SANDIA REPORT

SAND2017-10634
Unlimited Release
Printed August 2017

Evaluating Emulation-based Models of Distributed Computing Systems

Stephen T. Jones, Kasimir G. Gabert, Thomas D. Tarman

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2017-10634
Unlimited Release
Printed August 2017

Evaluating Emulation-based Models of Distributed Computing Systems

Stephen T. Jones
Cyber Initiatives
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0621
stjones@sandia.gov

Kasimir G. Gabert
Cyber Initiatives
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0621
kggaber@sandia.gov

Thomas D. Tarman
Emulytics Initiatives
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0621
tdtarma@sandia.gov

Abstract

Emulation-based models of distributed computing systems are collections of virtual machines, virtual networks, and other emulation components configured to stand in for operational systems when performing experimental science, training, analysis of design alternatives, test and evaluation, or idea generation. As with any tool, we should carefully evaluate whether our uses of emulation-based models are appropriate and justified. Otherwise, we run the risk of using a model incorrectly and creating meaningless results. The variety of uses of emulation-based models each have their own goals and deserve thoughtful evaluation. In this paper, we enumerate some of these uses and describe approaches that one can take to build an evidence-based case that a use of an emulation-based model is credible. Predictive uses of emulation-based models, where we expect a model to tell us something true about the real world, set the bar especially high and the principal evaluation method, called *validation*, is comensurately rigorous. We spend the majority of our time describing and demonstrating the validation of a simple predictive model using a well-established methodology inherited from decades of development in the compuational science and engineering community.

Contents

Figures

Tables

1 Introduction

Emulation-based models (EBMs) of distributed computing systems (we will define this term more carefully in Section 2) are used for a variety of purposes by many different government, commercial, and academic organizations. In some cases, these uses may have significant consequences like making resource allocation decisions, evaluating design alternatives, training staff, or performing test and evaluation. In other cases, the use may be less demanding, for example generating ideas or demonstrating a concept.

Regardless of the specific use case, the first question we should ask ourselves when employing such a model is how to determine and demonstrate that our use is credible and valuable. In more established model-using disciplines like the aerospace or automotive industries, evaluating a model's fitness for purpose is a natural part of using a model. In disciplines that use the relatively recent invention of EBMs, that habit has yet to emerge.

In this paper we will:

1. Describe what EBMs are
2. Describe some of the ways EBMs have been used to date
3. Discuss how one can generate convincing evidence that a specific use of an EBM is credible

There are a variety of uses for EBMs. The best way to ensure that we are using EBMs correctly will depend on our specific uses and model implementations. A range of evaluation techniques is available from informal to highly structured and rigorous. Choosing an evaluation method and then applying it to an EBM can be tricky because EBMs are often large and complicated with many moving parts.

In this paper, our discussions are based on the best information the authors have to date. No claim is made that the ideas or processes discussed are comprehensive or complete. Our main goal is to encourage careful and critical thinking about the uses of EBMs and to capture what we know today about evaluating EBMs and provide a basis for ongoing research and development.

2 Emulation-based Models of Distributed Systems

An EBM is a collection of virtualized or emulated computing, communications, or control system components that are connected by virtualized networks. Virtual machines [30] stand in for personal computers, servers, mobile phones, and embedded devices. Because the virtual machines or system emulators used within EBMs reproduce a large fraction of the hardware interface of a real computer system, they can run unmodified versions of real commercial or open source operating systems and application software. Virtualization is a mature technology and is widely used in production computing environments today. In an EBM, however, virtualization is used to create laboratory models of distributed computing systems for experimental purposes. Open source hypervisors like the Linux kernel virtual machine (KVM) [16] or Xen server [2] are often used to implement EBMs for maximum flexibility and to minimize licensing costs.

Each virtual machine in an EBM is configured in a specific way with a desired number of virtual CPUs, an amount of RAM, with attached storage devices like virtual disk drives, and with some number of network interfaces. The virtual machines that make up an EBM run on a collection, or cluster, of physical machines dedicated to experimentation. A virtual machine's network interfaces are connected to a virtual overlay network that is implemented on top of the computer cluster's physical network. The virtual overlay network is constructed using software switches like OpenVSwitch [25] or Linux bridges and other techniques like VLANs, tap devices, and point to point tunnels. Each virtual machine runs system software like Microsoft Windows, Linux, or OpenBSD, as well as a variety of application software. The detailed configuration of each virtual machine including the software it is running, the specific way the virtual machines are connected together in the network, and what events take place within an experiment depend on what the EBM is meant to accomplish.

For example, Figure 1(a) shows a simple example of an EBM design for testing a hypothesis about whether a load balancer can continue to service requests given the failure of one of its back-end servers. The abstract experimental topology includes client machines $C_1 - C_4$ which provide load, a switch to connect the clients, a load balancer, two servers $S_1 - S_2$ across which the load will be distributed, a switch connecting the pieces of server infrastructure, and a router that connects the two otherwise isolated network segments. Each of the abstract components, including clients, servers, router, load balancer, and switches, have properties attached to them that describe their configuration.

Our fictitious experiment will put the claims of our hypothetical load balancer vendor to the test, which are that service performance will degrade in proportion to the number of failed servers. This is a good candidate for an EBM experiment because the outcome depends on the software implementation details of the load balancer, a large, complicated system.

To actually run the experiment, the components of the design need to be instantiated as virtual machines and other components on a set of physical computers. Figure 1(b) shows one possible distribution of virtualized components across two physical hosts. The switches are shown dashed to indicate that, in this design, they are instances of Linux bridges running on the physical host to which virtual machines are connected rather than virtual machines themselves. There are many

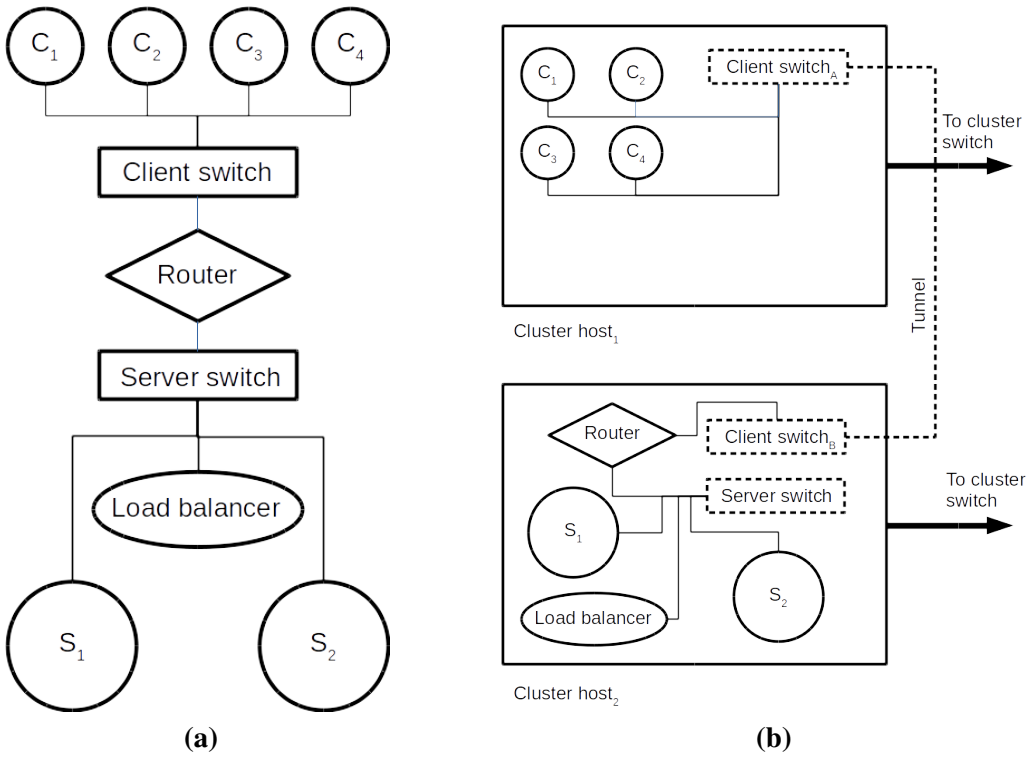


Figure 1: *An example experimental topology (a) and its instantiation (b) on the physical hosts of a small, two-node cluster.*

implementation details that the EBM orchestration software which creates and configures virtual machines must handle. One example in this case is the fact that the logical client switch must be split between the two physical cluster hosts since there are virtual machines located on both hosts that must attach to that switch. A tunnel runs from $client\ switch_A$ to $client\ switch_B$ to unify the two switches.

During the experiment, virtual machines will be created, operating systems will boot, software will be installed and configured, the clients will begin generating load through the load balancer, and, at some point, one of the servers will intentionally fail. Most EBMs are highly orchestrated and all of the preceding will happen automatically, carefully managed by a set of orchestration tools. The instantiated experiment will have been instrumented to record the behaviors we are interested in, which, in this case, are the continued availability of the load-balanced service and how the failure changes the performance of the service in terms of response latency or throughput.

Most EBMs are variations on the simple theme just described. An abstract topology is designed to answer a question. That design is instantiated on physical computers using virtual machines and other carefully configured emulation components. Scripted events implement experimental actions. Finally, the systems we are interested in are thoroughly instrumented to answer a question or achieve some other operational result.

We have experience building EBMs to represent enterprise networks, large internet services, and control systems sitting at the heart of sections of regional electric grids. Assuming the information

about the distributed system in question is available, an EBM is a convenient and relatively inexpensive way to build models of complicated computing systems and to observe and explore their behavior and performance in a safe, low-consequence, experimental environment.

2.1 Beware

Because EBMs are based on virtual machines and networks that look and act much like their physical counterparts, it is natural to believe that their performance and behavior will be very similar to comparably configured physical systems. After all, if virtualized systems performed dramatically worse or acted noticeably different from the physical systems we are more familiar with, it is less likely that they would be widely adopted, which they have been. For this reason, using EBMs for modeling purposes can be seductive. Because they intuitively seem to reproduce the behaviors of non-virtualized systems and networks, we are lulled into the belief that their detailed behavior will naturally meet our modeling and experimental expectations.

Unfortunately, we have evidence that this isn't true. The detailed and sometimes large scale behavior of virtualized systems can be significantly different from equivalent physical systems. For example, Chertov et al. [5] show how the forwarding behavior of emulated switches differs from that of hardware switches and how it can affect a modeling application [6]. Paleari et al. [24] demonstrate that the CPU implementation of widely used and compatible full system emulators often diverge from the hardware specification they implement. Even when emulated behavior is quite complete, the performance of simulations and emulators can diverge significantly [34], potentially to the point where fundamental network properties are violated [10]. Finally, when building models of complicated systems, it is easy to leave out important details and come to false or misleading conclusions [9, 17]. Our own experiences trying to use EBMs mirror these reports.

EBMs are laboratory, experimental models. They stand in for some part of the real world that we cannot or don't wish to experiment on directly. This is a different approach than the one taken by large scale mathematical models of physical systems. They are typically software implementations of abstract mathematical representations, like differential equations, meant to describe selected, externally observable characteristics of a system. A laboratory model, on the other hand, does not try to abstractly describe the behavior of a system, it reproduces, usually in some reduced or simplified form, the underlying mechanism we want to observe.

Laboratory models, though simplified versions of the real world, include all kinds of messy details, inherent randomness, and error. Laboratory models do not, and often cannot, implement or include all the features of the real systems they represent. For a variety of reasons, including resource sharing, overhead, or simplifications, the performance of the emulated systems, on which EBMs are based, can sometimes diverge dramatically from similarly configured real world systems. Even when all relevant components are included in a model and performance parity is high, uncertainty about parameters and configuration values can lead to bias or large variability in the measured outputs of a model based on emulators.

Other laboratory models used in different disciplines exhibit the same kinds of problems and have

similar advantages as EBMs. When biologists use fruit flies, mice, or even primates to study the effects of medical treatments on humans they must be careful to eventually perform human trials on promising interventions. It is very common that treatments, which appear to work well and consistently in simpler laboratory models, fail when applied to the unique physiology of humans. Hence, laboratory models of humans are efficient early stage tools to identify potentially promising treatments. They help researchers generate ideas and perform tests to trim obviously unproductive lines of inquiry. They are never used as the sole basis for setting public health policy or adjusting human clinical practice without focused, and usually expensive, validation via human trials. In spite of their shortcomings, laboratory models are immensely useful tools. They allow us to quickly focus our attention on the most promising real world trials.

Similarly, wind tunnels, another type of laboratory model, have a long history of being used during the design of air frames, wings, and other structures subject to high speed airflow. Performing tests at reduced scale, but realistic air speeds, pressures, and temperatures has dramatically advanced our knowledge of fluid flow and helped spawn the entire computational fluid dynamics modeling field. However, wind tunnels exhibit variability and bias too [19]. Researchers and engineers must take such errors and uncertainties into account or pay the price of poor performance predictions and incorrect decisions later in the design and testing process.

Like any other tool, EBMs can be helpful time savers and extend our capabilities or they can lull us into a sense of unwarranted confidence and lead to mistakes or misunderstandings. The key question we want to begin addressing in this paper is "how do we go about discovering whether our particular uses of EBMs are justifiable?". We will discuss this problem for a variety of uses of EBMs that we are familiar with and will cover some initial experience we have had attempting to build evidence-based cases for or against the use of an EBM to answer specific questions or serve some other purpose.

3 Credibility Case for EBMs

When evaluating whether a particular use of an EBM is credible, our goal is to collect evidence and construct a convincing case that the EBM is useful for our chosen purpose. As in a debate, we know when our case is good enough when it can convince a skeptic of our thesis. There is no general-purpose measure we can apply to an EBM to determine if it is appropriate independent of its purpose. There is no general standard for adequate "fidelity" or "accuracy" without the context of how a model is to be used. In fact, the same model may be perfectly appropriate to use for some purposes, but fall hopelessly short for others.

In all of the cases we will discuss, the process for collecting evidence for or against using a model is based on measuring a model's *performance*. We will observe whether the EBM is able to meet a pre-defined performance threshold and then base a judgment about whether or not to use a model on that observed performance. In some cases the threshold will consist of achieving an operational objective. In others, we will observe the degree to which an EBM matches the behavior of a real world reference system. In all cases, the required threshold is driven by the specific model purpose.

A single test or measurement of an EBM will rarely suffice to build a good credibility case. EBMs include many sources of randomness and other uncertainty. As with any other experimental endeavor, it will be necessary to replicate model observations so we can measure how consistently our models meet or fail to meet our predefined credibility thresholds. Often, a modeling purpose requires an EBM to perform adequately at many points within a large parameter space. Multiple sets of tests at several different configurations ensure the model performs well enough for the parts of the real world we are interested in. The full suite of replicated tests at various locations is necessary before we can say anything meaningful about a model's suitability for the associated purpose.

A credibility case for an EBM will typically consist of:

1. A clear statement of the purpose of the model
2. The performance requirements of the model, driven by the purpose; these are usually quantitative or yes/no thresholds that must be met for specific, measurable outputs of the model at specified locations in a parameter space
3. A justification for the performance thresholds that were chosen; i.e., reasons we believe the model is good enough for our purposes if it meets the stated threshold
4. A description of the experiments we will perform to measure the model's performance
5. The actual model measurements and a characterization of their variability or uncertainty

Given a credibility case, a model user or accreditation authority can then more easily judge whether a particular use of a model is justified. Even when careful and extensive measurements have been made, the ultimate choice to use or not use a model comes down to the informed judgment

of a person. To ensure that incentives are appropriately aligned and that an honest assessment is possible, the person deciding whether or not to use a model should also carry some of the responsibility for the consequences of using a model inappropriately.

4 Uses of EBMs

EBMs are used for a many different purposes. In this section we'll discuss the principal uses of EBMs that we have seen.

4.1 Predictive Analysis

One of the most interesting and certainly one of the most challenging uses of EBMs is for predictive analysis. In this case, we expect a model of a software intensive distributed system to predict some aspect of its corresponding real world system's performance or behavior. The intent is for the model to stand-in for the real world system that we care about for experimental purposes. We use a model, rather than experimenting on the real world system directly for some combination of cost, risk, or availability reasons. When possible, predictive analysis allows us to do reliable, model-based experimental science to understand and predict the behavior of distributed systems.

Distributed software systems exhibit complicated, sometimes emergent behavior that is hard to foresee, much like the physical systems that other brands of experimental science study. This is true even when we have all of the software and hardware artifacts available to us from which the system is constructed. This fact shouldn't be surprising. Deciding whether a software system has any specific non-trivial property is undecidable in the general case [27]. Even relaxed versions of the problem become impractical when many, often independently developed and individually complicated software systems interact with one other over networks or other interprocess communications channels. EBMs provide one important way of systematically observing complicated distributed systems with the intent of understanding their behavior better.

While EBMs are constructed from deterministic computing systems, they may include explicit and implicit sources of randomness and uncertainty. Hence, the predictions we expect from a model used in this way must be more than a single point estimate of the value we seek. A prediction typically consists of:

1. A point estimate of the desired measure
2. A characterization of the point estimate's variability in the form of a distribution or a confidence interval for the summary statistic used to generate the point estimate
3. A measure of the estimate's error, i.e., to what extent does the estimate systematically differ from the real world?
4. A characterization of the error's variability

The measure used for prediction need not be numeric or continuous. We may be predicting whether a certain property will hold (a boolean) or what state a system will be in after a series of inputs (a category). We can still measure how often we are right and how much that success rate varies.

If we can show that a model is consistently capable of predicting the distributed system measures we are interested in with small enough error for our scientific purposes, we can have confidence that our use of the model for that purpose is credible.

Scientific use is a high bar for a model, especially an EBM, to achieve. The process that model users typically employ to gather evidence to support the predictive use of a model, called verification and validation (V&V) [23], is commensurately rigorous. Because of the nature and use of distributed computing systems, they will often not meet the requirements necessary to perform strict validation. If a model is not validatable, it probably shouldn't be used for predictive purposes, especially in situations where the consequences for false conclusions are high. Other less definitive techniques will be necessary to evaluate whether an unvalidated, but potentially still useful model should be used for other, non-predictive purposes.

4.2 Training

EBMs are often used to create flexible computing and network environments in which to train staff, especially those responsible for defending networks from external intrusion or for monitoring networks for internal abuse and malicious insider activity [33]. In some cases these environments consist of fabricated, generic enterprise networks. In other cases modelers attempt to reproduce an actual enterprise network configuration that includes the topology, services, and traffic expected in the real world network that the trainee works in. Training environments can extend beyond enterprise networks to include industrial control systems and simulations of the associated physical processes like electric power generation and transmission systems. Sometimes, more exotic collections of devices that include weapons systems or specialized command and control systems and networks are modeled. We have seen EBMs used as small-scale training environments for DoD cyber defense staff, as the basis for large scale, multi-organization cyber exercises, to train and test network defense operators' skills using specific tools, or by incident response organizations to war-game scenarios.

Virtualized training environments are especially useful because they provide a low consequence venue where students can freely make mistakes, compensate, and repeat without fear of damaging production systems or disrupting operations. EBMs are typically highly orchestrated, so it is easy to tear down and re-establish the baseline training environment in a short time and with little effort. EBMs also provide an easy way to instrument most aspects of a training environment including recording the content of console displays, collecting a trainee's keyboard and mouse usage, or making full network packet captures that can be used for post training review and evaluation.

In all cases, the generic goal of a training event is to provide a positive training experience for a trainee. By definition, a positive training experience contributes to the immediate or eventual improvement in a trainee's performance on the direct and indirect tasks related to the training exercise. More specifically, training and exercises often have a concrete training objective they attempt to achieve. For example, a cyber defense training exercise may want to familiarize staff with the features of a new tool and ensure that everyone achieves a baseline level of proficiency in

its use. The goal of a different training event might be to reduce incident response times to below an acceptable threshold. Later, we will see how important it is to have specific and measurable training goals when evaluating whether an EBM is an appropriate training tool.

4.3 Test and Evaluation

In a test and evaluation (T&E) use case, an EBM usually provides an environment in which a system under test (SUT) is evaluated according to a formal test plan. A test team takes detailed measurements of the SUT as it operates under a variety of conditions and eventually judges, based on test results, whether it meets its previously agreed upon performance specifications. A SUT can be a physical device, a virtual device, a software product, a complicated distributed system, or an organizational procedure or technical process.

T&E environments share some important properties of other use cases. A training exercise, for example, tries to *influence* the real world performance of an experiment participant, who isn't an explicit part of the EBM itself, by exposing them to events and stimuli within a virtualized environment. In T&E, we are interested in the behavior of a SUT which is also alien to the virtualized experimental environment. T&E is also much like a scientific experiment. In the T&E case, we want to verify that the behavior, of the SUT, as measured during tests in a virtualized environment, falls within predetermined bounds specified in the test plan. Verifying such a hypothesis is much like predicting the real world behavior of the SUT.

Most importantly, regardless of whether the outcome of a test performed in a virtualized environment is pass or fail, we expect the observed behavior of the SUT within an EBM to match its behavior in the real world closely enough that our test outcomes and judgments are valid in the real world.

EBM tests, however, will rarely be the sole source of information about the performance of a SUT. T&E within an EBM is often a lower cost pre-filter for performing more involved testing in a real world environment. Like the prequalification of an aircraft part in a wind tunnel prior to full scale free space atmospheric tests, performing one, or many, initial rounds of testing in the relatively low cost environment of an EBM can save test time and resources. A common assumption is that if a SUT fails in a virtualized environment, there is little hope that it will perform adequately in full-scale, real world testing.

We have used EBMs to evaluate the performance of a variety of network-connected or software systems including firewalls, intrusion detection systems, and network mapping tools.

4.4 Scenario and Idea Generation

In the early stages of an investigation of a system, we often don't really know where to start. We don't understand the system well yet and we don't have much of an idea about the range of

behaviors it might exhibit in response to various inputs or changes in its environment.

For systems whose behavior is largely determined by software, EBMs can be helpful because they allow us to instantiate the system and observe what it does in response to a large variety of inputs and when configured in many different ways. EBMs let us quickly explore many scenarios and start to get a feel for what some of the important questions are relating to our system of interest. Generating ideas, often by playing interactively with systems, is a common, maybe the most common, way EBMs are used today.

Using EBMs to generate ideas is low rigor. We don't expect a model used in this way to tell us truth about the real world. We just use the model to stimulate ideas and to be an aid to creative thinking about the behavioral possibilities of a system.

Using EBMs as an aid for idea generation can be productive, but it is also quite dangerous. It is a very common pitfall to move from this low-expectation mode of using an EBM to believing such a model is predictive, especially under time pressure. Before we use any model to predict something about the real world for decision support purposes, especially when the consequences of being wrong are non-trivial, we must do enough additional evaluation work to convince a skeptic that our use of the model is credible. A common red flag to be especially wary of is when an unevaluated EBM, used for informal, idea generation purposes, exhibits behavior that violates our expectations. Before coming to the conclusion that the surprise is real, additional testing is necessary.

4.5 Communication

Using a working model to demonstrate how a complicated mechanism works is a time honored approach. Cut-away displays of internal combustion engines, electric motors, or pocket watches communicate the principles at work much more effectively than pages of text or static diagrams. EBMs can be used in just this way to show how a distributed computing system works in a controlled environment that is easy to instrument and examine.

For example, we have used an emulated model of a fictitious enterprise network to show how a certain type of malware might spread through an organization and demonstrate how mitigations like reconfiguring important shared services or partitioning the network have an affect on the malware's spread. The purpose of this exercise was not to predict the behavior of specific malware, but rather to conveniently demonstrate some of the mechanisms and behaviors that have been observed in the real world.

As with idea generation, the user of such a model must make it very clear that the model is being used for demonstration purposes only and that, without additional evaluation work, we cannot use the model to reliably predict anything about the real world.

4.6 Operations

EBMs are usually highly orchestrated collections of virtual machines configured to approximately represent some aspect of a real network. Using EBM infrastructure and tools it is often easy to create large, complicated networks of realistic-looking machines and software services. There is at least one interesting use case where EBMs have been given an operational, production role.

Attacks on information systems networks often follow a common pattern of activity. Attackers begin with external reconnaissance, proceed to getting an initial foothold, elevate their privileges, perform internal reconnaissance, spread laterally to the most lucrative parts of a network, then collect and exfiltrate private information or do other malicious things like extort money from the legitimate user of the systems using a ransom scheme. If a defender can interrupt the early stages of that process, the cost of performing the attack may exceed an adversary's resources or patience [12] and prevent a successful compromise and prevent damage.

EBMs have been used to study and, in limited circumstances implement, large, on-demand networks of interesting-looking, but unimportant and sacrificial computers and services that are added to a real network's address space to deceive and distract would-be attackers [32]. One goal of such "deception networks" is making the malicious activity more detectable by raising its profile. Legitimate users of the network have no knowledge of and no need to visit the additional machines and services, so most such activity is suspicious. Another motivation is to impose additional cost and require additional effort during the initial phases of an attack with the hope of exceeding an adversary's budget.

The effectiveness of these kinds of defensive efforts is unproven, but, if carefully evaluated and shown to work, could be a potent tool in a defender's toolkit.

5 Prediction vs. Other Uses of EBMs

In the scientific and engineering domains, validation is the gold standard when building a case that a model is an appropriate substitute for the real world for a specific, predictive purpose. Validation, in the case of EBMs, consists of making careful comparisons between model predictions and observations of the real world or non-virtualized experiments [23]. We will discuss the process of performing validation on EBMs in Section 6.

When we use an EBM for scientific, predictive purposes, validation is appropriate. However, validation in the strict sense is quite demanding. Hodges and Dewar [11] define four requirements for a model *use* to be validatable. Note that validation applies to a use of a model, not the model itself.

1. The behavior we want to predict must be observable and measurable
2. The behavior must exhibit constancy of structure in time
3. The behavior must exhibit constancy across variations in conditions not specified in the model
4. The situation in which the behavior occurs must permit the collection of ample data

If we expect a model to be able to predict the outcome of a real world situation well enough for a scientific use, validation is the current best known tool. In Section 8 we will discuss some strategies for evaluating uses of EBMs that do not meet this bar and how we might gain confidence that our use of such a model is appropriate.

EBMs have the potential to be tripped up by any of the four requirements. Model uses that require a detailed representation of human behavior, for example, are especially prone to fall afoul of requirements 2 and 3 because of the natural adaptability of people and the fact that models of distributed computing systems are often inextricably connected to the rest of the world in complicated ways that cannot be represented in a model. Studies of internet services, on the other hand, may be unvalidatable for all but the service owner because they fail to meet requirement 1. Availability of adequate amounts of data, addressed by requirement 4, is less often an issue, but we have seen situations where data that is technically feasible to collect is not available for policy reasons.

Uses of EBMs that cannot be validated in the strict sense are not without value. We have outlined several possible uses of EBMs in previous sections and most do not meet the requirements for strict validation or do not merit the considerable extra work required for validation. However, they can be valuable to their users. Validation should be reserved for the special cases where we expect an EBM to stand-in for the real world and predict some aspect of its behavior reliably and when the results of such a study will be used for decisions whose consequences, if the results are incorrect, justify the additional labor of validation.

6 Verification and Validation of EBMs

Verification and Validation (V&V) is the most appropriate technique for evaluating EBMs that fit the pattern of a predictive model and that meet the requirements of validatability. The key ideas behind V&V are very simple. We ensure the components making up the model do what we expect and then we measure the predictive performance of our model by comparing predictions made by the model to observations of an equivalent real world situation. While conceptually simple, the details can be tricky and often depend on the informed judgment of an expert.

There are many thorough and readable discussions that cover the concept of V&V [22, 3, 23]. We will paraphrase the basic outline of general-purpose V&V to make the paper as self-contained as possible, but it is worth the effort to read a more complete discussion of the many interesting and thorny issues raised, especially by validation, which can quickly expand to encompass the philosophy of science and the nature of knowledge obtained via experiment and observation.

6.1 Verification

Oberkampf and Barone [22] define verification as:

Verification: *The process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.*

Verification, which we have not discussed much in this paper, deals with determining that a model is reliably doing what we asked it to do. In the computational science and engineering world where modeling often consists of software implementations of mathematical models of physical phenomena, verification deals with mathematics [28]. Verification generates evidence that the software which implements and solves systems of mathematical equations to represent the behavior we are interested in is doing the math correctly and repeatably. It answers the question "are we doing the math right?"

While some EBMs include components that themselves are mathematical models of physical phenomena, e.g., the power dynamics of an electricity distribution grid, most do not. Hence, verification in the EBM world tends to mean something different.

EBMs are constructed from experimental components like virtual machines and virtual networks that directly reproduce the behavior of the components they stand in for. The constituent pieces of an EBM are often instances of the same kinds of hardware like CPUs, network interfaces, and storage devices that occur in the real world. The behavior of an EBM can differ from that of a strictly hardware system because, in a model, these building block devices are wrapped using a software-based management technique, called virtualization, that allows one to oversubscribe a single component, like a CPU, such that many logical copies of that device appear to exist in the model. In reality, a single device is being shared among all the virtual instances using time or space multiplexing [7]. In other cases, differences come from the substitution of a software device,

like a software-defined network switch, for a hardware version of that device. These emulated or virtualized doppelgangers look similar enough to their hardware cousins that most software can use them transparently without changing much of its functionality, but may change the performance or behavior enough to impact the results of an experiment.

So, in the case of EBMs, verification consists of a modeler assuring themselves that the components making up a model perform in accordance with the modeler's expectations. That can mean that there are no major, relevant software bugs that influence the performance or behavior of the component in an unexpected way. But, it can also mean that an imaginary, completely bug-free implementation includes all the functionality and performance characteristics that the modeler will demand at runtime within an EBM. Software code reviews for quality assurance and functionality testing are two ways a modeler could gather relevant evidence that the model components from which an EBM will be constructed meet their expectations.

The verification task continues throughout the process of using and validating an EBM. It is often the case that, in order to maintain an adequate level of performance, a component, like a CPU or a network interface, must operate beneath an acceptable threshold of utilization or that a component, like a storage controller, must be able to achieve a minimum level of performance for the results of an EBM experiment to be considered useful. It is important that the modeler implement reliable runtime checks, which can assert that all such conditions hold throughout an experiment, before accepting the experimental results (e.g., see [10]). Ensuring that appropriate runtime assertions exist is also part of verification.

Verifying EBMs is essential for their credible use. There is a great deal of existing prior work related to software quality assurance [31], performance measurement, and specification testing [21] that a modeler can refer to when building a verification strategy for EBMs. Software verification is a useful activity that software users in general depend on. A modeler can often draw on a manufacturer's or open source developer's test plans and results to support the claim that the virtual machines and other components making up their model will perform according to their expectations, but in some cases new measurement studies will be needed [8] to ensure that key model components meet our needs.

There is far less work on what it means to *validate* an EBM and that is the focus of the following sections.

6.2 Validation

Oberkampf and Barone [22] define validation as:

Validation: *The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.*

In the computational science world, validation answers the question of whether we are doing the right math to answer our question. In the world of EBMs, validation tells us whether a model is

capable of answering the specific questions we are asking of it.

The general validation strategy is to test the predictive performance of a model. That is, one uses the model to make predictions about the behavior of the system of interest under carefully specified conditions, then one observes the behavior of the real world (or a real world surrogate) under identical conditions. The observations from each environment are compared and, ultimately, a judgment is made about whether the model's predictive performance is adequate for the purpose to which we want to put the model.

The basic idea of validation is simple, but there are a few complicating details that affect whether validation activities are providing useful evidence. For example, what specifically should we measure about the model and the real world? How do we choose which configurations at which to perform these validation comparisons? Exactly how do we perform the comparison? How does one choose the real world surrogate to compare to? We will begin to discuss some of these concerns in the following sections and demonstrate an end-to-end validation process via a case study.

6.3 Validation Process

Most validation efforts follow the same basic outline.

1. Understand the model's purpose
2. Prioritize important model features
3. Select observables of the model to compare, known as Quantities of Interest (QoI)
4. Identify the model parameters and value ranges
5. Select a real world referent to which to compare your model
6. Design validation experiments
7. Perform the validation experiments and collect data
8. Compare the QoIs using a pre-defined validation metric
9. Compare the metric results to validity thresholds
10. Collect a description of the full validation process and the metric results into a credibility case that can be used to judge whether a model is suitable for its purpose

Using case studies, which we introduce next, we will describe many of these steps and discuss how they apply to validating EBMs, focusing on those that are tricky or have important pitfalls.

6.4 Introduction to the Case Study

The case study we will use throughout our discussion of EBM validation is based on a model of part of a large enterprise network that was an unwilling participant in a distributed denial of service attack. The attack is known as an "amplification attack" and proceeds by finding instances of a service, available on the internet, to which an attacker can issue many, relatively small requests and which reply with relatively large responses. Ideally, from an attacker's standpoint, the request can be forged to claim to come from some other internet address. The forged source address of the many requests corresponds to the victim of the attack which will be bombarded with many large responses to requests it did not make. The ensuing traffic will consume the victim's network network resources and make it unable to perform its function.

In the specific case we will consider, the amplification mechanism was the domain name system (DNS) security extension (DNSSEC), a variant of DNS that provides additional security features. DNS is the service on the internet that translates names like "cnn.com", "npr.org", or "harvard.edu" into numeric internet protocol addresses which are used by software clients and servers to connect and communicate with one another. It works a bit like a telephone book which maps names of people and businesses to their numeric telephone numbers. DNS is one of the core, indispensable services on which the internet depends.

DNSSEC adds features that allow a user of DNS to verify that the information they receive about a name-to-address translation actually comes from the authorized source. It does this using a cryptographic mechanism called a digital signature. In addition to name-to-number translations, a user of DNSSEC can request a digital signature of a set of translations and, given access to the authorized publisher's public key, can verify that the translations are genuine. DNSSEC provides amplification to an attacker because the digital signature responses are substantially larger than the associated requests and the design of DNS allows a requester to forge the source address of a request.

In response to the attack on which our case study builds, the network operator quickly implemented a number of mitigations in combination to stop the effects of the attack. The mitigations included:

- Isolating the DNS servers from the rest of the core network
- Disabling query logging on the DNS servers
- Implementing DNS rate limiting

The EBM we will be validating was used, after the fact, to examine each of the mitigations individually and in combination to determine which were the most effective. One interesting outcome of the study was that the model predicted that mitigation two, disabling query logging, would actually make the attack worse by increasing its efficiency.

Throughout this discussion we will refer repeatedly to this case study and our efforts to validate this use of an EBM. The process included mistakes and failures, coverage of which we also include

because they are informative. In the following sections we will discuss each of the steps involved in performing validation of an EBM using examples from the case study as demonstrations.

6.5 Understanding the Model's Purpose

Because models, including EBMs, can be used for such a wide variety of purposes, each of which may have very different goals and requirements, there is little hope that a single validation process could convince a skeptic that a model is appropriate to use for all possible purposes. Hence, the goal of validation is to demonstrate that a use of a model for a particular purpose is appropriate.

Since it is the combination of a model and its purpose that one is validating, one must carefully understand that purpose prior to embarking on any validation effort. This seems like common sense, however, there is a consistent desire from funders and model users alike to have "validated" models sitting on a shelf, ready for any possible future use. Without a specific purpose in mind, however, validating a model devolves into trying to answer the question of if the model is good enough without knowing "good enough for what?". As clearly stated by Leslie Lamport, we should "state the problem before describing the solution" [18].

Without a stated purpose, even the development of a model turns into tail chasing as we endeavor to improve the general "fidelity" of the representation of a system, which results in an unguided effort to make a model achieve arbitrarily high levels of accuracy. An unlimited amount of time and effort can be devoted to improving the accuracy of a model without ever making it better for a purpose we are actually interested in.

Without a purpose, we don't know what to measure about a model and what to compare it to, so judging model quality or accuracy is reduced to expert opinion based on the perceived quality of the components or the virtue and pedigree of the researchers producing it.

A model's purpose drives all of the remaining validation process steps including what features will be important, what parts of the configuration space are relevant, exactly what measures will be compared to the real world, and how close model predictions must be to those observed in the real world to be useful.

In our case study, the purpose of the model is to predict the responses, both performance and behavior, of the DNS system as configured in this enterprise network while it was subject to the loads imposed by an amplification attack. We place special emphasis on the style and size of the attack that actually occurred. Since it was a particularly surprising result at the time, we are especially interested in the credibility of the model's claim that disabling logging on the DNS server leads to a more efficient attack.

6.6 Prioritizing Model Features

Without further guidance, a validation process can consume project resources without bound because there is nearly an infinite number of aspects of a complicated systems model that one could study and measure in an attempt to demonstrate suitability. One of the first stages of validation is deciding which aspects of the model are most important for your particular use. Importance is usually defined in terms of:

- How much an aspect or feature of the model influences experimental outcomes
- How closely related the aspect is to the model purpose
- Our ability to model a feature adequately using existing techniques and our uncertainty about that ability

By prioritizing the features you include in a model and understanding where you need to focus your validation attention, you can spend your time and effort validating the parts of the model most relevant to the model's purpose and very little time measuring irrelevant or already well characterized aspects of the scenario being modeled.

Multiple structured thinking processes have been invented to assist in this stage of the validation process. The details of the specific prioritization process you choose to follow are probably less important than the key idea, which is to systematically consider each potential feature of the model use and, using all available information, including previous validation or other experimental efforts and expert judgment, rank each feature along the following two dimensions:

1. The impact of the model feature on the quantities of interest you will be measuring
2. How much you know about the feature and your ability to model it adequately

Minimally granular labels like "Low", "Medium", and "High" are usually sufficient, but more detailed, quantitative labels can be used too.

The phenomena identification and ranking table (PIRT) process [29, 26], developed originally by Shaw et al. and subsequently refined by many others, focuses on feature impact and modeling uncertainty to choose priorities. Input/Uncertainty Maps [3], developed by Bayarri et al. do essentially the same thing for combined impact and uncertainty. Depending on the complexity of the system being considered, taking a hierarchical approach to organizing what the modeling and validation team knows about the impact and uncertainty of each relevant subsystem and component may be appropriate.

Small sensitivity experiments are helpful when determining how much influence a modeled feature has on the outputs of a modeled scenario. If experiments show that a including or excluding a

Feature	Impact	Modeling uncertainty
Core network	high	medium
DNS server software	high	low
DNS server platform	high	low
Attack load	high	high
Non-core network	low	high
Enterprise clients	low	medium
Attacker location	low	medium
Victim	low	medium

Table 1: *Simple feature prioritization table*

feature from the model or changing the values of an identified parameter have little impact on the parts of the scenario relevant to the stated model purpose, its impact can naturally be lowered in the analysis.

The key payoff of each of these structured thinking approaches is that, once the simple table or other representation has been filled out, the aspects of the model on which you should spend your limited time during the validation phase should be clear. They are the aspects that you have rated high importance or high impact and about whose model adequacy you are most uncertain, which is very intuitive.

Table 1 shows a simple priority ranking table that includes some of the model features we considered for emphasis in our case study. For the purpose of this validation exercise, we did not go through a formal prioritization process. The table captures our informal deliberations as a team. For more complicated validation studies, more rigor and broader participation by other subject matter experts would be warranted.

In the table, for each feature we identified, we assign a label indicating how much impact we believe it will have on the quantities we will be measuring. For example, we have rated "core network" high because the inclusion and status of the network connecting the computers in our model is obviously critical to our experiments. We also assign a label to each feature indicating how confident we are in our ability to model it adequately. If we believe our representation of a feature is adequate, it rates a low modeling uncertainty. If we have little experience with the corresponding model component or have less confidence in its adequacy we rate it high as we have done for "attack load" in Table 1.

For our case study, the prioritization process led us to believe that the key features for validation included only the DNS server, including operating system and application software, the network connecting the DNS server to the rest of the world, and the attack load. Other aspects of the larger model including the rest of the enterprise network, the geographic location of attackers on the internet, or details about the ultimate victim of the attack were deemed less important for the specific questions we wanted to ask.

When deciding how much to focus on variants of the DNS server platform configuration we performed a variety of small sensitivity experiments, varying the number of CPUs, the amount of RAM, and the network bandwidth and observed the effect on the DNS server performance as measured by response times for loads in the planned range. That exercise helped us come to the conclusion that, for this experiment and model, very little about the configuration of the machine running the DNS server made much of a difference to its performance in the range of loads we were considering other than the speed of the network. These conclusions will be reflected in the reduced model we use during our validation experiments.

6.7 Select Quantities of Interest

Validation is an empirical, performance-oriented process. It is based on measuring some quantitative feature of the model under specific conditions and inputs and comparing it with measurements of the same quantity in some form of the real world under the same conditions and with the same inputs. Hence, we must choose, based on our modeling purpose, what we will measure and why. This value or set of values is typically called the Quantity of Interest (QoI).

For the purposes of our case study, the question is whether the model can predict the performance of a DNS server under load. Hence, we need a way of measuring performance. What does performance mean for a DNS server? For our initial attempt at model validation, we chose to measure performance as the latency of DNS responses. This measures how long it takes the server to respond to a specific request when it is under load from many other background requests. Response time of the DNS server under load is our QoI.

Other example QoIs might include: the utilization of a server under load, whether a fault-tolerant service remains available after an induced failure, or the convergence time of a distributed consensus-based storage service.

It is important that the QoI you choose be available and measurable in the same way in both the EBM and the real world environments to which you will be comparing. If measurements from one of the environments must be post-processed before they can be used, the measurements from the other environment should be treated the same way. The validation process is going to perform direct comparisons between the values from each environment, so it is critical that confounding differences in the measurements resulting from experimental or data collection and processing artifacts are minimized.

6.8 Identify Model Parameters

To say something meaningful about the performance of an EBM, we must understand and, when possible, control the circumstances under which the EBM is used as precisely as possible. This is also true for other modeling techniques and applications. When using software simulations implementing partial differential equations, for example, it is critical to explicitly state and control

what the boundary or initial conditions are. The values in the model environment that are under control of the experimenter and which can change the measured outputs are model parameters. The selection of the parameters and the values they can take on specify the space in which the validation experiments are embedded.

In complicated EBMs, the number of knobs that could be varied can be huge. For example, in a model that may include thousands of virtual machines, every variable configuration setting for the platform, system software, and application software is a potential parameter. However, managing a large set of experimental parameters is impractical and, in our experience, almost never necessary. Hence, the researcher should narrow the set of explicit experimental parameters to only those relevant to the QoI and the study at hand. Sensitivity analysis may be helpful to determine which parameters are relevant.

For example, imagine that an EBM includes many virtual machines representing enterprise client computers. We have determined that we cannot simply eliminate such machines from the model because the outcome of the experiment depends on occasional interaction with them, but none of those machines is expected to participate in the experiment frequently or run at a high utilization. Instead of including the configuration settings of each of these low-interaction components of the model in our parameter set, we will simply fix each of the settings to a single value. The justification, which we can demonstrate using a few simple sensitivity experiments, is that differences in the configuration of these many machines will not materially affect the outcome of the overall experiment.

There are a variety of parameters types that may be handled differently when using and validating a model.

- Controlled parameters: parameters controlled by the experimenter that can take on a variety of known values
- Fixed parameters: parameters for which there is a single known value
- Uncertain parameters: values known to vary, but whose specific values are not known

Controlled parameters are at the heart of the design of validation experiments. By varying their values we specify what parts of the parameter space are relevant for our study. Because each instance of a validation experiment tends to be time consuming, we want to limit ourselves to values that actually tell us something new about the usefulness of the model for our purpose. We cannot usually afford to do all of the experiments we would like. Carefully choosing where to perform experiments in the, sometimes very large, parameter space is a key validation task, which we will cover in the discussion of design of validation experiments.

A fixed parameter may be a controlled parameter for which the experimenter has decided that only one value is relevant for the model's purpose. They can also represent some aspect of the model that, while controllable, has only one sensible value. While not variable, it is still important to

identify such parameters because their value could change the outcome of the experiment and they must be documented for repeatability.

Uncertain parameters are values that are known to exist, to affect the outcome of the experiment, and may or may not vary, but for which the experimenter does not know the value. In some cases, these could be inherently random variables that conform to some known or unknown distribution. In other cases, the value of an uncertain parameter may be fixed, but simply unknown. For example, when modeling a real network, one collects as much relevant information about the configuration of its elements and network topology as possible, but some values may remain unavailable and must be guessed.

While the values of controlled variables drive where in the configuration space a validation experiment occurs, the values of uncertain parameters drive, in part, the variability we see in the measured values of the QoI. For example, at a particular, fixed point in the multi-dimensional validation parameter space we will measure specific values of the QoI. We will usually replicate the experiment at that location a number of times, choosing values for uncertain parameters (or allowing random, uncontrolled variables within the experiment to vary according to some distribution) and capture the range of values of the QoI that we observe. The variance of the QoI tells us about the uncertainty with which our model predicts the real world QoI and is a key factor when comparing the performance of a model to acceptable thresholds. The observed variability is the sum of the intrinsic, uncontrolled randomness of the experiment, the uncertainty involved in our measurements, and the variability induced by the uncertain parameters.

Ensuring that parameter values match between situations that will be compared is critical. In cases where our real world referent is data from the real world, we must carefully collect the values of the parameters we have identified as they existed at the time our real world data was collected. Sometimes, that can be difficult and may lead to additional uncertainty. When uncertainty about measured real world parameters exists, it should be reflected in the model parameters during model runs. If we only know a real world value that is a parameter of the model was within some range, we should include multiple model runs that vary the uncertain parameter across that range. A design of experiments approach may be helpful to optimize parameter space exploration with multiple parameters.

In our case study, we performed a small set of sensitivity experiments that showed DNS server performance was not sensitive to any of the platform configuration values like number of CPUs or RAM, as long as they were in sensible, commercially available ranges. The exception was network speed. Since the network that was attacked had a specific, known configuration and this scenario was our main interest, we chose to fix all of the parameters of the DNS server, including network speed, to match the configuration of the server involved in the original attack as closely as possible. The only remaining parameter in our validation study is, therefore, the load level, i.e., the number of requests arriving at the DNS server per second, which varied during the actual attack.

It is important to remember that the number of locations at which you can choose to perform a validation experiment will be tiny compared to the number of locations where you could potentially locate an experiment. We will rarely have the luxury of testing the model at all locations in the

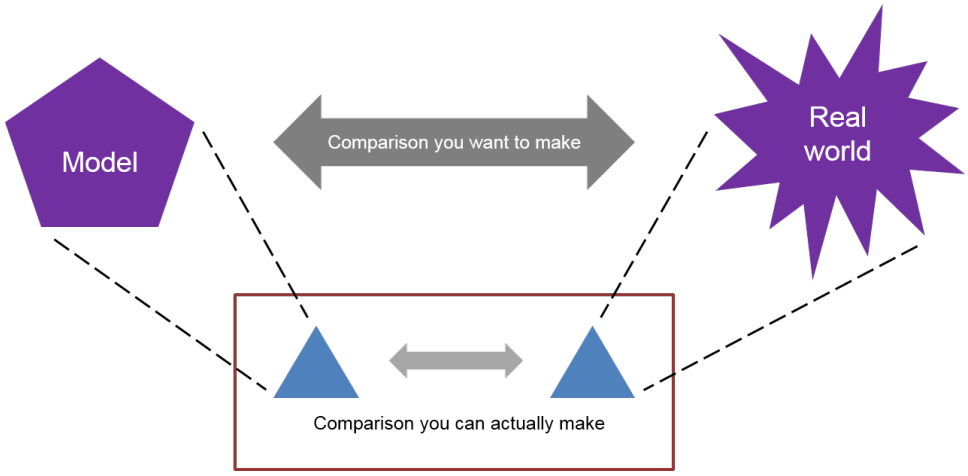


Figure 2: *Comparing a model to an experiment.*

parameter space that are relevant to our purpose. In fact, the number of such locations could be infinite. It is, therefore, common to have to interpolate between validation experiment locations or even extrapolate beyond the envelope of our experiments when judging the value of a model use. This topic will be discussed more in Section 7.4.

6.9 Select a real world Referent

The conceptual simplicity of validation is appealing. One just compares predictions made by a model to an equivalent set of measurements of the real world. Unfortunately, the choice of what to compare model predictions to is one of the trickier aspects of the validation process for EBMs. In most cases we will not be able to directly compare our model to the actual real world situation in which we are interested. We must usually compromise and compare our model to some, hopefully meaningful, subset of the real world. The exception is when we have the luxury of comparing our model outputs to data that has been directly collected from the real world systems we care about. We will discuss both situations below.

Figure 2, depicts one of the complications when choosing how to compare a model to the real world. Our desire is to compare the predictions from our simplified model to the behavior of the unvarnished real world as shown at the top of the diagram. If we don't have access to data from the real world, our next best option is to compare to a surrogate for the real world, or referent. A referent is usually a non-virtualized experiment. We cannot typically create an experiment that captures all of the complex behaviors of the real world. In fact, our ability to construct a real world experiment usually lags far behind our ability to construct a large, complicated model. The fact that it is hard to build rich distributed systems experiments without virtualization is one of the reasons we choose to use EBMs in the first place. Hence, our only option is to choose an experimental situation that is simple enough to implement within our resources and that still tells us something about the credibility of the model we want to use for our predictive purpose.

Our typical approach has been to *project* from the real world into a simpler, more manageable experimental space as depicted on the bottom right of Figure 2 then try to match the configuration of our model to the reduced experiment as closely as possible as shown on the left hand side of the diagram. Picking a reduced projection of the real world environment you are interested in as a validation referent has many implications about the claims you can subsequently make about a validated model. We need to carefully understand what aspects of the real world we are leaving behind when simplifying the referent and adjust our credibility case accordingly. If the reduced environment is not able to encompass all of the behaviors we are interested in validating, we may need to validate multiple sub models and make an argument that the collective set of validations provides adequate evidence for our use of composite model.

In the case study validation, our ultimate question is about mitigations for amplification attacks affecting a particular organization's network that is attached to the internet. The original results were based on a model that encompassed significant chunks of the enterprise network, its connections to the internet, and relevant pieces of the global routed autonomous system topology. In our validation, we have scaled our validation environment dramatically down from the full environment of the original enterprise network + ISP + internet model to a single DNS server under load. We then matched our EBM to the configuration of the referent so that we were comparing like to like.

We believe that the reduced environment, though much simpler than the original, incorporates the key features relevant to our question, namely how the performance of a DNS server changes in response to attack load. We justify the projection to the referent by noting that the performance of the DNS server is independent of what is happening elsewhere on the network given the load that reaches it. This DNS server is authoritative for several zones and can answer queries about those names without reference to any other system or service.

What if data collected from the real world is available to compare to? We certainly save ourselves the trouble and potential pitfalls of projection as described above, but there are other concerns. In the case when we use a reduced referent, we are ultimately comparing the results of a model and a referent that match closely as shown at the bottom of Figure 2. If we have real world data, it is far less likely that the model and referent will be obviously comparable. As shown at the top of Figure 3, we would like to compare data collected from the complex, messy, and all-details-included real world to an equivalently rich EBM. But, as a practical matter, we simply cannot include every aspect of the real world in a model. Hence, we must make the argument that the features we exclude from the EBM or the simplifications we make, as shown at the bottom of Figure 3 are irrelevant for our chosen purpose, which may not be easy or convincing.

6.10 Design of Validation Experiments

The design of a validation experiment is the selection of the locations in the parameter space where you will perform experiments. There is an extensive literature on designing good quality experiments that can effectively answer research questions about the effects of various levels of the parameters on the quantities of interest including parameter interactions and without having to

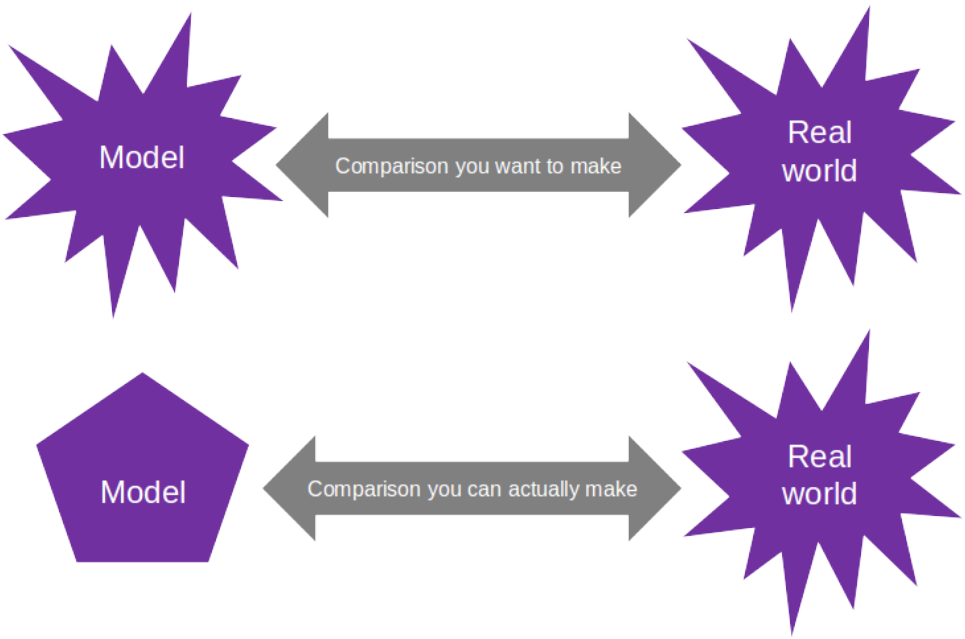


Figure 3: *Comparing a model to data collected directly from the real world.*

explicitly include all possible parameter combinations in the design. Montgomery [20] provides a thorough treatment of the design of experiments.

So far, our experience is that detailed and formal design of experiments has not been necessary, but two principles have helped us select where to perform experiments.

The first is simply an intuition that one should perform experiments in the parts of the parameter space that are important to the model purpose and about which one knows the least. If one part of the space is more important, perhaps because it is more common or the consequences for failure are higher there, then our attention should naturally focus there. If large parts of the space have already been considered and our ignorance is concentrated in some part of the space, perhaps we should focus our attention there.

We have no EBM-specific guidance for distributing the placement of experiments within the parts of the parameter space you have chosen to focus on. Whether one should focus on the extreme values, or evenly spread experiments throughout the chosen space, or place more experiments near an interesting transition is up to the experimenter based on the goal of the validation and what is already known. We have found it instructive to use the EBM itself to get a general flavor of the behavior of the QoI in the area where validation experiments will be executed. This is often feasible because running an EBM can be relatively inexpensive in terms of repeated setup and run time. Understanding how the QoI is changing often provides useful insights about where to place experiments to generate convincing evidence.

The second principle applies when the model purpose doesn't naturally drive us to choose experiments in one or another part of the space. Perhaps we are equally interested in the performance of

the model across the entire space, for example. However, we are interested in getting the best value from each of our limited validation experiments. In this case, we can take *informal* inspiration from Bayes formula that shows how new information updates a prior belief.

$$P(A|B) = \frac{P(B|A)}{P(B)} \times P(A) \tag{1}$$

In Equation 1, A is the event that a model is valid for our purpose and B is data collected during a validation experiment. $P(A|B)$ is the extent to which we believe, after observing a validation experiment, that a model is credible for its purpose. The change in our belief is totally determined by the first term. We increase our belief the most when the numerator $P(B|A)$ is large and the denominator $P(B)$ is small. Intuitively, a large $P(B|A)$ corresponds to observations, B, that consistently hold if the model is credible. Small $P(B)$, on the other hand, means that observing the outcome generally, i.e., whether the model use is credible or not, is rare. Hence, our informal intuition is that, all other things being equal, we get the most value by concentrating our experiments in areas where the model predicts consistent results that are also rare. These can correspond to rapid, but consistent changes in performance or pointed peaks in utilization, for example. If a model can correctly predict the fact of and location of such rare, but consistent, measurable features, our confidence intuitively grows rapidly.

We only use Bayes formula for intuition purposes and it does not allow us to calculate how much our confidence should grow given an experimental outcome. Others, however, are working on quantitatively prioritizing experiments using, for example, value of information techniques [15].

6.11 Perform Experiments

At this stage, one carries out the experiments described by the experimental design. The design specifies the initial conditions and other events that will occur at runtime during the experiment.

Collecting data is a key part of performing experiments that will be used as input to a validation process. The QoI must be measured at the appropriate place and the correct time in both the EBM and real world referent. As far as possible, the data should be collected in the same manner in both environments. For our case study validation experiments, we used exactly the same tools and techniques to collect our chosen QoI across the EBM and the real world experiments.

If time plays a role in the data collected, it is likely important to ensure that clocks are synchronized across the underlying cluster or, in some cases, within the experimental virtual machines.

Experimentalists know that the act of measuring can itself perturb the object being measured. When designing a data collection regime for a set of validation experiments, one should first endeavor to have as small an effect on the running experiment as possible. If the collection procedure does change what is being measured, which is often difficult to measure itself, one should try

to characterize that effect and take it into account during the comparison stage, especially if the effects are different in the EBM and the real world referent.

As with any experimental setup, it is dangerous to assume that an initial condition or runtime assumption holds during a validation experiment without verification. In fact, during certain stages of our case study experiments we showed that the nominal load to which we wished to subject the DNS server could not be sustained by the EBM environment. That fact called the results of the experiment into question and led to a diagnosis of why one variant of our validation exercises failed. All important experimental assumptions should be verified during validation experiments whenever possible.

For our case study, we used Firewheel [14], a Sandia-developed EBM orchestration framework that automates the creation and configuration of the various virtual machines and virtual networks that make up an EBM. Most of our research group's work happens in the context of Firewheel.

Other orchestration tools and approaches exist. While we have not gained much experience in using them, our intuition is that changes in the orchestration system or other parts of the experimental process, like the specific computer cluster used to host an experiment, could have significant side effects. Our current recommendation is, therefore, that validation-based credibility cases for a model be limited in their application to the orchestration framework and even the cluster that they were originally performed on until we have more experience ascertaining the true effect of those variables on experimental outcomes.

6.12 Compare QoIs

A validation case is based on our ability to compare measurements from a model to measurements made of some version of the real world. The technique for comparing those values is called a validation metric. The output of the validation metric is the error between the prediction made by the model and the, hopefully, more objective truth represented by the real world referent. Exactly how we perform the comparison depends on the form of the QoI. Originally we imagined that the QoIs we would measure for EBM-based validations would be large, multi-dimensional vectors of values with types ranging from continuous values to categorical values to booleans. Consequently we developed more involved validation metrics that could work for these complicated QoIs based on concepts borrowed from cryptographic security games [4]. In fact, we have not yet had a need to use them as all of the QoIs we have been exposed to have been simple continuous scalars.

Oberkampf and Barone [22] do a thorough job of describing the features that make up a good validation metric for comparing mathematical models to experiments. In the case of EBMs, we are comparing measurements from one set of experiments, the model, to the measurements of another set of experiments, the referent, or to data collected from the real world. Oberkampf and Barone cite six features of a good metric. The key characteristics that directly apply to EBMs are:

1. A metric should be quantitative (vs qualitative)

2. A metric should take experimental uncertainty into account

The metric should be quantitative because we will eventually compare its value to a pre-defined threshold that defines how accurate the model must be to fulfill our modeling purpose. If all we have, for example, are two plots showing the QoI for each of the model and the real world referent, it will be difficult to apply this filter.

The metric should take uncertainty into account for the same reason. If the point estimate of the mean error across multiple replications of our validation experiment lies beneath our threshold, but the variance is high enough that individual instances of the experimental results would fail the threshold test, we need to know that.

While many validation metrics have been proposed. We have found that a simple statistical hypothesis test for the difference of the means works well for the limited number of EBM validation experiments we have performed. The test simply shows that the difference of the means of the QoIs from each environment is less than a predefined threshold value by rejecting the null hypothesis that the difference of the means is greater than or equal to the threshold. The test naturally takes the variance of the two sample sets into account and returns a high confidence result that the threshold was not exceeded. This is the approach taken in our simple case study.

Oberkampf and Barone emphasize the need to segregate the characterization of the error between model prediction and experimental observation from the judgment about whether that error is acceptable for the given purpose. One reason is that the decision about whether to use a model or not depends on many factors, only one of which is the quantitative error between prediction and measurement. The simple hypothesis test we describe could be said to violate this principle. A similar procedure that strictly segregates characterization of error and judgment about the error would be to produce a confidence interval around the estimated error and use that as input to the credibility judgment.

6.13 Create a Credibility Case

A credibility case is the artifact provided to whomever will decide whether or not to use a model for a particular purpose. A credibility case based on validation should include:

1. A clear statement about the intended use of the model
2. The PIRT or other process output that was used to select what features of the model to prioritize for validation
3. A description of the real world referent and a justification for any reduction or deviation from the environment in which the model will be used
4. The validation experiment design including each configuration space location where a validation experiment was performed and why

5. The definition of the QoI, a justification for how it relates to the intended use of the model, and how the measurements were collected
6. The definition of the validation metric
7. The experimental data that are the input to the validation metric
8. A statement of the performance threshold and a justification for why we believe a model is useful for the given purpose if it can attain the corresponding level of performance
9. The results of the validation metric and how they compare to the performance threshold

The credibility case is one input for the decision to use a model along with a variety of other, sometimes non-technical concerns. Some examples of external factors include organizational appetite for risk, the consequences of model failure, or whether and how an intended use must extrapolate away from established validation experiment locations. Ultimately, the decision to use a model involves engineering judgment. The results of a good quality validation study, however, can have an appropriately large influence on the decision.

7 Validation Case Studies

In this section we will detail two end-to-end, but very simple, EBM validation exercises and present each in the form of a credibility case. The first attempt fails our stated credibility threshold. The second succeeds by restricting the model purpose.

7.1 Validating an Absolute Performance Prediction Use

Our emulation-based model was used to evaluate a variety of mitigations applied to counter a historical DNSSEC-based amplification attack on an actual enterprise network. Results from the model were used to compare the effectiveness of multiple mitigation strategies. One important measure of the effectiveness of a mitigation was the extent to which the ability of the DNS server to respond to queries that were unrelated to the attack were slowed down or prevented. For this case study, we wish to use an EBM to predict the absolute performance of a specific DNS server under a large amplifying load implemented using requests for DNSSEC signatures.

Intended use of the model

Note that our first attempt at validating this model focuses on a relatively broad question about whether an EBM is able to predict the absolute performance of a DNS server in a configuration comparable to that used during the attack described in our case study discussion when the server was under loads seen during that attack. In other words, would the DNS server within the EBM exhibit the *same performance* as an identically configured DNS server in the (non-virtualized) real world?

We believe this use case satisfies each of the four criteria for validatability described by Hodges and Dewar [11]. The value we will predict is DNS server performance as indicated by response times which are easy to measure. We expect performance to remain consistent across time and be independent of elements that we chose to exclude from the model. Finally, it is easy for us to collect ample data from both the EBM and our real world referent.

Prioritizing Features

As discussed earlier, we eliminated most of the original model from consideration because our validation question had to do only with the performance of the DNS server under load. The topology of our validation experiment is shown in Figure 4. The same topology is used in both the EBM and real world referent cases. The configuration of the DNS server machine is fixed to approximately match the configuration of the server that was in use during the attack in terms of the number of CPU cores (16), the amount of RAM (16GB), and the attached network (1 Gbps). The experimental outcomes were not sensitive to the configuration of the loader and prober machines. For the

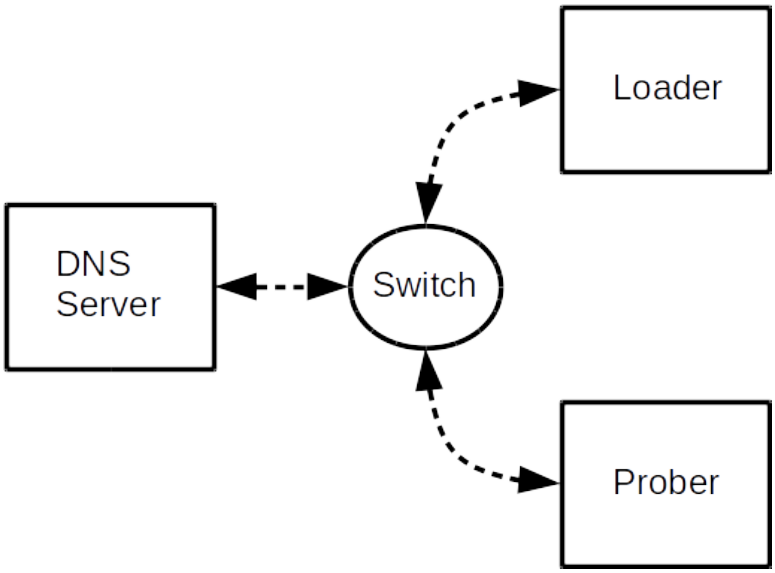


Figure 4: *Performance validation experimental setup.*

purposes of this experiment their configuration was set to 16 CPU cores, 90 GB of RAM, and the same 1 Gbps network to which the DNS server was attached.

The single variable to which the experiment outcome appeared to be sensitive was the level of the network load. For the purposes of this experiment we will vary the load from 10,000 requests/second to a maximum of 100,000 requests/second in steps of 10,000 requests/second. This range encompasses the load observed during the actual attack.

Real World Referent

The referent to which we will compare the output of the EBM has the same topology as shown in Figure 4. It was implemented using physical nodes of the same computer cluster on which we performed the EBM experiments. The DNS server, loader, and prober were each a single physical machine. The configurations of the machines were limited to that of the corresponding virtual machine making up the EBM by selectively disabling CPU cores, making RAM unavailable, and forcing the network interfaces to use a reduced signaling rate.

Experimental Design

For the machine configuration described above, in each of the EBM and real world referent environments, we vary the load between 10,000 requests/second and 100,000 requests/second in steps of 10,000. The load is sustained at a given load level for 150 seconds, enough time to generate 30 DNS probe requests 5 seconds apart.

Quantity of Interest

The quantity that we will measure and compare between the EBM and real world referent environment is the response time of each probe request. This is the interval between the moment a request is sent and the instant that a response is received. If a response is not received within 5 seconds, the request is considered dropped. 5 seconds is the default timeout interval for the Linux DNS resolver library. Dropped requests are represented in the output data as negative response times.

Latency, which is what we are measuring here, is the natural measure of performance for a server of this type. Each DNS request is typically considered independent. Other potential measures like throughput make less sense for a DNS server.

Validation Metric and Performance Threshold

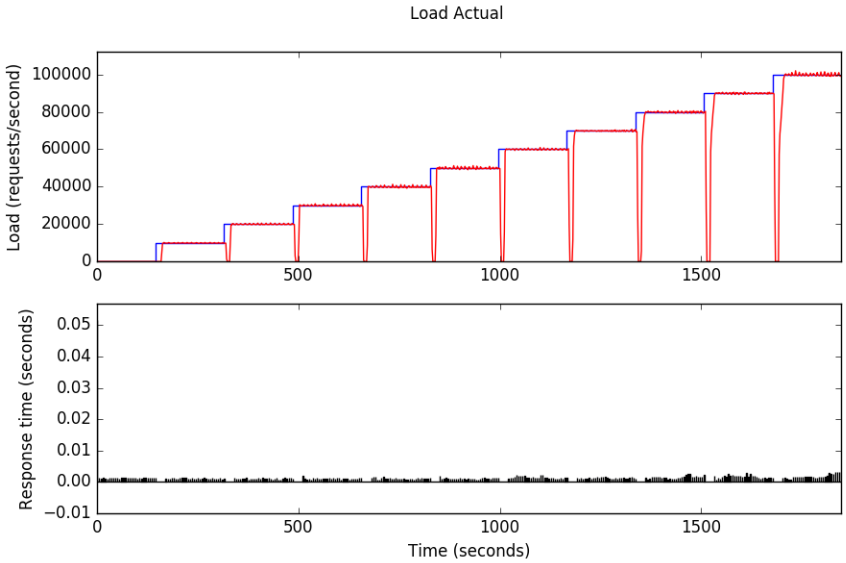
Our validation metric in this experiment will consist of comparing the mean response time for each load level between the EBM and the real world. Our threshold is indistinguishability, i.e., for the model to be useful as a performance predictor, the output from the EBM and the real world should be statistically indistinguishable. We define indistinguishable in this case as the inability to reject the statistical null hypothesis that the difference between means is zero at each load level.

Experimental Results

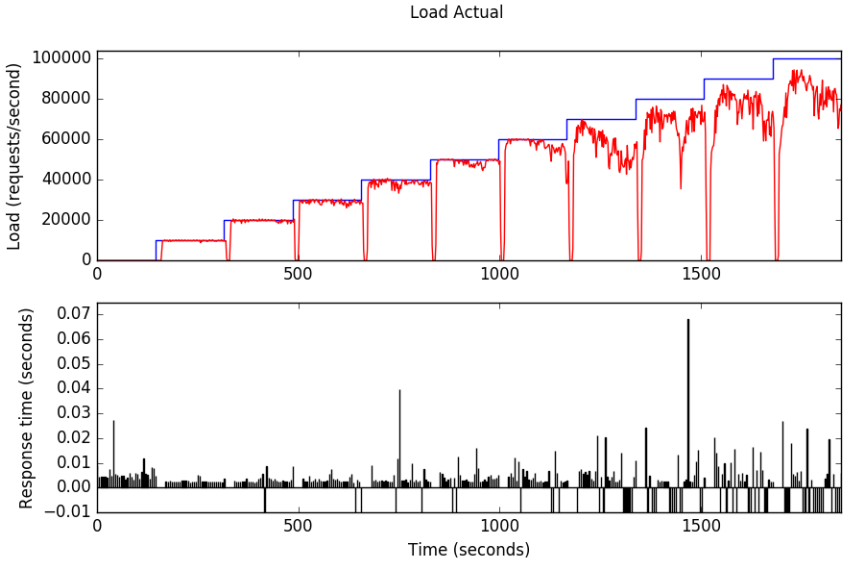
Figure 5 includes two pairs of graphs that show the resulting data from the performance validation experiment. Figure 5(a) shows the results from the real world, non-virtualized version of the experiment. Figure 5(b) shows the EBM results. The top graph within each pair shows load. The blue line indicates the requested load, the red line indicates the load that we measured actually reaching the DNS server. The bottom graph within each of the pairs shows the response times for each probe request. Bars with negative values in the bottom graph indicate a timed-out, assumed dropped, request.

Our goal with this validation experiment is to compare the response time results and, if similar, use that as evidence that our model is capable of predicting absolute DNS server performance under load. We exercised our validation metric and find that it is possible to distinguish the mean response times with high statistical significance ($p \ll 0.01$). Looking at the plots it is easy to see that the response times across the two environments are very different. In the EBM case, there are many dropped requests and the requests that are not dropped have much higher response times than in the real world case.

If we look at the data a little more closely, we can begin to see what is happening. In the real world, as depicted in Figure 5(a), our experiment is easily capable of sustaining the loads we request. The red and blue lines match each other closely. In the EBM case as depicted in Figure 5(b) it is clear that we are unable to sustain the requested loads. This is the first indicator that our EBM is



(a)



(b)

Figure 5: *Performance validation data.*

experiencing a performance problem, but not exactly what that performance problem is.

With additional investigation we found that the core issue has to do with small packet network performance under KVM, our chosen virtualization platform and the platform used by the original experiments which we are attempting to validate. The load we supply consists of many small DNS requests, whose size is about 50 bytes per packet. Because of the additional interrupt overhead imposed by virtualization and the very large numbers of interrupts imposed by numerous small packets, our system begins to bog down, lose packets, and is unable to match the small packet throughput of a real world system. This effect was verified using other loading and measuring tools outside our validation experiment. It is important to note that this problem is not a characteristic of all virtualization platforms. We performed experiments with other hypervisors that do not exhibit this performance issue. The difference is likely due to aggressive interrupt aggregation. The problem does not extend to flows with larger packet sizes and commensurately lower interrupt rates. KVM is capable of easily saturating a 1 Gbps link using larger packet sizes.

Judgment

Our validation experiment provides evidence **against** using this model to predict the absolute performance of a DNS server under heavy load. It fails to meet our pre-defined performance threshold.

7.2 Validating a Logging Effect Prediction Case

The previous simple validation example showed an instance of failing to provide evidence supporting the use of a model for a specific purpose. In fact, we generally advise against using EBMs to predict absolute performance for just this reason. There are many performance pitfalls that are sometimes non-intuitive and that have not yet been completely characterized.

In this section, we make another attempt at validating a nearly identical model for a related, but more focused, purpose. It demonstrates that a model which is invalid for one use, may be perfectly suitable for another, usually more restricted, use. This is one of those cases.

Intended Use of the Model

As we discussed earlier, one of the key outcomes of the original research effort described in the case study was that disabling DNS query logging made the amplification attack significantly more severe. In this set of validation experiments, we ask whether our model is capable of predicting two aspects of this effect:

1. The presence of the effect (i.e., does the severity of the attack change in response to whether query logging is enabled or disabled?)

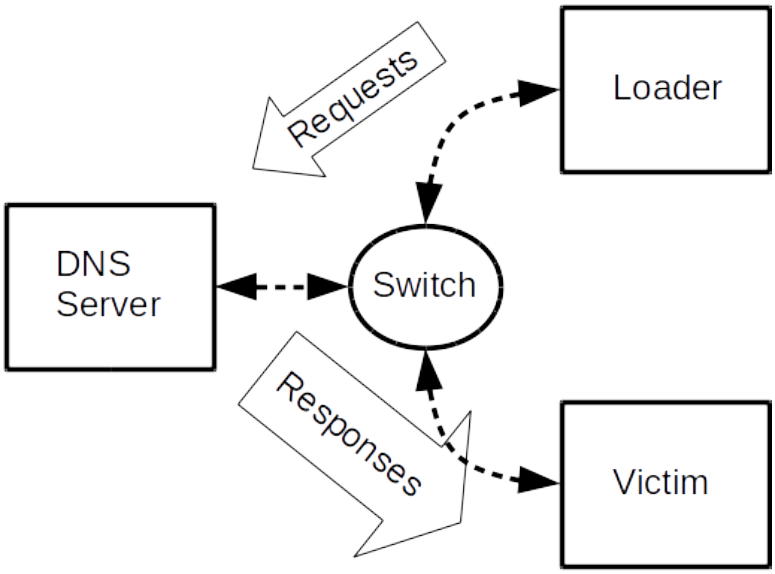


Figure 6: *Logging validation setup.*

2. The direction of the effect (i.e., does the severity of the attack get better or worse?)

Prioritizing Features

We use the same set of features and justifications for choosing those features as in the previous case study. Our topology changes slightly to accommodate the new validation goal as shown in Figure 6. We still maintain three machines, but their roles and titles change slightly. The DNS server still serves DNSSEC signatures in response to attacker requests supplied by the loader. Instead of a prober machine that directly measures DNS server performance under load, we introduce a victim machine that is the ultimate recipient of the DNSSEC signature responses and which will be the basis of a new quantity of interest described below.

The configuration of each of the EBM virtual machines remains the same as for the previous case study.

Real World Referent

The referent to which we will compare the output of our EBM is identical to the topology shown in Figure 6. We match the configurations using the same hardware limiting techniques described previously.

Experimental Design

The experiment scenario is that the loader issues requests for DNSSEC signature records to the DNS server just as before. For this experiment we have fixed the target load to 100,000 requests/second as this corresponds approximately to the peak of the real world attack and our question is about the effect of logging at that peak rather than how DNS server performance changes with load as before. The source address of the requests is spoofed by the loader to that of the victim, so responses will be sent from the DNS server to the victim.

Our experimental design has one controllable variable, namely whether query logging is enabled or disabled. Query logging is a feature of the Bind DNS server [1] which writes a record of every query request to disk. During the experiment, the loader makes requests to the DNS server for a period of 120 seconds. During that time the count of response packets seen by the victim is recorded each second. We replicate the experiment 10 times for each case of query logging enabled and query logging disabled.

Quantity of Interest

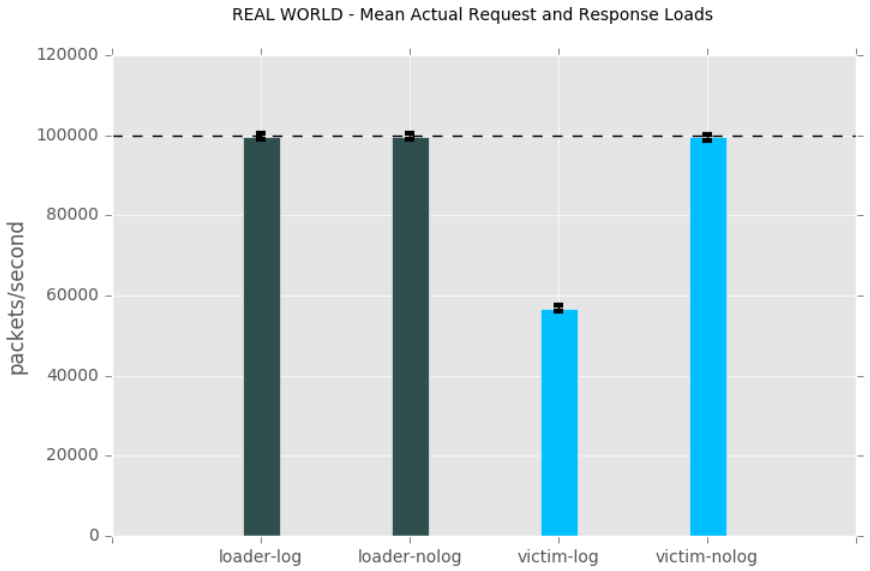
Our new quantity of interest is the severity of the attack as measured by the rate of attack response packets incident on the victim measured in packets/second. This matches our new purpose which is focused on measuring the difference in the intensity of the attack in response to changes in query logging. We are no longer measuring or comparing the absolute performance of the DNS server. We are measuring a change in the severity of the attack that occurs as a result of enabling or disabling query logging.

Validation Metric and Performance Threshold

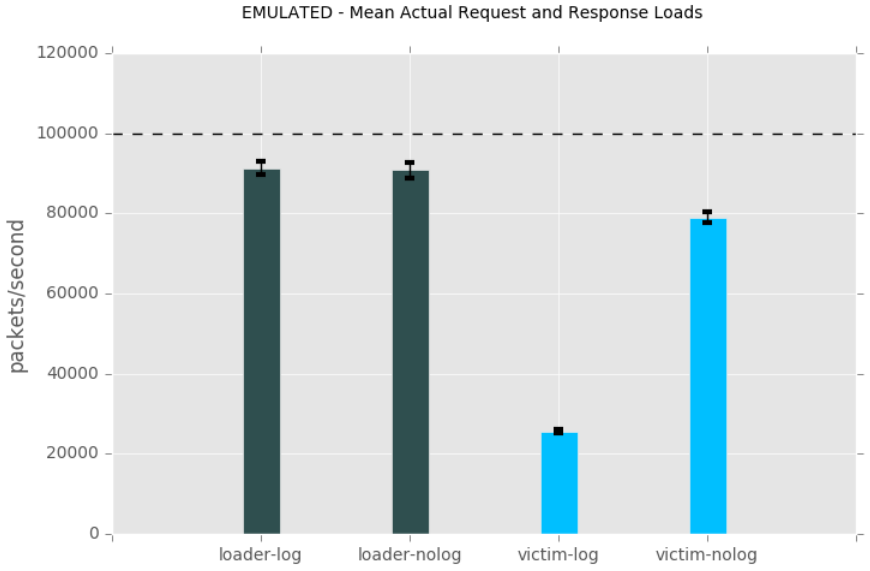
The performance threshold against which we will judge the model has three components:

1. Whether the predicted effect exists in both the EBM and the real world experiment
2. If the effect has the same direction in both environments
3. If both are "noticeable", defined as at least 25% more severe under the no logging case as under the logging case

Our validation metric is a simple Welch's t-test across our 10 samples that measures our ability to reject the null hypothesis that the difference between the no logging and logging attack severity is less than 25% of the logging case.



(a)



(b)

Figure 7: *Summary logging validation experiment results.*

Experimental Results

Summary results for the experiment are shown in Figure 7. There is one plot for each of the real world (Figure 7(a)) and emulated (Figure 7(b)) cases. The left two bars show the load in requests/second as measured arriving at the DNS server for each of the logging and no logging cases. The right two bars show the load in responses/second as observed arriving at the victim for the logging and no logging cases. The graphs show that both the real world and emulated cases exhibit the same kind of logging effect that was predicted by the original study, namely that disabling query logging increases the efficiency of the amplifier and makes the attack more severe at the victim. A t-test strongly rejects ($p < 0.0001$) the null hypothesis that the difference between the no logging and logging attack severity is less than 25% of the logging case.

Judgment

This second validation case study provides supporting evidence that our model is appropriate to use for the more limited predictive purposes we set out at the beginning of the section. This is despite the fact that the same small packet performance problems exist, which can be seen in the reduced load the emulated case is capable of supplying and in the significantly smaller load observed at the victim in the emulated case vs the real world case in Figure 7.

We believe the key difference that allows the second case study to succeed where the first failed is the fact that, in the second case, the prediction we expect the model to make concerns a relative, rather than an absolute, performance effect. Reproducing relative performance and basic system behaviors that are not particularly sensitive to the performance bias imposed by virtualization is something that EBMs are often suited for. Because of the tolerance that most software has to virtual time [13], many interesting distributed systems questions not related to absolute performance prediction are actually good candidates for EBM studies.

7.3 Case Study Summary

The validation cases we have presented are intentionally simple and easy to follow. There is a lot of room to perform more capable and, consequently, more involved and complicated validation tests. The key point we want to make with this paper, however, is that, as a community, we are very early in our thinking about EBMs in this more critical and rigorous way. Our first steps should be to encourage and support the notion that we need to justify our uses of EBMs to ourselves as creators of EBMs and to those who want to use their results.

7.4 A Note on Interpolation and Extrapolation in EBMs

Interpolation and extrapolation involve deducing values at locations we have not measured. Interpolation imputes values that occur between measured points and extrapolation does the same for points that lie outside the envelope of our measurements. For smooth, continuous processes, interpolated values can represent the underlying process well. Extrapolation of trends beyond what we have measured is always fraught with danger and uncertainty.

EBMs include some features that might help us have more confidence when interpolating between measurements and some that may cause greater uncertainty. EBMs exhibit a lot of internal structure. They are strong explanatory models that directly replicate many of the inner workings of the systems they represent rather than tracking an externally visible response. Hence, EBMs could be more resistant to the simplification effects that strictly descriptive, statistical, or response surface models fall prey to.

While a computing system is capable of exhibiting arbitrary, non-continuous behavior, they are often designed to avoid exactly that, i.e., to be predictable. Hence, in the typical performance regimes that systems often inhabit, highly unpredictable behavior is less likely by design. But, when using EBMs to study extreme behaviors outside the normal design space or when investigating the behaviors of combinations of systems that developers may not have foreseen, interpolation should be used with great care and little confidence.

8 Evaluating Non-predictive Uses of EBMs

EBMs are used in many situations that will not be amenable to strict validation. In fact, some of the most useful applications of EBMs like generating ideas, training environments, or doing demonstrations fit into that category. Being able to generate some confidence in these uses of EBMs is just as important as validation is in the predictive modeling case. In the following sections we consider other, non-predictive uses of EBMs and introduce some ideas for how one might evaluate such an EBM use.

8.1 Training

Evaluating whether an EBM is providing an effective environment for a particular training use case is a sub task of evaluating whether the training itself is effective. The first step in that process is being able to clearly state and understand what the training objective is. Unfortunately, this is more of a problem than it should be. Often training occurs because of a top-down mandate to "do training" without specific, well-reasoned goals attached.

Assuming a clear training objective does exist, the next question is whether we can detect if the objective has been achieved. Some objectives may not lend themselves to being measured or even observed, especially in the short term. An objective like "improve cyber defense skills" is too vague for any sort of immediate measurement to be possible. Other, more specific and measurable objectives like "reduce average incident response time from detection to deployed mitigation for internet-based denial of service attacks from the current 3 hours to 10 minutes" are specific and measurable by testing multiple instances of response times before and after training. As with other evaluation processes, the first, most important, but often missing, step is simply stating what the purpose of an activity is and ensuring that there are values we can measure that tell us something about the extent to which that purpose is being achieved.

There are at least two additional complicating factors when evaluating whether EBMs are being used effectively for training. The first is whether a training effect achieved using the EBM will manifest itself in the real world. It may be that the performance of a trainee increases measurably within the training environment, but the performance of the same person or group remains unchanged or even gets worse in the real world. This problem is not unique to training exercises using EBMs. In fact, skills learned in one real world situation are not guaranteed to be portable to a different real world situation. In any case, it is important to measure training effectiveness in the real world environments we actually care about in addition to measuring performance within the training environment.

The second complicating factor when evaluating EBM use for training is diagnosing whether a training failure, that is, a failure to measurably improve the performance of a trainee, is attributable to the EBM environment or some other factor. Training can be ineffective for many combinations of reasons and it can be difficult to assign causality to each potential factor. Once again, this is not a problem specific to EBM use in training environments. The same issue comes up in any

multi-factor measurement study. With enough data and carefully designed experiments we can demonstrate the relative contribution of various treatments to a measured outcome using well-understood statistical techniques. While possible, it seems unlikely that training organizers will spend the required effort and resources to rigorously diagnose why a training exercise failed and what was to blame.

Luckily the difficult parts of both of the complicating factors described above for evaluating EBM use in training scenarios only occur when we fail to meet the overall training objective and the exercise provides unacceptably low benefit. As long as one can show, via measurements of performance in the real world, for example, that a training exercise meets its objectives, no additional effort is required. In the case of failure, other heuristic or intuitive techniques will, in most cases, be cheaper and more appropriate for generating alternative training scenarios to test.

Operations

We include a discussion of evaluating operational uses of EBMs in the section on testing because they share the same underlying question, namely "Did it work?". When using an EBM to create a deception network, for example, we are far less interested in whether it is an accurate representation than if it actually worked to reveal or dissuade an adversary. Instead of evaluating the EBM, we evaluate the operational purpose of the EBM directly.

8.2 Test and Evaluation

Using an EBM as an environment in which to perform T&E is like training in some senses and more like prediction in others. Like training we are simply using an EBM as an environment in which to perform some other activity. When thought of this way, the T&E application of EBMs falls prey to some of the same pitfalls as training. If the T&E process fails, to what do we attribute the failure: the EBM or some other aspect of the test plan?

If, however, one thinks about the T&E application as a whole, where the EBM is an integrated part of the full testing and evaluation process, we can think about it more in terms of a predictive process where validation might apply. A T&E study using an EBM is predictive in the sense that results from tests carried out in the virtualized environment are expected to hold in the real world. There would be little purpose in performing the tests if we didn't believe the tests would indicate what the performance of the system under test would be when it is deployed.

In just the same way we perform a limited number of validation experiments to build a credibility case for the continued use of scientific models, we can perform a small number of paired validation tests for the full, end-to-end T&E process in both an EBM environment and in the real world. If the agreement satisfies our carefully chosen performance thresholds, we gain confidence that a more extensive testing regime, executed solely within an EBM, will satisfy the larger T&E requirement. So, T&E in a virtualized environment can't be a complete substitute for real world testing, but it

can be used to reduce the number of (often expensive) real world tests in a justifiable way that doesn't sacrifice our ability to test under a usefully large set of conditions.

The same T&E philosophy would seem to apply to other kinds of testing that use a reduced experimental model of reality for testing purposes. Wind tunnels are used extensively for this purpose to test air frames, wings, air sampling systems, and many other devices. Our informal survey of the literature citing wind tunnel testing, however, rarely mentions evaluating whether a specific use of a wind tunnel is credible. Part of the lack of evaluation may be due to the regular and extensive calibration process that many large wind tunnels are subject to. This approach doesn't apply to EBMs. EBMs are typically custom built for a specific purpose. A standard calibration process would not work generally for the broad diversity of possible EBMs.

If we change our perspective on T&E using EBMs we get some distinct advantages. By switching away from the view that EBMs simply provide a convenient environment in which to execute an independently developed test plan that could just as easily be performed in a non-virtualized environment, to one where we think about the full, end-to-end test as an integrated unit, we transform the T&E process into a predictive use of a model. If we fulfill the requirements for validatability, we can then bring the full power of validation to bear to build confidence in the virtualized, and much less expensive, T&E outcomes.

8.3 Idea Generation and Demonstration

There are two other main uses of EBMs we have discussed. They both have more limited needs when it comes to evaluation, but still require some vigilance.

The first of these uses is idea generation. EBMs are often used in this way and it is an excellent and productive way to use the technology. Idea generation is typically an early stage, rapid turnaround, low rigor activity. The ability to try something out, make some observations, and iterate quickly is much more important than convincing a skeptic that your early stage musings are correct. Exploratory uses of EBMs like these will all need additional work before they are ready to withstand much criticism. It doesn't make sense to spend a lot of time evaluating such disposable models. However, thinking about verification is still useful. Having sharp tools whose properties we understand can enhance even early stage, exploratory uses of EBMs.

Building demonstrations is another use case we considered. In contrast to idea generation, demonstrations typically come much later in an investigation process. Here we are using the convenience of EBMs to demonstrate an idea that has been proven out in some other way. The existence of an EBM-based demo should never be used as evidence that the underlying ideas are true any more than a cut-away model of a complicated mechanical process tells us anything about its reliability or performance. The point is to communicate an idea. Any evidence for or against the demonstrated idea must come from a separate, and carefully considered, process.

In both of these cases, the most important factor is consistently communicating the true status of a model to those who are consuming its results. For an outsider, it will be difficult to distinguish

whether a model has been carefully evaluated or not. The responsibility falls on the model developer or user to honestly represent the level of confidence we should have in a model and be prepared to provide supporting evidence.

Finally, we must always resist the very common demand to use an unevaluated model in a predictive way or for evidenciary purposes. The pressure to do so can sometimes be enormous. An unevaluated EBM can be a convenient and powerful tool, but we should not misuse it.

9 Conclusion

Emulation based models of computing, communication, and control systems are powerful tools for observing, learning about, and, in some cases, predicting the behavior of complicated systems that we don't understand well. We have seen EBMs used in many different ways spanning training, test and evaluation, evaluation of engineering design alternatives, or even making credible predictions of distributed systems' behavior.

As with any model, we should ask ourselves if we are using EBMs in a credible way. We can convince ourselves and others by building a credibility case based on techniques that range from rigorous comparison of model predictions and real world observations to careful thinking about what we expect of a model and its ability to deliver what we want.

In this paper, we have laid out a few approaches to building such a case, spending the most time on validation, the preferred technique when we expect a model to reproduce the behavior of a system in the real world. Other model uses don't merit the extensive effort demanded by validation. Nevertheless, careful credibility thinking is warranted to ensure that we use our tools in the appropriate way.

In all cases we should be ready to justify how we have chosen to use these flexible tools and honestly communicate to what extent we should trust their results.

References

[1] Bind open source dns server. https://www.isc.org/downloads/bind. Accessed: 2017-08-07.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, volume 37, pages 164–177. ACM, 2003.

[3] Maria J Bayarri, James O Berger, Rui Paulo, Jerry Sacks, John A Cafeo, James Cavendish, Chin-Hsu Lin, and Jian Tu. A framework for validation of computer models. *Technometrics*, 49(2):138–154, 2007.

[4] Mihir Bellare and Phillip Rogaway. Introduction to modern cryptography. *Ucsd Cse*, 207:207, 2005.

[5] Roman Chertov and Sonia Fahmy. Forwarding devices: From measurements to simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 21(2):12, 2011.

[6] Roman Chertov, Sonia Fahmy, and Ness B Shroff. Fidelity of network simulation and emulation: A case study of tcp-targeted denial of service attacks. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 19(1):4, 2008.

[7] Robert J. Creasy. The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490, 1981.

[8] Jonathan Crussell, Thomas M. Kroeger, Aaron Brown, and Cynthia Phillips. Virtually the same? the empirical differences between physical and virtual networks. Technical report, Sandia National Laboratories, 2017.

[9] Sally Floyd and Eddie Kohler. Internet research needs better models. *ACM SIGCOMM Computer Communication Review*, 33(1):29–34, 2003.

[10] Brandon Heller. *Reproducible network research with high-fidelity emulation*. PhD thesis, Stanford University, 6 2013.

[11] James S Hodges, James A Dewar, et al. *Is it you or your model talking?: A framework for model validation*. Rand Santa Monica, CA, 1992.

[12] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[13] David R Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(3):404–425, 1985.

[14] Stephen T Jones, Kasimir G Gabert, and Adam Vail. Intro to distributed system experimentation using firewheel. Technical report, Sandia National Laboratories, 2017.

[15] Jaydeep M Karandikar, Christopher T Tyler, Ali Abbas, and Tony L Schmitz. Value of information-based experimental design: Application to process damping in milling. *Precision Engineering*, 38(4):799–808, 2014.

[16] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux symposium*, volume 1, pages 225–230, 2007.

[17] Dmitri Krioukov, Marina Fomenkov, Fan Chung, Alessandro Vespignani, Walter Willinger, et al. The workshop on internet topology (wit) report. *ACM SIGCOMM Computer Communication Review*, 37(1):69–73, 2007.

[18] Leslie Lamport. State the problem before describing the solution. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/State-the-Problem-Before-Describing-the-Solution.pdf.

[19] W. J. McCroskey. A critical assessment of wind tunnel results for the naca 0012 airfoil. Nasa technical memorandum 100019, National Aeronautics and Space Administration, 1987.

[20] Douglas C Montgomery. *Design and analysis of experiments*. John Wiley & Sons, 2017.

[21] Glenford J Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[22] William L Oberkampf and Matthew F Barone. Measures of agreement between computation and experiment: validation metrics. *Journal of Computational Physics*, 217(1):5–36, 2006.

[23] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[24] Roberto Paleari, Lorenzo Martignoni, Giampaolo Fresi Roglia, and Danilo Bruschi. A fistful of red-pills: How to automatically generate procedures to detect cpu emulators. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2009.

[25] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. The design and implementation of open vswitch. In *NSDI*, pages 117–130, 2015.

[26] Martin Pilch, Timothy Trucano, JL Moya, Gary Froehlich, Ann Hodges, and David Peercy. Guidelines for sandia asci verification and validation plans-content and format: Version 2.0. *Sandia National Laboratories, SAND2000-3101, Albuquerque, NM*, 2001.

[27] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.

[28] Patrick J Roache. *Verification and validation in computational science and engineering*, volume 895. Hermosa Albuquerque, NM, 1998.

[29] RA Shaw, SZ Rouhani, TK Larson, and RA Dimenna. Development of a phenomena identification and ranking table (pirt) for thermal-hydraulic phenomena during a pwr large-break loca. *INEL, NUREG/CR-5047*, 1988.

[30] Jim Smith and Ravi Nair. *Virtual machines: versatile platforms for systems and processes*. Elsevier, 2005.

[31] Jeff Tian. *Software quality engineering: testing, quality assurance, and quantifiable improvement*. John Wiley & Sons, 2005.

[32] Vincent E. Urias, William Stout, and Caleb Loverro. Computer network deception as a moving target defense. Technical report, Sandia National Laboratories, 2015.

[33] Vincent E. Urias, Brian Van Leeuwen, William Stout, and Han W. Lin. Dynamic cybersecurity training environments for an evolving cyber workforce. Technical report, Sandia National Laboratories, 2017.

[34] Bart Van den Broeck, Pieter Leys, Jan Potemans, Johan Theunis, Emmanuel Van Lil, and Antoine Van de Capelle. Validation of router models in opnet. In *Proc. of OPNETWORK*, 2002.

DISTRIBUTION:

1	MS 0359	D. Chavez, LDRD Office, 1911
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories