Isolated virtualised clusters: testbeds for high-risk security experimentation and training*

Joan Calvet[1,2], Carlton R. Davis[1], José M. Fernandez[1], Wadie Guizani[2], Mathieu Kaczmarek[2], Jean-Yves Marion[2], and Pier-Luc St-Onge[1]

[1]École Polytechnique de Montréal, Montréal, QC, Canada
[2]LORIA, Nancy, France

Abstract

Adequate testbeds for conducting security experiments and test under controlled, safe, repeatable and as-realistic-as-possible conditions, are a key element for the research and development of adequate security solutions and the training of security personnel and researchers. In this paper, we report on the construction and operations of isolated virtualised testbeds used in two separate security research labs in Canada and France, as part of a joint collaborative effort. The main idea was to use mid- to large-scale isolated computing clusters to obtain high levels of scale, manageability and safety by heavily leveraging virtualisation technology, open-source cluster management tools and a network architecture separating experiment and control traffic. Both facilities have been used for conducting different types of security research experiments, including in-lab reconstructions of botnets, denial-of-service attacks, and virus detection experimentation. They have also been used for teaching and training students in experimental security methods. We describe these facilities and the criteria that we used to design them, the research and training activities that were conducted, and close by discussing the lessons learned and the pros and cons of this approach.

1 Introduction

Experimentation is a keystone of the scientific method and of technology innovation and development. From a scientific and knowledge discovery point of view, it is not necessarily an end in itself, but when used in combination with other methods such as direct observation and theoretical modelling, it provides soundness to the knowledge and understanding of the phenomena under study. At the same time, experimentation plays a key role in the training and education of highly qualified personnel in scientific and technical fields. Additionally, from an engineering point of view, experimentation allows us to determine the viability, applicability and reliability of the tools we develop and intend to use in the real world, while at the same time providing us with the opportunity to determine how to best use these tools, i.e. by finding out their optimal operational parameters for deployment.

In this paper we describe our approach for building and operating testbeds for conducting computer security experimentation, in order to meet these goals. The key idea behind these testbeds is the use of isolated, special-purpose computing clusters, which heavily use virtualisation technology, physical and logical separation, and cluster mananagement tools in order to attain a high level of flexibility, realism, and scalability.

At its inception, this concept was initially developed at the École Polytechnique de Montréal in 2005 as part of a grant proposal to the Canadian Foundation for Innovation. Once obtained, this 1.2-million$ infrastructure-only grant made it possible to build and equip the experimental facilities of the *Laboratoire de securité des systèmes d'information* (SecSI) that became operational in 2008. That same year a joint collaborative effort was started with the *Laboratoire de Haute Sécurité* (LHS) of the LORIA in Nancy, where a similar facility was being designed. The collaboration started with the exchange of technical personnel and eventually of graduate students. The primary goal was to try to attain an "economy of effort" by sharing know-how and try to develop a joint concept of operations covering system architecture, experimental procedures, security policies, administration tools and procedures, thus hoping to avoid duplicating efforts in both labs. Furthermore, given our many research interests in common, it was hoped that sharing tools and procedures would become an enabler for joint research projects, where experiments could be designed in one facility and run or replicated in the other.

Today, both the SecSI and LHS labs have operational isolated testbeds that share the same design and operational philosophy, as well as many of the same tools. In

[*] Authors listed in alphabetical order

both cases, the facilities have been used to conduct security experiments in various research areas and successfully used to train highly qualified personnel in experimental security methods.

It is important to note that neither the facilities nor the approach described in this paper are entirely unique. Indeed, they share many elements of commonality both in terms of objectives and construction with other computer security testbeds and approaches such as the well known DETER facilities (based on the Emulab platform) [1, 2]. Nonetheless, for many reasons it was not, and is not, a viable option for the co-authors of this paper to use.

Initially, DETER was not fully operational and open for business when our facilities first started. The planning and construction of these facilities and that of DETER have since followed mostly separate paths, with many of the same ideas having been adopted in both. As the DETER project progressed other reasons became apparent that motivated us to stay on a separate path. First, we wanted to be able to conduct experiments at a scale of thousands, or even tens of thousands of systems, something in principle unachievable without virtualisation, which Emulab did not implicitly support. Second, due to the fact that DETER is a distributed and remotely accessible facility, its security policy did not allow (at the time) the conduct of high-risk security experiments that the authors intended to perform as part of their research programme. Third, for non-technical reasons it was deemed important to provide alternative facilities to Canadian, French, and non-US researchers, beyond those of the US-funded DETER.

Thus, the main contribution of this paper is not to report on "newer" and "better" testbed facilities and approach, but 1) to report on an *alternative* approach that was developed in parallel to others such as DETER, and shows some advantages and disadvantages with respect to those, 2) to propose and describe a unified list of criteria that we believe all such computer security research facilities should have, and 3) to report on our experiences and lessons learned from the use of such facilities. In Section 2, we describe the criteria that were the driving force behind the choices we made in terms of equipment, architecture, security policies, tools and methods used for their administration. We then describe in Section 3 the technical details of the actual testbeds, pointing out similarities and differences between the two labs in Canada and France. We briefly describe in Section 4 some of the research and educational projects that have been conducted, including some projects that are being planned. Finally, we conclude by discussing the advantages and disadvantages of this approach and the lessons learned so far in Section 5.

2 Requirements for Security Testbeds

From the onset, the main objectives that influenced the design and operational philosophy of these facilities were precisely some of the difficulties that have been attributed to security testbeds, namely scale, risk, realism, rigour and setup complexity. We explain below the various criteria that influenced the decisions that were made in the design of these facilities. While many of these criteria are probably the same that have influenced the design of other facilities like ours, we are not aware of such an explicit and comprehensive list of criteria. Beyond using it to explain the rationale behind our design choices, we propose it (and open it for criticism) to the community as a potential *template* for designing and even evaluating adequacy of computer security facilities.

2.1 Risk Analysis and Risk Management

Depending on the kinds of research activities performed, security research can involve risk to the existing information infrastructure and ultimately to the public at-large. From a confidentiality point of view, it is important to protect data on software vulnerabilities and malware collections from use for nefarious purposes. In addition, certain data collections that are used in security research, such as network traffic traces, could potentially contain private information. Similarly, details about network architecture or configuration of defensive mechanisms from real networks that are being emulated in a testbed facility, should be protected from unauthorised access, as knowledge of these details could jeopardise their security. Most importantly, experiments involving live malware could "spill" into a real computing environment, affecting it from an integrity and availability point of view, for example, in the case of accidental releases of malware samples on previously uninfected machines. Alternatively, the effects of such actions could be indirect, such as those caused by researchers interacting with infected and criminal-controlled systems. In that case, this action could potentially trigger a premature and unnecessary arms race between the criminals and security researchers and security product vendors, by alerting cybercriminals that someone is "onto them", or on weaknesses in their tools and approaches.

Thus, the responsible researcher must take all appropriate measures in order to minimise these risks to acceptable levels, at least proportionate with the expected benefits of such research. In some rare cases, academic institutions have put in place a formal framework for evaluating and managing such risks to the information infrastructure, in a similar fashion with high biological or environmental risks. This is the case at the École Polytechnique de Montréal where in addition to restrictions self-imposed by the researchers, a committee and a procedure have been setup to 1) evaluate risk associated with these projects, 2) assess their relative advantages and disadvantages, and 3) evaluate and vet the use of adequate risk-mitigating counter-measures [3]. Hence, the SecSI lab's design and operational procedures had to be sufficiently safe to convince the members of the oversight committee. To that end, a tight security policy was adopted involving three aspects: physical security, logical security and personnel security.

In terms of physical security, the installations hosting these testbeds (both at LHS and SecSI) are protected by strong physical barriers, surveillance systems, and separate physical access control systems using multi-factor authentication. The facilities are further segregated into separated zones, concentrically arranged, so that lower-level security zones protect inner zones with more sensitive information assets (data storage, servers, etc.). In terms of logical security, these testbeds are in principle isolated from outside computer networks. For the higher-risk experiments, the testbeds used have no gateways, at any level of the protocol stack, with the outside world; in other words, they are *air gapped*. For lower-risk experiments and for experiment design tasks, lower security *development testbeds* are used that can be connected to external networks through appropriately configured, restricted and access-controlled gateways. Finally, some personnel security measures have been adopted such as sensibilisation training of lab users (students and staff) to potential risks, the signing of a security policy outlining accepted and unauthorised use, and even in some cases voluntary background checks of personnel with access to the more sensitive data and portions of the facilities. Of course, the physical, logical and personnel security aspects of the security policies are all linked together. For example, higher risk experiments involving higher sensitivity assets are conducted only in the inner zone of the facility (e.g. experimentation with network-capable, self-propagating malware), which is subject to the strictest logical separation rules (air gap), and only by personnel on which we have been able to assess a low risk of unauthorised behaviour.

In the case of the SecSI lab, the administration of École Polytechnique had to review and approve the lab's security policy and technical procedures, not only because of the above mentioned risk assessment committee, but also to ensure the safety of the personnel using the laboratory, and also to make sure that the personnel security policies did not infringe on privacy or the right of access to education.

In summary, the physical and logical layout of the facilities was designed so that different levels of risk could be managed by using layered security policy and procedures, and only applying counter-measures where and when necessary, in order to minimally hinder the flexibility needed for this kind of research.

2.2 Realism, Scale and Flexibility

The evaluation of security solutions or performance analysis of threats can be essentially conducted in four fashions: mathematical modelling, stochastic simulation, in-laboratory emulations, or analysis of real world data. The latter has often been the prefered method in security research, especially with regards to malware analysis. For example, the "malware enthomologists" of this world, whether working for anti-virus companies or academic security research labs, go about the Internet Jungle to catch new bugs, bring them back to their labs and disect them with sophisticated tools (reverse engineering with unpackers, disassemblers and debuggers). While this approach is commendable, it has limits in that it does not provide a complete picture of the malware design space. In particular, observation of the real world often only provides one or only a few data points in the space of parameters that could change both in the space of security solutions and that of threats alike. In that respect, mathematical modelling and stochastic simulations provide more flexibility, in that they can efficiently provide performance predictions for a wide-varying range of parameters and design choices. However, the main problem with these approaches is that it can be difficult to validate that the predictions obtained through them correspond to what would be observed in reality; in other words, it is hard to know *a priori* how realistic these modelling and simulations results can be.

Laboratory experimentation offers an interesting compromise. On the one hand, the controlled conditions in which they are ideally conducted provide 1) the ability to validate previous experimental results obtained by others, i.e. *repeatability*, and 2) the ability to vary the parameters and characteristics of security solutions or threats being studied, i.e. *experimental control*. On the other side, the use of the same or similar pieces of hardware (whether real or emulated) and the same software that is present in the real world, whether offensive or defensive, adds a level of *a priori* realism that is hard to attain just by modelling and simulation. However, in order for security testbeds to provide such an adequate level of realism they must obey the following criteria:

Versatility. The ability to emulate or natively run a large variety of software. In particular, it should be possible to run various kinds of operating systems. While this level of variety is not so crucial for hardware, the inclusion of new types of hardware (especially at the network level) should be possible without overly disturbing the facility's layout or the experimental design.

Synchronisation. Whatever technology is used to deploy and run the various levels of hardware and software, the notion of time should not be more than linearly distorted, and should be equally distorted for all emulated/recreated elements of the experimental setup.

Soundness. Measurements made during the experiment should not appreciably affect the behaviour of the emulated system, and in particular should not appreciably affect the values of any of the observed quantities.

Transparency. It should not be possible for software running on the testbed to easily detect that it is running on an experimental testbed. This applies to both malicious and legitimate software running on the testbed.

Scale. The number of elements (machines, subnets, processes, etc.) that are being emulated or recreated in the experiment should be large enough to approach the numbers in the real world, or at the very least large enough so that statistically significant results can be obtained.

Environment. The static conditions describing the environmental setup should be as close as possible to those of typical equivalent environments in the real world. This includes network topology, server configurations, proportion of machines and equipement in various roles.

Background. The time-varying conditions (the dynamics) of the emulated environment should approach that of real world systems. In particular, the baseline behaviour of the environment in the absence of emulated threats or attacks should approach the behaviour of equivalent real systems, in as much as an equivalent baseline (i.e. *ground truth*) can be established and is well known, including time-varying conditions such as user-generated network traffic, network delays, CPU load factors and processing delays, etc.

Lastly, since research budgets are never infinite and resources are limited, additional efficiency requirements on security testbeds are necessary in terms of flexibility, manageability and usability. We postulate that security testbeds should exhibit the following properties:

High-level Experimental Design. It should be possible for the experimental designer (researcher, student, practicioner) to create a high-level specification of what components are to be part of the emulated environments, how they are connected together and how they should behave. In particular, the design process should be scalable (i.e. the designer should not have to individually specify the parameters of each emulated machine). Ideally, some graphichal or high-level language should be used.

Deployability. A high-level description should be enough to allow automated and semi-automated tools to generate a set of lower-level configurations settings for all relevant testbed equipment and components.

Manageability. From a complete set of configuration settings, it should relatively simple to automatically reconfigure the whole testbed, within reasonable delays (hours or minutes rather than days or weeks), thus allowing rapid transitioning between different experiments and sharing of the facility by different research projects.

Portability. Ideally, it should be possible to port with relative ease the high-level design of an experiment, or even possibly its low-level configuration description, from one testbed to another testbed independently of actual hardware and equipment being used.

Sterilisability. For scientific soundness reasons, it must be possible to efficiently sanitise the testbed between experiments, so that previous configurations do not contaminate the results of future experiments. Furthermore, for security reasons it should be possible to do so with reasonable assurance that no resident malware can remain in the testbed after this process.

3 Isolated Virtualised Clusters

Given, these criteria for risk management, realism, scale and flexibility, the natural initial choice was the use of a sizeable cluster of homogeneous machines, used solely for the purposes of computer security experimental research. This concept by itself was not new at the time of our initial grant proposal, as the main ideas behind the DETER testbed had been proposed just before. However, there are several important differences with respect to the DETER concept, namely in terms of hardware, virtualisation, networking and configuration management.

Hardware. The DETER testbed is a set of distributed facilities, that may contain a collection of heterogeneous machines. These facilities are linked together and accessible via the Internet, so that researchers at various locations may access them. In contrast, both the LHS and the SecSI lab use two separate and isolated clusters each, composed of homogeneous hardware: 1) a development cluster, used for lower-risk activities such as testing and development of experimental setups, training, and running lower-risk and smaller experiments, and 2) a production cluster, where higher-risk and large scale experiments are performed. In Montréal, the development/low risk cluster is composed of one 11U rack, containing 14 blades with a single quad-core processor, 8 Gb of RAM, dual 136 Gb SCSI disks, and a network card with four separate gigabit Ethernet ports. This rack is mounted on wheels, so that it may be moved from the lower to the higher security zones of the laboratory or vice-versa, as required, after the prescribed sanitisation procedures have been performed. On the other hand, the production cluster contains in total 98 blades, identical to those of the development cluster. The blade modules occupy two full-height 42U racks, with two extra racks hosting the networking equipment and a server with 12 Tb of storage, used for control and storage. In Nancy, the LHS also has two clusters. The development cluster is mainly used for low-risk research on anti-virus detection using morphological analysis [4]. The production cluster is comprised of 24 identical servers, each with 80 Gb of disk, 16 Gb of RAM, and two gigabit Ethernet ports.

Virtualisation. In order to attain the criteria of sterilisability, portability and deployability, we decided from the onset that our testbeds should make heavy use of virtualisation technology. In order to meet the versatility criterion and be able to emulate both Linux and Windows, we opted for VMWare products to the detriment of linux-only or Windows-only virtualisation solutions. Furthermore, and because of the sterilisability and transparency criteria we set aside virtualisation solutions, such as Xen, that (at the time) required the guest OS to be re-compiled in a VM-aware fashion. This is in contrast with the DETER testbed which does not integrate virtualisation *per se* in its experiment design and deployment process. While some researchers have used virtualisation in Emulab for network research purposes [5], what was being virtualised were the network elements. Also, one group of security researchers has been able to use VMs in testbed for worm propagation experiments [6]. However, the DETER platform does not allow for an experimental design process with a granularity level down to the VM. While it is possible to design and specify an experiment down to the host, there are no embeded mechanisms in the DETER/Emulab platform to specify or automatically deploy, start or configure individual VM.

Network. A key feature of the testbeds that was introduced to support the soundness and synchronisation criteria, was the separation between emulated traffic (that could transit on virtual or real switches) and the control traffic used to configure the testbed, start and stop experiments, and gather measurement data from the emulated elements. In particular, we wanted to avoid having conditions in which such control traffic could affect network delays of the experiment traffic, or conversely that the timely delivery of control or measurement data be jeopardised by experiment traffic (as would be the case in a DoS attack experiment, for example). To that purpose, two physically separate networks were created, with a separate set of switches, network ports and patch panels connecting them, such as illustrated in Figure 1. More concretely, the first port of a given host/blade (e.g. `eth0`) is assigned as its *control port*. It is wired separately to the *control cabinet*, where it is patched onto an access switch, connected itself to the core switch of this control network. This control cabinet also includes the control and storage server, as well as network connections to the control consoles, located in another zone of the lab with equivalent level of security but higher level of comfort (not air conditioned). The other interface cards on the blades are separately wired to the *experiment cabinet*, that contains separate patch panels, and access and core switches. The experiment cabinet can also host network equipment such as routers, security appliances, etc., that are to added to the emulated environment.

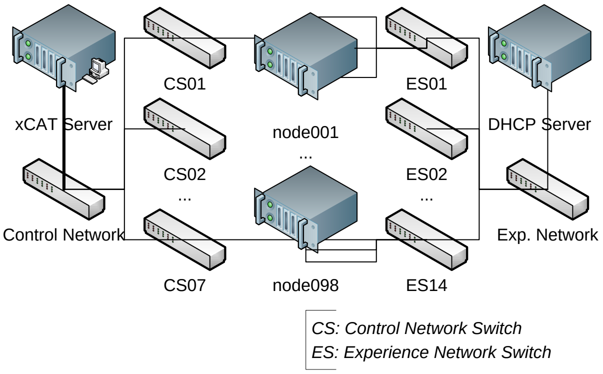

Figure 1: Cluster architecture

Experiment design and configuration management. The DETER platform uses the tools developed by Emulab to describe experimental setups and generate configuration information that can be pushed to the DETER hosts. In the case of the SecSI lab, an alternative solution had been suggested based on the Extreme Cloud Administration Toolkit (xCAT), [7], an open-source cluster management tool. This option was made especially attractive when VMWare functionality was developed into xCAT. After careful consideration and a detailed comparative analysis [8] with the Emulab/DETER solution, we chose to follow the xCAT path.

From a management point of view, xCAT does essentially two things. It contains foremost a database of host configurations, including details such as machine template (i.e. location and name of the ghost image), hostname, IP address, etc. This information is stored in tables, that can be edited with a text editor or through a very primitive command-line interface. It then uses machine configuration image deployment and control commands, that take the information from this database and uses remote boot technology, such as PXE, to force active hosts to reboot and retrieve the new image and configuration. The beauty of the xCAT code developed to support VMWare ESX is that a natural extension of the database information and deployment commands can be applied to VMs, in an almost transparent fashion.

Thus, the management, experiment design and deployment processes look like this. First, the researchers come up with an abstract, high-level description of the desired environments. Typically, this includes 1) building a handful of VM templates or ghost images (e.g. machines to be infected, gateways, innocent bystanders in a DoS attack, etc.), and 2) coming up with a network topology, addressing plan and host naming convention. Depending on the size of the experiment, the population of the xCAT tables can either be done manually or automatically. In the case of our larger experiments, this was done by writing perl scripts that generated the appropriate xCAT tables, given the high level specification of the environment. At the LHS, however, xCAT is not used at all, and instead custom-made scripts are used to express the high-level design, push VM images and start them, all with the same tool.

In both cases, however, there is a choice. If the experiment is to be run completely virtualised, it is necessary to first deploy and install appropriate host images containing ESX . This is done using xCAT and fortunately the corresponding tables only need to be filled once (since the hardware configuration of the cluster never changes). The only experiment-specific operation is the definition of virtual networks within the host, and their assignment to the experiment physical ports; this is a relatively simple task since it involves pushing a single ESX configuration file onto the appropriate hosts and can be easily accomplished with xCAT commands. If the experiment is going to be run *on the metal*, i.e. with no virtualisation, then the machine templates and configuration generated above as ghost images are pushed onto the respective machines, as we would do for the default ESX host configuration. In this case, care is taken so that the control port is only visible to non-experiment processes and programmes (e.g. only visible at boot time or only accessible by monitoring processes in kernel mode), so that it cannot be taken over by malicious software or accidentally polluted by experiment traffic.

In general, our policy is to make maximum use of virtualisation in all experiments. However, two sets of reasons can push us to run experiments on the metal. One is the possibility of experimenting with VM-detecting or VM-breaking malware. This is one reason why our current security policy does not allow us to run unknown and not carefully analysed malware on our production cluster. If reverse engineering of a high-risk, unknown malware requires at-scale live experimentation, then we would do it on the development cluster run in high-risk mode in the high security zone. This has not been the case so far, as we were able to do all reverse engineering with desktop machines. In particular, our analysis of the Waledac botnet showed that even though it did employ virtual environment detection techniques, these were only used during the code unpacking phase. We thus deemed it acceptable to run the unpacked version of Waledac on VMs in our production cluster, without compromising the transparency criterion or accepting unnecessary risk. The second reason that could force us to run experiments on the metal are soundness and synchronisation considerations. This has been the case in our DoS research. Since in that context we were flooding SMTP servers to the limit of their CPU and memory allocations, we did not want to falsify results by adding to the targeted machine the burden of VM context switching and overhead. Hence, in that case both targeted servers and legitimate clients were implemented by hosts running directly on the metal, because of the need for precise response time measurements.

One last point concerning experimental methodology that we will not discuss in detail is that of measurement and result gathering. In general, the method of choice due to its simplicity is to have legitimate software or instrumented malware log the necessary information onto local files on the VM, that can then be centrally gathered and copied *a posteriori* to the server with xCAT commands. Since the virtual disks can be mounted as read-only partitions by the host, it is not necessary that these VM be started during this data collection phase, hence minimising risk. For almost all experiments conducted and planned, this is an adequate method. However, it is possible that in certain cases this might not be viable, e.g. if instrumenting malware is undesirable or impossible. In that case, there would be other options such as monitoring processes running either in the same VM or directly on the host machine, the latter being more transparent, but potentially harder to implement.

4 Case studies

These testbeds have been used in three different types of research projects and for individual and formal training.

In Nancy, the low-risk cluster has been used for testing new approaches in malware detection. While relatively low-risk because of logical separation, this kind of research does involve large malware sample collections, which is why it was conducted within the confines of the LHS. The production cluster has been operational for several months, but it has not yet been used for large-scale research projects. However, it is scheduled to be used for a reproduction of the botnet experiments run in Montreal as described below.

In Montreal, two different research projects have used the production cluster. The first one, completed in the summer of 2009, involved a comparative analysis of the resilience of various SMTP server applications under denial-of-service attacks. In that case, 42 blades were used for the experiment. All machines were run on the metal, including the targeted server and the legitimate client. The remainder 40 blades were used to attack the server by opening as many STMP sessions as possible and keeping them active for as long as possible. From a testbed evaluation point of view, the experiment was sucessful. In addition to an example of how xCAT could be used to run an on-the-metal experiment, it also provided proof that the production cluster could be easily logically separated (both control and experiment networks were divided into two air-gapped segments) by altering our patch panel configuration. In this case, the other 4 blade modules were used during that same period to conduct a comparative analysis of Deter/Emulab vs. the xCAT-based solution [8].

The second research project is probably our most impressive accomplishment with the testbed so far. In our previous work on the Waledac botnet [9], we discovered a sybil attack that could significantly cripple its command and control abilities. Using our testbed, we first recreated an in-lab version of Waledac involving 3,000 bots. This also included the *protector* layer of proxy servers and the top-level command and control HTTP server, that was providing properly formatted malicious commands to the bots, including spamming commands. Our environmental setup also included SMTP servers that were used to receive the spam and hence measure the *yield* of the botnet. For example, we were able to establish that a Waledac botnet of such size was able to send 13,200 spams per minute! In order to test our theory about the efficacy of the Sybil attacks, we then deployed VM containing sybil code of our own brew to redirect the "real" towards a "fake" command and control server that we also put in place. We were then able to verify that the attack does indeed work. More interestingly, however, the experimental results showed that the load constraints on the both the "real" and "fake" command and control servers were severe. This fact, which could not have been discovered in simulation or in-the-wild study, has helped us make significant progress in understanding some of the design decisions made by the botmasters of Waledac.

Finally, we have used the development cluster of the SecSI lab in the teaching of a graduate class in advanced topics in computer security. One of its main goals is to teach graduate students about quantitative methods in security research and product evaluation, including modelling, simulation and, of course, in-lab experimentation. A first assignment had them learn how to use VMWare, xCAT and the other related tools. Since we were not able to give physical access to the lab to all students in the class (background checks would not have been a viable option for students attending a regular class), the development cluster had to be connected to the university network for the duration of the course. One of the blades was accordingly setup as a secure gateway. A second assignment asked the students to reproduce the results of a previous simple experiment on malware propagation [10] that was performed in our lab in 2007. For this, each student team was assigned two blades, that they could configure independently. For their final class project, teams were given the option of using the cluster for designing from scratch and running a computer security experiment. Two teams decided to do so, one creating their own "concept" botnet and the other conducting a worm detection experiment using the NetADHIC tool [11].

5 Lessons Learned and Conclusions

Generally speaking, our experience has shown us that isolated facilities such as ours have some significant advantages with respect to remotely-accessible distributed facilities such as DETER. First, it has allowed us to safely conduct higher-risk experiments such as the Waledac experiment, and that at a scale comparable to that of the actual botnet. It is important to note that this experiment has allowed us to discover facts otherwise unattainable by either reverse engineering or in-the-wild observation.

Nonetheless, there are some disadvantages to isolated facilities. First, access by other research groups not located in the same university or city is limited. While there are no administrative restrictions on their use by researchers from outside academic or industry researchers (this is actually encouraged), in practice the restrictions of the security policy make it such that only those who are remotely collaborating with on-site researcher or have temporarily physically moved to the facilities have been able to use them. Thus the total number of users so far in Montreal is in the low tens, and under that for Nancy. However, while economies of scale on equipment are lost, sharing of tools, procedures and experimental know-how is still possible and beneficial, as the international collaboration between our labs has shown.

Another drawback of a more operational nature is the fact that experimental design and development on isolated facilities is more tedious and requires more careful preparation. For example, in both the DoS and botnet experiments, VM deployment and initialisation of the software on them was not always flawless, either due to configuration errors or network overload. Without direct access to the experiment network it was hard to find the faults and correct them. This was resolved in the botnet case by the installation of a python-based monitoring application, to which network commands could be sent to report on the status of the nodes and their programmes. Similarly, students and researchers had to carefully plan their work, since they did not have access to the Internet while doing a lot of this development and debugging work, due to the strict physical and logical access control policies. One of the main lessons learned was that the use of the low-risk cluster (potentially connected to external networks) in the initial phases of such development is not a luxury and should be encouraged.

Beyond the fact that our facilities are isolated, the other main differing characteristic with other testbeds is the use of machine virtualisation. In terms of the list of criteria for security testbeds we introduced in Section 2, the use of virtualisation introduces major advantages with respect to non-virtualised testbeds. The chief advantage is in terms of scalability because the same number of physical nodes can, for most experiments, support up to 1 or even 2 orders of magnitudes more emulated (virtual) machines. The price to pay is that other realism criteria could be negatively affected. For example, time synchronisation could be affected in the case of high CPU utilisation. Also, transparency could be lost if one does not take appropriate measures to re-configure default VM configurations to prevent VM detection by obvious techniques (e.g. the red pill programme). In the experiments run so far this was not a problem, as we deliberately limited the number of VM to keep CPU loads small enough to prevent real-clock to virtual-clock drift, and we were able to determine that none of the malware did active virtual environment detection once unpacked.

On the other hand, virtualisation does also provide benefits in terms of the flexibility criteria. As long as VM containment conditions are safely maintained, virtualisation offers some advantages in terms of deployability and manageability, but most importantly in terms of sterilisability. While all of these conditions can be achieved in testbeds where experiments run on the metal by physical interventions on the hardware, they become much simpler to enforce in virtualised environments. Finally, in terms of portability between our two labs, the use of virtualisation does provide us quite a bit of flexibility as VM templates can be transparently migtrated from one facility to the other, with minimal risk of incompatibilities. However, portability at the high-level experiment design is still an issue due to the use of different tools.

Beyond these pros and cons, there are some other limitations of the testbeds as they stand today that we intend to address in the future. As far as we know, all of these shortcomings are common to other types of testbeds as well. In terms of manageability, for one, while xCAT does allow configuration and control down to the VM, this level of granularity is not quite sufficient for development and debugging purposes. Beyond the python monitor mentioned above, it would be nice to have a standardised method for monitor or altering running programmes or processes through a centralised interface. Second, the deployment times for that many VM, while acceptable, is still in the order of hours. We are exploring network optimisation solutions (including multicasting) to be able to significantly reduce such setup delays. Third, we need to continue to explore and develop new avenues for efficiently and transparently collecting measurement results.

But the most important limitation concerns the realism of the testbed environment with respect to network topology. The topologies that we have used so far are very simple and primitive (essentially star). This is in part due to the fact that, unlike DETER/Emulab, there is no mechanism in xCAT for managing and configuring switches and other network elements. This is something that we will address in the future, possibly by re-using or getting inspiration from the DETER work in that area. The other reason, of course, is that it is not at all obvious from a modelling point of view what a "typical network" for such experiments should look like. This is in itself another whole science of experimentation research topic.

References

[1] Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K., Ostrenga, R., Schwab, S.: Experience with DETER: A testbed for security research. In: Proc. IEEE Conf. on Testbeds and Research Infrastructures for the Dev. of Networks and Communities (TridentCom). (March 2006)

[2] Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K., Ostrenga, R., Schwab, S.: Design, deployment, and use of the DETER testbed. In: Proc. DETER Community Work. on Cyber Security Experimentation and Test (CSET). (August 2007)

[3] Roehrig, C., Fernandez, J.: The supervision of research projects entailing computer risks within an academic context: the case of école Polytechnique de Montréal. In: Proc. INEER Int. Conf. on Eng. Education and Research (ICEE/ICEER). (August 2009)

[4] Bonfante, G., Kaczmarek, M., Marion, J.Y.: Morphological detection of malware. In: Proc. Int. Conf. on Malicious and Unwanted Software (Malware). (Oct. 2008) 1–8

[5] Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale virtualization in the Emulab network testbed. In: Proc. USENIX Ann. Tech. Conf. (June 2008)

[6] Li, L., Liu, P., Jhi, Y., Kesidis, G.: Evaluation of collaborative worm containments on DETER testbed. In: Proc. DETER Community Work. on Cyber Security Experimentation and Test (CSET). (August 2007)

[7] xCat: xcat: Extreme cloud administration toolkit. http://xcat.sourceforge.net

[8] Benzamane, H.: Analyse de systèmes de gestion de clusters: Xcat, Emulab. B.Eng. Thesis, Dept. of Comp. & Software Eng., École Polytechnique de Montréal (April 2009)

[9] Calvet, J., Davis, C., Bureau, P.M.: Malware authors don't learn, and that's good! In: Proc. Int. Conf. on Malicious and Unwanted Software (MALWARE 2009). (October 2009)

[10] Bureau, P.M., Fernandez, J.: Optimising networks against malware. In: Proc. Int. Swarm Intelligence and Other Forms of Malware Workshop (Malware). (April 2007)

[11] Inoue, H., Jansens, D., Hijazi, A., Somayaji, A.: NetADHICT: A tool for understanding network traffic. In: Proc. USENIX Large Installation System Administration Conf. (LISA). (Nov. 2007)