

LEXIS: an EXam Invigilation System

Mike Wyer and Susan Eisenbach
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ
mw@doc.ic.ac.uk

June 5, 2001

Abstract

University exams require a large amount of dedicated preparation, planning, and careful execution. This extended abstract briefly covers the issues and technologies involved in systematically securing a teaching lab to enable an exam to take place. It describes LEXIS the EXam Invigilation System, which has already been used successfully to conduct a programming exam this year, and it looks likely to be used heavily in the coming academic year.



1 Background

People learn to program by sitting in front of a machine and typing. However, formal programming examinations are usually hand written on paper. So the skill that is being tested is not the same as the one being learnt.

At Imperial College there was a lot of experience in having online programming *tests*. These were felt to be

sufficiently secure for testing programming skills, since logs were always available if there was any concern that a student had used unacceptable resources.

For our online tests our systems were run unrestricted. This means that there were no hardware or software constraints to prevent students from accessing their own files, sending or receiving email or even surfing the internet for help with their test. These activities were restricted by student honesty (and the threat that cheating would be detected) and invigilators' diligence.

Our regulations are such that one of the necessary conditions for passing the programming course is that a student must pass the final examination. It was felt that the system of checking logs was not workable for the examination as it would place too much onus on the lecturer to check huge logs looking for possible cheating.

The experience of the online tests created a demand for examining large numbers of students, concurrently, online. Fortunately, the tools available for a standard Linux platform are such that writing a system for running examinations with limited network access seemed feasible [1]. LEXIS, the *EXam Invigilation System*, is a rigorously developed combination of standard tools and management systems which provides an extraordinary level of security on a standard Linux workstation.

The target systems are well understood [2]. They consist of about two hundred general access PCs, all running RedHat Linux 6.2 [3] and directly connected to the Internet. A way of severely restricting the network was needed, and the most obvious and effective method would have been to simply disconnect the network during any

exam. Our network topology and hardware are such that this was a fairly straightforward option. The target machines would then be required to function correctly without any network. This raised several concerns about reliability, synchronisation, and monitoring. What would happen if a machine had a fatal problem during the exam, say a head crash? How long would it take to recover any data, if it was possible at all?

These issues encouraged us to look at other solutions to the problem. Leaving the network connected also introduces problems. There were still reliability issues, cheating might be easier and the whole exam could be open to external attack. It was vital that the worst-case failure of any of the constituent systems would not invalidate the exam. In order for LEXIS to be a success, the safety, security, and reliability of pen and paper had to be matched.

2 Requirements

Although most people are familiar with the security arrangements which accompany an official examination, they are not often encountered in a systems administration context. LEXIS needed to be developed to meet requirements specified by academics, who were not interested in altering requirements because of implementation issues that were raised.

2.1 Aims

The software must provide:

- familiar lab-like environment during exams,
- all resources necessary to complete exam,
- secure environment for completing exam,
- secure means of collecting exam answers.

The software must prevent:

- access to unauthorised data,
- access to other users on the network,
- distraction or interference from other users on the network.

Some exams may involve providing students with templates, stub code fragments, or other data. Likewise, the student will be required to create or modify files as part of the exam. The students will not have access to shared network volumes, so the files will need to be provided by LEXIS.

Each completed exam submission must be securely stored and associated with the right candidate number. Exam submissions must be available only to the LEXIS administrator.

As with any other examination, students will only be allowed access to permitted resources. In addition to the usual physical precautions of a written exam (no books, paper, phones, pdas, etc), the student should not have access to unauthorized stored data or communication systems on the workstation. It is also important to make sure that other users on the network do not interfere with the student during the exam, the on-line equivalent of the noisy students in the corridor outside an exam.

3 More detailed requirements

LEXIS needs to be managed from a master server, which will only be accessible by authorised users (examination officers and trusted administrators). While a workstation is in exam mode, only connections to and from the master server will be allowed.

Students will log in with their candidate number as their username and password. This means there will be a separate system account for every student taking the exam (where a candidate number is not available, some other unique identifier must be used). All files created will be uniquely identifiable as belonging to that student. The exam accounts have no special privileges. All work will be done in the /exam area on the disk.

There is not enough disk space on lab machines for a separate clean “exam” installation, so the normal lab setup will have to be secured. This is less of a disadvantage than it appears, as it removes the task of synchronising normal and exam setups. Likewise, it will highlight any areas of concern with the security of the standard lab setup. Securing the machine involves several stages:

- warn and log out any existing users,
- prevent any further logins,

- display warning on screen,
- shut down all daemons (especially all network services and the automounter),
- enable firewalling rules (no incoming or outgoing packets, except one specific link to the LEXIS server),
- unmount all non-essential filesystems,
- remount root directory with 'nosuid' option - this turns off all set-userid bits, which means that the exam accounts will not be able to affect system devices (such as the network) or mount filesystems,
- clear /exam area,
- clear all other directories which have been writeable (/tmp, /var/tmp and so on). This will be done by traversing the whole directory structure on the hard disk and clearing any directories with write access, rather than relying on a fixed list.

Once the machine is secure, it needs to be setup so that the student can take the exam:

- transfer account details,
- transfer exam files,
- display exam login screen.

When the exam starts, logins will be enabled again (but only for exam accounts). When the student logs in, the system will display all messages set by the examiner. During the exam, the /exam area will be backed up securely across the network to the LEXIS server at an interval specified by the examiner (default is every five minutes).

At the end of the exam (at the time specified by the examiner), write access to the /exam area will be revoked and login access will be removed. The files in /exam will be copied securely across the network to the LEXIS server. Once the files are confirmed as being on the LEXIS server, the student will be logged out (if still logged in) and the /exam area will be cleared. The standard lab setup will then be restored.

4 First stages of development

As is inevitable with all system administration tasks, there was only short development time and a large area of concern. Therefore, it was important to investigate currently available solutions. Although the requirements of the project were very specific, there was a good chance that some of the individual tasks could be covered by one of the many security tools, packages, and utilities available for Linux [1].

Firewalls seemed the most important issue that had to be tackled so the first step was to investigate the Linux kernel level firewalling. Linux 2.2 was the stable kernel at the time, so the ipchains interface was evaluated [4]. The evaluation proved to be very positive.

Using ipchains would give a precise control over the network traffic to and from each workstation involved in the exam. While this is not a novel idea to anyone who has been using ipchains, the key factor is that using ipchains provides an easy way to achieve *temporary* network security while still allowing certain connections such as OpenSSH [5] connections to a central server.

Most firewalling schemes are designed to be permanent; the rules are designed to be in place for perpetuity. With LEXIS, the rules are in place for a few hours. Not only do the rules have to be automatically applied, they have to be removed as well. While the techniques involved are straightforward, they have to be absolutely reliable: a client machine's network connectivity cannot be restored during the exam period if we lose contact.

We could now write software to enable the management and monitoring of exams in real time. Regular remote copies of all student work could be kept in case of catastrophe. If there was something wrong with the exam questions new versions could be sent to all student machines. This would require a file transfer mechanism.

Since security and reliability were of paramount importance, any file transfer had to be safe from interception and corruption. Since there was already an OpenSSH connection present, we needed a way to safely transfer files over that connection. This was achieved by uencoding the file to prevent any accidents with control characters, and an MD5 checksum was taken. The protocol we decided to use was to send the filename, number of lines, MD5 checksum, then the data. This would be implemented on the client and server side for 2-way transfer

of data (sending exam stubs and environment files to the client, and sending dumps back to the server).

Having taken care of external threats, network abuse, backups, synchronisation, and disaster recovery it was also important to prevent straightforward cheating.

Most of the activities which constitute illegal behaviour by students are privileged operations on the system. Operations such as mounting disks and creating trusted network sockets require either root access or set-uid root file permissions. By remounting the root filesystem without set-uid bits active, most cheating is eliminated. A positive side effect of this operation is that some system binaries that are installed setuid root (notably *man* and *ssh*) are also disabled. This would prevent the student logged into the machine from using the OpenSSH client to attack the only open network channel (ie the ssh link to the LEXIS server).

5 What was achieved

LEXIS used run mode 4. This enabled us to finely control the running services. It also means the system boots into a secure mode if it is restarted during an exam. Run mode 4 is currently unused by RedHat, although all the infrastructure is in place for using it.

LEXIS is a collection of Perl [6] with XML [7] configuration files and Ruby [8] scripts which cover all of the activities which constitute exam invigilation. It currently consists of 2000 lines of Perl and 200 lines of Ruby.

LEXIS server and clients communicate via an OpenSSH2 link. The LEXIS DSA public key is already installed on all client machines. The private key is stored, encrypted, on the LEXIS server only.

A system-wide LEXIS group is created. All exam accounts (which are newly created) are to be in the LEXIS group. All files to be accessed by the students during the exam have LEXIS group access privileges.

The details of the exam life-cycle include:

5.1 Exam life-cycle

Server startup The server reads config file, with list of files, candidates and client machines. The server connects to all client machines.

Client configuration All files needed for the exam are transferred to the client. New passwd and shadow files are created for the candidates for this session. The system PAM config files are restricted to the minimal set needed for the exam. Copies are taken of the previous configuration. Any logged in users are warned. The run level is changed to run level 4.

Client ready Firewalling is turned on. Unauthorised processes are killed. Non-essential filesystems are unmounted. All remaining filesystems are re-mounted without set-uid bits and registered with server.

Exam in progress Backups are taken of all student work at the rate specified by the config file, typically every 5 minutes.

Exam over The final state of all user files is taken. The normal lab configuration is restored. The system is reverted to previous run level (usually 5).

6 Conclusions

On 21st March 2001, LEXIS was used to invigilate a programming exam for 110 students on over 150 machines. The exam ran for two hours, 6600 dumps were taken, totalling 60 MB of data. A CD of final exam answers was created for the lecturers to mark.

The exam was administered from one server, although the machine in question was running very low on resources having to maintain 300 concurrent ssh connections (an admin connection, and a separate connection for dumps). The batching / forking code was added after the first full-scale test, where it took over 30 minutes to initialise 100 machines. Adding in parallelism brought the system start-up time down to about 5 minutes.

No problems arose during the examination. The pass rate was higher than would have been expected had the same exam been sat on paper.

LEXIS is available under the GPL from <http://www.doc.ic.ac.uk/~mw/code/lexis/>.

References

- [1] <http://www.linux.org/>

- [2] www.doc.ic.ac.uk/csg/
- [3] <http://www.redhat.com>
- [4] <http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html>
- [5] <http://www.openssh.org>
- [6] <http://www.perl.com/pub>
- [7] <http://www.w3.org/XML/>
- [8] <http://www.ruby-lang.org/en/>

7 Biographies

Mike Wyer is a recent graduate of Imperial College who currently works as a Systems Administrator in the Department of Computing. He has had a long-term interest in examinations and computers, having worked on exam registration in a final-year group project.

Susan Eisenbach is a reader in the Department of Computing where she is responsible for the teaching programme. Her research interests are programming languages for distributed computing.