

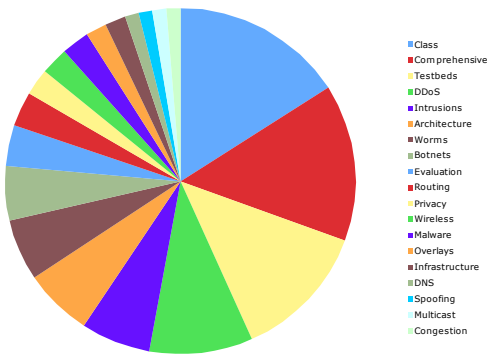


Elevating the Science of Cybersecurity

DETER at a Glance

DETER is an open community-based facility available to academic, industrial, and government organizations for research, testing, and computer security evaluation.

Types of DETER Projects



Where is DETER Used?



Useful Links

DETER Project Web Page:
<http://www.isi.edu/deter>

DETER Testbed Web Page:
<http://www.deterlab.net>

User Support:
testbed-ops@isi.deterlab.net

DETERlab: Looking Back and Looking Forward

The DETER cyber-security testbed, also known as DETERlab, provides a general-purpose, remotely accessible, and partitionable experimental infrastructure to support the development and demonstration of next-generation information security technologies. Since it began operation in the Spring 2004, DETERlab has been continuously in operation as an open research facility, with major funding from the Department of Homeland Security and with important contributions from the National Science Foundation and the Air Force. The completion of the most recent DHS grant for DETERlab provides an occasion for looking back at what has been accomplished over this 6+ year effort to support and enhance high quality research on cyber security.

At the same time we want to look forward, as we embark on a new 5 year program for DETERlab operation, development, and extension. The major components of this new DETECT effort are described later.

DETERlab: The First 6 Years

Looking back, we see the following advances in the DETERlab cyber security testbed since 2004.

- The project provided an experimental platform as a highly available, shared, and remotely accessible service to more than 100 researchers or research teams, working on a wide range of security problems.
- The hardware resources were increased from 64 750 Mhz experimental nodes to 400 nodes with speeds ranging up to 3 Ghz. We also attached a few special-purpose hardware boxes, including a few commodity routers and a netFPGA unit.
- The Emulab software base from the University of Utah was modified and extended by ISI and UCB to provide a safe environment of “risky” code testing, to improve performance, and to extend user facilities.
- Operation of the facility required a continued effort to install software upgrades, to track bugs, and to respond to user questions.
- Administrative procedures and software extensions were developed to accommodate a rapidly increasing use of DETERlab for teaching college-level courses in cyber security. With a curriculum development grant from the NSF, ISI collected course materials for integrating DETERlab use into classroom laboratories.
- A library of useful tools, most developed by DETERlab users, was created and maintained.
- A prototype “workbench” was built to integrate tool choice and configuration, experiment monitoring and visualization, and post-experiment analysis, within a unified GUI-based framework. This workbench is called SEER (Security Experimentation Environment)
- Annual workshops were held to bring together the community interested in cyber security research. Papers were refereed and selected papers were published.

During this period, user input and questions led us to start research on several area of the testbed software. We developed initial prototypes of the following facilities:

- “Federation,” the ability to construct and operate experiments that span multiple testbeds under different administrations, including dynamic allocation of high-performance bit-layer connectivity;
- Monitoring the “health” of an experiment; and
- Allowing controlled exceptions to the normal DETER experiment isolation, for experiments that need to communicate with specific hosts in the general internet.



DETECT, the Next Generation of DETER: Advancing the Science of Cyber Security Research and Experimentation

The DETER (cyber DEFense Technology Experimental Research) team has received a new five year contract from the Department of Homeland Security to carry out an ambitious program of research, development, and operations .

The primary goal of this award is to advance the science of cyber security, by developing and supporting transformative methodologies and tools for advanced cyber security experimentation and research. We plan to meet these goals through extensions and enhancements building upon the existing DETER Testbed capabilities.

The DETER testbed will be transformed to:

- support even larger scale and more complex experiments,
- advance scientific rigor through guides to verify accuracy of experimental results,

- provide adaptive and domain-specific tools to help users create new experiments and to deal with the great complexity of large-scale cyber security experiments,
- build a knowledge base of experimental designs and results,
- incorporate an extensible software architecture,
- provide a user-friendly interface for both novice and experienced users,
- and, as a result of these advances, support a significantly larger and more diverse research community.

We are excited by the opportunity afforded to us. As we perform the research and development to achieve these goals, we look forward to a tighter coupling with the cyber security community. We invite you to collaborate with us in these new research areas and help us evaluate some of these new testbed functionalities. Please contact any of the researchers on the team, or the PI, Terry Benzel, directly.

Effective Model Checking of Message-Passing Distributed Protocols

By Peter Bokor

Prof. Neeraj Suri and his group at TU Darmstadt in Germany, including graduate student Peter Bokor, work in the area of dependable systems with special focus on distributed algorithms and mobile networks. In recent work they are using DETERLab for the correctness verification of message-passing distributed systems.

Context & Motivation Message-passing distributed protocols are being employed in a wide range of systems such as embedded or large scale systems. For example, they are used to implement coordination tasks at service providers like Google, Yahoo or Amazon. Bugs in these protocols can lead to system outages, inconsistencies, or other violations of the service specification. Manual efforts to guarantee that the protocol satisfies its specification are not only tedious but also error prone given the *concurrency* in these systems. An alternative is *model checking*, i.e., the automatic and exhaustive simulation (or testing) of the system. In general, model checking is only feasible for small systems, as otherwise the number of possible simulation runs depletes the available resources. In this work we address optimizations of model checking to mitigate its time and memory requirements.

Faithful & Symmetric Models Message-passing distributed protocols are usually specified using simple languages similar to pseudo-code. These conceptual designs help protocol developers to abstract away from implementation details and to focus on concurrency and the protocol's high-level properties. A protocol description, call it M , written in such a simple language can be directly model checked to find fundamental design errors overlooked by the developer. However, sometimes even these models turn out to be too complex. An option is to create a *faithful* model M' which preserves the desired properties (or specification) of M . Given that M represents a message-passing protocol, we define M' such that the event of delivering a message is implicit and cannot be interleaved with other events. We have shown that M' is faithful for a practical class of specifications [2]. Our DETERLab experiments (see below) show that using M' can save model checking time and memory by up to *one order of magnitude*.

It is often the case that the model checker does redundant work. A common source of redundancy is *symmetry*. Intuitively, two different states of the system are symmetric if the system behaves identically when launched from these states except for permutation of some components in the system. Model checking can exploit symmetries so that only one of these two behaviors is simulated, a technique called symmetry reduction. We detect symmetries by restricting the syntax of the programming language such that the system contains certain symmetries by construction, i.e., invalidating these symmetries results in a compilation error. However, the restrictions are often non-intuitive, which limits its usability. Therefore, we have proposed such a language tailored to message-passing protocols, where the restrictions tend to be intuitive for protocol designers [1]. The main observation is that the more fine-grained the decomposition of the system into replicated components, the more symmetries can be detected and, thus, the better the gain of symmetry reduction. We show that symmetry reduction can help save *another order of magnitude* of model checking resources.

DETERLab Setting We can run model checking within a local machine because M is a model of a distributed system. In fact, we use single DETER nodes to run the verification of different models in parallel. No information is sent between these nodes. We measure the time and memory required to model check M , i.e., single models of different example protocols. Our experimental setup consists of Linux nodes. We note that the increasing number of DETER nodes can increase the throughput, i.e., the number of models verified per time unit. Our experimental setup consisted of around a dozen models, which were feasible to be model checked using two nodes. Each of our DETER nodes is equipped with 4GB memory. The memory utilization of our model checking tasks often exploit > 3 GB of memory. Another reason of using dedicated nodes is that single model checking tasks can last up to several days. Also, it is desired that no other application be executed on the nodes, to allow precise measurement of the gain of the optimization.

References

1. P. Bokor, M. Serafini, N. Suri, H. Veith. Role-Based Symmetry Reduction of Fault-tolerant Distributed Protocols with Language Support. *ICFEM*, pp. 147–166, 2009.
2. P. Bokor, M. Serafini, N. Suri. On Efficient Models for Model Checking Message-Passing Distributed Protocols. *FORTE*, 216–223, 2010.