Original citation:
Bhardwaj, J. (2012) The cyber security learning and research environment. Coventry, UK: Department of Computer Science, University of Warwick. CS-RR-451

Permanent WRAP url:
http://wrap.warwick.ac.uk/59533

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

A note on versions:
The version presented in WRAP is the published version or, version of record, and may be cited as it appears here.

For more information, please contact the WRAP Team at: publications@warwick.ac.uk

http://wrap.warwick.ac.uk/

The Cyber Security Learning and Research Environment

Technical Report CS-RR-451

Julian Bhardwaj
Department of Computer Science
University of Warwick, Coventry, CV4 7AL, UK
j.bhardwaj@warwick.ac.uk

October 26, 2012

Abstract

This report outlines the design and development of the Cyber Security Learning and Research Environment (CLARE). It explains how such a system can be implemented with minimal hardware, either on a single machine or across multiple machines. A prototype implementation is outlined, and the details of the design of the components that constitute the environment are provided alongside configuration documentation to allow for replication of the environment.

1 Motivation

Cyber security laboratories are becoming more common within academic institutions as computer science education refocuses on developing a new generation of cyber security professionals to meet the massively growing demand within the industry [1]. However, the majority of these labs consist of a dedicated set of workstations for students to use [2, 3]. These require a considerable commitment of resources, both hardware and space, by a university department. Other systems make use of virtualisation technologies to provide a server-centric environment that can be accessed from thin clients, however these come with substantial hardware overheads still [4, 5]. This is generally due to use of hypervisor technology which relies on hardware-assisted virtualisation (see Section 4.2). The Cyber Security Learning and Research Environment, known as CLARE, aims to be a lightweight approach to a cyber security laboratory making extensive use of modern virtualisation technologies on low end hardware, which perhaps does not have hardware-assisted virtualisation support.
The primary aim of CLARE is to remove the requirement of high-end server grade hardware which many virtualisation solutions rely on to provide a more lightweight approach to a cyber security lab. This allows for the use of graveyard/decommissioned desktop machines and therefore drastically reduces financial overheads. Additionally, CLARE is designed such that it is scalable to allow for the most efficient use of the resources available. Free and open source software is used wherever possible to provide a flexible framework which can be developed and customised further to meet the needs of a particular environment.

2 Design Overview

CLARE is designed to operate with the hardware that is available, and therefore can run on a standalone desktop machine or across multiple servers. Whilst this report focuses on a very minimal setup consisting of a single machine providing all components of the environment, the individual components can be split across different machines or duplicated for load balancing. The principle components are as follows:

- **Host Gateway** Provides access to the virtualised environment from the physical network. Responsible for managing containment of the environment and interfacing with the hypervisor.
- **Virtual Simulator** Consists of the hypervisor and associated virtual networking components. Provides the isolated environment to host security challenges and competitions.
- **Web Interface** The frontend system responsible for user and Virtual Simulator administration, as well as managing user interaction with the environment. Consists of a web server and database.

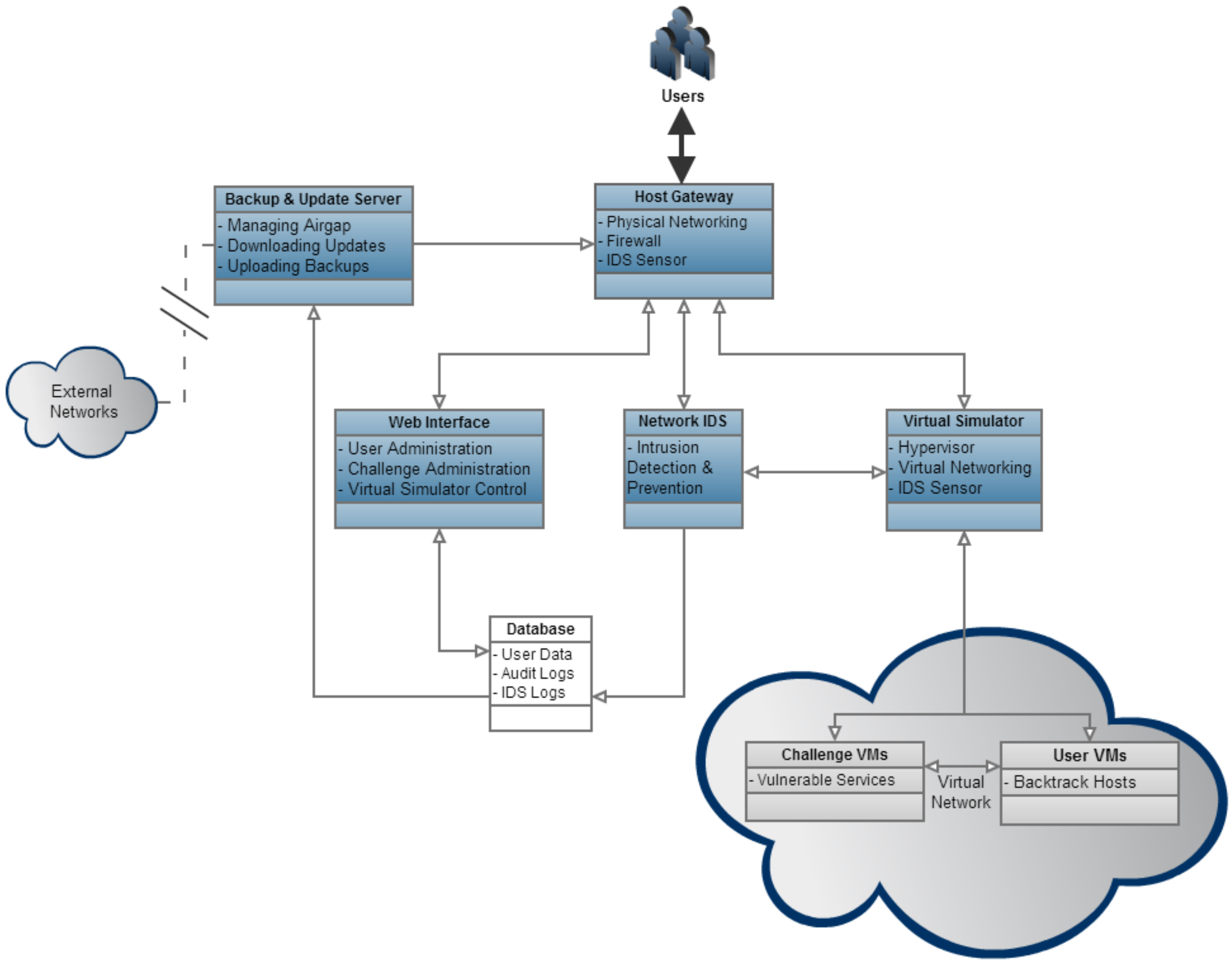

Figure 1: CLARE Design Overview

Other optional components include:

- **Network Intrusion Detection/Prevention System** Monitors containment of the system and ensures the damage resilience of the environment's infrastructure. Also can be used for providing audit logs of a users actions within the Virtual Simulator.

- **Backup & Update Management Server** Manages the air-gap between the isolated environment and external networks to allow for download of updates to the environment and backup of the backend database and audit logs.

These components can be configured in three different ways:

1. A single host system providing all components natively.

2. One or more host systems providing components through virtualisation (separate virtual machines for each component).

3. Dedicated host systems for one or more components.

Configuration 1 is recommended for a low resource environment, whilst configuration 2 is preferred when a single high-spec machine is available and configuration 3 is best suited to a high resource environment.

3 Prototype Implementation

To demonstrate how CLARE operates within a low resource environment with minimal budget available, the prototype detailed below used configuration 1: a single machine providing all components natively. Configuration files provided in Section 4 are taken from this prototype implementation.

Host Gateway

The machine currently used for CLARE is a desktop PC sporting an Intel i5-2400 CPU (3.10Ghz quad-core) and 8GB of RAM.

Physical Network Infrastructure

A Cisco Aironet access point (wireless 802.11g) is used in conjunction with a 4 port gigabit switch to provide an isolated network with no external connectivity to the university network nor Internet. Wireless is not required as a wired networked could be used for the isolated network, though it was decided that this was preferable so that a dedicated space is not required to access the environment.

Host Operating System

As all of the services provided by the Host Gateway run natively, an Ubuntu Server (11.10) installation is currently used. However, it is preferable to use a more lightweight operating system when different components/services are to be separated into separate virtual machines (configuration 2). Either a minimal Debian installation, or something as small as SliTaz[6] is suitable and capable of running a hosted hypervisor.

4 Software Configuration

Common packages available from the Ubuntu repositories are used, opting for open source where possible. Though it should be noted that some packages or parts of packages contain non open source components.

4.1 Host Gateway

The OpenSSH package is used for remote administration as well as Webmin[7] for ease of configuration and management, though these are both optional. Where Webmin is used it is recommend that it is configured to bind only to localhost and to be accessed by port forwarding through SSH - see Listing 6 for suggested firewall rules.

4.1.1 Networking

A dual-homed machine with two network interfaces is not required when configuration 1 is used. Instead, a wireless access point can be connected direct to the physical adapter or through a switch. It is intended that all routing is done by the Host Gateway in this configuration, therefore a separate router is not used and the host provides a DHCP and DNS. A dedicated router would be preferable where multiple physical machines are in use to prevent network traffic bottle-necking on the Host Gateway (configuration 2/3).

4.1.2 DHCP

The **isc-dhcp-server** package is used to provide DHCP services on both the physical and virtual interface (see Section 4.2.2 for details of the virtual interface). This allows for an IP and DNS to be pushed to user's machines that connect to the physical network and removes any custom configuration of user's machines. It is more efficient to have the Host Gateway provide a single DHCP (and DNS) for both the physical and virtual domains for a configuration 1 setup, however arguably having separate services provides better containment and should be used in configuration 2/3.

4.1.3 DNS

The **bind9** package provides the DNS service to the physical and virtual networks. This reduces the complexity of a user connecting to the Web Interface and the Virtual Simulator as domain names can be configured to resolve to these addresses. Whilst there is no Internet connection within the environment, it is useful to be able to provide access to chosen resources that have been mirrored locally on CLARE. The DNS therefore can be configured to redirect the domain names for these resources to the Web Interface instead. Moreover, the DNS can be used to resolve host names within the virtual environment.

4.1.4 Firewall

The **ufw** package provides simplified management of an iptables based Netfilter firewall, and is configured to block access across the physical and virtual domains (eth0 <-> tap1).
In addition to allowing access to the user facing services provided by the Host Gateway, it is also necessary to configure access to the RDP ports used by the Virtual Simulator to provide remote desktop connections to the client virtual machines. (see Section 4.2.4). Rather than allowing these rules to be configured dynamically by the Web Interface on a per user basis (which would introduce security concerns), a block range of ports is opened (3389-3489) and it is instead the responsibility of the Virtual Simulator to provide authentication to the services running on these ports. A suggestion to what this rules could look like is provided in Listing 6

4.2 Virtual Simulator

Type I (bare-metal) hypervisors are generally designed to run on servers and normally have very specific hardware requirements making it problematic to install them on desktop hardware. Type II (hosted) hypervisors are generally a lot more flexible and therefore in most cases the Virtual Simulator will run on a Type II hypervisor. It is intended that CLARE be fairly hypervisor-independant to allow for adoption of different and multiple vendor-products (whether they be open source or proprietary licensed). This said, VirtualBox is currently in use due to its flexibility and vast functionality.

4.2.1 VirtualBox License

As of version 4.0, there is not a separate Open Source Edition of VirtualBox [8]. Instead, closed-source components are distributed in a separate ‘extension pack’ which is governed by the ‘VirtualBox Personal Use and Evaluation License (PUEL) instead of the GPL version 2. The PUEL however allows for ‘Educational Use’ by an academic institution (schools, colleges and universities, by teachers and students) [9].
The VirtualBox extension pack is needed to provide VRDP remote desktop connections to the client virtual machines (see Section 4.2.4)

4.2.2 Virtual Networking

Whilst many hypervisors like VirtualBox provide their own set of virtual networking options, a hypervisor independant approach is taken here to allow for multiple hypervisors to be used in conjunction with each other. This also makes it easier for the Network Intrusion Detection/Prevention component to monitor both the physical and virtual networks.
VDE (Virtual Distributed Ethernet)[10] is a set of libraries and tools providing virtual hubs, switches, and routers that interface with physical adapters and tap/tun devices allowing for the creation of virtual networks that can be distributed across multiple physical machine.
The simplest network topology used here is a vde_switch which is bound to a tap device. Virtual machines can then be bridged to this tap device running on the host. Listing 1 shows this configuration. **/tmp/vs0** is the switch data socket used here and **/tmp/vs0m** is the management socket for the switch. **tap0** provides a forward interface for the switch but as for this simple topology we are containing the whole virtual network with this switch, this interface isn’t used. **tap1** is the interface with virtual machines are bridged to.
VDE can actually run natively with VirtualBox just using the sockets created by vde_switch [11], however using a tap interface allows for cross hypervisor networking, expansion across different physical machines by combining virtual and physical networking, and for the Network Intrusion Detection/Prevention System to be able to be run from the host.

4.2.3 VirtualBox Web Service

In addition to a COM/XPCOM API, VirtualBox provides a SOAP web service (**vboxweb-srv**) to allow developers to interface with the hypervisor. This allows for hypervisor control without use of the command-line interface. The SDK includes WSDL files defining the web service bindings along with a service wrapper library for PHP (amongst other languages) [12]. This acts as an interface between the Virtual Simulator and the Web Interface allowing the web app control of the hypervisor. The details of this are contained in Section 4.3.3
Listing 7 shows the configuration to bind the vboxweb-srv to localhost:18083 running under the 'vbox' user. Only virtual machines created under this user are accessible, it does not currently seem very feasible to manage virtual machines across multiple users.

4.2.4 Remote Desktops

To avoid users of CLARE needing to install security testing software on their own machines (placing requirements on the machine used), a VDI (Virtual Desktop Infrastructure) approach is taken. Though instead of the user's virtual desktop being pushed down onto their machine, it is instead hosted within the Virtual Simulator. Users access their personal virtual machine through a remote desktop connection and interact with the virtual environment through this. Whilst this adds to the resource requirements of the Virtual Simulator, it provides a more accessible and contained environment.
VirtualBox provides support for remote display through VRDP, which is the VirtualBox implementation of Microsoft's Remote Desktop Protocol. This is compatible with the majority of RDP viewers (**mstsc** on Windows, **rdesktop** or **krdc** on Linux for example) [13]. There are performance benefits to having the remote desktop provided by the hypervisor instead of by the guest operating system, and it allows for sessions to be managed through the hypervisor (and hence Web Interface as discussed in Section 4.3.3).

4.3 Web Interface

The Web Interface manages the user experience and interaction with the rest of the environment by both providing control over the Virtual Simulator and managing the provision of challenges and competitions that run on CLARE. This allows for delegation of control over individual personal virtual machines to each user through the website without them needing access to the server. Source code for the VirtualBox Interface Library (CLARE/VBOX-LIB) and the Web App (CLARE/WWW) is available on Github (`https://github.com/jbhardwaj/clare`), licensed under the GNU General Public License Version 3 [14].

4.3.1 Web Server

Apache is the web server of choice here, providing a stable platform to host web services. Other servers such as **lighttpd** are more suited to a very low resource environment. It was decided that the popular CakePHP[15] web framework would be used to add extra resilience to the Web Interface. This allows for fast development of a MVC class infrastructure without the need to write the whole Web App bespoke.

4.3.2 Database

MySQL was the database of choice for ease of compatibility with CakePHP and other services wishing to use a database backend (for example Snort - Section 4.4.1). The **phpMyAdmin** package is also used running localhost only for ease of configuration.

4.3.3 VirtualBox Interface Library

A custom PHP class library has been written to sit over the top of the PHP bindings provided by the VirtualBox SDK for use with vboxweb-srv. The benefit of this middle layer is that it makes the codebase of the Web App simpler and allows for multiple hypervisors (whether they are VirtualBox or otherwise) to be managed through this custom class library. Therefore, the Web App does not need to be concerned with hypervisor specific implementation, nor with how many hypervisors the Virtual Simulator uses and how to access them.

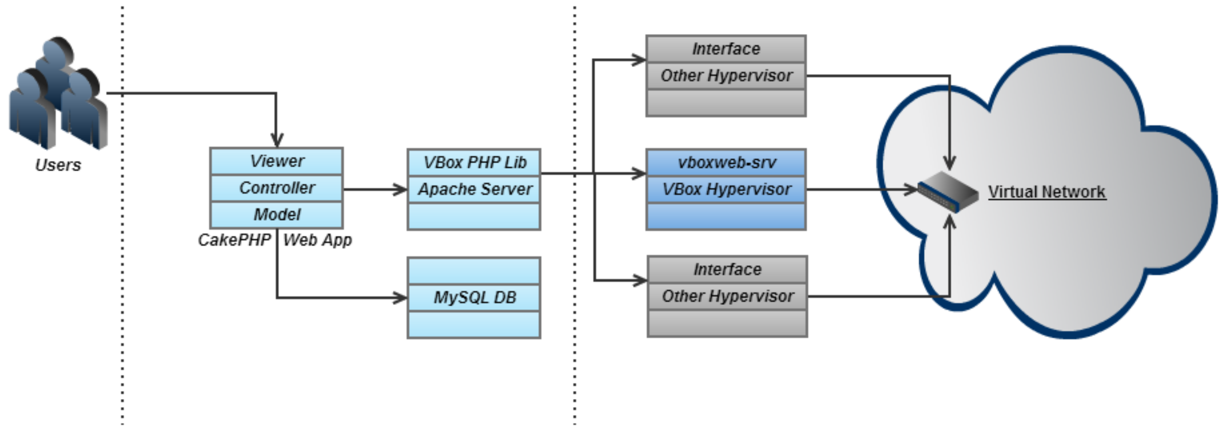

Figure 2: Web Interface and Virtual Simulator Interaction

4.3.4 Web App

The Web App allows for student users to access challenges, submit challenge reports and control their own client virtual machine. Additionally, it gives administrator users further access to control the whole virtual network and machines running on it and give feedback for submitted challenge reports.

4.4 Network Intrusion Detection/Prevention System

The system used for managing and monitoring containment of CLARE and providing additional logs for providing user feedback from security challenges can be either a very straight forward IDS running on the Host Gateway (for instance just using Snort) or as sophisticated as a multi-sensor distributed setup. When this component is implemented as in configuration 2/3, Security Onion[16] provides one such 'turn-key' (out of the box) solution that can be used to accomplish this.

4.4.1 Snort IDS

Snort is a very popular IDS which can quite quickly be configured on a Ubuntu host to run with a MySQL database using the **snort-mysql** package [17]. The configuration in Listing 8 is for Snort running on the Host Gateway listening on both the physical and virtual interface.

4.4.2 Web Frontend

It is optional whether the web frontend to this component of CLARE is run as a separate service hosted by the Web Interface, or whether a web frontend is used at all. Snorby[18] is a particularly nice web app that works with Snort (as well as Suricata and Sagan) running on Ruby on Rails, but AcidBase[19] is the more traditional lightweight choice. On the prototype this component is provided natively by the Host Gateway so AcidBase was chosen to avoid needing Ruby on Rails and the other prerequisites required for Snorby.

4.5 Backup & Update Management Server

Segregation between CLARE and any external network is a key requirement of the environment. The approach taken to ensure this is to run an isolated network with an 'air gap' to the external network and the Internet. This creates complications in management, primarily in updating software packages and backing up the live database. A separate physical system can act as the Backup & Update Management Server by manually switching its physical network connection to/from CLARE and an external network. As long as this server is never connected to both networks, containment can be reasonably maintained. This server would contain a mirror of the package repository being used and a backup of the live database. It is not necessary, however, to have a separate physical system for this. Sections 4.5.1 and 4.5.2 detail how containment can be maintained whilst still being able to provide updates and backups using just a USB memory stick.

4.5.1 System Updates

Offline package management is greatly simplified in Debian with the **apt-offline** tool described in [20]. This allows for the APT database on the host to be updated and packages (and their dependencies) to be installed and upgraded using the three step procedure outlined. Updates to the Web App and VirtualBox Interface Library can be performed by pulling from a offline copy of the git repository as described in [21].

4.5.2 Database Backups

Whilst there are many backup and sychronisation tools which would be suitable for updating a backup database dump to a memory stick, probably the most powerful and versatile is **rsync**. This combined with a cron job to generate a database dump locally is probably the most straight forward approach to take, for example Listing 9.

4.5.3 Health Monitoring

From a system administrator point of view, an isolated network is harder to maintain as it requires you to connect to it to be able to view logs/alerts etc. Additionally, the Network Intrusion Detection/Prevention System is not as useful without the ability to provide real-time alerts to an administrator. One method to provide limited alert functionality is to implement a GSM gateway using a cheap mobile handset to allow for SMS messages to be sent out to an administrator (either via their phone or another GSM-enabled system). Gammu[22] is a command line tool built on libGammu that allows for interfacing with handsets in this manner. This functionality was not implemented on the prototype of CLARE.

5 Further Development

Based on the design of CLARE detailed and the prototype implementation described in this report, further work is taking place to develop and test this infrastructure with different configurations and levels of available resources. The focus so far has been to demonstrate how lightweight and low resource such an environment can be when configuration 1 is implementation. The next stage will be to evaluate how well this design scales to efficiency use the available resources in a configuration 2/3 implementation. The current development version of the source code for the Web App and VirtualBox Interface Library will be updated on the public repository mentioned in Section 4.3. Moreover, it is envisioned that sample virtual machines and custom installation disks for each component will be published to allow for the ease of replications of CLARE. Additional pedagogic research will evaluate how effectively this environment can be used in designing security challenges and exercises for undergraduate computer science students.

APPENDIX - Configuration Files

Networking

Listing 1: /etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        gateway 192.168.1.1

auto tap1
iface tap1 inet static
        address 192.168.3.254
        netmask 255.255.255.0
        gateway 192.168.3.254
        pre-up /usr/bin/vde_switch -F -d -t tap0 -t tap1 -s /tmp/vs0 -M /tmp/vs0m
           --pidfile /var/run/vs0_vde.pid
        post-down kill -s HUP `cat /var/run/vs0_vde.pid`
```

DHCP

Listing 2: /etc/dhcp/dhcpd.conf

```
default-lease-time 3600;
max-lease-time 7200;

ddns-update-style interim;

subnet 192.168.1.0 netmask 255.255.255.0 {
 interface eth0;
 range 192.168.1.3 192.168.1.100;
 option routers 192.168.1.1;
 option domain-name-servers 192.168.1.1;
 option domain-name "dcs.warwick.ac.uk";
}

subnet 192.168.3.0 netmask 255.255.255.0 {
 interface tap1;
 range 192.168.3.1 192.168.3.253;
 option routers 192.168.3.254;
 option domain-name-servers 192.168.3.254;
}

host clare.dcs.warwick.ac.uk {
 hardware ethernet 00:22:4d:52:76:22;
 fixed-address 192.168.1.1;
}

host ap {
 hardware ethernet 00:13:80:14:F0:28;
 fixed-address 192.168.1.254;
}
```

```
host ap2 {
 hardware ethernet 00:13:80:07:17:79;
 fixed-address 192.168.1.253;
}
```

DNS

Listing 3: /etc/bind/named.conf.local

```
zone "clare.dcs.warwick.ac.uk" {
        type master;
        file "/etc/bind/db.clare";
};

zone "." {
        type master;
        file "/etc/bind/db.blackhole";
};
```

Listing 4: /etc/bind/db.clare

```
$TTL    604800
@       IN      SOA     clare.dcs.warwick.ac.uk. admin.clare.dcs.warwick.ac.uk.(
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@       IN      NS      .
@       IN      A       192.168.1.1
```

Listing 5: /etc/bind/db.blackhole

```
$TTL    604800
@       IN      SOA     . admin.clare.dcs.warwick.ac.uk. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

        IN      NS      .
.       IN      A       192.168.1.1
*.      IN      A       192.168.1.1
```

Firewall

Listing 6: /lib/ufw/user.rules

```
[lines 1-17 omitted]
#### RULES ###

### tuple ### allow tcp 22 0.0.0.0/0 any 0.0.0.0/0 OpenSSH - in_eth0 -A ufw-user-
    input -i eth0 -p tcp --dport 22 -j ACCEPT -m comment --comment 'dapp_OpenSSH'

### tuple ### allow tcp 80 0.0.0.0/0 any 0.0.0.0/0 Apache - in_eth0 -A ufw-user-
    input -i eth0 -p tcp --dport 80 -j ACCEPT -m comment --comment 'dapp_Apache'
```

```
### tuple ### allow any 53 0.0.0.0/0 any 0.0.0.0/0 Bind9 - in_eth0 -A ufw-user-
    input -i eth0 -p tcp --dport 53 -j ACCEPT -m comment --comment 'dapp_Bind9' -A
    ufw-user-input -i eth0 -p udp --dport 53 -j ACCEPT -m comment --comment '
    dapp_Bind9'

### tuple ### allow any 3389:3489 0.0.0.0/0 any 0.0.0.0/0 in -A ufw-user-input -p
    tcp --dport 3389:3489 -j ACCEPT -A ufw-user-input -p udp --dport 3389:3489 -j
    ACCEPT -m comment --comment 'dapp_VRDP'

### tuple ### allow tcp 10000 127.0.0.1 any 127.0.0.1 in -A ufw-user-input -p tcp -
    d 127.0.0.1 --dport 10000 -s 127.0.0.1 -j ACCEPT -m comment --comment '
    dapp_Webmin'

### END RULES ###
[lines 38-56 omitted]
```

VirtualBox

Listing 7: /etc/default/virtualbox

```
VBOXWEB_USER=vbox
VBOXWEB_HOST=localhost
VBOXWEB_PORT=18083
VBOXWEB_LOG=/var/log/vboxweb.log
```

Snort IDS

Listing 8: /etc/snort/snort.conf

```
var HOME_NET 192.168.1.0/24, 192.168.3.0/24
var INTERFACE eth0,tap1
var EXTERNAL_NET any
```

Backup

Listing 9: /etc/crontab

```
15 2 * * * root mysqldump -u root -p PASSWORD --all-databases | gzip > /mnt/backup/
    database_`data '+%m-%d-%Y'`.sql.gz
```

References

[1] Shiva Azadegan, M. Lavine, Michael O'Leary, Alexander L. Wijesinha, and Marius Zimand. An undergraduate track in computer security. In Vassilios Dagdilelis, Maya Satratzemi, David Finkel, Roger D. Boyle, and Georgios Evangelidis, editors, *ITiCSE*, pages 207–210. ACM, 2003.

[2] John M. D. Hill, Curtis A. Carver Jr., Jeffrey W. Humphries, and Udo W. Pooch. Using an isolated network laboratory to teach advanced networks and security. In Henry MacKay Walker, Renée A. McCauley, Judith L. Gersting, and Ingrid Russell, editors, *SIGCSE*, pages 36–40. ACM, 2001.

[3] J.H. Schafer, D.J. Ragsdale, J.R. Surdu, and Jr. C.A. Carver. The IWAR Range: A Laboratory for Undergraduate Information Assurance Education. *The Journal of Computing in Small Colleges*, 1964:223–232, 2001.

[4] C. Willems and C. Meinel. Online assessment for hands-on cyber security training in a virtual lab. In *Global Engineering Education Conference (EDUCON), 2012 IEEE*, pages 1 –10, april 2012.

[5] William I. Bullers Jr., Stephen D. Burd, and Alessandro F. Seazzu. Virtual machines - an idea whose time has returned: application to network, security, and database courses. In Doug Baldwin, Paul T. Tymann, Susan M. Haller, and Ingrid Russell, editors, *SIGCSE*, pages 102–106. ACM, 2006.

[6] Slitaz. `http://www.slitaz.org`. Accessed on 1st October 2012.

[7] Webmin. `http://www.webmin.com`. Accessed on 1st October 2012.

[8] VirtualBox and open source. `https://www.virtualbox.org/wiki/Editions`. Accessed on 27th September 2012.

[9] VirtualBox Personal Use and Evaluation License (PUEL). `https://www.virtualbox.org/wiki/VirtualBox_PUEL`. Accessed on 27th September 2012.

[10] VDE - Virtual Distributed Ethernet. `http://vde.sourceforge.net`. Accessed on 1st October 2012.

[11] VDE native support for VirtualBox. `http://wiki.virtualsquare.org/wiki/index.php/VDE_native_support_for_VirtualBox`. Accessed on 28th September 2012.

[12] Oracle VM VirtualBox Programming Guide and Reference. `http://download.virtualbox.org/virtualbox/vboxsdkdownload.html`. Version 3.2.12, Accessed on 27th September 2012.

[13] VirtualBox Manual: Chapter 7. Remote virtual machines. `http://www.virtualbox.org/manual/ch07.html`. Accessed on 28th September 2012.

[14] GNU General Public Licence Version 3. `http://www.gnu.org/licenses/gpl.html`. Accessed on 22nd October 2012.

[15] CakePHP. `http://cakephp.org`. Accessed on 1st October 2012.

[16] security-onion. `http://code.google.com/p/security-onion`. Accessed on 1st October 2012.

[17] SnortIDS. `https://help.ubuntu.com/community/SnortIDS`. Accessed on 28th September 2012.

[18] Snorby. `http://snorby.org`. Accessed on 1st October 2012.

[19] Basic Analysis and Security Engine (BASE) Project. `http://base.secureideas.net`. Accessed on 1st October 2012.

[20] Offline Package Management for APT. `http://www.debian-administration.org/article/Offline_Package_Management_for_APT`. Accessed on 28th September 2012.

[21] Synchronizing Git repositories without a server. `http://blog.costan.us/2009/02/synchronizing-git-repositories-without.html`. Accessed on 22 October 2012.

[22] GWammu. `http://wammu.eu/gammu`. Accessed on 1st October 2012.