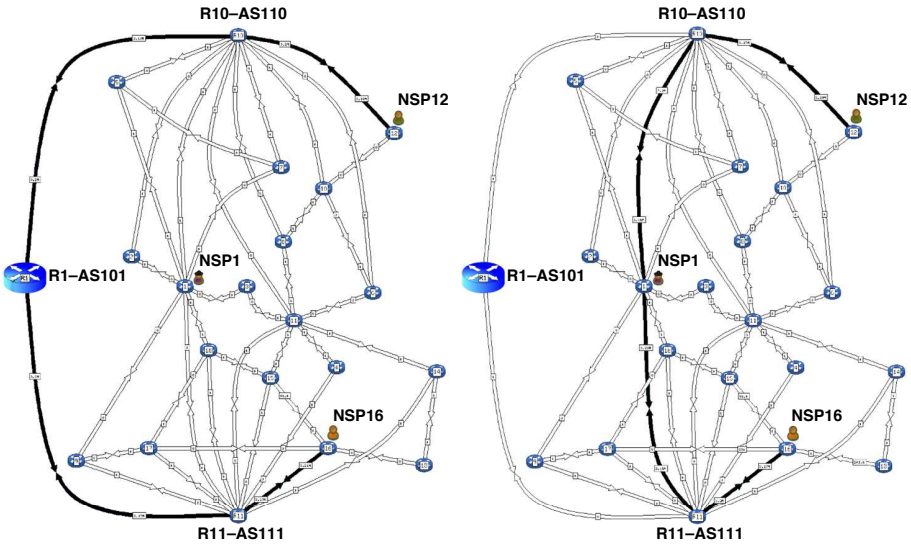

Figure 7. Sensitive traffic between NSP12 and NSP16.

terms of ASs) was: 12 – 110 – 101 – 111 – 16, corresponding to the ASs of NSP12, R10, R1, R11 and NSP16, respectively:

```
NSP12# show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.16.0.0/16     10.1.20.2           0 110 101 111 16 i

NSP16# show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.12.0.0/16     10.1.25.2           0 111 101 110 12 i
```

After the attack, the BGP tables were manipulated and included more specific prefixes that followed a different path through NSP1 (AS1). The use of "AS-path prepend" by the attack made the new entries seem legitimate, just as if they were announced by the destination network service provider:

```
NSP12# show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.16.0.0/16     10.1.20.2           0 110 101 111 16 i
*> 10.16.0.0/24     10.1.20.11          0 110 1 111 16 i

NSP16# show ip bgp
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.12.0.0/16     10.1.25.2           0 111 101 110 12 i
*> 10.12.0.0/24     10.1.25.11          0 111 1 110 12 i
```

During the exercise, the participants could react to and choose one or a combination of two response strategies:

- **Filtering Strategy:** The victims of the attack contact the peering network service providers and ask them to take action. In the exercise, the core routers R10 and R11 receive harmful announcements from NSP1 and filtering must be applied to block the announcements.

- **More Specific Prefix Strategy:** The victims combat the attack by announcing even more specific prefixes. The victims act and coordinate their activities as in the filtering strategy, but they do not need to contact other network service providers.

Although the strategy of announcing more specific prefixes seems less complex and requires less coordination with other network service providers, it may not be the best technical and long-term strategy for several reasons. For example, providers that are upstream of the victims might deploy prefix filters that do not allow the use of more specific prefixes, or the victims could compete with the attacker in announcing prefixes of increasing specificity. A detailed discussion of these issues is outside the scope of this paper.

In the case of the hijacking attack and assuming that the filtering strategy is applied by the network service providers that are directly connected to the attacker, then the total time T_t that the victims would spend on the telephone to ask all the network service providers to filter the attack is given by:

$$T_t \propto (N_p \times N_v) \times T_c$$

where N_v is the number of victims, N_p is the number of network service providers that peer with the attacker and T_c is the time required for two participants to coordinate their actions. This represents the "cost" of mesh communications between uncoordinated victims and network service providers that are directly connected to the single man-in-the-middle attacker. This time is different from the actual time required to mitigate the attack because the latter depends on factors such as the availability of concurrent communications, the time needed to apply filtering by network service providers due to internal procedures, and even BGP convergence times. Furthermore, the formula assumes a constant time T_c for each communication but, in reality, T_c depends on operator experience, contact networks, operators language skills, etc.

The value of a real operational exercise based on this use case goes beyond theoretic constructs to a deeper understanding of the operational reality where decisions and reaction measures follow administrative procedures. This often translates to a series of communications, possibly involving third parties (e.g., a regional Internet registry (RIR)) to confirm information related to the announced routes. Therefore, in the context of preparedness, the execution of an operational exercise on top of an emulation testbed (enhanced with voice communications) with multiple participants from different network service providers would not only support training, but also provide input to researchers about network service provider coordination in terms of procedures followed, typical values of T_c and the need to automate administrative procedures. Given

the complexity of the Internet, coordination between organizations, institutions and stakeholders is a key factor in any response to a contingency.

7. Conclusions

Organizing multi-party operational cyber security exercises using an emulation testbed offers several advantages. Exercises can be conducted without interfering with production networks while offering a realistic environment that supports voice and data. Remote access to the testbed supports real-time exercises of long duration that actively involve large numbers of participants. Exercises can include architectures, technologies and policies that are not yet deployed; and monitoring and data collection can be very detailed with limited privacy concerns. Finally, investing in reusable components simplifies the task of organizing future cyber exercises while reducing costs.

Our future work will analyze the effectiveness of the paradigm in real exercises. Other areas of focus include enhancing the fidelity of the platform, and developing and conducting exercises that cover multiple critical infrastructure sectors.

References

[1] W. Adams, E. Gavas, T. Lacey and S. Leblanc, Collective views of the NSA/CSS cyber defense exercise on curricula and learning objectives, *Proceedings of the Second Conference on Cyber Security Experimentation and Test*, p. 2, 2009.

[2] European Commission, Protecting Europe from Large Scale Cyber-Attacks and Disruptions: Enhancing Preparedness, Security and Resilience, COM(2009) 149, Brussels, Belgium (ec.europa.eu/information_society/policy/nis/docs/comm_ciip/comm_en.pdf), 2009.

[3] Federal Emergency Management Agency, Homeland Security Exercise and Evaluation Program (HSEEP), Washington, DC (hseep.dhs.gov).

[4] Flux Research Group, Emulab bibliography, School of Computing, University of Utah, Salt Lake City, Utah (www.emulab.net/expubs.php).

[5] Flux Research Group, Emulab – Network Emulation Testbed, School of Computing, University of Utah, Salt Lake City, Utah (www.emulab.net).

[6] C. Hepner and E. Zmijewski, Defending against BGP man-in-the-middle attacks, presented at the *Black Hat DC Conference*, 2009.

[7] H. Jones, Network Weathermap (www.network-weathermap.com).

[8] K. Lahey, R. Braden and K. Sklower, Experiment isolation in a secure cluster testbed, *Proceedings of the Conference on Cyber Security Experimentation and Test*, 2008.

[9] Y. Li, M. Liljenstam and J. Liu, Real-time security exercises on a realistic interdomain routing experiment platform, *Proceedings of the Twenty-Third Workshop on Principles of Advanced and Distributed Simulation*, pp. 54–63, 2009.

[10] M. Liljenstam, J. Liu, D. Nicol, Y. Yuan, G. Yan and C. Grier, RINSE: The real-time immersive network simulation environment for network security exercises (extended version), *Simulation*, vol. 82(1), pp. 43–59, 2006.

[11] J. Mirkovic, A. Hussain, S. Fahmy, P. Reiher and R. Thomas, Accurately measuring denial of service in simulation and testbed experiments, *IEEE Transactions on Dependable and Secure Computing*, vol. 6(2), pp. 81–95, 2009.

[12] J. Mirkovic, P. Reiher, C. Papadopoulos, A. Hussain, M. Shepard, M. Berg and R. Jung, Testing a collaborative DDoS defense in a red team/blue team exercise, *IEEE Transactions on Computers*, vol. 57(8), pp. 1098–1112, 2008.

[13] R. Ostrenga and P. Walczak, Application of DETER in large-scale cyber security exercises, *Proceedings of the DETER Community Workshop*, 2006.

[14] RIPE Network Coordination Center, YouTube hijacking: A RIPE NCC RIS case study, Amsterdam, The Netherlands (www.ripe.net/news/study-youtube-hijacking.html), 2008.

[15] B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, W. Adams, C. Morrell and G. Conti, Toward instrumenting network warfare competitions to generate labeled datasets, *Proceedings of the Second Conference on Cyber Security Experimentation and Test*, p. 9, 2009.

[16] S. Schwab, B. Wilson, C. Ko and A. Hussain, SEER: A security experimentation environment for DETER, *Proceedings of the DETER Community Workshop*, p. 2, 2007.

[17] R. Stapleton-Gray, Inter-network operations center dial-by-ASN (INOC-DBA), A resource for the network operator community, *Proceedings of the Cybersecurity Applications and Technology Conference for Homeland Security*, pp. 181–185, 2009.

[18] The White House, The National Strategy to Secure Cyberspace, Washington, DC (www.dhs.gov/xlibrary/assets/National_Cyberspace_Strategy.pdf) 2003.

[19] A. Turner, Tcpreplay (tcpreplay.synfin.net).

[20] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb and A. Joglekar, An integrated experimental environment for distributed systems and networks, *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, pp. 255–270, 2002.