Exploiting Emulation Testbeds for Security Experiments

Andreas Grau, Klaus Herrmann and Kurt Rothermel
Universität Stuttgart, Institute of Parallel and Distributed Systems (IPVS)
Universitätsstr. 38, D–70569 Stuttgart, Germany
Email: {grau,herrmann,rothermel}@ipvs.uni-stuttgart.de

Abstract

Security, resilience and stability test are an important step to protect the information and communication-based digital society. Experimental platforms allow researchers to evaluate todays and tomorrows internet architectures, protocols and applications. However, selecting the most adequate platform is difficult. In this paper we first present a common used approach based on real world testbeds. We identify a couple of problem coming around with this approach and show how emulation-based testbeds can avoids them. Finally we present an hybrid approach to combine the benefits of both systems to provide a scalable, repeatable, controllable, flexible and secure experimental platform.

1 Introduction

Digital information and communication are the foundation of todays economy. Data loss and disruption of communication cause serious damage to business, administration and private lifestyle. Therefore, architecture, protocols and applications have to be secure, resilient and stable. Researchers and application developers, therefore, need a experimental platform to evaluate their software.

The remainder of this this paper is structured as follows. In Section 2 we discuss a common approach for experimental platforms based on real world testbeds and identify problems coming with it. Section 3 presents an approach to avoid the problems using emulation-based testbeds. An combination of both approaches is presented in Section 4, before we conclude in Section 5.

2 Real World Testbeds

A common approach to run evaluate security, resilience and stability of software systems are based testbeds like PlanetLab and Onelab, which we name in the following *real world testbeds*. The main motivation to use these type of testbeds is the connectivity of the nodes to a real network like the Internet. Due to the real network between the nodes, this approach uses realistic (real) network conditions including real bandwidth, delays and message loss. However, the usage of a real network leads to a couple of problems with this approach:

The first problem is the **controllability** of experiments. Since we cannot specify the link parameters between nodes or the amount of background traffic in the internet, we are unable to run experiments with conditions differing from the current situation in the real network. This, for example, prohibits running experiments with on a lightly loaded network (which occurs at night) at daytime. Also related to this topic is the impossibility to run experiment with userdefined topologies e.g. a local area network.

A related problem is the **repeatability** of experiments. Since we are unable to specify the network conditions of the real network, we cannot test the influences of different software parameters or software components against the same network conditions. Since we also cannot repeat experiments with same parameters and same network conditions, we can hardly make statements concerning the statistical significance of the observered behavior or stability issues of software. Since nodes in real world testbeds are typically shared between many users, overloaded resources may also biases experiment results. In case the experiments themself generates high load, e.g. while testing DoS countermeasures, the experiments may also bias experiments of other users.

The **flexibility** of real world testbeds is limited by the introduced abstraction level. In these testbeds typically users have no root access on the nodes and, therefore, they cannot make any changes to the operating system. This limits the possible software under test in experiments to application layer software which prohibits experiments like, for example, countermeasures against TCP-SYN attacks. Also many security related software like firewalls requires operating system access and, therefore, cannot be included in the experiments.

The last, but may most important problem is related to **security** itself. Testing security related system in the real world may result in security risks for the internet itself. Possible risks include distribution of malware while testing countermeasures against them, or testing attack patterns against routers can harm the internet routers. Finally testing DoS attacks might result in attacks against the testbed itself.

3 Emulation-based Testbeds

To conquer all abovementioned drawbacks, we propose the exploration of emulation testbeds for security experiments. In an emulation testbed, like the Network Emulation Testbed (NET) [1] or the Emulab, dedicated

nodes and network infrastructure is used for experiments. In the following, we discuss how the problems of real world testbeds are handled by emulation-based testbeds.

Controlability An arbitrary, user defined network topology between nodes can be formed, including topologies from small LANs to large scale networks with multiple autonomous systems as well as mobile adhoc networks. The definition of the topology includes attributes like bandwidth, delay, message loss. Scenario specific network conditions can be generated using background traffic generators.

Repeatability Same conditions on the nodes as well as in the network can be repeated for different experiment runs. This allows to deduce the influences of these network parameters.

Flexibility Nodes are time shared, meaning that nodes are used exclusively. Therefore the user has full control on the environment. Using virtual machine technology, it is possible to run nodes with arbitrary software. This includes the execution of different operating systems like Linux, Windows, BSD as well as different versions of these systems. In general, network emulation testbeds consider all layers above Data Link Layer (Network, Transport and Application) as software under test.

Security Since emulation testbeds uses dedicated hardware, the emulated network is completely separated from other networks. In case of security-related experiments this enables, for example, evaluation of attacks against routing protocols as well as transport protocols like TCP and firewall-based countermeasures. This feature also allows to evaluated even scenarios with unknowing potential risks and sideeffects of the used software (e.g. malware).

Scalability The size of scenarios, namely the number of nodes, can be increased far beyond the number of physical available testbed nodes, by multiplexing several nodes to a physical testbed node using node virtualization. Even with node virtualization, the experimenters freedom to define arbitrary network conditions are not restricted. Another concept in emulation testbeds is the time virtualization. This concept allows to virtually increase the available physical testbed resources, which allows to increase the number of nodes in the scenario or emulate nodes with resources beyond the physical available resources, e.g. high speed network links.

4 Hybrid Experimental Platform

Beside the benefits of using an emulation testbed as an experimental platform for Internet resilience, security and stability tests, the emulation approach as one fundamental limitation. The opportunity to model arbitrary network conditions, leads to the following challenges:

- What are the **relevant** network topologies and network conditions to consider?
- How to model **realistic** network topologies?
- How to **share** network topologies between different experiments?

To solve these challenges we propose to build up a repository storing experiment setups. An description of the experiments allows other experimenters to search for experiment setups that match up their requirements. Based on the stored network topology and network conditions a selected experiment can be automatically deployed in an emulation testbed. Using a standardized description language testbed providers can implement adaptors to setup the experiment in their testbed. This allows the experimenters to select a testbeds, based on constraints like the currently available resources in that testbed.

In order to get realistic results, we propose to combine emulation testbeds and real world testbeds using a three phase approach. In the first phase the conditions of a real network are recorded and stored in the repository. To record the data the experiment is executed in a real world testbed. In the second phase the experiment is deployed in an emulation testbed. In this phase the experiment can be repeated with different parameters using the same network conditions. Using the knowledge gained from running the software in an emulation testbed, the software under test may be adjusted and the experiments in the emulation testbed could be repeated. After finding a satisfying solution the software could be again deployed on a real world testbed to confirm the results of the emulation environment.

5 Conclusion

In this paper we argued, the demand of using emulation testbed as an experimental platform for Internet security, resilience and stability tests. The main arguments to prefer emulation to real world testbeds are the gained controllability, repeatability, flexibility and security issues. After that, we proposed an approach to fulfill the demand on using realistic network conditions in emulation testbeds. Here, we propose to record the conditions of the real network using a real world testbed and utilize the captured conditions as an foundation for experiments in an emulation testbed.

References

[1] A. Grau, S. Maier, K. Herrmann, and K. Rothermel. Time Jails: A Hybrid Approach to Scalable Network Emulation. In *22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008)*. IEEE Computer Society; ACM, June 2008.