

Design of the DETER Security Testbed

Version: 27 May 2004

***USC Information Sciences Institute
University of California at Berkeley
McAfee Research***



Cyber DEfense Technology Experimental Research (DETER) Network Evaluation Methods for Internet Security Technology (EMIST)

USC Information Sciences Institute • University of California, Berkeley • University of California, Davis • Penn State University
Purdue University • International Computer Science Institute • Stanford Research Institute (SRI) • McAfee Research • SPARTA

Design of the DETER Security Testbed

Version: 27 May 2004

***T. Benzel, R. Braden, E. Fraser, A. Joseph,
D. Kim, J. Mehringer, C. Neuman,
R. Ostrenga, S. Schwab, Dan Sterne***

Abstract:

This document describes the design of the DETER testbed, a project funded by NSF and DHS to provide an experimental environment for research in network security. The design is based upon a mesh of clusters of experimental nodes. Each cluster is based upon Utah's Emulab hardware and software, with additions and modifications to provide the security and isolation that is a unique requirement of the DETER testbed. This design document was prepared by the members of the DETER Architecture Working Group, drawn from USC Information Sciences Institute, University of California at Berkeley, and McAfee Research.

Design of the DETER Security Testbed

Version: 27 May 2004

1 Introduction

The Cyber Defense Technology Experimental Research (DETER) testbed and the associated Evaluation Methods for Internet Security Testbed (EMIST) research projects are jointly funded by DHS ARPA and NSF. The project objectives are to build an effective experimental and testing environment and to develop a corresponding experimental methodology, for Internet security issues and defense mechanisms. The centerpiece of the experimental environment will be a safe (quarantined) but realistic network testbed. This document describes the design of the DETER testbed. This design is itself an important research and engineering problem in the DETER effort. This is a ‘living’ document describing the DETER design plans; it will be updated as the work unfolds.

Among the recommendations in the “Report of NSF Workshop on Network Research Testbeds” [NSFnr02] is “Recommendation 5: Network research testbeds should be driven by the needs and goals of networking researchers.” This implies the primary directive for the design of the DETER testbed: meet the special needs of researcher in issues of network security. Important examples of application areas for the DETER testbed are distributed denial of service (DDoS) defense, worm propagation and defense, and defense of the network control plane, e.g., routing and the DNS. In addition to the DETER testbed, the project will create a supporting software environment of attack, defense, traffic generation, measurement, and analysis tools.

Section 2 of this document describes the requirements and how they shaped the design of the DETER testbed. Section 3 presents an abstract architecture for the testbed, while Section 4 outlines the actual design. Section 5 discusses the security and isolation issues of the DETER testbed in detail.

2 DETER Design Choices and Rationale

2.1 Background: NDDTEF Study

The DETER testbed design was initially informed by a DARPA-funded study on the requirements for a National DDoS Defense Technology Evaluation Facility (NDDTEF) [NDDTEF03]. We expect that the testbed technology developed for DETER will find application when and if an NDDTEF is built. The primary technical requirements laid out for the NDDTEF were:

- a) (Realistic) network topology
- b) Non-production (“breakable”) network, with support for repeatable experiments
- c) Rapid reconfiguration
- d) Realistic network traffic
- e) Data archiving
- f) Network management, monitoring, and analysis

g) Physical and network access control

This list is generally applicable to DETER; the specific DETER requirements will be discussed below.

However, it should be noted that NDDTEF objectives differ in some important ways from the objectives of the DETER testbed. The proposed NDDTEF would be a “vendor-neutral shared laboratory in which researchers, developers, and operators from government, industry, and academia can evaluate potential DDoS defense technologies under realistic conditions...” Its objectives emphasize product testing of real DDoS defense devices in a realistic environment, by a broad user community including government agencies, network device vendors, DDoS defense product vendors, and ISPs. On the other hand, the primary focus of DETER is on network security research, to develop understanding and measurements of Internet attack and defense mechanisms. DETER is also intended to address a much broader class of security issues than NDDTEF, not only DDoS.

The NDDTEF report describes five different design options:

Option 1: Isolated single-site test network

Option 2: Single-site network fully connected to Internet

Option 3: Isolated distributed multi-site test network

Option 4: Distributed multi-site network fully connected to Internet

Option 5: Distributed multi-site network offering transit to the Internet

For reasons to be discussed below, the basic DETER design corresponds closely to Option 3. Options 1 and 2 were too limited, and Option 5 could create unacceptable impact on the Internet. We will investigate the possibility of a carefully controlled version of Option 4. For example, an attractive objective would be to safely gather Internet viruses and worms into the testbed while not allowing any to propagate back out. This requires further study.

2.2 DETER Testbed Requirements

The following specific requirements were determined for the design of the DETER testbed.

a) Versatility

The testbed must support a wide range of security test scenarios. This may include, for example, experimenting with the propagation of viruses and worms, measuring the effectiveness of DDoS defense mechanisms, or testing network infrastructure protection (e.g., DNS or routing protocol security). Experiments may use attack mechanisms developed for the experiment, or they may test “live” pathogens captured from the real Internet.

b) Fidelity

For some experiments, fidelity to “real” networks will be an important requirement. There are several dimensions to fidelity: (1) the number of nodes (routers and end systems), (2) realism, i.e., reproducing real router and end-system behavior, and (3) realistic heterogeneity of hardware and software, and (4) a realistic mix of link bandwidths and delays.

c) Economy

There is a real-world limitation in the available funds to build and operate the testbed. We must make realistic compromises that will maximize the scientific output from testbed experiments for the available funds.

d) Efficiency

Following from economy is experimental efficiency. The testbed will be an expensive resource that must be shared among a potentially large community of experimenters. Just as a major particle accelerator needs to have multiple beam-lines, so the DETER testbed needs to support multiple simultaneous experiments. It must be possible to partition the physical resources of the testbed in a flexible and dynamic manner. This is sometimes called “space sharing” of the testbed facilities.

e) Containment and Security

Security is not only the object of research using the testbed; it is also a vital requirement for the testbed itself. Security for the DETER testbed is critical, and the threats are both internal and external. Internal threats come from virulent code that is tested within DETER and threatens to either shut down the control plane or escape into the Internet. Containment is required. The external threat will come from hackers who see the testbed as a tempting target for an exploit.

The DETER testbed is being designed to be safely run experiments that present a wide range of threat levels. The most dangerous level might be represented by “live” testing of a contagious attack program whose attributes are completely unknown, for example. The traditional approach to testing such dangerous programs has used a completely isolated laboratory consisting of dedicated systems whose disk drives and memory chips never leave the laboratory and are never reused. Experimenters must be physically present in these laboratories and must be specially trained.

The approach of the DETER project, on the other hand, is to build a single multi-level safe testbed that can change its operational mode to match the threat level of the experiments. It will provide a shared laboratory facility for those experiments whose threat level is low enough to allow sharing, but it can be reconfigured (or will reconfigure itself) to exclusive use by more dangerous experiment. It will also allow remote experimenter access for all but the most dangerous experiments. Building such a multi-level safe testbed is itself a research project undertaken by DETER, and a result will be an understanding whether there are some experiments that are so dangerous that they will require the more traditional approach of complete physical isolation.

f) Accessibility

The testbed must be readily accessible to the experimenters, who must be able to remotely set up and monitor their experiments using a set of easy-to-use tools. In addition, an experimenter must be able to exert control even when the test network is congested or broken.

Note that remote accessibility may clash with the security and containment requirements.

g) Programmability

A significant set of DETER experiments may involve testing new monitoring, filtering, and diagnosis mechanisms within the network. This will often imply adding or modifying router algorithms.

Past experience with network testbeds has shown that this programmability requirement will often conflict with the realism requirement (b)(2). Router vendors are not anxious to open their platforms to experimental modifications, and real router internals are typically complex and would be expensive to modify even if it were allowed.

h) Repeatability

The emphasis in DETER is on scientific experiments concerning network security. A central requirement for many experiments will be repeatability. The dynamics of the real Internet cover a wide range of time and space, and measurements on the real Internet can vary widely from day to day and place to place. An important objective of the testbed, on the other hand, is to allow the experimenter to reproduce experimental conditions precisely, modifying them only in a controlled manner.

i) Functionality

The testbed needs rich functionality to support the experimental program. This includes a rich set of traffic and topology generators and experimental profiles, and a comprehensive instrumentation facility.

It is clear that fidelity has costs for purchase, maintenance, and operation of hardware and software. The hardware-related costs increase linearly with the number of nodes and perhaps faster than linearly with increasing heterogeneity. It will never be possible to faithfully reproduce the real Internet in the testbed, and in fact such ultimate fidelity is questionable from a scientific viewpoint. A central aspect of the experimental science will be constructing idealized abstractions of the real Internet with *enough* fidelity for specific experiments.

For example, suppose that an experiment requires 50 router nodes. The scientific question is: would the results be qualitatively different, or even quantitatively different, if the experiment were repeated with 100 routers? With 500 routers? With 5000 routers? In short, how does a security experiment *scale* with network size? A similar consideration arises with the propagation parameters of bandwidth and delay; do we have to run an experiment with 1 Gbps (or 10 Gbps) links, or can we scale up measurements made at 100 Mbps?

2.3 Router Parameters

A central design issue for the testbed is how routers will be realized. Three different approaches might be considered. The first is to use a homogeneous collection of identical PC platforms, as in Utah's Emulab testbed. At the opposite extreme, the testbed could use only real vendor router hardware and software. A third possibility, intermediate between the first two, would use network processor platforms like the Intel IXP.

Figure 1 compares these three approaches using five attributes: number of nodes (that the project can afford), the degree of heterogeneity that can be achieved, the realism that can be achieved, the programmability of the platform, and the achievable throughput.

Fig 1. Router Parameters

	Emulab Cluster	Intel IXP	Vendor Router
Number of Nodes	Moderate # – O(100s)	Small # - O(10)	Small # - O(10)
Heterogeneity	None	A little bit...	Moderate
Router Realism	Low (PC routers)	Intermediate	High
Programmability	High (completely programmable)	Moderate (not easy)	None
Throughput	Low-moderate < 1 Mpps, 1 Gbps	Intermediate	High

2.4 Meeting the Requirements

Network testbed architecture falls into two general classes: *distributed* and *cluster*. Early network testbeds (e.g., DARPA’s DARTnet and CAIRN) were primarily distributed, i.e., they were composed of routers sited at individual research campuses scattered across the US. A cluster testbed, in contrast, puts many nodes in a single laboratory and interconnects them using a programmable patch panel to construct (nearly) arbitrary topologies. Some nodes can be used as traffic shapers to emulate desired link characteristics.

There are two strong arguments in favor of a cluster testbed for DETER: *security* and *repeatability*. Consider first security. An important objective for the design of DETER is to allow safe testing of worms and viruses that could cause great damage if let loose in the Internet. To provide highly reliable containment for pathogenic code, the testbed operators must be able to exercise very tight control over the physical hardware, system interconnections, system configuration, and firewalls. It is not possible to exert such tight control over the nodes of a distributed testbed; rather, the nodes need to be clustered at a few physical locations, under a single administration and with physical site security

Repeatability will be required for an important class of DETER experiments. Repeatability cannot be achieved if the commodity Internet is used for connectivity between experimental nodes. To achieve repeatability, a distributed testbed would need dedicated wide-area pipes. However, the DETER project, which must serve researchers all across the US, does not have funding for dedicated pipes of this magnitude. The DARTnet testbed succeeded because it was able to use economical T1 (1.5 Mbps) link; however, today’s DDoS and worm experiments generally need bandwidth that is significantly higher than T1. Fortunately, bandwidth within the same room is cheap and can be dedicated to a particular experiment, so a cluster testbed can provide predictable and high bandwidth links.

Our fundamental design choice is therefore to base the DETER testbed upon a cluster architecture. This satisfies the economy, versatility, programmability, and repeatability requirements, at some expense of fidelity. By using well-designed control software for the cluster, we can achieve efficiency and accessibility as well.

The DETER design represents a compromise to meet budget limitations. *Fidelity* is very desirable in a testbed, but it is inherently expensive. As noted earlier, the issues in fidelity are node count, realism, heterogeneity, and link properties. Thus, fidelity to Internet operation would require large and realistic topologies, heterogeneous and realistic nodes, and heterogeneous and realistic links. Our plan is to begin with a homogenous PC cluster and to add limited heterogeneity and realism, chosen to get the most scientific effectiveness possible for the limited funds that are available. Specifically, DETER requirements are reflected in fidelity in the following manner.

- Numbers of nodes

The initial objective of DETER is a cluster of roughly 100 nodes, in service early in the project. We hope to get significant hardware donations to expand this to at least 300, with 1000 as a likely upper limit. Some software virtualization of nodes will allow us to effectively multiply the number of experimental nodes by a small factor, e.g., by 4 to 10.

- Realism

PC routers are an excellent choice for some kinds of experiments; the programmability and generality of PC routers are a significant win. On the other hand, some experiments may depend upon having real router hardware. For this purpose, we plan to add a small number of real routers as nodes in the testbed. Again, we will be largely dependent upon equipment donations.

- Hardware Heterogeneity

It is not desirable (or possible) to purchase all the experimental nodes with the highest available CPU power. Moore's law creates an "expanding universe" of CPU performance; today's leading edge CPUs are tomorrow's moderate performers, but the leading edge CPUs are significantly more expensive than the moderate performers. Furthermore, CPU performance bottlenecks represent one of the important variables in some security experiments, particularly in worm experiments. On the other hand, there is a need for some CPUs in the cluster that are faster than the bottleneck machines, to provide extra capacity for control and measurement of the experiment.

These considerations lead to (at least) a two-tiered CPU performance profile: the great majority of the experimental nodes should be economical moderate-performers, but a few (e.g., 10 – 20%) of the nodes should be leading-edge machines.

Otherwise, the experimental node hardware within a cluster will be homogeneous. This is probably not a major problem as long as DETER is mostly devoted to scientific experiments; indeed, a large degree of homogeneity will often be desirable in this case. If and when the DETER testbed or a later clone of the DETER technology is used for consumer testing of actual security software products, the limited heterogeneity of DETER may become a problem. However, since there are significant costs associated with heterogeneity, we are forced to accept limited heterogeneity in the testbed for the present.

- Link Properties

To provide controlled link characteristics, a cluster testbed experiment uses some of the nodes as traffic shapers. However, it may be useful to insert some uncontrolled long-delay (i.e., wide-area) paths into the testbed. The plan for DETER is therefore to build a few widely separated clusters instead of a single centralized cluster.

This *mesh of clusters* architecture has organizational and operational advantages; it represents, in a sense, a sensible compromise between the cluster testbed and the distributed testbed approaches.

Although we expect that the great majority of experiments to be controlled remotely over the Internet, there will be occasions when physical presence is desirable, e.g., for demonstrations.

3 Architecture for a Security Testbed

In the parlance of computer networking, “architecture” is a high-level design that is used to guide and inform specific technical decisions. This section presents an architecture for a security testbed that is consistent with the requirements of the previous section. This architecture defines the logical functions and flows that are required for a secure experimental environment. It will also clarify the required security mechanisms and policies. It is an abstraction that can be compared with the actual design to be presented in later sections..

The architecture consists of four primary control “planes”: Facility Control, Experiment Control, Data Loading and Collection, and Experiment. The Facility Control Plane is global to all experiments running in the testbed, while the other three are instantiated for each active experiment.

3.1 Facility Control Plane

The Facility Control plane controls and monitors the operations of the testbed. It controls project setup, including the allocation and the loading of the experiment’s nodes. This plane has complete authority over all nodes within the facility. Its functions include:

- Monitoring the safe operation of all nodes within the testbed.
- Allocating node resources to specific experiments as requested by users.
- Loading the experimenter-specified operating system on each allocated experimental node.
- Loading experiment-specific data such as node configuration, accounts, keys, DNS configuration, routing tables, etc.
- Monitoring access to the facility.
- Monitoring any data that leaves the facility.
- Pre-empting Experiment Control to shut down out-of-control experiments.

3.2 Experiment Control Plane

The Experimental Control plane gives experimenters control over the nodes that are allocated to their experiment by the Facility Control plane. Experiment Control plane functions include:

- Experiment configuration
- Experiment control
- Experiment monitoring
- The loading of experiment-specific data

Experimenters can access the Experiment Control plane in various ways:

- Remote Terminal Service Access via an SSH over the Internet to each node’s serial console (TTY) port.

- Remote interactive shell access via an SSH over the Internet to each node's control interface. (Not available to high risk experiments)
- Local onsite access to the nodes and the serial consoles, in exceptional circumstances.

3.3 Data Loading & Collection Plane

The Data Loading and Collection plane provides passive data collection. It is isolated from the Experiment Control plane; in other words, there are no routable interfaces from the Data Loading and Collection plane to the Experiment plane during a live experiment. Interactive access is allowed to this plane only from the Experiment Control plane or from the Facility Control plane. Depending on the experiment's risk rating, data collected by this plane may have to be scrubbed for mal-ware before the data is allowed to leave the facility. This plane may contain, for example:

- Unix machines promiscuously collecting data (e.g., using TCPDUMP)
- Sniffer machines monitoring links between selected nodes
- Devices collecting router flow data (e.g., NetFlow)
- RMON monitoring devices
- Intrusion Detection Systems
- Network management systems
- Data scrubbing devices such as Anti Virus machines

3.4 Experiment Plane

This plane is the emulated network in which the experiment will run. Its scope includes all experimental nodes within a single experiment, assigned to the experiment by the Facility Control plane. Direct connectivity to the Internet will not be allowed in general (although specific exceptions may be made for low-threat experiments). This plane allows unrestricted interactions between any and all nodes within a given experiment. This plane includes:

- Hosts, routers, and emulated LANs.
- Traffic shapers, e.g., using *Dummynet*.
- Standard NS traffic generators

3.5 Phases of an Experiment

Figures 2a – 2e illustrate the interactions of DETER planes during the various phases of an experiment.

Phase 0: Proposal Review

Because of the possibility of dangerous experiments, DETER requires administrative approval for every experiment [Policy04] Approved experiments are registered in the Facility Control plane.

Phase 1: Experiment Initiation

Given a script file describing the desired configuration, the Facility Control plane allocates nodes, generates all the node configuration files, and downloads the OS and configuration files into those

nodes. At the completion of this phase, it informs the user that the experimental setup is complete, hands off user control to the Experimental Control plane, and enters phase 2.

Phase 2: Experiment Ready

The experimenter is now in control of the experiment and can load any experiment-specific data and configure any data collection devices that were not configured in phase 1. If specified by original script, this phase may be executed automatically without experimenter intervention.

Phase 3: Experiment Running

Execution of the experiment. Data may be collected in the Data Collection plane.

Phase 4: Result Capture

Experimental results may be retrieved over Internet.

Figure 2a:

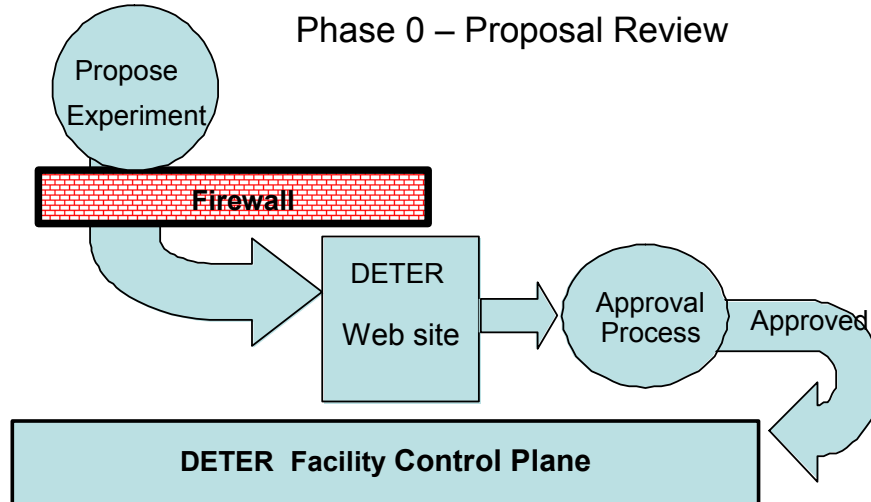


Figure 2b:

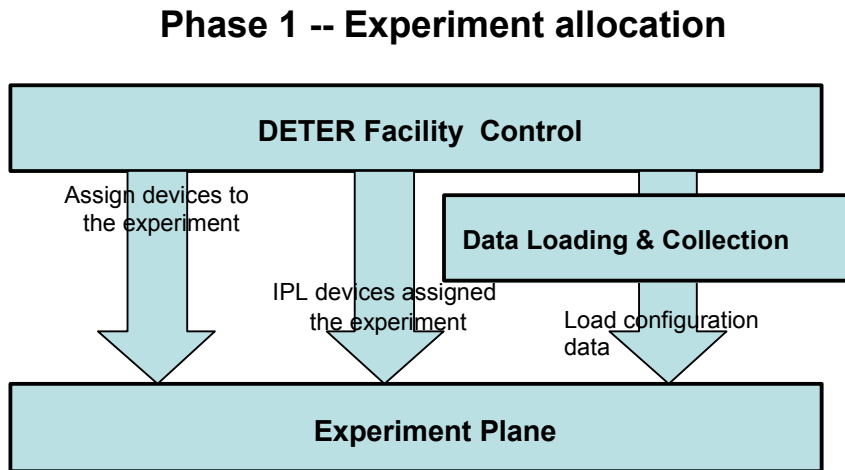


Figure 2c:

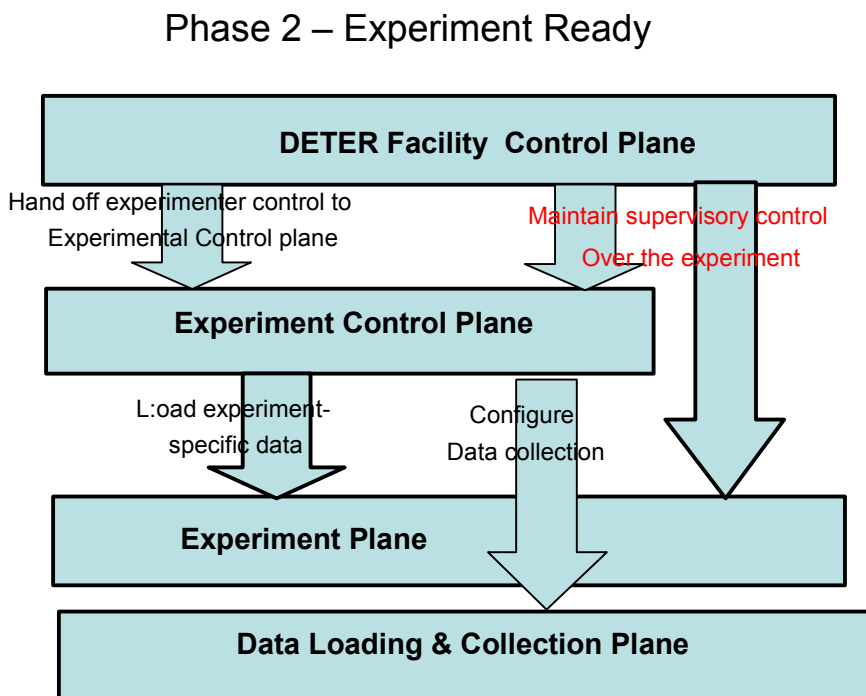


Figure 2d:

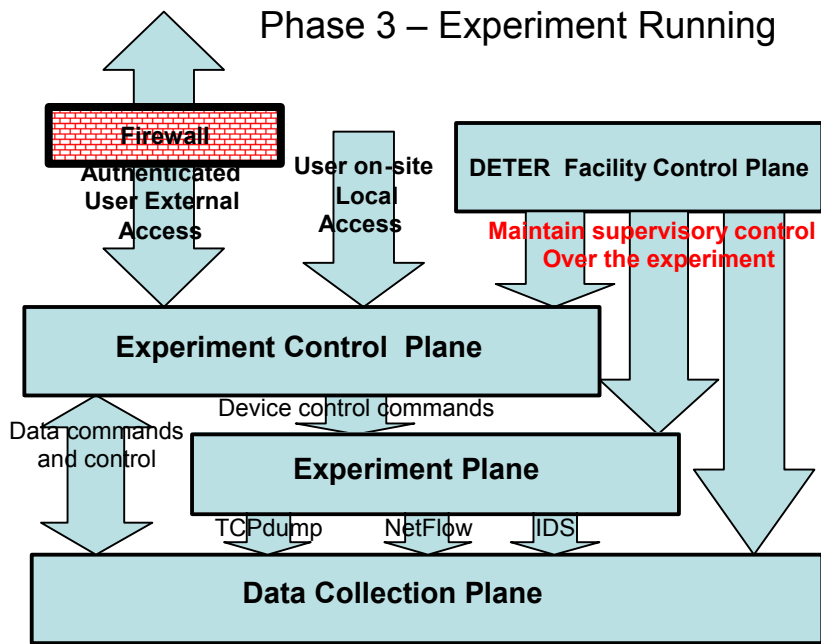
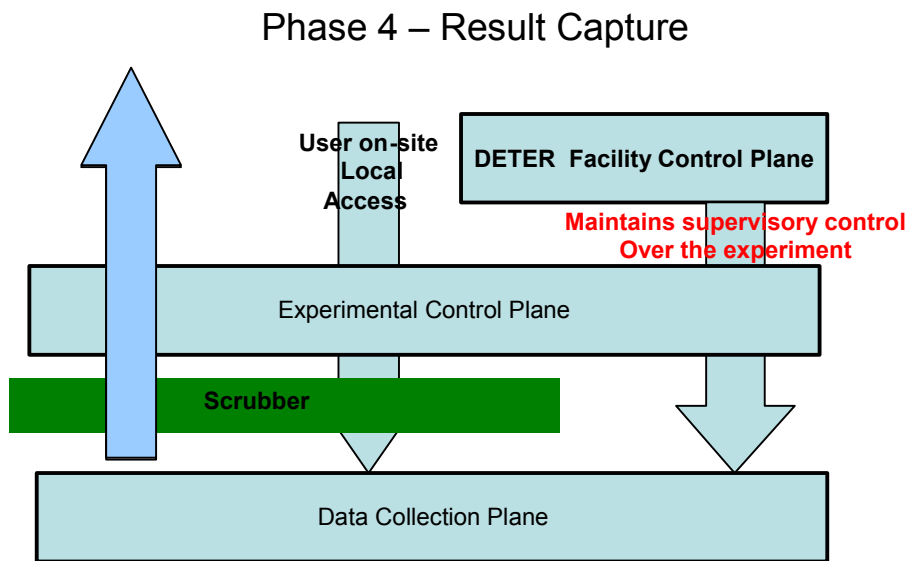


Figure 2e:



*Experimental Plane is not active during this phase

4 DETER Testbed Design

4.1 The Basic DETER Plan

The basic development plan for DETER is as follows.

1. Initially build a homogeneous, flexible, and controllable experimental facility using a cluster-based design. Each node in a cluster can be configured to emulate an end node, a router, a traffic generator, a traffic shaper, a monitor, etc. The design is based on Utah's Emulab [Emulab02] cluster testbed and leverages from their administration and control software. Utah's Emulab software package is large, well designed, and well maintained, and many researchers have found it easy to use.
2. Later, provide mechanisms to allow the integration of (a small number of) commercial routers and/or network processors with the cluster(s), to increase realism for particular experiments.
3. Add containment, security, and usability features to the testbed control software.
4. Build a software environment – tools and libraries -- useful for security experiments.

Specifically, the DETER project plans to build and operate a testbed consisting of experimental node clusters at three different physical sites: ISI-West, UC Berkeley, and ISI-East. The DETER and EMIST projects will design and build a variety of experiment generation, control, and measurement software specialized for security research on this testbed. The DETER design objective is to support a total of O(1000) nodes across the three clusters. Hardware planning is further designed to maximize future flexibility and versatility, to adapt to a rapidly evolving problem.

Leveraging from the Emulab hardware and software architecture provided a major programmatic advantage for the DETER project. First, experiments could begin on the existing Emulab testbed at Utah in parallel with building the first DETER cluster at ISI. This provided valuable experience that was put to use in designing the DETER cluster, and it provided an immediate set of test applications. In addition, the Emulab software allowed the rapid achievement of an initial operating capability in DETER. The DETER project plans to continue to coordinate closely with the Utah group in the future, to maintain as much software compatibility as possible between the DETER testbed and Emulab. Since our objectives differ somewhat, there are already some areas of technical divergence, and in the future we expect this divergence to increase. However, we should allow DETER to diverge from Emulab only when the benefits to DETER objectives are manifest and clearly outweigh the additional development and maintenance costs that may result.

4.2 Cluster Architecture

Figure 3 shows the planned architecture of a DETER cluster. The design and terminology are based upon the pioneering Utah Emulab [Emulab02]. See Appendix A for a list of the specific equipment configurations at the cluster sites.

The major components of an Emulab cluster are as follows.

- Cluster

A cluster occupies one physical site in the DETER testbed and consists of one or more pools of identical experimental nodes. For example, the initial ISI cluster contains 64 identical high-end PCs, while the initial UCB cluster contains 32 high-end PCs (see Section 4.5).

- **Experimental Nodes**

Each experimental node has four high-speed data ports to the VLAN switch that is used to create arbitrary emulated topologies. In addition, each node has a medium-speed network port to the control network, a serial console line, and a switched power source. A user has root access to each node allocated to his experiment.
- **Programmable Patch Panel (VLAN switch)**

The VLAN switch serves as a programmable patch panel to set up the desired experimental topologies. The intent is for this switch to have enough capacity that each virtual Ethernet link will be completely isolated from the other links.
- **‘User’ Server**

The User (or Ops) server supports Unix login accounts to all experimenters using the testbed. The User server (or simply User) also supports a large NFS file storage facility in which users can store their data and programs.
- **‘Boss’ Server**

The Boss server (or simply Boss) is the master machine that supports the administrative and scheduling functions of the testbed and maintains the testbed control database. Users invoke these functions via a web server that runs on this machine.

Boss sets up the VLANs in the programmable patch panel to create the virtual configurations. It also drives the power controllers that allow individual experimental nodes to be power-cycled. It configures the User NFS server to allow file system mounts from active experiments. Finally, Boss configures and supports a DHCP server for booting experimental nodes, a DNS server, and a trusted disk-image server for downloading standard disk images into experimental nodes. Downloading under the Emulab software uses a high-performance multicast-based program called Frisbee.
- **Control Network**

The *Control Network* is actually a set of independent Ethernet segments, created by VLANs. The component segments shown in Figure 3 are the User VLAN, the Boss VLAN, the Control Hardware VLAN, and the Experiment Control Network that includes the experimental nodes. A router joins these segments and includes a firewall to further compartmentalize communication within the testbed.

Major uses of the Control Network include:

 - Boss server contacting User server, using NFS and login.
 - DHCP and DNS calls from experimental nodes to Boss.
 - Frisbee downloads of disk images from Boss to experimental nodes.
 - Control actions from Boss to remote power controllers and VLAN switch.
 - NFS access to user accounts on User server, from experimental nodes.
- **External Firewall – ‘Gatekeeper’**

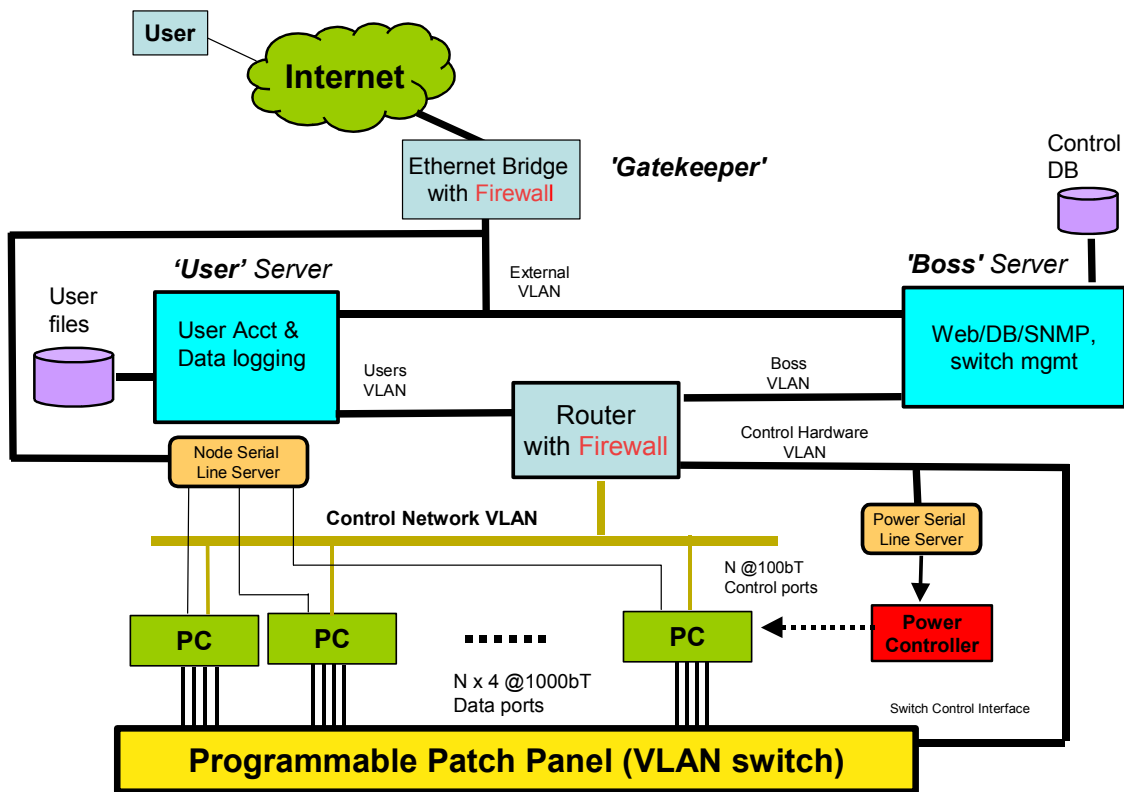
A firewall between the testbed and the Internet serves both to protect the testbed from external attacks and to help quarantine dangerous programs within the testbed. This is implemented by a machine that is configured as an Ethernet bridge. Hence, the interfaces on the External VLAN (Figure 3) all have real external Internet addresses, assigned by the service provider.

- Serial Line Servers

A serial line server provides access to the console ports of the experimental nodes. Emulab allows experimenters to SSH (only) to any of their allocated nodes, via the serial line server. Experience has shown that serial line access to experimental nodes is also critical to maintenance of the testbed.

In the ISI cluster, the 'User', 'Boss', and node serial line servers are all directly addressable on a provider network and have public IP addresses from the provider (Los Nettos). The internal VLAN segments use private IP addresses. The data ports on the experimental nodes are allowed to use any IP addresses, public or private. Some experiments may use their nodes to emulate regions of the real Internet, with the actual Internet addresses, for example.

Figure 3: DETER Testbed Cluster Design



It should be noted that the design in Figure 3 differs in an important way from that of Emulab. Emulab allows an experimental node direct access to the Internet, while the DETER testbed design has no direct IP path between an experimental node and the Internet. Depending on the categorization of an experiment, certain links may be physically disabled, providing even greater containment. These security issues are discussed further in Section 5.

It is useful to understand how the planned cluster testbed design shown in Figure 3 differs from the layered abstract architecture for a security testbed that was previously described in Section 3. In the abstract architecture, running a security experiment could be compared to the operation of a possibly noxious chemical process: you pour the ingredients into a robust container, close it up tight, and let it cook. You monitor the process only through very narrow openings that cannot let the brew escape. When it is cooked, you open it up and remove the contents, possibly encapsulating them for safe removal. Finally, you detoxify the vessel for future use. In contrast, the cluster testbed in Figure 3 is more analogous to leaving the reaction container open but within a larger closed vessel. Thus, data is introduced into an experiment by using NFS to the User file server, not by pre-staging it into the experiment. The NFS-based design, taken from Emulab, was chosen to provide user convenience and flexibility. Experience with the DETER testbed will show whether a closed container with complete pre-staging of input would have been more advisable.

4.3 Multiple Clusters

The three clusters of experimental nodes are intended to form a common facility for security experimentation, but there is a range of possible administrative and technical models for accomplishing this integration. The three clusters might be administered independently; in this case, a user would have to make an early binding of a project to a particular cluster, at the time of setting up a project account. A more cost-effective and convenient model would provide a single set of project accounts for all DETER users, and allow users to defer binding to the nodes of a particular cluster until they start an experiment. Finally, all the nodes might be treated as a common pool so that a single experiment could span multiple clusters.

Each of these models has technical implications for the hardware and software of the clusters. In particular, in the independent model, each cluster would have its own independent control plane, including a User server, Boss server, Control Network, Power Controller, Serial Line Server, etc. The more integrated models require integration of the control plane in some manner.

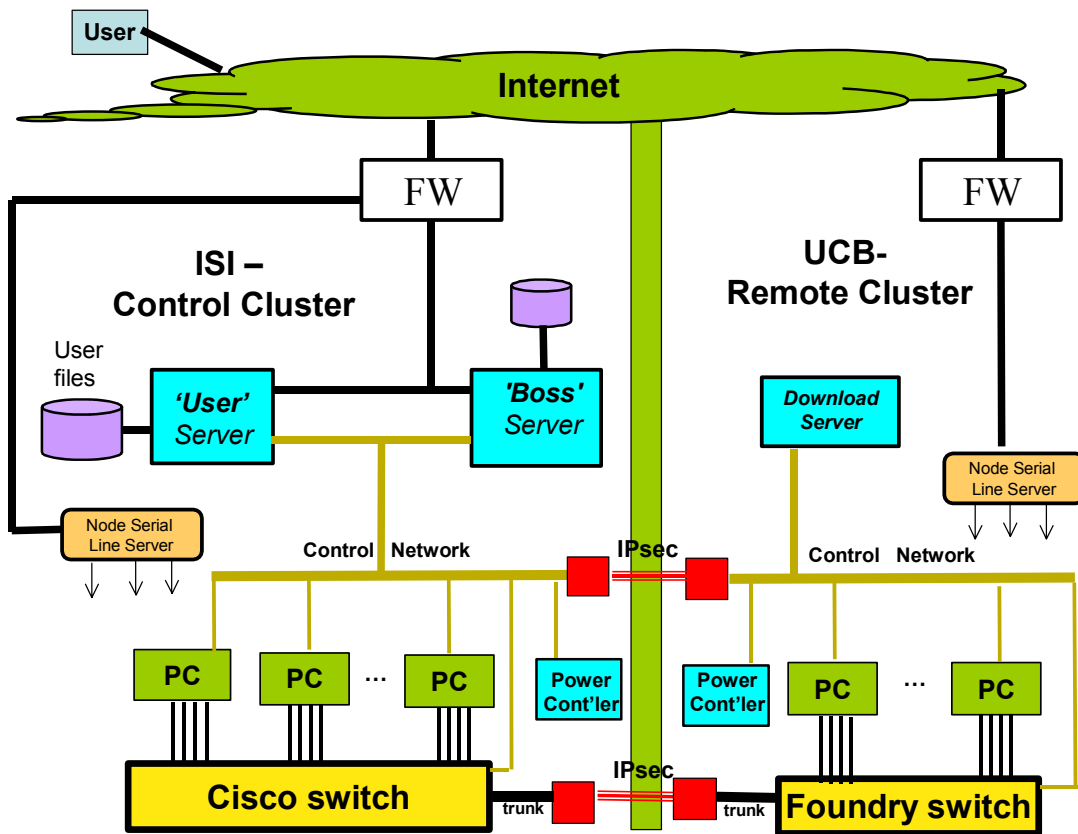
The initial DETER approach will use the most integrated model with centralized control. There will be a single administrative and access point for all DETER users, at a particular *testbed control site* (probably ISI). The Boss and User servers will be instantiated only at the control site and they will control the other cluster nodes remotely. Other sites will contain only experimental node clusters and a local programmable patch panel. This extreme centralization of control and access will avoid complex coordination and database synchronization problems that would arise from a distributed cluster control plane. It is also expected to simplify security and administration. The downside is performance limitations, to be explored.

To a user, the available nodes will be distributed into homogeneous pools at the different sites. A user's experiment request may draw nodes from any of the pools, or it may specify that the nodes are to be allocated entirely from a particular pool at one site. Thus, an experimenter can determine whether the uncontrollable characteristics of the inter-cluster link across the Internet will be part of an experiment.

Fig 4 shows schematically the planned initial interconnection between the control site at ISI and the UCB cluster. Two IPsec tunnels are used. One tunnel links the control network segments at the two sites, while the other provides a bridged VLAN-knowledgeable layer-2 interconnection between the switches forming the programmable backplanes. Figure 4 is only schematic, omitting the details of the Control Network and of the external connections to the Internet. Note also that the machines at each end of the IPsec tunnels must also act as firewalls, to allow only the IPsec tunnel traffic.

Figure 4 also suggests two possible modifications to the simple centralized scheme. First, the serial line servers may be local to the particular sites. Users gain access to these servers using URLs generated by the Emulab control software, and it should be easy to generate the appropriate IP address for a specific experimental node, transparently to the user. Second, a major load on the Control Network tunnel will be downloading disk images using the Frisbee program. It should be possible to use a local copy of the standard disk images in each cluster site, as shown by the 'Download Server' box. The Download server could be considered to be a first, very small, step towards distributing the function of the Boss server, or it could simply be considered a performance enhancement.

Figure 4: Cluster Interconnection



4.4 Portal Nodes

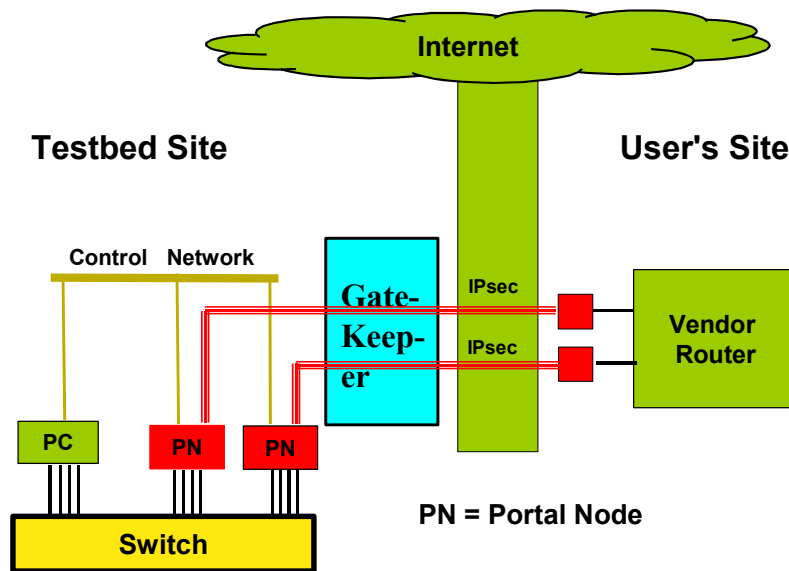
Applications of the DETER testbed may include experiments that involve specialized hardware (e.g., vendor routers) or software (e.g., covered by restrictive licensing agreements) that cannot be wired or loaded directly into the testbed. To support such experiments, we propose to allow a logical extension of a cluster to an experimenter's site. [This is a form of what Utah calls a *federation*]. This will be accomplished by equipping a few cluster nodes with additional hardware ports that are connected directly to the Internet, using IPsec tunnels.

An example of the proposed portal scheme is shown in Figure 5. Here there are two experimental nodes connected as portal nodes, connecting to the user site as shown. A single vendor router is shown at the experimenter site, although in fact the experimenter could have arbitrary local topology. Note that nodes at the experimenter's site are not known to the Emulab control plane at the cluster site; the experimenter must manually configure all user-site nodes.

Like all connections to the Internet, these connections go through the Gatekeeper firewall, which must have specific rules to allow this traffic. When they are not in use as portal nodes, these two special nodes should be available for allocation as normal nodes, in which case the firewall must block access to the Internet.

There are serious security implications of the portal scheme, which are discussed in Section 5 below.

Figure 5: Example: Using Portal Nodes



4.5 Hardware Configurations

The full hardware configurations are contained in Appendix A. The following describes the experimental nodes in particular, since this is the hardware of direct relevance to an experimenter.

- **ISI Cluster Configuration**

The initial 64 experimental nodes of ISI's cluster have the configuration:

- IBM Netfinity 4500R dual 733MHz PIII processor, in a 3U case.
- 1GB of RAM
- One 17GB SCSI disk
- PCI slots (3 @ 64 bit, 2 @ 32 bit)

- Two 10/100bT Ethernet interfaces, one with PXE boot support enabled on a PCI slot.
 - Two dual-port 1000bT Ethernet interfaces (4 interfaces total), in 64bit/33MHz PCI slots. Intel PRO/1000 MT Dual Port Server Adapters.
- **UC Berkeley Hardware Configuration**

The initial 32 experimental nodes of UCB's cluster has the configuration:

 - Sun V60x dual 2.28Ghz Processor
 - 2 GB of RAM (2 x 1GB DIMMs)
 - Two 36GB 10K RPM Ultra320 SCSI disks
 - PCI-X slot (Full height/full length 64 bit/100Mz)
 - PCI-X slot (Half height/half length 64 bit/100 Mhz)
 - Two 10/100/1000bT Ethernet ports (on board)
 - One 1000bT Intel Pro 1000 MT dual-port Ethernet port
 - One 1000bT Intel Pro 1000 MT Ethernet port (control VLAN)
 - CD and floppy drives
 - **ISI-East Hardware Configuration**

(To be supplied later)

4.6 Testbed Software

4.6.1 Experimental OS

The initial testbed software will support Linux and FreeBSD as the basic operating systems in the experimental nodes. Assuming that we can obtain sufficient cooperation from Microsoft, we will later add Windows as an alternative. Windows support in DETER is important because it is the target of choice for network exploits.

Another useful software base for experimental nodes used as routers would be the Click modular router software [Click00] from MIT. This provides efficient and flexible packet forwarding at the interrupt level, with consequent high performance.

4.6.2 Support Software

Early efforts in the DETER project have focused on building a testbed. However, a larger project goal is to build a complete environment that will both make the security researcher's job easier and also improve the scientific quality of the results. It will be the objective to foster a research community building on each other's work. For example, we intend to create a library of scripts that define particular experimental conditions, which can be used by various experimenters to compare different attack and defense mechanisms under the same conditions and using common metrics. The technical aspects of this experimental archive will be designed and created by the DETER project, but the scientific guidance will come from the synergistic EMIST project.

The DETER staff will create an index of useful traffic generators, attack generators, traffic shapers, and measurement and data capture tools, and visualization tools, and to the extent feasible, make these tools

readily available to experimenters. There is a balance between convenience and security here; for example, we may not want to leave attack tools readily accessible online.

An important set of DETER experiments is expected to need topologies that emulate a subset of “real” Internet topology and routing. For this purpose, BGP protocol code will be required. Suggested code packages are XORP and Zebra.

4.6.3 Experimental Data Sets

Another aspect of the experimental environment for security research is the trace data sets that drive many experiments. Trace data may be taken from real network situations, or it may be synthetically generated. There are often commercial and legal barriers to obtaining real trace data and sharing it among researchers, although the government is currently trying to develop a process that will resolve these issues sufficiently to allow the use of ISP-provided data traces in DETER. The resulting process will have an impact on the technical and procedural rules for introducing such data into the DETER testbed.

Ideally, the DETER project hopes to be able to build up a database of standard test cases that can be used by many experimenters to provide common metrics for comparison of different algorithms. We do not yet know to what extent this will be possible.

We do know that there will be technical issues with using such trace data. One simple issue is that the IP addresses in a trace will in general need to be mapped into a different range for a particular cluster experiment. The DETER project plans to provide a mapping program for this purpose.

4.7 Instrumentation

Instrumentation is an important requirement for the DETER testbed. Figure 6 shows five possible approaches to instrumenting the traffic between experimental nodes. A sixth method is to make SNMP queries to gather traffic statistics collected by the nodes. This requires that the nodes have enough CPU capacity remaining to respond to the queries.

The five methods are as follows.

1. "Splitter" boxes (promiscuous taps), connected to monitor machines (M1).
These could be permanently wired into particular links between experimental nodes and the pluggable patch panel switch and connected to distinct monitor machines. The Emulab allocator would need to treat these links specially when it mapped a virtual topology into a real configuration. This approach would add no additional latency between experimental nodes.
2. Switch hardware that supports multiple "mirror" ports, i.e., splitters built into switch (M2).
Emulab would have to be adapted to support this switch, including the mirroring. The monitor machines could be allocated from the experimental nodes. This approach would add no additional latency between experimental nodes.
3. Dynamic link monitor node (M3)
Upon request, Emulab could allocate one of its pluggable experimental nodes and insert it as a monitor between two functional nodes. This would be analogous to the insertion of link emulation (*Dummynet*). In fact, it has been suggested that *Dummynet* software should itself be extended to include the monitor/filter function as well as its current traffic shaping function. Inserting dynamic link monitoring nodes would add to latency.
4. Static link monitor machine (M4)

This is essentially the same as (3), except the monitor machine would not be one of the cluster nodes. It would be permanently wired to particular experimental nodes. This would economize on experimental nodes, but it would probably require (perhaps complex) changes to Emulab. It would add to latency.

5. Linux OS bridging (M5)

The Linux OS has a bridging feature that could be used to attach a separate monitoring machine to a cluster node. This would take CPU time in the cluster node, but it would not add additional latency to the monitored link.

Fig 6: Instrumentation Approaches

Key:

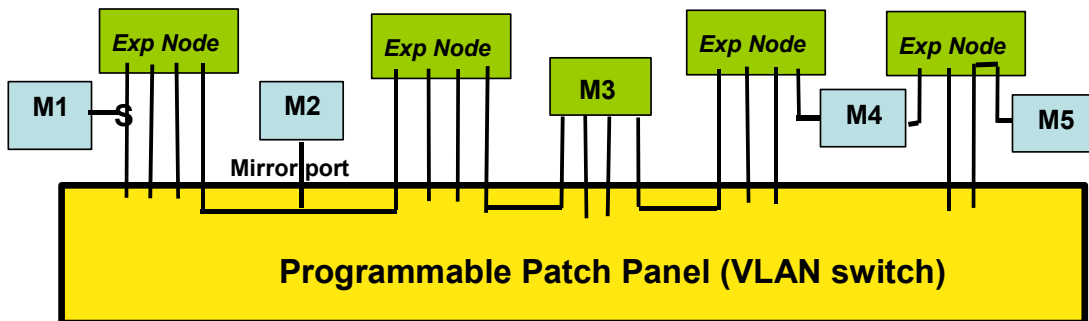
M1: Monitor machine wired to hardware splitter (hub) S.

M2: Monitor node allocated to mirroring port on switch by Emulab.

M3: Monitor node allocated to link by Emulab, like shaper node.

M4: Monitor machine wired between 2 experimental nodes.

M5: Monitor node connected using Linux OS bridging support

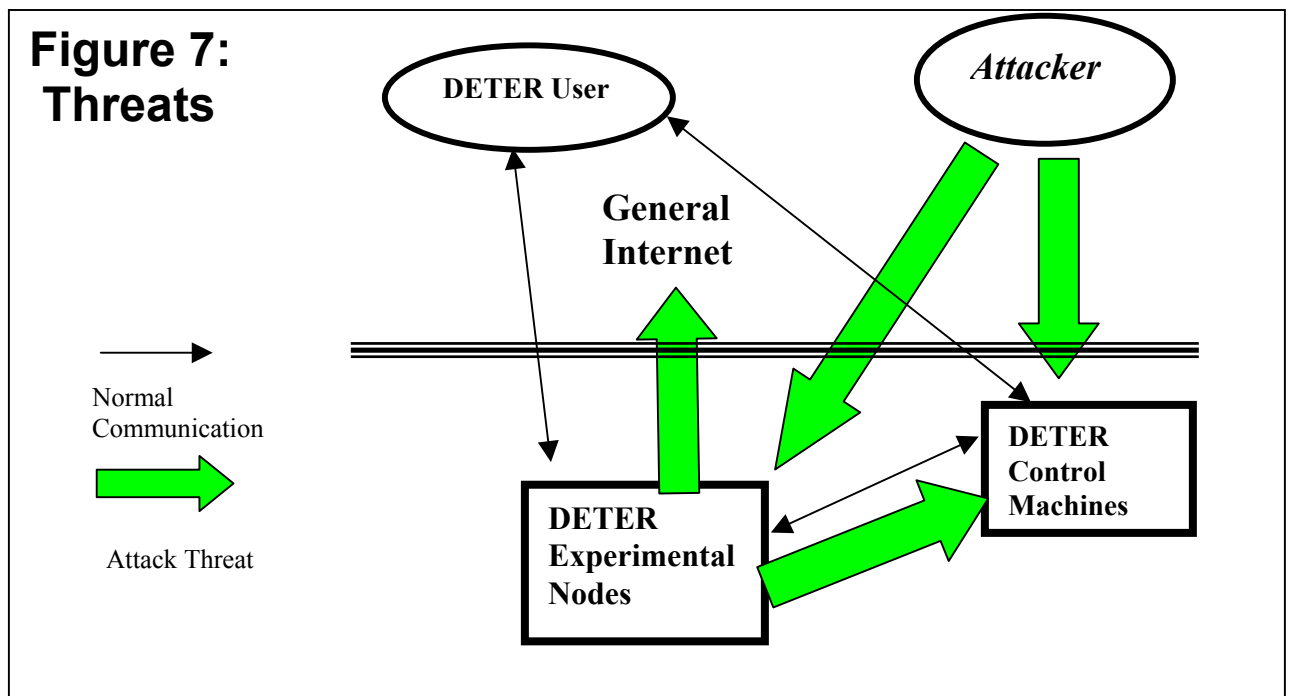


5 DETER Security and Isolation

This section outlines technical approaches to security and isolation in the DETER testbed. *Security* is required to provide protection from harm, while *isolation* refers to maintenance of independence among different experiments. This document describes the resulting technical issues, while a separate document [Policy04] specifies policies and procedures.

Some of the security requirements for the DETER testbed are unique, while other security requirements are more stringent than is typical in the Internet. Security issues for DETER include protection, containment, and confidentiality. *Protection* is required against attacks from the external Internet, while *containment* is required to prevent testbed experiments from attacking the testbed control mechanisms or escaping into the Internet (these threats are illustrated in Figure 7). Finally, some experiments may require *confidentiality*.

While protection and containment deal with active attacks, there is another class of vulnerabilities: *passive threats*. When an experiment involves a virus or a worm, any piece of data extracted from the experiment may contain pathogenic code that could be later activated, e.g., by accidentally opening a file for execution. This dangerous code might be stored in a user account on the User server, or it might be left in a cluster node that is subsequently used by a different experiment. A *decontamination* (or sterilization) procedure is needed to meet these passive threats.



5.1 Technical Requirements

Effective use of the testbed requires that the containment, isolation, and confidentiality requirements be balanced by usability and manageability for experimenters. DETER experiments will be categorized as described in the operational procedures document [Policy04], and this categorization will be used to select the technical measures used for security. Many DETER experiments will execute code that deliberately pushes against the boundaries of valid functionality; we must assume that it will search for, and find, any security hole that exists. Experiments that are particularly dangerous will require more intrusive measures to achieve the required security, impacting the ease or remote experiment setup and operation.

In more detail, the general security and isolation requirements for DETER are as follows.

- **Isolation**

To be scientifically sound, experiments run on the Deter testbed must be protected from inadvertent or intentional interference from other experiments and other experimenters. Performance isolation is significantly harder for DETER because some experiments are going to try to saturate links or nodes to the point of stressful failure. We need to clarify when and how performance overloads will become isolation failures.

There is also a security aspect to isolation: experiments must also be protected from, or made aware of, intentional or inadvertent interference by traffic outside the testbed. For example, denial of service attacks against the management infrastructure of the testbed, or against the capacity of links carrying traffic between remote sites in the testbed, could affect the results of an experiment. Running completely separate physical infrastructure to connect testbed sites would minimize the latter impact, but this solution could be cost prohibitive.

- **Containment**

Experiments run on the Deter testbed may involve malicious code or the deployment of traffic generation or attack tools that could seriously affect the operation of the open Internet if such code were to be leak out of DETER. The function of containment is to prevent the release of virulent programs or hostile data into the Internet.

In addition, the DETER testbed must protect itself against attacks originating from experiments. We can generally assume that no DETER experiment will be specifically designed to exploit weaknesses peculiar to the testbed, but we must assume that it may attack any host it can reach, including the testbed control machines. This raises the possibility of a virulent experiment infecting a control machine and then using that machine as a platform to launch itself into the Internet.

- **Protection**

Most DETER experimenters will load, manage, and collect the results of their experiments remotely from their home machines, across the open Internet. Any facility connected to the Internet is subject to attack from Internet worms, viruses, etc. This problem will be especially acute for the DETER testbed, which is likely to form an especially attractive target.

Furthermore, DETER's support for remote experiments broadens protection problem to possibly include the experimenters' home machines. An experimenter is required to use SSH and SSL to access the User and Boss servers, respectively; this should prevent snooping and active attacks on these connections. However, we must be concerned with the possibility of

an attacker taking over the experimenter's home machine(s) and thereby gaining DETER access (e.g., using Trojan horses to find passwords).

Finally, we note that containment and protection are intertwined. If protection fails and a DETER server (User or Boss) is taken over, it may be used as a platform for an attack back to the Internet, if containment fails.

Confidentiality

Experiments may require confidentiality of input, output, and the tools and countermeasures in use for the experiment. Confidentiality for tools is important when traffic generation tools might themselves be useful to attackers if released. Confidentiality of the experiments is necessary when vulnerabilities not yet known to the attacker community are being tested. Confidentiality of traffic traces is important, even though it is expected that such traces will be sanitized of identifying information before use on the testbed. Experiments may be required to sign Non-Disclosure Agreements to obtain traffic traces. Finally, companies whose products are under test may also impose confidentiality constraints on the experimenters. The testbed policies, procedures, and technical design must support all of these possible requirements for confidentiality.

5.2 Security and Isolation Issues

This section analyzes the security and isolation aspects of the design plan for the DETER testbed (Section 4). We organize this analysis in terms of the significant components of the testbed.

- **Experimental Network (ENet)**

For each active experiment, the programmable patch panel dynamically creates a set of network interconnections with a requested topology. Each link is a separate VLAN, to isolate it from other links in the same or different experiments. We refer here to the set of emulated links for a given experiment as an *Experimental Network* (ENet). Experimental node addresses on an ENet may use either private addresses or addresses from the open Internet; the latter may be necessary for proper emulation of the external environment on which such nodes would normally run.

An ENet instance is isolated from other ENet instances, from the Control Network (see below), and from the Internet.

- **Server Nodes: User, Boss, and Serial Line Servers**

These three servers are the only externally-visible components of the testbed, and therefore they are the primary focus for protection against outside attacks. They are connected to the outside Internet through the External VLAN (EVlan) and the Gatekeeper firewall, which is configured to limit access to legitimate nodes and addresses.

Furthermore, the User and Boss servers are accessible from inside the testbed; compromise of either of these machines would provide a path for information to exit the experimental network into the Internet, even though there is no direct forwarding path for IP packets between an ENet and the external Internet. Hence, protection of these machines is essential for containment as well as for protection.

Here is a possible threat that results from this design. An experimental node must be able to send packets to the Boss server for initiation and monitoring of experiments. Suppose that a worm running in a legitimate testbed experiment discovered the IP address of Boss on the Control

Network, and is able to penetrate Boss due to some exploit. Since Boss has access to the Internet, the worm may be able to propagate outside the testbed.

From the Internet, users can access Unix login accounts on the User server. This feature of Emulab is a considerable convenience to users, but it provides Internet access to a full Unix platform with all its possible vulnerabilities. Therefore, access to the User and the Serial Line servers is limited to SSL sessions using keys administered by the Emulab control plane. However, these keys may be widely distributed among research groups, and the resulting fragility of User security is a cause for concern. If User were compromised, it could be used to attack users' files or running experiments, or worse, as a platform to attack the Internet.

The User server also contains the experimenters' file storage, which is subject to isolation and perhaps confidentiality requirements.

The Boss server machine is particularly critical since it contains the administrative and control machinery and databases for the testbed. On the other hand, its interfaces are already somewhat "narrower" than the full Unix capability of Users. Remote users access Boss only as a web server, to register projects, configure, start, and monitor experiments. This web access must use SSL. Currently access control uses clear text passwords, although it may be possible to use the authentication facility that SSL potentially provides.

- Control Network

Control communication within the testbed uses a set of logical links (VLANs) called collectively the Control Network (See Figure 4). This network links, the User and Boss servers, the control ports of the experimental nodes, and the control hardware communicate over. For security and flexibility, the Control Network is segmented into VLANs that are interconnected through a central router (see Figure 3). All interfaces on the Control Network use private IP addresses. The Control Network may be vulnerable to DoS attacks as well as layer-2 attacks, from experiments.

The Control Network is used by both Boss and experimental nodes to access the User server. In particular, an experiment may mount NFS file systems stored on the User server, while Boss sets up export configuration files in User to limit NFS access to file systems belonging to the project running the experiment.

The Control Network is used by the Boss server to communicate with the control hardware VLAN (see below) and with running experimental nodes, providing services that include downloading disk images, a DHCP server, and a DNS server.

- Control Hardware VLAN (CHvlan)

This VLAN segment of the Control Network is accessible only to the Boss server, which uses it for configuring the programmable backplane and driving the remote power controller.

- Control Network VLAN (CNvlan)

This VLAN connects to the control interface on each experimental node. It is used by Boss for downloading disk images and monitoring node health, by experimental nodes to access services on Boss, and by experimental nodes to mount file systems stored on the User server. The range of uses of this VLAN create a significant exposure. For example, if an experiment can use it to take over one of the servers, it can then launch itself into the Internet. Firewall rules in the central router can constrain the traffic to the extent possible, but for highly virulent experiments this may not be sufficient.

- Cluster Interconnections

The discussion so far has concerned security in a single cluster, but the DETER testbed includes multiple experimental clusters interconnected across the Internet (Figure 4). With the single control site that characterizes the initial design (Section 4.3), the security issues described earlier are mainly relevant to that control site. The cluster interconnections will use IPsec tunnels that should provide effective protection and containment for these interconnections. However, these links across the Internet cannot provide experimental performance isolation; the interconnection links will be subject to uncontrollable variations in delay and loss, and even to DDoS attacks.

If and when the testbed design model is extended to use a distributed testbed control plane with some subset of a Boss server at each site, then the design and configuration at each site must provide equivalent provisions for isolation, containment, protection, and confidentiality.

- **Experimental Portals**

At the opposite extreme from disconnected operation for severe-threat experiments, benign experiments may be allowed to use portal nodes as explained in Section 4.4.

However, portal tunnels may still create a threat. If portal users set up additional connections to their local test node(s), e.g., for control and monitoring, then these nodes will be outside the DETER security boundary. Then the IPsec tunnels, which penetrate the security boundary, might conceivably form channels for attacks on the cluster or breach the cluster containment to allow dangerous code to escape from DETER.

5.3 Defense Methods

We turn now to a consideration of approaches and mechanisms to defend against the threats discussed in the preceding section. It is not practical to attempt to provide absolute security, due to the complexity of the testbed and the need for a reasonable amount of user convenience. We instead apply mechanisms that provide at least partial security against the most likely threats, and we adopt a defense-in-depth strategy with multiple overlapping mechanisms and approaches. There will be a real possibility of mis-classifying the threat level of experiments (or of experimenters taking inadvisable short cuts on “safe” versions of potentially dangerous experiments). Therefore, DETER should always invoke those additional security mechanisms that don’t have a devastating impact on usability.

5.3.1 Firewalls

Firewalls are critical to security of the DETER testbed. The Gatekeeper firewall, implemented as an Ethernet bridge, is deployed between the external VLAN and the open Internet to control all traffic into or out of the testbed. There is another firewall within the Control Network router to constrain the internal testbed control traffic on the Control Network.

The firewall is statically configured via SSH login from a specific trusted local network. We expect to need a mechanism for dynamically updating the Gatekeeper filters. For example, the portal mechanism will require “punching a hole” through the firewall for a specific experiment (see Figure 5).

The Gatekeeper is configured to block both incoming and outgoing traffic that does not conform to rules for the communication necessary for external control of testbed experiments. It may also enforce other security mechanisms, e.g., require the use of IPsec tunnels to encrypt and protect traffic into and out of the testbed. In addition, the Gatekeeper could be configured to allow access to the External VLAN only from designated network addresses; these filters would have to be dynamically updated by the administrative mechanisms of the testbed. It is presently unclear whether the additional safety of limiting IP addresses in this way would be justified by the administrative and user headaches it might cause. The

router in the control is configured to allow only known patterns of communication among the components on the testbed Control Network, described earlier in Section 5.2.

Example: Gatekeeper should disallow outgoing SYN's from Boss or User to the Internet.

5.3.2 Virtual Private Networks

As indicated earlier, remote user access to the testbed must use virtual private networks created by SSH and SSL to authenticate access and encrypt testbed traffic that crosses the Internet. This protects traffic from disclosure and modification during experiment setup and retrieval of results and provides stronger assurance that messages destined for Boss, User, and the Serial-line servers are legitimate. Virtual private network technology in the form of IPSec tunnels will be applied also to interconnect the control network and the experimental networks between testbed sites

For isolation and containment, CNvlan will be further segmented into an individual per-experiment control VLAN. The current Emulab control software does not provide this isolation, but this must be an objective for the DETER project.

5.3.3 Physical Private Networks

A fundamental part of the DETER testbed design is the use of VLANs to provide (virtual) physically private networks, for both experimental isolation and containment. As noted earlier, the use of VLANs needs to be extended to create per-experiment CNvlans.

An issue of concern: can the switch hardware be sufficiently overloaded to the point where it starts acting as a hub, ignoring VLANs?

If available, separate wavelengths of light on fiber networks will be used to build dedicated network links interconnecting the testbed sites. Such physical (spectral) separation will provide us with stronger isolation of the performance effects of attacks since bandwidth consumed between sites will separate from the pool of bandwidth available for production use of the Internet. This will provide isolation of these effects in both directions, protecting the internet from the effect of high bandwidth consumption by experiments on the DETER testbed, and will eliminate communication bottlenecks created for communication between testbed sites when the internet is subject to external denial of service attacks (or even heavy congestion caused by other factors).

5.3.4 Narrowed User Interface

Emulab was designed to give high priority to user convenience. It therefore allows users unrestricted access to Unix accounts on the User server, access to the web server on Boss, and direct IP connectivity to their running experimental nodes. DETER must alter the balance of user convenience vs. security to place more emphasis on the latter. At present, DETER disallows direct IP connectivity to experiments. A design issue is whether the DETER testbed ought to “narrow” the user interfaces to User and Boss to better protect them from external and internal threats.

For example, a bug in the web server could allow a successful penetration of Boss. In addition to supporting the web server for users, Boss supports all the critical control plane functions of the cluster, so this would give the attacker complete control of the testbed, a potentially disastrous situation. One defense against this threat [Lepreau, Private communication] would separate Boss into a front end that contains the web server and a separate backend that administers and controls the testbed. The assumption is that a carefully constrained “narrow” interface between them would be much easier to defend than a general HTTP interface. However, this change would be a significant programming task.

Similar, but somewhat less severe, hazards attend the User server. This system is prone to a much broader class of vulnerabilities, but penetration of the machine would be less catastrophic to the testbed. It is an open question whether we need to narrow the user interface to Users. Logging into the User server allows an experimenter to, in turn, log into a particular DETER node. DETER allows this form of relayed connectivity instead of the direct IP connectivity that Emulab allows.

An alternative to relaying through a User account might use an explicit relay proxy developed just for this access. This might be software that could be booted on a PC to serve as a relay proxy to communicate with an experimental node. This relay proxy would open an encrypted channel through the external firewall and connect with the User server using this restricted interface. This is sort of a NAT-like function, since private addresses are used for the control interfaces of the experimental nodes, while public addresses would be used externally. However, the explicit application-layer gatewaying of the relay proxy should give better control and enhanced security in both directions.

Another, more complete, approach to narrowing the interfaced to the Users server would develop a complete remote control interface that allowed only specific control actions. It could be implemented as a bootable CD that will boot its own OS and application interface, configured to preclude any communication mode except the control actions for DETER.

5.3.5 NFS Security

The Emulab control software rewrites the `/etc/exports` file of the User server as experiments come and go, to limit an experiment to mounting only its own file systems. However, NFS security depends upon using the IP address of the node's control interface as the authentication token, which threatens isolation and might threaten containment. Segmenting CNvlan into a VLAN per experiment will provide isolation.

Even with a CNvlan segmented into per-experiment VLANs, an experiment will still be able to clobber its own files on User, which may (or may not) be a cause for concern. A possible approach would pre-stage" files into the experimental environment (see the remarks at the end of Section 4.3). There might be a small set of private NFS servers connected to the programmable patch panel. Before an experiment was started, its allocated private server would be downloaded from the user's file system on User. Then a dynamic filter update in the Control Network router would disallow NSF traffic to User while the experiment runs. When the experiment is complete, the private file server's contents would be uploaded back to User.

The long-term storage of confidential files on the User server is a cause for concern about confidentiality. If there is any failure of isolation at any time, the persistence of this data could allow disclosure. Similarly, there may be long-term storage of virulent software such as worms and viruses in the same file systems. Again, any failure of isolation could unleash one of these agents within Emulab. Both of these cases can be handled by encryption of the stored data. Decryption might occur when an experiment begins and after the data has been copied to a private NFS file server, as suggested above; encryption of result data could take place before it is copied back to User from the private file server. Then the decrypted data would exist only within the closed (and high isolated) environment of a single Experimental Network.

5.3.6 Disconnection

For experiments that deal with malicious code or that are otherwise categorized as dangerous, the standard fire walling and encrypted tunnel mechanisms described earlier may not provide sufficient protection. It may be necessary to run such experiments with a logical, or even a physical, disconnection from the Internet and perhaps from the Control Network. For such experiments, any user services required for the experiment must be pre-staged on the experimental nodes themselves; they will be

accessible only through the Experimental Network. It may also be advisable to give exclusive use of the testbed to an experiment that poses a severe threat. By implication, no access should be provided to the testbed while such an experiment is running. Further consideration is required on whether dangerous experiments can be allowed to span multiple clusters.

Control of such a disconnected experiment must be provided via scripts. These scripts may run on the experimental nodes or on a new set of “mini-Boss” private control nodes. There would be a limited set of such nodes that could be allocated to disconnected experiments that were space-sharing the cluster. Scripting will be aided by the development of a security-oriented experiment control API, which the EMIST project is developing for DETER

Once disconnected, the experiment may not be reconnected until (1) the threatening experiment is concluded, (2) shutdown procedures have completed, (3) all data generated by the experiment has been rendered safe via encryption (see Section 5.3.9), and (4) the testbed node disks have been reloaded with a default image that is known to be clean.

As noted in Section 5.3.7, monitoring within the testbed may raise an alarm about a possible attack or breach of containment, to cause an automatic disconnect.

How will disconnect be achieved? Controlled? Enforced?

5.3.7 Security Monitoring

Despite application of the technical means described earlier, it must be expected that loss of containment might occur due to mis-categorization of an experiment, to a bug in the software implementing the containment techniques, to failure of VLAN switching due to overload, or through some other unforeseen event. For this reason it is important to monitor all components of the deter testbed looking for unexpected behavior that may represent breaches of security or isolation.

For example, the Gatekeeper and Control Network firewalls should run Intrusion Detection software to detect anomalous patterns of traffic. In the Gatekeeper, this will help identify external threats and penetrations in incoming traffic and breaches of containment in outgoing traffic. The Control Network router may also detect attempted, and perhaps successful, breaches of containment. This task is simplified because, with the exception of the experimental network itself, all the packets flowing across other components of the network are intended for a single application, managing and running the experiment. We can exhaustively list all the communication that we should see in terms of origin, destination, ports, and protocol. Any packets that don't confirm to this list may indicate an attempt to breach containment through the control network interface.

At an advanced stage of our understanding of these security issues, it may be possible for a detected intrusion to automatically isolate the testbed from the Internet. Currently we must depend upon the operations staff to recognize and react to the report. A detection of such communication may also lead to the turning off the experimental side of the control network to prevent communication with the User or Boss servers. As we get more experience with such attempted breaches we may be able to develop countermeasures that are less intrusive than complete shutdown of the testbed as a whole. Once shutdown, the testbed must be restarted from a known safe state.

Another form of monitoring would create leak detectors. For example, to help detect leaks of information from a particular experiment, we might create message sinks outside the experimental network with IP packet marking that is known only within the experiment. This marking might be IP addresses, IP header options, or some higher-level addresses like email addresses. If a message (IP datagram or email message) makes it to a sink, there is a leak. Again this might generate an alarm or actually shut down the leaky experiment.

5.3.8 Portal Security

The threat posed by portal tunnels, described earlier in Section 5.2, can be ameliorated by extending at least a partial protection boundary around the remote experimental machine(s). This might take the form of a CDROM that boots a firewall that is configured and controlled from DETER operations. The remote user would be required to interpose a machine running this firewall code between the experimental machines and the local network, preventing any back-door connection to the Internet. Of course, some experimenter may fail to completely isolate his/her nodes using the prescribed firewall, but the risk may be acceptable since tunneling into DETER via the portal tunnel seems like a relatively unlikely threat.

5.3.9 Decontamination

At the termination of an experiment, Emulab normally wipes the disk of each experimental node by downloading a default disk image that is known to be clean and reboots the node. This node decontamination procedure needs to be performed infallibly, and it assumes that the node hardware prevents a program from modifying the BIOS.

Task: examine this mechanism in Emulab and check for holes or exceptions

However, the downloading process is likely to write only those disk blocks that are needed by the new system. This may leave some blocks unwritten, still containing contamination from the earlier experiments. Furthermore, for complete certainty it is necessary to write all zeros and then all ones into each disk block.

In addition, decontaminating the experimental nodes will not remove all traces of an experiment. Data files generated during an experiment may be corrupted by malicious code, e.g., a live virus or worm. A mechanism is needed to prevent accidental infection by data removed from the testbed. There are two major threats here.

- During the experiment, virulent code might leak out to the user site through the relayed login connectivity (Section 5.3.4). The only defense known against such leakage is to ensure that high-threat experiments are executed with the experimental nodes disconnected from the outside world.
- The experiment may write the malicious code file into the User file server, and the user may later FTP it to his site. A general defense against this threat is to use encryption to “envelope” data that is exported from a User file system onto the Internet. The envelope might also contain additional administrative information about the project, experiment ID, run time, etc.

When the enveloped data reaches a user site, the user can decrypt it for local use. This could expose the user to danger, but the testbed would have no liability for this. Note the danger that automated file-handling software may invisibly decrypt the file upon reception and leave an unmarked dangerous file on the user’s machine; explicit user action should be required to remove the envelope. A safe procedure could supply the decryption key to the user on paper, either via postal mail or via FAX.

- Since the malicious data may also persist in the User file system, it may be useful to store all data in the file system with encryption. Then exporting would involve double encryption.

Design issue: modify NFS to encrypt every disk read/write, or do a post-experiment pass to encrypt only specified files?

There has been some discussion of sanitizing persistent data that is collected during an experiment by scanning such files for viruses, worms, and other dangerous code. With encryption, this would give an additional layer of protection.

5.3.10 Protecting Remote Experimental Nodes

It is infeasible for the DETER staff to provide the same degree of protection to remote experimenter machines as we provide within DETER. Nevertheless, there must be some concern about the possibility of breaking into DETER via a user machine, although the threat seems narrower than the threat of a direct attack on DETER.

- The lowest line of defense is to require by policy that experimenters follow good security practices on the local systems that are used to access Boss or User [Policy04].
- Experimenters could be required to boot their local systems from a clean and properly configured system disk supplied by the DETER operations staff, before accessing DETER remotely. This would provide much stronger assurance with only modest impact on usability of DETER.
- Finally, the greatest security (and least convenience) would result from banning remote experiment access and requiring that experimenters travel to a testbed site. We hope that we will not need to go this far.

5.3.11 Gathering Information from the Internet

An experiment in DETER may need access to the real Internet DNS, and it might need to peer with external BGP speakers. Proxies will be required within Boss for these functions.

5.4 Attack/Defense Matrix

Much of the preceding discussion is summarized in the Attack/Defense matrix of Figure 8. Here rows represent the origination point of an attack and columns represent the target of the attack.

This needs to be recast in terms of the components and terminology of the actual plan.

Figure 8: Attack/Defense Matrix

Key:

- EVlan: External VLAN
- TCN: Testbed Control Network
- ECN: Experiment Control Network
- ENet: Experimental Network

	Internet	EVlan	TCN	ECN	ENet
Internet	N/A	External firewall limits incoming ports, dests, and address origins to testbed users. Access inbound subject to DOS attack on network links. VPN protects content inbound and outbound.	No routable connection and defense on firewall, User, Boss servers prevent path in.	No routable connection and defense on firewall, User, Boss servers prevent path in.	No connection. Vulnerable to DOS on VPN tunnels. May use separate lambdas.
EVlan	External firewall limits ports, dests, and address dests of packets to testbed	Boss and User servers apply host-based firewalls to block packets received on	No routable connection, and defense on firewall, User,	No routable connection, and defense on firewall, User,	No connection.

	users. Subject to release of testbed data if authorized testbed users export data they should not. Interface and encapsulation tools mitigate inadvertent release.	unexpected interfaces. Systems disable all service not necessary for their function. ID system to monitor traffic to serial line server and shut down system if packets seen that indicate attack.	Boss servers prevent path in.	Boss servers prevent path in.	
TCN	No routable connection. Defense on User, Boss servers prevents path out. Control router/firewall to defend User and Boss servers.	No routable connection. Defense on User, Boss servers prevents path out. Control router/firewall to defend User and Boss servers.	Control firewall router limit access to known port address pairs, ID system looks for anything else and shuts down testbed if detected.	Control firewall router limit access to known port address pairs, ID system looks for anything else and shuts down testbed if detected.	No connection.
ECN	No routable connection, defense on User, Boss servers prevents path out. Control router/firewall to defend User and Boss servers. Intrusion response to shut down connection for malicious experiments.	No routable connection, defense on User and Boss servers to prevent path out. Control router/firewall to defend User and Boss servers. Intrusion response to shut down connection for malicious experiments.	Control router firewall allows only limited access from experimental network. Attempts at other ports or addresses detected and shuts down connection entirely.	Control router firewall blocks access on ECN between nodes.	No connection.
ENet	Rate limitation on any VPN tunnels to be applied, or use separate lambdas.	No connection.	No connection.	No connection.	Switch provides isolation.

5.5 Security in DETER Operation

This section reviews the security provisions of DETER in terms of the experimental cycle. The operation of an experiment occurs in several phases: proposal review, experiment initiation, execution, and result capture.

5.5.1 Proposal Review

Experimenters propose an experiment that is evaluated for technical merit and for safety by two separate committees. The procedures for this are described elsewhere. A series of experiments may be proposed and considered together so long as the safety categorization is the same for all experiments.

In this phase, the investigator is required to identify the potential consequences of breach of containment, isolation, or confidentiality, categorize the experiment according to the taxonomy described elsewhere, and explain why the categorization is correct. The safety review panel will consider the explanation and either accept or re-categorize the experiment. This categorization will determine the configuration(s) of the testbed during the following three phases of the experiment and whether the experiment will be authorized for use of the testbed

5.5.2 Experiment Initiation

During the initiation phase of the experiment, investigators will be granted controlled access to the User server to stage materials that will run on the experimental nodes. Some of this data will be truly staged, i.e., moved to the nodes executing the experiment when the nodes are allocated. Other data will remain on the user server for access by the testbed nodes (for experiments that are categorized as having minimal danger).

5.5.3 Experiment Running

Prior to initiation of the experiment, external access to file areas available to the experimental nodes will be disabled. Data will be loaded onto the testbed nodes, including perhaps a private NFS file system, through the control network. The experimental network will be configured to support the desired topology of interconnection for those nodes participating in the experiment. Depending on the categorization of the experiment, the experiment control network may be disabled, preventing access by the nodes to the testbed control network. Finally, the experiment will be initiated through the console ports to the individual machines, or automatically as the machines are booted.

Intrusion detection tools on the control network, in the external VLAN, and outside the testbed will monitor for any indication of successful egress, or for the control network, any unexpected activity. Ideally, detection of such activity will automatically cause the firewalls to close down and perhaps power down the control router. In any case, the testbed operators will be notified. (Note that false positives will be more tolerable in the DETER environment than in the normal Internet case. DETER users must be prepared for paranoia in DETER operation.)

Upon conclusion of the experiment, all of the nodes under test for an experiment will be shut down and their disks will be wiped clean (“decontaminated”).

5.5.4 Experiment Conclusion – Capturing Results

Depending on the categorization of the experiment, some result data may have been written to the User file server, where it will be available for retrieval as described later in this section. Since the experimental nodes will be decontaminated, any data to be saved must be moved to the User server. This data will be encrypted on disk and further encoded in a manner that will prevent any possibility of accidental execution. This is important because malicious code that had been running on the experimental node might have modified the data to cause easy execution if these procedures were not followed.

Once the experiment has concluded and all experimental data moved to the User server, the investigator will be allowed to retrieve data from his or her experiment through a controlled interface. All data retrieved will be encapsulated and encrypted to prevent inadvertent execution by any software that might come across it. The policies and procedure document [Policy04] describes constraints on what kinds of data is allowed to leave the testbed, but it is recognized that if we allow the investigator to retrieve any data from the testbed that we cannot prevent a determined investigator from violating these guidelines.

5.5.5 Severe Threat Operation

For some experiments, any existing testbed path to the Internet may constitute an unacceptable risk. When the threat category of an experiment is judged sufficiently high, the Gatekeeper can be set to remove all IP connectivity to the External network, for the duration of the experiment.

A specific mechanism for this needs to be designed. See also what UCB is doing.

Once disconnected, the Gatekeeper may not be reconnected until (1) the threatening experiment is concluded, (2) shutdown procedures have completed, (3) all data generated by the experiment has been rendered safe via encryption, and (4) testbed re-initialization actions have been completed.

Since disconnection prevents all remote access to the testbed, dangerous experiments must use batch mode rather than execute interactively. Since no access is provided to the testbed while an experiment of this threat category is running, such experiments must execute under control of the Emulab's batch facility, or, in extraordinary circumstances, interactively by an experimenter physically present at the cluster site. An experiment of this threat category precludes the use of experimental nodes from multiple clusters, and it may preclude sharing a single cluster with simultaneous experiments.

5.5.6 Experiment archive

Configurations and experimental data that is archived by DETER will be encrypted, so that reading the data will require the manual provision of an encryption key. Such encryption may use a combination of public and conventional techniques, and depending on policy, might require the participation of multiple participants (e.g. the testbed operator and the investigator, or the testbed operator and some other entity) to decrypt the data.

5.6 Emulating the Internet

Some DETER experiments will require an environment that looks like the real Internet. For example, this may be necessary when examining the spread of malicious code that has hard-coded addresses or address ranges, that depends on the domain name system for its operation, or that depends on other services such as routing. Some of these services are specific to an experiment and must be run on experimental nodes allocated to the particular experiment. Internet addresses may be assigned on this segment that duplicate addresses outside the testbed, allowing the code under study to see the same address space as in the real world.

For services such as DNS or BGP that depend on data from outside the testbed, proxy servers run on the internal nodes can return pre-loaded cached data to the experiment. Preloading can be based on traces of the experiment, e.g., to determine what names the DNS can resolve. Alternatively, they might be allowed to make realtime queries into the real Internet. Such queries would be submitted on the external VLAN through a pared down caching name server configured to chain queries from a single hard-coded internal node to a single external name server.

5.7 Future Plans

This section has described a number of security mechanisms to meet the specific DETER requirements. While the set of mechanisms was motivated by a consideration of the potential vulnerabilities, the mechanisms themselves overlap in complex ways to increase the security and isolate of DETER. The intent is to develop a defense-in-depth, to significantly reduce the possibility of accidental compromise due to configuration errors, for example.

We plan to use a combination of analysis and red-teaming to look for further vulnerabilities and to set priorities on the security mechanisms. Many of these mechanisms require software extensions to the base Emulab system, and it will be necessary to choose implementation priorities wisely.

6 Conclusions and Acknowledgments

This document has described an abstract model and a concrete design for a network testbed suitable for experiments in network security, including the ability to safely execute dangerous code. The detailed design is based upon the pioneering work of the Emulab group at the University of Utah. The testbed is an important part of the DETER project's larger goal of developing a powerful experimental environment to enhance the practice and science of security research.

The design described here was developed by a Working Group on DETER Architecture, composed of staff from USC ISI, UC Berkeley, and McAfee Research. We are very grateful to Jay Lepreau and to his testbed staff at Utah for their generous and expert help in installing the Emulab system.

7 References

[Click00] E. Kohler, R. Morris, B. Chen, J. Janotti, and M. Kaashoek, The Click Modular Router. ACM TOCS 18(3), August 2000;

[Emulab02] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, *An Integrated Experimental Environment for Distributed Systems and Networks*. OSDI 2002, December 2002.

[NDDTEF03] W. Hardaker, D. Kindred, R. Ostrenga, D. Sterne, and R. Thomas, *Requirements and Justification for a National DDoS Defense Technology Experimentation Facility*. Network Associates Laboratories, March 2003.

[NSFnrt02] J. Kurose et al, *Report of NSF Workshop on Network Research Testbeds*. NSF Workshop report, http://gaia.cs.umass.edu/testbed_workshop, November 2002.

[Policy04] *DETER Testbed Policies and Procedures*, DETER Project staff, March 2004.

8 Appendix A: Full Hardware Configurations

A.1 ISI Cluster Configuration

- 64 IBM Netfinity 4500R experimental nodes with the configuration:
 - Dual 733MHz PIII processor, in a 3U case.
 - 1GB of RAM
 - One 17GB SCSI disk
 - PCI slots (3 @ 64 bit, 2 @ 32 bit)
 - Two 10/100bT Ethernet interfaces, one with PXE boot support enabled on a PCI slot.
 - Two dual-port 1000bT Ethernet interfaces (4 interfaces total), in 64bit/33MHz PCI slots. Intel PRO/1000 MT Dual Port Server Adapters.
- Eight Sun V65X experimental nodes with the configuration:
 - Dual 2.8 Ghz Xeon processors in 2U cases
 - 2GB of RAM
 - 6 36GB Ultra320 10,000rpm disks configured as one file system
 - 2 on-board 1000bT Ethernet interfaces
 - 1 Intel PRO/1000 MT Dual Port Server Adapter
 - 1 Intel PRO/1000 MT Server Adapter
- VLAN Data Switch
 - Cisco 6509 switch chassis with a 720 supervisor blade and 6 blades of 48 1000bT ports.
This supports the 4 x 64 GigE Ethernet ports to the nodes. The choice of Cisco was dictated by Emulab support; the cost of modifying the software for another vendor's switch is expected to far exceed what one might save in hardware.
- Control Router
(TBD)
- Control Network Switch
 - Foundry FastIron Edge Switch 9604
- 'User' Server
 - Intel XEON 3.06GHz 533 FSB 512 K, in a 4U case
 - 2GB of RAM
 - 2 TB (250 GB x 8) RAID storage
- Master ('Boss') Server
 - Intel XEON 3.06GHz 533 FSB 512 K, in a 4U case
 - 2GB of RAM
 - 0.96 TB (120 GB x 8) RAID storage

- Firewall
 - A two-interface machine to serve a dedicated firewall and access point between the cluster and external networks:
 - Dell PowerEdge 650: Pentium 4 2.4GHz, 512K, 544 MHz FSB in a 1U case.
 - 256MB DDR 266Mhz
 - 40 GB 7200rpm IDE
 - Intel Pro 1000MT Gigabit dual-port NIC
 - A 1U, Serial Line Servers
- Serial Servers
 - Two generic 1U machines for serial line servers:
 - Pentium 4 2.66Hz 512K 533 FSB processors in 1U cases
 - 512 MB DDR PC2100 RAM
 - 80GB 7200 RPM IDE
 - Two 1Gb Ethernet ports, on board.
 - One serial server machine, same except 2.8Ghz.
- Remote Power Controllers
 - Five RPC28-30NC with twistlock connectors
 - One RPC27-20NC with twistlock connectors.

In the future, ISI plans to add an additional 64 nodes -- 1U servers with 5 interfaces -- and a second Cisco 6509 switch. Eventually we will want to purchase two 4-port 10GE blades to link the two switches. However, the optics of the 10GE cards is expensive, even the multimode fiber ports that we would use for short distances, and we don't need the high inter-cluster capacity in the early stage. We can use 1GE initially.

A.2 UC Berkeley Hardware Configuration

- Each of the 34 experimental nodes of UCB's cluster has the configuration:
 - Sun V60x dual 2.28Ghz Processor
 - 2 GB of RAM (2 x 1GB DIMMs)
 - Two 36GB 10K RPM Ultra320 SCSI disks
 - PCI-X slot (Full height/full length 64 bit/100Mz)
 - PCI-X slot (Half height/half length 64 bit/100 Mhz)
 - Two 10/100/1000bT Ethernet ports (on board)
 - One 1000bT Intel Pro 1000 MT dual-port Ethernet port
 - One 1000bT Intel Pro 1000 MT Ethernet port (control VLAN)
 - CD and floppy drives

- VLAN Data Switch
 - Foundry FastIron 1500 with 15 slots, 10 16-port Gb line cards (160 ports total)

A.3 ISI-East Hardware Configuration

(To be supplied)