CDEST – A Cyber Defense Exercise Scoring Tool

John Franco (franco@ucmail.uc.edu)

Dept. Electrical Engineering and Computer Science
University of Cincinnati
Cincinnati, OH 45221-0030

November, 2023 – version 1.1

Table of Contents

Overview

CDEST is a team competition that is intended to enhance the Cyber Operations (CO) community by reducing time and cost of apprenticeship training and by improving the motivation and quality of the CO workforce.

Critical to improving quality and reducing time and cost of apprenticeship training is ensuring a greater depth of knowledge and utility of CO skills in personnel entering the workforce. CO specialists 1) protect data, networks, and net-centric capabilities; and 2) neutralize the digital capabilities of adversaries including their ability to communicate, initiate attacks, and control infrastructure. Critical CO skills therefore include low-level programming, operating system weaknesses and vulnerabilities, network protocols, encryption, authentication, integrity protection, reverse engineering, and forensic analysis of traffic and data. The I-Wars competition can exercise some or all of those skills.

CDEST supports a variety of Cyber Defense Exercises and consists of two major components: a modified operating system (OS) and scoring software (Scorer). The OS is distributed by a Competition Administrator (CA) to Teams some number of days, determined by the CA, before a CDEST competition commences. Teams examine the OS for configuration errors, protocol weaknesses, and any vulnerabilities the OS may have and attempt to mitigate or fix such problems. Meanwhile, the CA configures a competition – see Appendix A for details. At competition start the Teams put the OSes online to provide a collection of internet services that are specified by the CA in advance. Also at competition start the CA initiates the Scorer: at 1 minute intervals the Scorer checks to see if the services of a Team's OS are up. The Scorer adds 1 point to a team's cumulative score for each service that is deemed working when a check is made. A real-time scoreboard is maintained during the competition. The scoreboard is publicly accessible by default although it can be made private by the CA. At competition end the team with the highest cumulative score wins. More importantly, an assessment is made as to how well a Team was able to defend its OS.

There are several ways CDEST can be used. The supplied OS may be replaced by another at the discretion of the CA. The services checked may be changed by modifying the CheckServices.java file and recompiling the Scoring software. The time between checks may be changed in the same way. Teams may be allowed to attack the OSes of other Teams in a variety of ways as well as defend their OSes from attack. The CA may require Teams to defend only and enlist other Teams, having seen the OS in advance, to attack the OSes. Finally, the CA may ask Teams to choose their own OS and require that a collection of services must be running on particular ports.

Not included with CDEST are the many publicly available tools for analyzing, modifying, and controlling network traffic, for example firewalls. Teams should use those tools to anticipate and mitigate attacks. The CA will use those tools to analyze the performance of Teams after the competition is ended.

CDEST is best used at the end of a course on cyber defense. It helps if Team members have also taken a course in vulnerabilities analysis.

CDEST Details

CDEST has two components: the modified OS and the Scorer. Information about the supplied OS is given below. The CA is free to choose another one. The scorer not only checks OS services during the competition but also provides configuration management and a real-time scoreboard. Details of these functions are given below.

Compromised Operating System

Teams are given an operating system with many vulnerabilities including misconfiguration; memory corruption such as buffer, stack, and heap overflow; various race conditions arising from time-of-check, null pointer dereference, and others; privilege escalation problems; integer overflow, underflow, truncation; pointer arithmetic problems; pointer problems of various kinds such as double free, use-after-free, and out-of-bounds references; type checking and conversions among others.

Configuration Management
The CA has the following tasks to perform to prepare for a competition:

1. create a list of competitors and files containing data that enables competitors to join a competition and the Scorer to know where in the competition network they are located (their competition IP address).
2. set competition parameters such as scoreboard location, times of competition start and end, logging verbosity, whether to allow an account to be created remotely, whether to recover a database if the Scorer is restarted
3. establish a private network over which the competition is to be run
4. distribute all credentials and competition information such as usernames, IP addresses, etc. to the competitors so that they may have their OSes be seen by the Scorer during the competition.

The next section provides details that are needed by the CA to configure a competition successfully.

Getting and Installing the CDEST Package

There are two versions – one for Linux and one for Windows. However, since the package software really needs to be run in a VM, the Linux version is suitable for Linux, Windows, or Mac hosts. To get the Linux version from a Linux host do this from a command shell:

    wget http://gauss.ececs.uc.edu/cryptol-course.7z

Then this:

    7z x cryptol-course.7z

The result is a directory `ElemCourse` with top level files `intro.pdf`, `run`, `run-gui`, `run-txt`, `README.md`, `index.txt`, and `game.jar`. Get the Windows version by pointing to `gauss.ececs.uc.edu/saw-course.zip` from Edge or another browser, downloading, and choosing 'Extract All' when opening the downloaded file. The result is a directory `SawCourse` with top level files `intro.pdf`, `run.bat`, `run-gui.bat`, `run-txt.bat`, `README.md`, `index.txt`, and `game.jar`. Run `./run-gui` to run the CDEST Control Panel GUI in Linux from the command line. Run `./run-txt` to open the Control Panel as a text-based application. From a Windows File Explorer double click on the `run-gui.bat` or `run-txt.bat` icon. All scripts run `java -jar cdx-contest.jar`, a Java archive, from a version of Java that is supplied in directory `jdk-lin` (Linux) or `jdk-win` (Windows). See the scripts to find out where the `jar` files are located.

The CDX Control Panel

A CA prepares for a competition using the CDX Control Panel which configures a competition and creates and distributes files needed by competitors to join a competition network, and possibly the host of an OpenVPN server where the competition network is created (for OpenVPN details see Appendix C). The CDX Control Panel is started by running the 'run-gui' script in Linux or 'run-gui.bat' script in Windows. The CDX Control Panel GUI is shown at startup in Figure 1. The Panel window is divided into four sections. The large blank area on the right displays help, the results of configuration, and the status of all competitors up to the current point in time. The area on the left contains buttons for activating commands and fields for setting parameters for commands. Some of the commands, such as 'Add Player' are used during configuration and some commands such as 'HTTP Query' are used by the CA while the competition is in progress. The left four buttons at the top are used to 'Start', 'Stop', 'Suspend', and 'Resume' a competition. The right three buttons at the top are used to 'Load' the competitor database file (players.db is the default name) into memory, 'List' the current competitor status (from memory) and 'Save' the competitor status from memory to the competitor database file. A listing of competitors and status is printed in the right text area when 'List' is clicked such as is shown in Figure 2 where the 0s are scores obtained so far by each competitor (this shot was taken before the start of the competition so all numbers are 0). The four buttons at the bottom of the GUI on the right are 'Help' which displays a description of the commands invoked from clicking buttons in the Panel; 'Configure' which brings up the Configurator GUI, outlined in the next section; 'Reset' which resets the status of all competitors to what they are at the start of the competition; and 'Exit' which quits the CDX Control Panel. On the left is the 'Time to start:' field showing a competition has been set to start in 0 days, 7 hours, 18 minutes, and 22 seconds. This field changes to 'Time to end:' during a competition.

The Configurator

Click the 'Configure' button in the CDX Control Panel to bring up the Configurator. The Configurator allows the CA to customize settings of the parameters associated with the competition. Many settings are selected by drop

down menus, however the scoreboard URL and scoreboard title are intended to be edited in. An example Configurator window is shown in Figure 3. Descriptions of the parameters can be found in Appendix G as well as by clicking 'Show Help' and hovering the mouse over the menu associated with a parameter or button.

CDX Control Panel

| Start | Suspend | Resume | Stop | Load | Save | List |

Add | | Player | | IP address | | email

Notice: no player database exists in memory
  A player database should be created before contest is started,
    either automatically via the timer or manually via START.
  To create a player database in memory invoke ADD commands or
    load a database from the 'players.db' file, if it exists,
    using the LOAD command
  To save the player database to file 'players.db' invoke the SAVE command
  To list players that are in the database invoke the LIST command
----

Delete | | Player

Check | | Player | | OpenVPN server location

Interrupt | | Player

Bump | | Player | | Score

WP | | Player

HTTP | | Player | | Payload

| | Phrase | | Port

Hit | | Player

Time to start: 0:7:18:22 | Help | Configure | Reset | Exit

Figure 1: The CDX Control Panel at startup

CDX Control Panel

| Start | Suspend | Resume | Stop | Load | Save | List |

Add | franco | 10.8.0.200 | franco@localhost
Player | IP address | email

Player	IP address	Score
NG Fellows B	10.8.0.122	0
NG Fellows C	10.8.0.123	0
No Name Yet A	10.8.0.127	0
No Name Yet B	10.8.0.128	0
No Name Yet C	10.8.0.129	0
Sneed's Feed And Seed A	10.8.0.115	0
Sneed's Feed And Seed B	10.8.0.116	0
Sneed's Feed And Seed C	10.8.0.117	0
Team 3 A	10.8.0.124	0
Team 3 B	10.8.0.125	0
Team 3 C	10.8.0.126	0
The Boys A	10.8.0.112	0
The Boys B	10.8.0.113	0
The Boys C	10.8.0.114	0
The Pink Panther Squad A	10.8.0.100	0
The Pink Panther Squad B	10.8.0.101	0
The Pink Panther Squad C	10.8.0.102	0
Two Musketeers A	10.8.0.130	0
Two Musketeers B	10.8.0.131	0
Two Musketeers C	10.8.0.132	0
No Attack Pls A	10.8.0.133	0
No Attack Pls B	10.8.0.134	0
No Attack Pls C	10.8.0.135	0
franco	10.8.0.200	0

Delete | | Player

Check | | Player | | OpenVPN server location

Interrupt | | Player

Bump | | Player | | Score

WP | | Player

HTTP | | Player | | Payload

| | Phrase | | Port

Hit | | Player

Time to start: 0:7:17:44 | Help | Configure | Reset | Exit

Figure 2: The competitor 'franco' has been added to the competitor database and the database has been loaded and listed

The Configurator GUI is partitioned into sections, each of which represents some functionality that is needed for the competition to run smoothly. Each section is boxed with a title above the box on the left. The 'Web Server' section serves to configure the Scoreboard so it can be seen by all participants and refreshed properly. The 'Web server directory' holds the place from which files are served to the web; the 'Scoreboard title' will appear at

3

the top of the scoreboard; the 'Web server URL' is edited into the scoreboard header to make sure automatic refreshes show a correct update of the Scoreboard page. The section labeled 'Database' has a menu 'Player DB state' which allows anyone to establish their own Client account remotely if 'Dynamic' is chosen by the CA or not if 'Static' is chosen.   In the 'Log' section the verbosity may be selected (the name of the log file is fixed at 'cdx.log'). Contest dates and times may be set by choosing month, day, year, hour, and minute from the 'Start:' and 'End:' menus, then choosing the time zone from the 'timezone' menu, finally clicking the 'Set Dates' button to record the dates and times. This information is used by the CA and is distributed to the competitors so everyone knows when the competition begins and when it ends. The recorded times automatically account for Daylight Savings Time.

Figure 3: *The Configurator*

The remaining three sections of the Configurator GUI are used to prepare and, optionally send, files to the OpenVPN host, the Scorer host, and the competitors[1]. OpenVPN keys and certificates are made in the 'OpenVPN Key & Certificate Maker' section. Keys and certificates serve as credentials for admission to the competition VPN if one is used. A description of the competition location and contact must be filled in. Use the '#keys' menu to select the number of keys and certificates to make. Click the 'Make Keys' button to make the keys. This does not have to be done for every competition – it is up to the CA to decide whether existing keys have become stale and should be replaced. Most GUI features are disabled when keys and certificates are being made. Making keys takes a very long time. Key making can be interrupted by clicking the 'Stop Making Keys' button. Made keys are always saved. The 'Save' button saves only the competition location and contact information to file `game-id.txt` In directory `cdest/contest`.
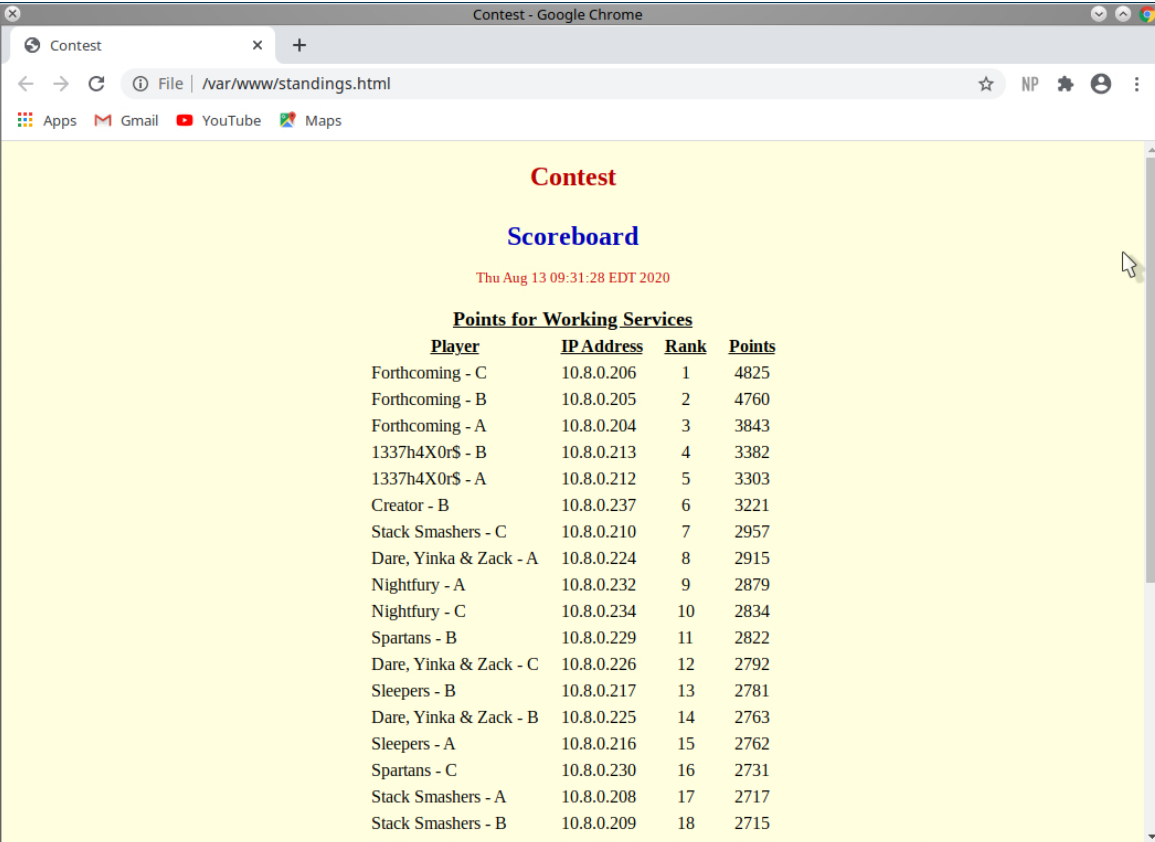
The 'Prepare Files' section has a button 'Prepare without OpenVPN keys' for saving all files to be distributed when OpenVPN is not being used and a button 'Prepare with OpenVPN Keys' for saving all files for distrbution including those needed by a competitor to connect to the VPN and those needed by the OpenVPN server to establish the VPN. Either way, the result is a 'Contestants' directory with subdirectories, named for each competitor and containing a competitor's files, a 'Parameters' directory containing a file `Parameters.txt` for the Scorer host, and, if OpenVPN is used, a subdirectory of `contest` named `server` containing all files needed by the OpenVPN server to establish a private network.

---

1   The Windows version of the Configurator does not support management of an OpenVPN network.

The 'Controls' section contains a button 'Send Files and Quit' which archives the competitor directories, sends them to competitors, and, in case OpenVPN is being used, archives the directory containing OpenVPN server files and places the result in a directory that is created called 'VPNServer' (the CA will manually send the contents of this directory to the server host). There is also a button 'Quit, Do Not Send Files' which quits the Configurator without making any additional changes. The 'Help' button enables help tips to be displayed when the mouse pointer hovers over a menu or button in the Configurator. If the 'Cancel' button is clicked, the Configurator is exited with no changes. See Appendix F for how to set parameters for different competitor skill levels. See Appendix E for all the immutable (fixed) parameters.

Scoreboard

The Scorer runs when the Competition is started – this could happen from the CDX Control Panel 'Start' button or automatically when a configured start time is reached. The Scoreboard is maintained by the Scorer. Its location is set by the CA in the Configurator in the editable parameter fields called 'Web server URL' and 'Web server directory'. The 'Web server directory' default is /home/httpd/html and the 'Web server URL' default is file:///home/httpd/html. The default path to the scoreboard file will then be /home/httpd/html/standings.html as the file that is served to the web containing the scoreboard is named standings.html. The standings.html file is updated directly in the Web server directory at 45 second intervals so permissions must be set to prevent permission denied errors. The reason the Web server URL is a parameter in the Configurator is that the value in the Web server URL field is embedded in the standings.html file and causes an automatic and periodic refresh to the scoreboard. A scoreboard example, representing the result of a short mock competition, is shown in Figure 4.
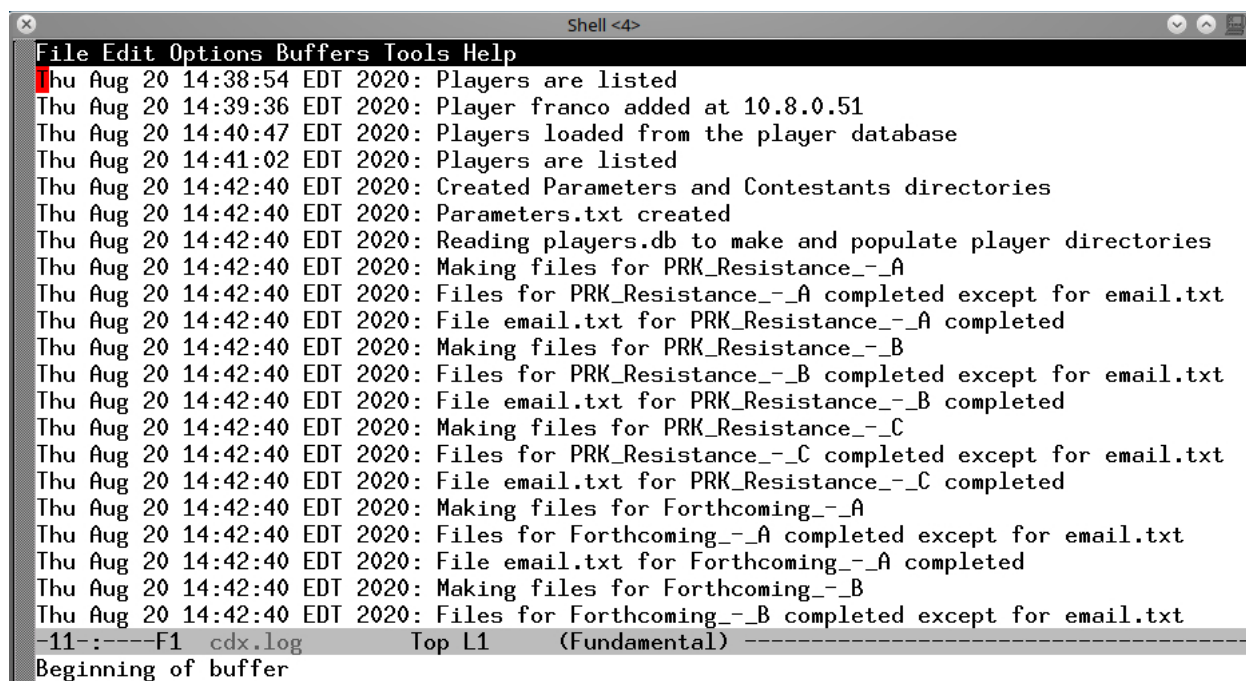

Figure 4: Example scoreboard during a competition

Log File

The Scorer generates a log file that contains all events and the times they occurred. The CA may choose to make the log file available to the participants of the competition. This will be especially useful in low level competitions where participants lack familiarity with advanced forensic tools such as wireshark.   Figure 5 shows the beginning of a log file that was generated by a Scorer. The format of the log is date and time followed by an event, one event per line. Events are categorized as a) Directory and file creation, file reading; b) Configuration events plus competition commands and services; c) Checking service results plus other commands and results during a competition; d) Only the results of checking services. Events that appear in the log are determined by the CA who chooses a number from the 'Logging Option' menu in the Configurator before the competition begins: possibilities are 1) all events are logged; 2) events in categories b-d are logged; 3) events in categories c-d are logged; 4) only events in category d are logged. If the log file is made public a link should be set from the same directory containing the scoreboard to `cdx.log` so it may be served to the web.

```
                                      Shell <4>
File Edit Options Buffers Tools Help
Thu Aug 20 14:38:54 EDT 2020: Players are listed
Thu Aug 20 14:39:36 EDT 2020: Player franco added at 10.8.0.51
Thu Aug 20 14:40:47 EDT 2020: Players loaded from the player database
Thu Aug 20 14:41:02 EDT 2020: Players are listed
Thu Aug 20 14:42:40 EDT 2020: Created Parameters and Contestants directories
Thu Aug 20 14:42:40 EDT 2020: Parameters.txt created
Thu Aug 20 14:42:40 EDT 2020: Reading players.db to make and populate player directories
Thu Aug 20 14:42:40 EDT 2020: Making files for PRK_Resistance_-_A
Thu Aug 20 14:42:40 EDT 2020: Files for PRK_Resistance_-_A completed except for email.txt
Thu Aug 20 14:42:40 EDT 2020: File email.txt for PRK_Resistance_-_A completed
Thu Aug 20 14:42:40 EDT 2020: Making files for PRK_Resistance_-_B
Thu Aug 20 14:42:40 EDT 2020: Files for PRK_Resistance_-_B completed except for email.txt
Thu Aug 20 14:42:40 EDT 2020: File email.txt for PRK_Resistance_-_B completed
Thu Aug 20 14:42:40 EDT 2020: Making files for PRK_Resistance_-_C
Thu Aug 20 14:42:40 EDT 2020: Files for PRK_Resistance_-_C completed except for email.txt
Thu Aug 20 14:42:40 EDT 2020: File email.txt for PRK_Resistance_-_C completed
Thu Aug 20 14:42:40 EDT 2020: Making files for Forthcoming_-_A
Thu Aug 20 14:42:40 EDT 2020: Files for Forthcoming_-_A completed except for email.txt
Thu Aug 20 14:42:40 EDT 2020: File email.txt for Forthcoming_-_A completed
Thu Aug 20 14:42:40 EDT 2020: Making files for Forthcoming_-_B
Thu Aug 20 14:42:40 EDT 2020: Files for Forthcoming_-_B completed except for email.txt
-11-:----F1  cdx.log         Top L1    (Fundamental) -----------------------------------
Beginning of buffer
```

Figure 5: Sample beginning of a log before the beginning of a competition. Entries show time and date as well as an event

Proxy Server

If OpenVPN is used to establish a competition network, the OpenVPN server is behind an organization's firewall, and some competitors intend to enter the competition from outside the organization's perimeter then it is likely that a proxy server is needed to channel outside connections to the OpenVPN server. If an `ssh` server is operating behind the firewall then this can be done as follows from the competitor's Linux VM:

```
ssh -N -f -T -D 8080 visitor@helios.ececs.uc.edu
```

where it is assumed a ssh server is running on `helios.ececs.uc.edu` and an account named `visitor` has been established with reduced privileges to allow a connection to be established between the server and a competitor's OS. See Appendix M for instructions to set up such a `visitor` account. If a proxy server is used by a competitor then that competitor must modify `client.conf` by uncommenting the line:
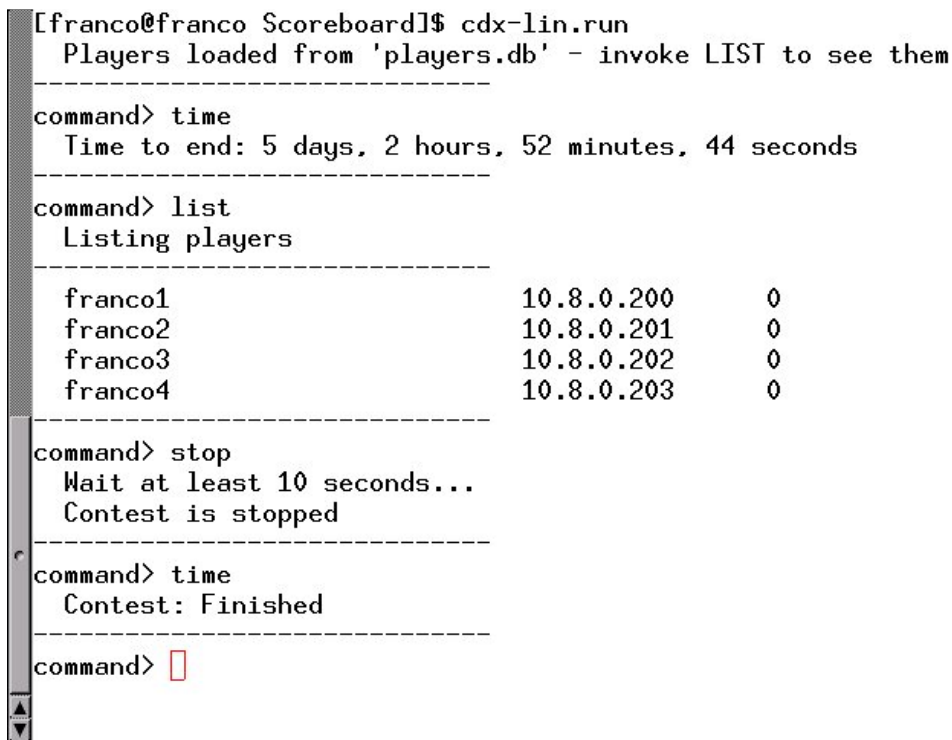
```
;socks-proxy 127.0.0.1 8080
```

This is done by removing the ';' from the beginning of the line.

Running a Competition Without the GUI

Although a competition can be run from the CDX Control Panel discussed above, it may be much more convenient to run a competition from an application that is not graphical. This is because the CA likely needs to issue commands during a competition and because it is further likely that the Scorer will be run remotely with respect to the CA. Thus, the CA will usually start the Scorer as a remote process, which it detaches, and later reattaches to the Scorer, as needed, to issue commands. In Linux this is facilitated by an application called 'screen'[2]. In Windows 10 'powershell' has commands to run detached remote processes and 'screen' can be used if the Scorer is run in the Ubuntu shell. But a detached process cannot be run graphically. For this reason there is a Scorer application that runs only with command line inputs. This Scorer does not have the CDX Control Panel – the buttons and argument text fields are replaced by commands with arguments. This Scorer also does not include the Configurator.

Use of the command line Control Panel is detailed in Appendix L . An example screenshot of the Control Panel, at startup, is shown in Figure 6. The lines beginning 'command>' show the command prompt. The first 'time' command, invoked by the CA, shows 5 days, 2 hours, 52 minutes, 44 seconds to the end of the competition. Since the competition start time was before the execution of the Scorer, the competitor database was automatically loaded when the Control Panel was run. The scores, shown as a result of invoking the 'List' command, are all 0 because the `Parameters.txt` file indicated that all scores should be reset upon initial loading. Invoking the command 'Stop' causes the competition to end immediately, stopping the Scorer. The second invocation of 'time' shows the competition is finished.

```
[franco@franco Scoreboard]$ cdx-lin.run
    Players loaded from 'players.db' - invoke LIST to see them
    -----------------------------
command> time
    Time to end: 5 days, 2 hours, 52 minutes, 44 seconds
    -----------------------------
command> list
    Listing players
    -----------------------------
    franco1                      10.8.0.200      0
    franco2                      10.8.0.201      0
    franco3                      10.8.0.202      0
    franco4                      10.8.0.203      0
    -----------------------------
command> stop
    Wait at least 10 seconds...
    Contest is stopped
    -----------------------------
command> time
    Contest: Finished
    -----------------------------
command>
```

Figure 6: The command-line Scorer has been started with four competitors. and a running competition. The 'time' command shows how much time is left until the end of the competition. The 'stop' command terminates the competition immediately as the second call to 'time' shows.

Files and Directories

There are many files that need to be created, modified, and distributed by the Competition Administrator. This is made simple by commands 'Prepare with OpenVPN keys' and 'Prepare without OpenVPN keys' in the

---

2   There are other applications that do this as well or better.

Configurator (see Figure 3), which make the files, and the command 'Send Files and Quit' which distributes the files. There is also command 'Make Keys' in the Configurator which makes OpenVPN keys and certificates if OpenVPN is used. More on OpenVPN in Appendix C. Command 'Make Keys' may be skipped if the competition being prepared has been preceded by another competition with which keys and certificates can be shared and keys and certificates have already been created for the preceding competition. In addition to the files that are made for distribution there are a number of other files that are permanently part of the Scorer package and exist to run the Scorer and create files for distribution. Below, the files and their locations are described. The contents of many of these files are in Appendix H, Appendix I, Appendix J, and Appendix K.

Files that are part of the Scorer package as distributed

Directory	File(s)	Description
`contest/bin`	`tar` `tar.exe`	Used to archive all files in a competitor's directory as a single file which is sent to a competitor. Archived files exist in directory '`Contestants`'. This executable is statically compiled so runs independently of system libraries. `tar` is the Linux command and `tar.exe` is the Windows version.
	`openssl`	Used to create keys and certificates. This is called by the `easyrsa` bash script in directory `contest`. This is an ELF executable that is statically compiled. The Windows version of this software does not use openssl because there is no provision for using OpenVPN in Windows. However, the Linux version will work in the Ubuntu Shell in Windows along with the XMing X server for Windows. Also a Linux VM in VMWare or Virtualbox may be used to run the Linux version of the software if OpenVPN is intended to be used.
	`mutt`	Statically compiled mailer. Called by bash script `mailit`. The Windows version of this software does not use a mailer – so mailing must be done manually, one competitor at a time. Using the Linux version in Ubuntu Shell or a VM in a hypervisor allows the possibility of mailing in Windows but if the host is forbidden to email then all files (now archived as `tar` files) need to be sent manually
`contest/client`	`run.client`	Calls `run.vpn` to join an OpenVPN network. A copy is placed in each competitor's directory if OpenVPN is used to establish a competition network. Available in Linux only.
	`stop.client`	Uses `vpn.pid` which was created by `run.vpn` to stop the running `openvpn` process that has made a connection to an OpenVPN network. A copy is placed in each competitor's directory if OpenVPN is used to establish a competition network. Available in Linux only.
	`run.vpn`	Makes a connection to an existing OpenVPN network using `/sbin/openvpn` on the `client.conf` that is created specifically for a competitor. Saves the process number of the `openvpn` process in `vpn.pid`. A copy is placed in each competitor's directory if OpenVPN is used to establish a competition network. Only available in Linux.
	`client.conf`	A template of the file needed by `openvpn` to join an existing OpenVPN network. A customized `client.conf`

		is made for each competitor by appending a competitor's OpenVPN keys, certificates, and the location of the OpenVPN server to the template. This is accomplished in function 'appendToClientConfig' in Java class 'vpnFrame'. The custom client.conf is placed in the competitor's directory if OpenVPN is used to establish a competition network. Available only in Linux.
	JAVA.txt	Contains information about the use of Java
	README.txt	Contains general information for a competitor that may be useful when using the files from this directory
	instructions.pdf	Instructions for joining a competition if OpenVPN is used. A copy is placed in each competitor's directory and included in the tar file sent to each competitor if OpenVPN is used. Available only in Linux.
contest/keys	xxx.crt xxx.key xxx.pem xxx.req ca.crt dh2048.pem	Created by the Configurator when preparing for a competition that is run over OpenVPN. These are the credentials that get distributed to competitors and the OpenVPN server.
	openssl-easyrsa.cnf safessl-easyrsa.cnf	Configurations files for easyrsa use of ssl.
	extensions.temp	X509 extensions added to every signed cert
contest/examples	xxx.db xxx.example	Sample competitor database files and parameter files
contest/ x509-types	xxxx	Files that could be edited to add values that every certificate should have and other things that are stated in the comments of each file – used by easyrsa
contest/doc	xxxx.md	Easyrsa documentation
contest/server	server.conf.tmpl	A template of the server.conf file that is missing the name and location of server keys and the IP address of the server host. Used to construct server.conf for the OpenVPN server.
	run.server	Calls run.vpn to create an OpenVPN network. This script is intended for Linux.
	stop.server	Uses vpn.pid which was created by run.vpn to stop the running openvpn process that creates and sustains an OpenVPN network. Intended for Linux.
	run.vpn	Creates and sustains an OpenVPN network using openvpn on the server.conf that is created specifically for the OpenVPN host. Saves the process number of the openvpn process in vpn.pid. Intended for Linux.
	server.conf	The completed OpenVPN server configuration file that is created from server.conf.tmpl when the 'Prepare with OpenVPN keys' button in the Configurator is pressed.
	server.tar	Archive of OpenVPN files that is sent to the OpenVPN server if an OpenVPN network is to be used. This is created from directory contest/server when the 'Send Files and Quit' button is pressed in the Configurator.
	run.proxy	Shows an example of how to create a connection to a firewalled OpenVPN server via a socks proxy using ssh.

	`stop.proxy`	Stops the socks proxy connection
	`README.txt`	Contains general information for a competitor that may be useful when using the files from this directory
	`ipp.txt`	File mapping Common Names to IP addresses. This file is created by the Configurator when OpenVPN is used and keys are made. Its purpose is to ensure a competitor, given keys of prefix `clientXX`, XX a number, always have the same, given IP address.
`contest/server/ ccd`	`client0, client1 ... client149`	Files, one for each set of credentials created by the Configurator when making keys, that map credentials to IP addresses and provide a netmask. Used with `ipp.txt` to ensure that a competitor always has the same IP address in an OpenVPN network. This directory is copied from `contest/ccd` when the 'Prepare with OpenVPN keys' button is pressed in the Configurator.
`contest/server/ keys`	`ca.crt dh2048.pem server.key server.crt`	Credentials for the OpenVPN server. This directory is created and populated when the 'Prepare with OpenVPN keys' button is pressed in the 'Prepare Files' section of the Configurator.
`contest/server/ allkeys`	`clientXX.crt clientXX.key ca.crt dh2048.pem`	Complete sets of credentials for as many key sets as made in the Configurator in the 'OpenVPN Key & Certificate Maker' section, also credentials for the OpenVPN server. Credentials are created when the 'Prepare with OpenVPN keys' button is pressed in the 'Prepare Files' section of the Configurator.
`contest`	`easyrsa`	Script for producing keys and certificates. Relies on `bin/openssl`. Available only for Linux.
	`cmd1 - cmd9`	Bash scripts for making OpenVPN keys and certificates. Makes use of the `easyrsa` script. Invoked from within the running Java jar file, method of the '`MakeKeys`' class in directory `src-lig`. In addition, files that associate IP addresses with keys and certificates are produced and placed in directory `contest/server/ccd` and file `contest/server/ipp.txt`. See the next Section for more details on those files.
	`vars1,vars2`	Two halves of the `vars` file which are concatenated with competition information supplied in the 'OpenVPN Key & Certificate Maker' Section of the Configurator to produce the '`vars`' file that `easyrsa` uses to make certificates.
	`openssl-easyrsa.cnf`	Configuration file for `easyrsa`.
	`gpl-2.0.lic`	Software license
	`contest-setup.txt`	Body of the message that is sent to competitors along with files needed by the competitor to participate in the competition.
	`workaround`	Sets up an environment so `easyrsa` can generate a server certificate request and key. This is a kludge and probably can be eliminated by moving this command to another script such as `make-keys` where the environment must be set differently.
	`vpnKeyIds.txt`	List of names of directories that are created from

		competitor names in the competitor database file. When a button in the 'Prepare Files' section of the Configurator is pressed, these directories are created to hold all the competition files that will be sent to the competitors.
	`game-id.txt`	Identifies particulars of the location that becomes part of the generated certs when making keys. Created or modified by clicking 'Save' in the 'OpenVPN Key & Certificate Maker' section of the Configurator.
	`README.txt`	Contains general information for a Competition Administrator
`config`	`players.db` `players.db.bak`	Files containing competitor information such as competitor IP address, score, email address, etc. Created by multiple 'Add' commands in the Configurator and modified during competition or by the CA. The .bak file is a backup created when clicking 'Save' in the Control Panel and at the launch of the Scorer.
	`cdx.log` `cdx.log.old`	Log of a competition and backup produced when launching the Scorer.
	`Parameters.txt`	Contains all setting made in the Control Panel. See Page 41 for contents.
`src-lig` (Linux) `src-lin` (Linux) `src-wig` (Windows) `src-win` (Windows)	`xxxxx.java`	Java source files that compile to class files that are archived as `cdx-contest.jar` in directory `src-xxx`. Available only to developers.
	`run`	Script that invokes java on `cdx-contest.jar` in the Linux source directories.
	`run.bat`	Script that invokes java on `cdx-contest.jar` in the Windows source directories.
	`cdx-contest.jar`	Executable java archive of the competition software
	`compile.bat`	Script that compiles the Java source files and archives the resulting class files as a Jar file. For Windows. Available only to developers.
	`Makefile`	Instructions used to compile the Java source files and archive the resulting class files as a Jar file. For Linux. Execute 'make' in one of the Linux source directories to do this. Developers only.
	`nmap`	Files to support `nmap` in Windows. In the Windows directories only.
	`README.txt`	Explanation of compiling code, running the Configurator assuming OpenVPN is used, what competitors should do with the tar files they receive.

`doc`	`cdest_manual.pdf`
	`description.pdf`
	`scoreboard.pdf`
		Documentation in the form of a manual and in the form of a short description of the software.

`HowToJoinVPN`	`instructions.pdf`	File to be distributed to competitors to tell them what to do with the tar file they receive from the CA.
	`what-to-do-OCR.pdf`	Instructions for operating in a Cyber Range

`.`	`cdx-lig.run`
	`cdx-lin.run`
	`cdx-wig.bat`
	`cdx-win.bat`
		Script that runs the Scorer in one of four ways – `cdx-lig.run` invokes the GUI which looks like Figure 2 and Figure 3. The GUI supports configuration plus manufacture and distribution of OpenVPN credentials. This is a Linux script that may be run in the Ubuntu shell of Windows 10 if the XMing X server is installed and

| | | running. The Windows batch file `cdx-wig.bat` is the counterpart of `cdx-lig.run` except that it does not bring up anything related to the OpenVPN credentials. Both scripts have the ability to start and stop competitions, manually and automatically. `cdx-lin.run` opens a command-line only utility that can manage a competition but cannot configure one in Linux. The `cdx-win.bat` counterpart can do the same. |

Table 1: Relevant files that are part of the competition package

Directories created for file distribution to competitors

Directory	Description
Contestants	Contains a subdirectory for each competitor in the competitors database that is created after 'Send Files and Quit' command is activated in the Configurator. The name of each subdirectory is the name of the competitor in the competitor database where blanks are replaced with underscore. Thus, if a competitor's name is given as 'John the Farmer' then the name of the subdirectory is 'John_the_Farmer'. Below, '<competitor>' is used to represent any competitor's name. The contents of a competitor's directory depends on whether OpenVPN is employed to establish a competition network. If not, the directory contains a file specifying a competitor's competition IP address and start and end times. If so, additional files allow making a connection to the competition network via OpenVPN.
Parameters	Contains Parameters.txt for the CA – see Appendix I for contents
VPNServer	Contains server.tar for the OpenVPN server host – see Page 13 and Appendix I for contents

Table 2: Repositories for distribution

Files made via Configurator commands and distributed to Competitors and others

Directory	File	Description
`Contestants/<competitor>`	`Parms`	Information for a competitor – competition start time, competition end time, IP address for the competition.
	`email.txt`	The email address of the competitor. This comes from the 'Add Player' command which requires an email address.
	`run.client`	Calls `run.vpn` to join an OpenVPN network. Exists only if OpenVPN is used to establish a competition network.
	`stop.client`	Kills an `openvpn` process that connects the competitor to a competition network if OpenVPN is used.
	`run.vpn`	Makes a connection to an existing OpenVPN network by running `openvpn` on the `client.conf` that is created specifically for a competitor. Saves the process number of the `openvpn` process in `vpn.pid`. Exists only if OpenVPN is used.
	`client.conf`	A customized configuration file that `openvpn`

		is applied to for connecting to a competition network established by OpenVPN. Does not exist unless OpenVPN is used.
	`JAVA.txt`	Notes on the use of Java. Exists only if OpenVPN is used.
	`README.txt`	General operating notes. Exists only if OpenVPN is used.
`Contestants/<competitor>/ keys`	`ca.crt`	Certificate of the OpenVPN server. This directory and its contents exists only if OpenVPN is used.
	`clientXX.key`	Competitor's key for a secure connection to the competition network if OpenVPN is used. `XX` is a number no greater than 255. The name `clientXX.key` matches the name stated in `client.conf` and this location is also part of the customization for a competitor. The OpenVPN server uses `clientXX` as well in directory `ccd` to coordinate matching an IP address with a key.
	`clientXX.crt`	Competitor's OpenVPN certificate.
`Parameters`	`Parameters.txt`	File intended for the Competition Administrator with information regarding the competition including start date and time, stop date and time, IP address of the Scorer, scoreboard URL, scoreboard location, log verbosity, database recovery mode, and whether competitors can be added remotely.
`VPNServer`	`server.tar`	An archive of all files needed by an OpenVPN server. These include the following: - `allkeys` – a directory that contains all keys and certificates that were made when '`Make Keys`' is clicked in the Configurator. - `keys` – a directory containing keys and certificates specifically for the server including server.crt, server.key, ca.crt, and dh2048.pem. - `ccd` – directory that contains a collection of files named '`clientXX`', XX a number, with contents '`ifconfig-push <IP-address> 255.255.255.0`' that maps a key to an IP address so that a competitor getting a key/certificate pair is tied to a given IP address. This is done so the Scorer knows who it is scoring. - files for starting and stopping the network connection including `run.server`, `stop.server`, `run.vpn`, and `server.conf`. - example files for starting and stopping a socks proxy including `run.proxy`, and `stop.proxy` - `ipp.txt` – a file whose lines map a key (common) name to an IP address – it seems both `ipp.txt` and the files in directory `ccd` are necessary for the mapping to work as expected (mapping is retained after restarts). - information files – `JAVA.txt` and `README.txt`

Table 3: Relevant files that are created for a competitor with information needed to join a competition

The `contest` directory contains `openvpn`, `easy-rsa` files, some templates for vpn configuration, starting and stopping scripts, and Java source code for the CDX Control Panel and Configurator package. The `client` subdirectory contains the openvpn start and stop scripts which are to be distributed to competitors for connecting their OSes to the competition private network in case OpenVPN is being used for that purpose. The supplied `run.client` script shows how to connect to an OpenVPN server. It also shows how to set up a connection to a proxy server that is inside an organization perimeter - those lines are commented in the script. The first commented line of `run.client` would cause all running `openvpn` processes to be killed, if uncommented, before being connected to the VPN. This may not be desirable if more than one vpn address is needed on a host and, in that case, leave it commented but alert competitors that they may need to kill running OpenVPN processes on their own before connecting to the VPN. The `client` subdirectory also contains a `readme.txt` file, which explains the use of the entire package that will be distributed to competitors, and a template for `client.conf`, the configuration file for setting up a connection to a running `openpvn` server via `run.client`. Three lines need to be added to `client.conf` before distribution – these lines are unique to each Client and are explained below.

Other files include `game-id.txt` which contains certificate identification information for the X.509 certificates used in the competition; the directory `keys` which holds all the openvpn keys and certificates used for connecting to the VPN; and `vpnKeyIds.txt` which is a list of competitor names that is used to create directories containing information that is distributed to competitors.

Another subdirectory is `server` which contains the template for `server.conf`, `ipp.txt` and the `ccd` subdirectory: the latter two ensure a stable, predictable match between the keys and certificates to be distributed and VPN IP addresses assigned to competitors.

Files in the `src-xxx` directory that are created by CDX Control Panel and Configurator operations include `Parameters.txt` and the competitor database, the default name of which is `players.db`. Parameters.txt is used by the CA to set up the competition. It contains competition start and end dates and times so that Scoring can be started automatically plus logging and competitor database options, and scoreboard setup information. The competitor database file, the default name of which is `players.db,` contains a list of names of competitors, their IP addresses, their current scores, and email addresses.

OpenVPN Server Location

This is a text field in the CDX Control Panel that is filled in with the IP address or domain name of the host on which the OpenVPN server is to be run if OpenVPN is to be used (see Appendix C). If OpenVPN is not to be used then this field may be left blank. Otherwise, the contents of this field become part of `server.conf` and each competitor's `client.conf.` See Page 41 for the contents of `server.conf` and Page 39 for the contents of `client.conf.`

Competitors Database

The default name for the competitor database is `players.db`. The name can be changed by editing the file `GameParameters.java` in `src-xxx` and recompiling using `make` in Linux and `compile.bat` in Windows. This file may be created manually or by means of the 'Add' command in the CDX Control Panel. The file may also be edited manually or by means of the 'Delete' or 'Bump' commands in the CDX Control Panel. Information in `players.db` consists of current score, an unused field, the IP address that a competitor's OS takes (so that the Scorer knows who it is scoring), an email address of the competitor, and the competition identity of the competitor's OS. All of this information occupies one line of `players.db` for each competitor. Table 4 shows a few sample lines from a `players.db` file.

VPN Credentials

The section of the Configurator called 'OpenVPN Key & Certificate Maker' sets up the files needed for an OpenVPN server to establish a network and for competitors to connect to that network. There is no need to do anything with this section if OpenVPN is not used. Error: Reference source not found shows this section. Information seen in this window comes from file `game-id.txt` a default version of which is supplied. The fields

may be edited and changes will be stored when 'Save' is clicked. File `game-id.txt` is unchanged unless 'Save' is clicked. All fields must be populated if keys are to be made. Set the menu '`#keys`' to a number of keys to be made: default is 150. Click 'Make Keys' to make the keys. The keys and certificates for the server, certificate authority, and competitors are made in subdirectory `keys`. Click 'Stop Making Keys' to interrupt the key making. Keys that are already made are not erased.

```
...
1957 0 10.8.0.-46 franco@localhost Stack Smashers - C
3303 0 10.8.0.-44 franco@localhost 1337h4X0r$ - A
2437 0 10.8.0.-54 franco@localhost PRK Resistance - C
3843 0 10.8.0.-52 franco@localhost Forthcoming – A
...
```

Table 4: A few lines from a players.db file. The fourth octet in the IP address in each case is a negative number – add that number to 256 to get the positive octet. For example, the IP address for competitor Forthcoming – A is 10.8.0.204, for PKR Resistance – C is 10.8.0.202, and for Stack Smashers – C is 10.8.0.210.

Figure 7: *Make and distribute keys and certificates for connecting to an OpenVPN network*

Transfer Files to Competitors

If OpenVPN is to be used to establish a private network keys and certificates must first be made by selecting an appropriate number of keys to make and then clicking 'Make Keys' in the 'OpenVPN Key & Certificate Maker' section of the Configurator (see Error: Reference source not found). The keys are distributed to competitor directories when 'Prepare with OpenVPN keys' is clicked in the 'Prepare Files' section of the Configurator (see Figure 3). Keys are distributed to recipients, whose email address is part of `players.db`, when 'Send Files and Quit' is clicked in the 'Controls' section of the Configurator. In the Windows version the competitor files need to be distributed manually using the email addresses stated in the `email.txt` files.

Scoring

Scoring begins manually when the 'Start' button in the CDX Control Panel is clicked or automatically when 'Time to start:' becomes 0. Scoring ends manually when the 'Stop' button is clicked or automatically when 'Time to end:' becomes 0. Scoring may be suspended and resumed when those buttons are clicked. A real-time scoreboard is maintained by the Scorer. For details see the Scoreboard section above.

Help Boxes

In the Configurator hovering the mouse pointer over a button, menu, or text box will bring up an information box that explains what action the button is intended to perform or what the value of a text box means. To enable this feature click the 'Show Help' button in the bottom right section of the Configurator. An example is shown in Figure 8. Click 'Hide Help' to disable this function.
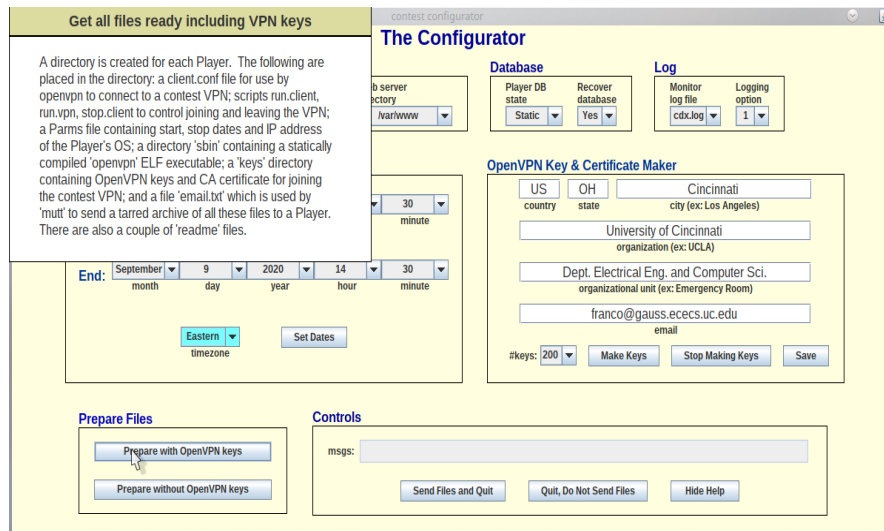
Figure 8: *Example of a help box opened after hovering over the '*Prepare with OpenVPN keys*' button*

Accessibility

The following are accessibility features in the ready-made Client:

1. Little actual typing is necessary. The setup parameters are provided by the CA or other person in a file and the file populates most editable fields and all necessary fields in the GUI. New passwords are generated by one button click. A list of all competitors is obtained by one button click – all commands, such as TRADE_REQUEST or GET_CERTIFICATE, that name a competitor use the list: the competitor merely clicks on the list entry to identify another competitor.

2. Hovering over a button or field displays a popup that describes what the button or field is for. This 'help' feature can be turned on or off and is off by default.

3. Text to speech is supported. All transactions are text based and are printed to the console to the console to allow a text-to-speech translator to transform the text to speech if the 'Showing Messages' label on a Command Panel button is present. The text that is printed to the console does not contain all the words in the transactions, rather the console messages are designed to be understood with minimal dialog. Clicking the "Showing Messages' button disables this feature and changes the label to 'Not Showing Messages'. Clicking that button again enables the feature.

Acknowledgments

This project was influenced by the iWars project that was designed and written by several students including graduate students Jonathon Meeks, Hrishikesh Bhide, and Brad Kuhn plus undergraduate students Coleman Kane, Robert Sexton, and Greg Larson.

Appendix A - How an Administrator Sets Up a Competition

Collect Competitor information

Assume the CA is beginning from scratch without a competitor database. The CA collects the following information from people who will be competitors: their preferred competition name and their email address. If OpenVPN is to be used to support the competition network the CA needs to know the IP address of the host for the OpenVPN server.

Install the software and/or run it

The software is bundled as part of a verification course that is archived in file `cryptol-course.7z`. This file can be obtained by doing this in a Linux terminal:

```
prompt$ cd ~/Downloads
prompt$ wget http://gauss.ececs.uc.edu/cryptol-course.7z
```

or by pointing a browser to `http://gauss.ececs.uc.edu/cryptol-course.7z` and downloading the file to directory `Downloads`. Uncompress the archive in a Linux terminal like this:

```
prompt$ 7z x cryptol-course.7z
```

to get directory `ElemCourse` with subdirectories `jdk-lin` and `cdest`, among others. The competition software includes a CDX Control Panel, a Configurator, and a Scorer. These are described below. All three exist in directory `cdest/src-lig`, compiled as `cdx-contest.jar`, and run from script `run-gui` or `cdest/cdx-lig.run`. When run, the CDX Control Panel appears (Figure 1) from which the competition can be controlled and configured using the Configurator (Figure 3). The Scorer is not visible but is controlled from the Control Panel and may even start automatically from information provided in file `Parameters.txt` which is located in directory `cdest/config`. Directory `cdest/src-lin` contains a second version of `cdx-contest.jar` that is text-based for remote control of a competition. That version has only functions of the Control Panel and the Scorer (Figure 6). The `ElemCourse` directory should be moved to a suitable directory, say a CA's home directory. It is self contained so no changes in the `PATH` variable are needed. Directory `cdest` can be copied to another location but `jdk-lin` or a link to `jdk-lin` must accompany it, wherever the copy winds up.

Create a Competitor database, assign IP addresses, and save to file

Using information collected from competitors, for each competitor, in the line beginning with the 'Add' button of the CDX Control Panel, enter the competitor's name (usually as requested by the competitor), the competitor's email address, and the IP address to be given to the competitor by the CA. It is important that the competitor sets its IP address to what is assigned or else the Scorer will not be able to probe that competitor's OS resulting in a loss of points during a competition. If OpenVPN is used the IP address will be `10.8.0.XXX` where XXX is a number from 100 to 249 (depending on the credential sets that have been made and exist in directory `cdest/contest/keys`) that is chosen by the CA and should be unique with respect to all other competitors, the OpenVPN server, and the Scorer. If OpenVPN is not used then the CA chooses IP addresses consistent with the requirements of the network the competition will be run in. After a competitor's information is filled in, click the 'Add' button. At any time click the 'List' button to see what the competitor database looks like. Click the 'Save' button to copy the competitor database to a file with name specified in the source file `GameParameters.java` (`players.db` by default). The 'Save' button should be clicked at least once – when the competitors have all been added. If some change to the competitor database must be made before the competition begins it is easiest to edit the competitor database file using a text editor. Otherwise, use the 'Delete' button in the Control Panel and then 'Add' the updated information for the competitor. Use the 'Bump' button if a competitor is to begin the competition with a non-zero number of points.

Configure a competition

The first and important thing to do, if OpenVPN is going to be used to establish a competition network, is type the local IP address of the host for the OpenVPN server into the CDX Control Panel field named 'OpenVPN server location'. The local address is the address inside an organization's perimeter, or internal address. If the address that is visible outside the perimeter, or external address, is used (and is likely different from the internal address) the network will be unusable. Click the button named 'Configure' at the bottom of the Panel. This brings up the Configurator as shown in Figure 3. Edit the fields 'Web server URL', 'Scoreboard title', and 'Web

server directory' as described in Section The Configurator on Page 4. Choose a Player DB state which is either Dynamic or Static. If Static, only the CA can edit the competitor database using the 'Add' button in the CDX Control Panel, or by directly editing the competitor database file (likely `players.db`). If Dynamic, anyone can enter a competitor in the database remotely (see Appendix B). Choose whether restarting a competition should return state to what it had been when the competition was stopped. Choose how verbose the log is. The competition start and end times and days are set in the section 'Competition Dates and Times' – choose the dates and times and be sure to <u>choose the correct time zone</u> and then <u>click the 'Set Dates' button</u>. If OpenVPN is to be used the CA may fill in the fields in the 'OpenVPN Key & Certificate Maker' section, and click the 'Save' button. Choose 'Make Keys' to create new credentials. It is up to the discretion of the CA to make credentials before any particular competition. If credentials have to be re-distributed due to some failure, possibly a network failure, making new keys should probably be avoided as that would change credentials for everyone, surely leading to some problems connecting to the server for some or all competitors.

Create files to be sent to Competitors

When satisfied with the configuration choices, and after dates and times are set, click the 'Prepare with OpenVPN keys' button if OpenVPN is to be used or the 'Prepare without OpenVPN keys' button. This action will create a directory called `Contestants` which contains a subdirectory for each competitor in the competitor database with name the same as the competitor's name and the contents of each competitor subdirectory contains at least a file with the email address of the competitor plus a file named `Parms` with IP address and competition start and end times in unix and human readable format. If OpenVPN is used each competitor subdirectory also contains OpenVPN credentials, a configuration file for `openvpn`, and some scripts for starting and stopping the `openvpn` client. Directory `Parameters` which contains `Parameters.txt`, the configuration file to be used by the Control Panel and Scorer, is also created at this point. That file may be sent to the person who operates the Control Panel and Scorer during the competition in case 1) the Scorer is to be run on a machine other than the one used to configure the competition or 2) the Scorer is on the same machine but run from a directory that is different from `src-xxx` (`src-xxx` has the `Parameters.txt` file as a result of this step).

Archive and send competition files to Competitors

Assuming no errors were encountered in creating the above directories click the 'Send Files and Quit' in Linux or 'Tar Files and Quit' in Windows. The result is each competitor subdirectory of directory `Competitors` is archived as a tar file. In either Linux or Windows the CA may individually send those tar files to competitors using the email address that is placed in `email.txt` in competitors' directories. In the case of Linux, an attempt is made to automatically email the tar files using `mutt`. This will not work unless the machine used by the software is not blocked from sending email. In both Linux and Windows 10 competitors can apply a `tar` command to unarchive the tar file they receive.

If an error occurs, it likely has something to do with OpenVPN. Check the competitor database file – particularly the IP addresses – and make sure keys and certificates have been created for all competitors (there will be errors if too few credentials were created when making keys).

In case OpenVPN is used, a directory named `VPNServer` is created and a tar archive of all files that should be sent to the server host is placed in that directory. That tar file needs to be sent to the person who will maintain the OpenVPN server during the competition.

Create the competition network

The competition should be run on a private network. The CA must supervise the establishment of this network before the competition begins. See Appendix C for information about establishing a private network with OpenVPN. If OpenVPN is to be used to create the network the file `server.tar` will be sent to the person responsible for maintaining the OpenVPN server. That person will untar the file received in a convenient location and must be able to use administrative privileges to run the script `run.server` as root. Doing so creates a network interface, most likely named `tap0`, that associates with the IP address `10.8.0.1`. It may be necessary to additionally create a socks proxy in case the OpenVPN server operates from behind an organization's perimeter. This may be accomplished using an OpenSSH server and establishing an account, for example named `visitor`, with a password to be distributed to competitors that need to access the proxy. However, care

must be taken to make sure that account will only allow access to the proxy and nothing else. It is recommended to put the OpenSSH server on a second machine behind the firewall to greatly limit the possibilities of an attacker to cause damage to organization machines. The proxy server could be started just before the competition and be terminated just after it ends for extra protection. More details on setting up and using a socks proxy are in Appendix M.

Start the competition and make checks

A competition may be started automatically or manually and may be controlled by the CDX Control Panel GUI or command-line version. To start a competition manually from the CDX Control Panel just click he 'Start' button or run the Start command in the command-line version. Click the 'Stop' button or run the Stop command to stop the competition or click 'Suspend' then 'Resume' to suspend scoring then resume it. A stopped competition can be restarted only by exiting and re-running the Scorer software. When doing so care must be taken to make sure that the Recover option is set in the `Parameters.txt` file that is loaded when the Scorer is re-run so that scores will be retained (it is safe for Recover to always be set because, if it is desired to reset all scores to 0 on the re-run, this can be accomplished by clicking the 'Reset' button). If dates and times were set properly in the Configurator, they will occupy two lines in the `Parameters.txt` file which is read by the CDX Control Panel when the Scorer is run. If the start time is in the future there will be text at the bottom left of the Control Panel that says 'Time to start:' followed by a string formatted like this: `dd:hh:mm:ss` where dd is number of days, hh is number of hours, `dd` is number of days, and `ss` is number of seconds – this string counts down the time to the beginning of the competition. The Scorer does not probe for active services during this countdown. Probing begins when the countdown reaches 0. At that moment the text changes to 'Time to end:' and a countdown to the scheduled end of the competition, with same format as above, begins. When that countdown reaches 0 probing ceases and the text changes to 'Contest: Finished'. The 'Start' button may be clicked to start the competition before the scheduled start time and the 'Stop' button may be clicked to stop the competition before the scheduled stop time.

The command-line Scorer performs the same operations as the CDX Control Panel except for Suspend and Resume. It is ideal for scoring a competition, especially if the Scorer is on a remote machine. Both Windows and Linux allow a command-line application (free of any use of graphics) to be started remotely, then continue running after the connection to the remote machine is cut, then reconnected at a later time when it is desired to issue commands, in this case to observe or control the Scorer. One of the oldest such applications in Linux is called `screen`. One logs in remotely then runs `screen` which, after typing space twice, brings up a shell. The Scorer can be started in that shell. To disconnect yet keep the Scorer running type control-A then type d. You will see something like `[detached from 66011.pts-0.gauss]`. The number in that string is a process number used to reattach to the running process. You can log out then log in later and run the command `screen -list` which presents a message that looks something like this:

```
There is a screen on:
        66011.pts-0.gauss       (01/24/2021 02:55:57 PM)        (Detached)
1 Socket in /run/screen/S-gauss.
```

To re-attach to the running process, for this example, run `screen -r 66011`.

An example of the use of the command-line Scorer is shown in Figure 6 and Appendix L contains a more detailed example.

During the competition the CA should check that the scoreboard is available and being updated at short intervals. If the CA wants to make the log file publicly available, it should also make sure that file is reachable as well.

Appendix B – What A Competitor Does To Compete

Receive files from the Competition Administrator

Competitors receive a `tar` archive containing files that are important to joining the competition. A competitor untars the archive in a convenient location. If OpenVPN is not used to establish a competition network then the only file in the archive is `Parms`. See Appendix H for the contents of `Parms`. With `Parms` a competitor will know what IP address it will be assigned, when the competition begins and when the competition ends. If OpenVPN is used additional files, as described in Appendix H, are used to join the competition.

Connect to the competition network

A competitor needs to make sure its IP address is set to the value that is given in the Parms file which is included in the archived set. This means DHCP (the usual default) is not used and a static IP address must be set instead. Doing this is straightforward with the tools that are available in Linux and Windows. If OpenVPN is used then the competitor needs to connect to the OpenVPN server which supplies the IP address. To do this the competitor must have administrative privileges which should be OK since it will likely be the competitor's personal machine or a machine supplied by the organization the competitor is associated with. If the competitor needs to connect to the network using a socks proxy the CA will have informed competitors what the account, password, and IP address of the socks proxy is. The competitor will connect to that proxy in a manner described by the CA. If the socks proxy is created In Linux, say with the account name `visitor` on machine `example.edu,` then the following line, followed by password when prompted, will open a proxy connection.

```
prompt$ ssh -N -f -T -D 8080 visitor@example.edu
```

To use the proxy the competitor must make sure the `socks-proxy` line in `client.conf` is uncommented (no semi-colon at the beginning of the line). To connect to the network invoke `run.client`. To quit the network invoke `stop.client`. See Appendix M for details about setting up access to a socks proxy and an example of its use.

Join the competition

Normally, a competition is joined after the CA has started the Scorer. That happens at precisely the day and time given by the CA in the `Parms` file. Before that time a competitor should already have put its OS online at the specified IP address (set manually – not with DHCP) and started serving. The task of the competitor is to ensure the specified services are up and running in its OS as often as possible and, possibly, to try to prevent an adversary's OS from having their services up and running.

Appendix C – Create a Competition VPN with OpenVPN

There are numerous details associated with creating a competition and getting all those details correct is a challenge. For this reason the configuration tool, discussed in detail in the Configuration Management Section, Page 2, was created. The configuration tool reduces the work in getting a competition set up to a few mouse clicks.   However, in some cases, for example setting up a socks proxy, some additional editing and/or distribution of files after using the tool will be necessary. This section presents what much of the tool does so as to enable a CA to work around any difficulties that are seen after the configuration tool is applied.

For OpenVPN to succeed it must cooperate with several other packages including OpenSSL, LZO, and EasyRSA, among others. But in our experience, one combination of versions of the above do not interact in the same way that another combination does rendering it difficult to create a lasting working environment for making OpenVPN credentials. Compounding this is that, at least for Ubuntu 22.04, the version of EasyRSA that is installed from repository is considerably out-of-date. Thus, installation of a specific, suitable collection of packages is detailed here before working with EasyRSA to create credentials is described. This will hopefully avoid many of the frustrations that we have experienced trying to get EasyRSA/OpenVPN to operate satisfactorily.

Assumptions

An OpenVPN server runs on a Linux host that is not inside an organization's firewall-protected perimeter
Competitor or Team OSes run on Linux machines, likely in VMs, that are outside the perimeter
The Linux host running the OpenVPN server is accessible via an OpenSSH server
Application `openvpn` is installed on all machines.

Create the key-making environment

The instructions of this section, for the Linux operating system, are adapted from the Anubisss github site

Get and install OpenSSL

```
prompt$ sudo mkdir /cdest              ← choose any name you like
prompt$ sudo chown user.user /cdest   ← user is the username of the person doing this
prompt$ mkdir /cdest/contest
prompt$ cd /cdest/contest
prompt$ wget https://www.openssl.org/source/openssl-1.0.2j.tar.gz
prompt$ tar -xvf openssl-1.0.2j.tar.gz && cd openssl-1.0.2j
prompt$ ./Configure gcc -static -no-shared –-prefix=/cdest/vpn
prompt$ make
prompt$ make install
```

check for the existence of `/cdest/vpn/bin/openssl /cdest/vpn/lib/libssl.a`.   Move `openssl` to `/cdest/contest/bin`.

Get and install EasyRSA

Download `EasyRSA-unix-v3.0.6.tgz` from the Easy RSA github site

```
prompt$ cd ~/Downloads
prompt$ tar xf EasyRSA-unix-v3.0.6.tgz
prompt$ mv EasyRSA-v3.0.6/* /cdest/contest/
```

Note: it may be necessary to install `bridge-utils` (ex: `sudo apt install bridge-utils`) you may as well do that to be on the safe side.

Note: consider disabling `openvpn` from automatic startup, if openvpn has been installed by a package manager, since a running `openvpn` may interfere with starting the server `openvpn` for the competition.

Create and distribute competition credentials and parameters

Assume the EasyRSA-v3.0.6 package has been moved to `/cdest/contest`.

```
prompt$ cd /cdest/contest
prompt$ cp vars.example vars
```

Edit the file `vars`:

```
#set_var EASYRSA_OPENSSL   "openssl"              ← replace this
set_var EASYRSA_OPENSSL "../vpn/bin/openssl"      ← with this
...
#set_var EASYRSA_DN  "cn_only"                    ← change this
set_var EASYRSA_DN  "org"                         ← with this
...
set_var EASY_REQ_COUNTRY "US"                     ← Choose country, city,
set_var EASY_REQ_PROVINCE "OH"                      province, organization,
set_var EASY_REQ_CITY "Cincinnati"                 organizational unit, and
set_var EASY_REQ_ORG "UC"                          email address that fits
set_var EASY_REQ_EMAIL "contestant@gmail.com"      your site
set_var EASY_REQ_OU "EECS"

# In how many days should the root CA key expire?
#set_var EASYRSA_CA_EXPIRE 3650                   ← 10 years default
set_var EASYRSA_CA_EXPIRE 365                     ← Suggest good for 1 year
...
# In how many days should certificates expire?
#set_var EASYRSA_CERT_EXPIRE 1080                 ← almost 3 years default
set_var EASYRSA_CERT_EXPIRE 365                   ← other certs good for 1 year
```

Make a Certification Authority certificate, server key, and server certificate from the command line

```
prompt$ ./easyrsa --batch init-pki
prompt$ ./easyrsa --batch build-ca nopass
prompt$ EASYRSA_REQ_CN=server ./easyrsa --batch gen-req server nopass
prompt$ ./easyrsa --batch sign-req server server
```

OpenVPN has a mechanism which maps a competitor's common name (`EASYRSA_REQ_CN`) to an IP address. This entails making a file whose name is the competitor's common name and whose content is the IP address and netmask and also adding an entry such as `client0,10.8.0.50` in a file named `ipp.txt`. To prepare for this do the following (**current directory is `/cdest/contest`**):

```
prompt$ mkdir server
prompt$ mkdir server/ccd
prompt$ mkdir server/keys
prompt$ cp pki/ca.crt server/keys
prompt$ cp pki/private/server.key server/keys
prompt$ cp pki/issued/server.crt server/keys
```

Then, for each competitor, make credentials that will be passed to the competitor. The credentials consist of two certificates and one key. Naming format is <Common-Name>.key and <Common-Name>.crt. Let <Common-Name> be `client0`. One certificate, `ca.crt`, has already been made in a previous step. To make and place the remaining credentials of `client0` do this:

```
prompt$ EASYRSA_REQ_CN=client0 ./easyrsa --batch gen-req client0 nopass
prompt$ ./easyrsa --batch sign-req client client0
prompt$ echo "ifconfig-push 10.8.0.50 255.255.255.0" > server/ccd/client0
prompt$ echo "client0,10.8.0.50" >> server/ipp.txt
```

Check directory `pki/issued/` for `client0.crt` and `pki/private` for `client0.key`. Repeat for all competitor Common Names (which do not have to match competitor identities). When done, if, say, 50 client keys and certificates were made with Common Names from `client0` to `client49` then directory `pki/issued` should have `client0.crt` to `client49.crt` plus `server.crt` and directory `pki/private` should have `ca.key`, `server.key` and `client0.key` to `client49.key`. Finally, run

```
prompt$ ../vpn/bin/openssl dhparam -out pki/dh2048.pem 2048
```

Check directory `pki` for `dh2048.pem`.

The OpenVPN server needs to have configuration files made. Enter the server directory.

```
prompt$ cd server
```

current directory is /cdest/contest/server

Create file `server.conf` with the following contents in directory /cdest/contest/server:

```
# the address of the machine which hosts the openvpn server
local 10.52.10.254    ← change this to match the IP address of the machine running
                         the openvpn server
# the port – 1194 is the usual
port 1194

# TCP or UDP server – I have run successful competitions with tcp
proto tcp

# dev tap means ethernet frames, dev tun means tcp packets
dev tap

# server credentials
ca keys/ca.crt
cert keys/server.crt
key keys/server.key  # This file should be kept secret
dh keys/dh2048.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt 0

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files
client-config-dir ccd
route 10.8.0.0 255.255.255.0

# There should be one file in ccd for every key
# that has been made for a Client and the name
# of the file should be the Common Name given to
# that key.  The format of the file in directory ccd
# is
#    ifconfig-push <ip-address-of-client> 10.8.0.1
#
# For example,for a Client key with Common Name client0
```

```
# a file client0 is created in ccd which has only the
# following line:
#    ifconfig-push 10.8.0.50 10.8.0.1
#
# where 10.8.0.50 is the IP address that has been
# assigned by the Competition Administrator to the
# Client who gets the key with Common Name client0.
# If X is the number of Client keys created there will
# be X files in directory ccd and each will
# have the one line format above.  For this distribution
# Common Names will range from client0 to client49.

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Allows different clients to see each other
client-to-client

# Ping-like messages are sent back and forth so each side
# knows when others go down.  The following causes a ping
# every 10 seconds, the peer is considered down if there
# is no response in 120 seconds.
keepalive 10 120

# Choose cipher
cipher AES-128-CBC    # AES

# Enable compression on the VPN link.
comp-lzo

# The maximum number of concurrently connected clients allowed
max-clients 200

# Reduce the OpenVPN daemon's privileges after initialization.
user nobody
group nogroup

# Avoid accessing resources on restart not accessible due to privilege downgrade.
persist-key
persist-tun

# Show current connections, truncated and rewritten every minute.
status openvpn-status.log

# log file: new on restart. Use log-append openvpn.log to append to existing log file
log openvpn.log

# verbosity 4 – reasonable for general usage
verb 4
```

Create server start script `run.server` in directory `/cdest/contest/server` with the following contents:

```
#!/bin/bash
# first line is likely necessary – other openvpn processes
# may interfere with the one that is being started here
```

```
sudo killall openvpn
sleep 2
sudo ./run.vpn
```

and create `run.vpn` in the same directory

```bash
#!/bin/bash
openvpn server.conf &
echo $! > vpn.pid
```

Create server stop script `stop.server` in directory `/cdest/contest/server`:

```bash
#!/bin/bash
if [ -e vpn.pid ]; then
    sudo kill `cat vpn.pid`
    rm -f vpn.pid
fi
```

Make all the scripts executable

```
prompt$ chmod a+x run.server stop.server
```

Directory `/cdest/contest/server` contains directory `ccd` with many files whose names are the Common Names given to competitor credentials plus files `ipp.txt`, `run.server`, `stop.server`, and `server.conf`. There is also a `keys` directory which now needs to be populated with the server's credentials as follows:

```
prompt$ cp ../pki/dh2048.pem keys
prompt$ cp ../pki/issued/server.crt keys
prompt$ cp ../pki/private/server.key keys
prompt$ cp ../pki/ca.crt keys
```

The `server` directory may now be archived and sent to the person that will maintain the OpenVPN server.

```
prompt$ cd ..
prompt$ tar cf server.tar server
```

For each Common Name a directory similar to that of `server` is to be created, populated, archived, and sent to the competitor for whom the Common Name applies. This is done here for `client0`.

```
prompt$ cd ..
prompt$ mkdir client0
prompt$ mkdir client0/keys
prompt$ cd client0
```

current directory is `/cdest/contest/client0`

Create file `client.conf` with the following contents in directory `/cdest/contest/client0`:

```
# Specify that this is a client
client

# Use the same setting as on the server – choose tap
dev tap

# Connect to a TCP server
proto tcp

# The hostname/IP and port of the server.
remote 10.52.10.254 1194 ← replace 10.52.10.254 with local address used in server.conf
                           unless a socks proxy is used (then use remote remote-host 1194)

# Keep trying indefinitely to resolve the host name of the OpenVPN server.
resolv-retry infinite

# Most clients don't need to bind to a specific local port number.
nobind
```

```
# Downgrade privileges after initialization (non-Windows only)
user nobody
group nogroup

# Try to preserve some state across restarts.
persist-key
persist-tun

# credentials
ca keys/ca.crt
cert keys/client0.crt        ← replace client0 with Common Name of client
key keys/client0.key         ← replace client0 with Common Name of client

remote-cert-eku  "TLS Web Server Authentication"

# cipher same as for the server
cipher AES-128-CBC

# compression
comp-lzo

# log file verbosity.
verb 4

# Do not use socks proxy if Client is inside perimeter:
# otherwise uncomment by removing ';'
; socks-proxy 127.0.0.1 8080  ← uncomment if using a socks proxy
```

Create a Client start script `run.client` in directory `/cdest/contest/client0` with the following contents:

```
#!/bin/bash
# consider uncommenting next line if Client is the
# only Client on host
# sudo killall openvpn
# uncomment the next 3 lines if a socks-proxy is to be used
# sleep 1
# ssh -N -f -T -D 8080 <login-name-of-proxy-host>@<openvpn-server-host-address>
# sleep 1
sudo ./run.vpn
```

and create `run.vpn` in the same directory

```
#!/bin/bash
openvpn client.conf &
echo $! > vpn.pid
```

Create Client stop script `stop.client` in directory `/cdest/contest/client0`:

```
#!/bin/bash
LINE=`pstree -paul | grep ssh | grep 8080 | grep <login-name-proxy-host>`
IFS=',' read -ra N <<< $LINE
PID=${N[1]}
if [ -z "$PID" ]; then
    echo no proxy running
else
    kill $PID
    echo proxy with pid $PID killed
fi

if [ -e vpn.pid ]; then
    sudo kill `cat vpn.pid`
    rm -f vpn.pid
fi
```

The `stop.client` script contains code that kills a socks proxy that would have been activated if the commented portion of `run.client` was uncommented. There should not be side-effects if `stop.client` is run when no socks proxy exists.

Make all the scripts executable

```
prompt$ chmod a+x run.client stop.client
```

Create the `Parms` file that is used to provide information to the Client that enables the Client to enter a competition. The following is an example.

```
# Start Time
1599516000                    ← POSIX time that competition will start
07.09.2020  18:00:00 Eastern  ← start time in human readable form

# End Time
1599775200                    ← POSIX time that competition will end
10.09.2020  18:00:00 Eastern  ← end time in human readable form

# Client Location
10.8.0.200                    ← IP address of the competitor's Server
```

Directory `/cdest/contest/client0` contains scripts `run.client`, `stop.client`, and files `Parms` and `client.conf`. There is also a `keys` directory which now needs to be populated with the competitor's credentials as follows:

```
prompt$ cp ../pki/issued/client0.crt keys
prompt$ cp ../pki/private/client0.key keys
prompt$ cp ../pki/ca.crt keys
```

The `client0` directory may now be archived and sent to the person that will operate the OS designated as `client0`.

```
prompt$ cd ..
prompt$ tar cf client0.tar client0
```

The above is repeated for all OSes in the competition.

Start the openvpn server

On the server host untar `server.tar` and change directory to `server`.

```
prompt$ tar xf server.tar
prompt$ cd server
```

Whatever machine the server is operated from, it should have access to an `openvpn` binary that is typically in directory `/sbin`. To start the VPN server run the following:

```
prompt$ ./run.server
```

Stop the openvpn server

To stop the VPN server run the following from the same directory that `run.server` was run from.

```
prompt$ ./stop.server
```

Competitor's OS joins the VPN

Assume that `/sbin/openvpn` exists. A competitor receives `<competitor-name>.tar` from the CA. The competitor untars the file.

```
prompt$ tar xf <competitor-name>.tar
```

This results in directory `clientXX` which contains `run.client`, `stop.client`, `client.conf`, `Parms`, and subdirectory `keys` which contains `ca.crt`, `clientXX.key`, and `clientXX.crt` where XX is a

number from 0 to 149 assuming the setup above was followed (otherwise the client key and certificate could be anything the CA chooses). To join the VPN the competitor executes

```
prompt$ ./run.client
```

This adds IP address 10.8.0.YY to the Client host's network interface.

Note: the server has file `clientXX` in its `ccd` directory and the contents of `clientXX` is

```
ifconfig-push 10.8.0.YY 255.255.255.0
```

Note: the server also has a line in `ipp.txt` that says

```
clientXX,10.8.0.YY
```

Then the competitor checks the IP address with the command:

```
prompt$ /sbin/ifconfig
```

Before starting its OS, the competitor makes use of the information in the `Parms` file to set up its OS – the IP address of the OS must be set as in the Parms file and the start time should be noted and the OS must be providing services by that time. Then the OS services can be started.

Appendix D – How to Join the VPN Network

Assumptions

The competitor will enter the competition through a Virtualbox VM that is a Linux variant.
The instructions below are written specifically for Ubuntu 22.04 but similar procedures apply for others.
The Ubuntu 22.04 desktop install iso has been downloaded from the <u>Ubuntu</u> website.
The competitor receives <competitor-name>.tar from the CA containing files in Appendix H.
The competition Scorer is run in a VPN which is behind some organization's firewall-protected perimeter.

Overview

The competition is run over an IPv4 VPN network which is facilitated by the OpenVPN package. IP addresses given to competitors are pre-assigned in the range 10.8.0.100 to 10.8.0.249. Since each competitor may get more than one competition account, depending on the decision of the CA, each competitor may be assigned more than one set of credentials and therefore more than one VPN IP address. Competitor (or Team) OSes are expected to be installed on separate VMs – one VM hosting one competitor (or Team) OS. Instructions for setting up those VMs are given below assuming Virtualbox is used as a (type 2) hypervisor.

What the Competitor needs to do

VirtualBox may be downloaded from the **<u>VirtualBox</u>** website. The extension needs to be obtained as well: click 'All supported platforms'. Installation for Windows or MacOS begins by clicking one of those links. To install in a Linux distribution click the 'Linux distributions' link and then scroll down until you reach instructions for your OS. To install the extension start Virtualbox (command line: virtualbox, menu: system). Open the 'File' menu and select 'preferences'. Click 'Extensions' to bring up the dialog box shown in Figure 9.
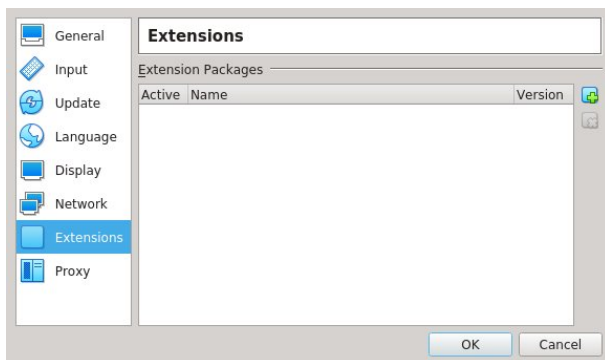

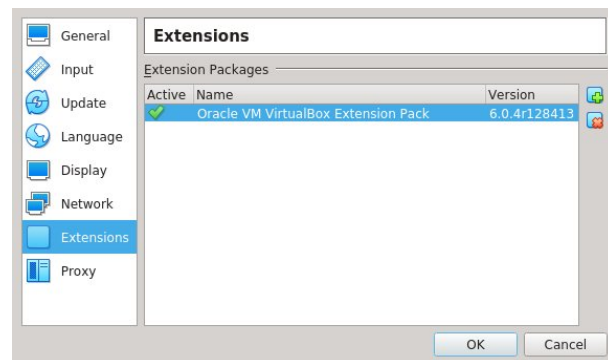Figure 9: *Click the blue-green square icon to load extensions*

Figure 10: *Extensions have been loaded*

Click on the small blue-green square icon with a plus in the lower right corner to bring up a file dialog box. Select the downloaded extensions file and then click on install, accept, and so on. If done correctly the extensions dialog box will be as shown in Figure 10.

Create an Ubuntu VM in Virtualbox

The first step is to prepare an environment for the OS. Click the 'New' button shown in Figure 12 top and fill in the details as shown in Figure 12 middle except change /home/franco to the competitor's home directory. Click 'Next' and choose a memory size as big as can be made (say 3GB if the host computer has 16GB of RAM). Click 'Next', then click 'Create', then click 'Next', click 'Next', choose, say 30GB, for the disk size (the disk actually uses much less storage - the 30GB is just a maximum size). Then click 'Create'. A new tag appears in the left margin of the Virtualbox screen as shown in Figure 12 bottom. With the new tag selected click the 'Settings' icon as shown in Figure 11 top and select 'Storage' to get the screen in Figure 11 bottom. Click the small blue disk with a plus to the right of 'Controller: IDE' as seen in Figure 11 bottom. Click 'Choose disk' - select the downloaded Ubuntu iso image (which should be in the Downloads directory). Then select the new

Ubuntu entry in the storage dialog, click 'OK', and click the green arrow facing right, shown in Figure 11 top. The Ubuntu install image will boot. Choose install - answer questions on the way - choose language, choose normal installation, choose 'Erase and install Ubuntu' and click 'Install Now'. Choose 'Continue' to write changes to disk, enter name, username, password, timezone etc. Reboot when a dialog box appears and says to do so.
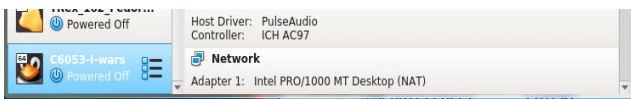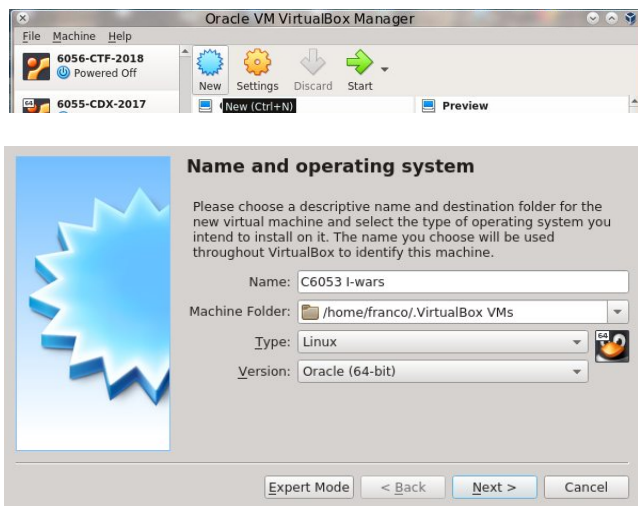

Figure 12: *Create an environment for the new VM*
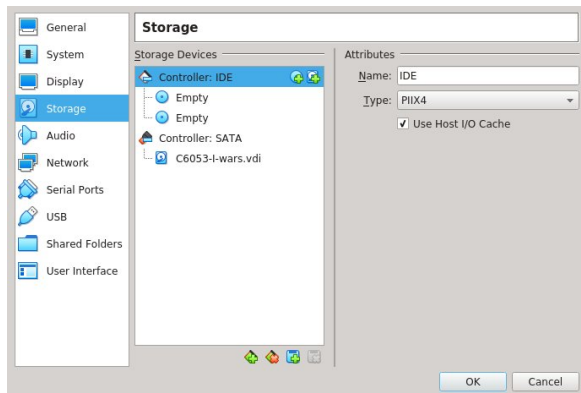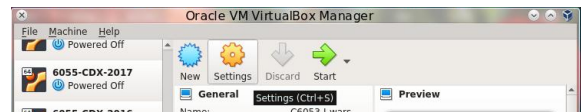
Figure 11: *Ubuntu ISO as a disk drive*

Configure the Ubuntu VM

With the VM powered down, configure the OS by selecting the VM tag on the left of the Virtualbox main dialog box and clicking 'Settings' as before. Then select the tags in the left side of the Settings box one by one and choose resources and settings. Most of the defaults are OK. Increase the number of cores dedicated to the VM, if possible. Changing the Graphics Controller to VBoxSVGA should provide increased screen area. The Network should be set to NAT as shown in Figure 13.
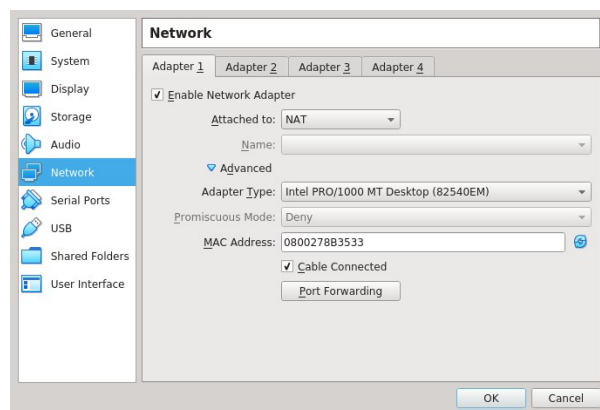
Figure 13: *Network setting*
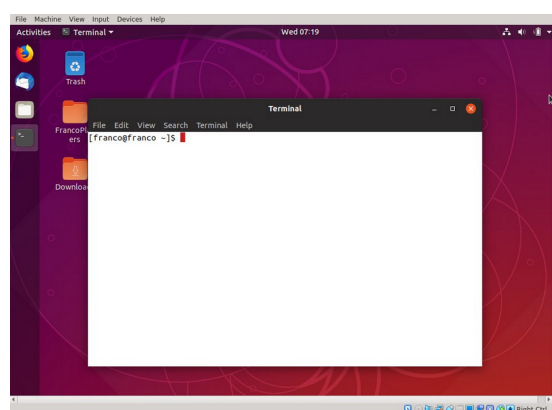
Figure 14: *Desktop*

Update and install software

Open Virtualbox, select the VM tag in the left margin, click the green arrow facing right to boot the OS. Once logged in, open the 'Activities' menu at the top left, search for 'terminal', and hit return. The desktop looks something like Figure 14. Update the OS using the commands below which are entered at the terminal prompt.

```
prompt$ sudo apt update
prompt$ sudo apt upgrade
```

The OS will authenticate by asking for the password specified during the OS install after the first `sudo` only. Next, install important software.

Make sure OpenVPN is installed with

```
prompt$ locate openvpn | grep bin
```

If it is not installed it can be obtained with

```
prompt$ sudo apt install openvpn
```

Also run

```
prompt$ sudo apt install net-tools bridge-utils
```

Next, place files obtained from the CA and use credential info.
Each competitor is given the following file (see Appendix H for the files it contains):

```
<competitor-name>.tar
```

The files are intended for a single VM. To get the files into the VM find out the IP address of the host that these files exist on. This can be done in Linux with `/sbin/ifconfig`. If the host is Windows use `ipconfig` from `powershell`. Say the host IP address is `192.168.1.107`. Also, determine the directory those files are in on the host. Say it's `~/Downloads`. From a terminal in the VM:

```
prompt$ mkdir ~/Competition
prompt$ cd ~/Competition
prompt$ scp <host-username>@192.168.1.107:Downloads/<competitor-name>.tar .
```

Next untar the tar file like this:

```
prompt$ tar xf <competitor-name>.tar
```

This results in a directory named `<competitor-name>` with subdirectory `keys` plus start and stop scripts and `client.conf`. The keys and certificates in the directory `keys` are unique to a single account and, among other things, they determine what the VPN IP address of the account will be. The `<competitor-name>` directory can be placed anywhere that can be accessed by a normal user, such as that user's home directory.

The next step is to connect to the VPN.

Join the VPN

Change directory to `<competitor-name>`. Look at `run.client`. Here are the contents with lines 2 to 5 uncommented:

```
#!/bin/bash
sudo killall openvpn
sleep 1
ssh -N -f -T -D 8080 visitor@example.edu
sleep 1
sudo openvpn client.conf
echo $! > client.pid
```

The `ssh` line establishes a socks5 proxy behind the organization's firewall to allow external communication to port 1194 of the `openvpn` server. See Appendix L for how to create a proxy server account and how to use it in this competition. For illustration only, suppose an account named `visitor` has been created on a machine named `example.edu` which is inside the organization's perimeter. Although the socks5 proxy is intended for external communication it should work just fine inside the organization's network so there is no need to change anything if the competition VPN is entered from outside or inside the organization's perimeter. Commenting out the ssh line is OK inside the organization's perimeter but then the last line in `client.conf` must be commented or removed as well. Normally, `run.client` would be invoked at this point but if something is wrong you would not see why so do this instead:

```
prompt$ sudo openvpn client.conf
```

The result is a long output which, if everything is fine, ends with something similar to this:

```
Wed Mar 20 09:56:29 2019 us=137912 do_ifconfig, tt->ipv6=0, tt->did_...
Wed Mar 20 09:56:29 2019 us=138002 /sbin/ip link set dev tap0 up mtu 1500
Wed Mar 20 09:56:29 2019 us=141321 /sbin/ip addr add dev tap0 10.8.0.104/24...
Wed Mar 20 09:56:29 2019 us=144110 GID set to nogroup
Wed Mar 20 09:56:29 2019 us=144175 UID set to nobody
Wed Mar 20 09:56:29 2019 us=144203 Initialization Sequence Completed
```

If `Initialization Sequence Completed` is not observed something is wrong. If so, the problem could be due to an already running instance of `openvpn` or `vpnagentd` that was not killed (most common problem) or the keys can not be found because their location does not match that stated in `client.conf` or `server.conf` is wrong because the wrong IP address was entered in the 'OpenVPN server location' field in the Control Panel or the OpenVPN server is not running.

Here are some additional checks. Run `/sbin/ifconfig` to get something like this:

```
tap0  Link encap:Ethernet  HWaddr 4e:f3:3f:3d:f8:b5
      inet addr:10.8.0.101  Bcast:10.8.0.255  Mask:255.255.255.0
      inet6 addr: fe80::4cf3:3fff:fe3d:f8b5/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:50 errors:0 dropped:0 overruns:0 frame:0
      TX packets:57 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:2100 (2.1 KB)  TX bytes:6414 (6.4 KB)
```

No `tap` interface means there is no connection to the VPN. If the Scorer is running and its IP address is known to be, say, `10.8.0.100`, try this to make sure you can connect to it.

```
ping 10.8.0.100
PING 10.8.0.100 (10.8.0.100) 56(84) bytes of data.
64 bytes from 10.8.0.100: icmp_seq=1 ttl=64 time=77.1 ms
64 bytes from 10.8.0.100: icmp_seq=2 ttl=64 time=37.4 ms
64 bytes from 10.8.0.100: icmp_seq=3 ttl=64 time=35.2 ms

--- 10.8.0.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 35.258/49.958/77.126/19.232 ms
```

Passing the above tests means the Client software is connected to the VPN. From this point on use `./run.client` to join the VPN.

Disconnect from the VPN

To disconnect run

```
prompt$ ./stop.client
```

Appendix E – Hard Coded Configuration Parameters

File	Line	Purpose
CheckServices.java	`srvports[0] = "13/tcp";`	Services to be monitored by the Scorer. These include, in order, daytime, ftp, http, printer, mysql
	`srvports[1] = "21/tcp";`	
	`srvports[2] = "80/tcp";`	
	`srvports[3] = "631/tcp";`	
	`srvports[4] = "3306/tcp";`	
ConfigFrame.java	`fos = new FileOutputStream(` `  "../Parameters/Parameters.txt");`	Fix "`Parameters.txt`" name
	`fos = new FileOutputStream(` `  "Parameters.txt");`	
	`fis = new FileInputStream("../config/"` `  GameParameters.PLAYER_DB_FILE);`	Fix "`players.db`" name
	`String ip="10.8.0."+(ip_count++);`	Set IP address 3 octets
	`FileOutputStream fos = new` `FileOutputStream(contdir+"/Parms");`	Fix "`Parms`" name for OpenVPN

	`FileOutputStream fos = new`
	`FileOutputStream(contdir+`
	`"/email.txt");`
		Fix "`email.txt`" name

DynamicListener.java	`socket = new ServerSocket(9180);`	Entry port for Dynamic mode
GameParameters.java	`public static String GAME_DIRECTORY` `= "./"`	Game directory
	`static final String PLAYER_DB_FILE` `= "players.db";`	Fix "players.db" name
	`public static int` `TIME_BETWEEN_PLAYER_PROBES = 10000;`	Fix time between probes in milliseconds
Monitor.java	`String logfile = "cdx.log";`	Fix name of log file

	`HTTPQuery q = new`
	`HTTPQuery(competitor,"wordpress/?p=4", 80,`
	`phrase, null);`
		Fix WordPress query

	`HTTPQuery q = new`
	`HTTPQuery(competitor,"index.html",80,`
	`"<head>", null);`
		Fix HTTP query

Appendix F – Sample Configurations and Skill Levels

Configuration	CA Decisions	Group/Machines
Basic	Ready made Clients are handed out to competitors Three VMs and three Clients per team No authentication, encryption, integrity checking Scorer source and binary is withheld Scorer log is made public	Middle School Scorer on desktop Clients on laptops Students do not code
Intermediate	Ready made Clients are handed out to competitors Ten VMs and Clients per team Authentication, encryption, integrity checking provided Scorer source code and binary are withheld Scorer log is made public	High School Scorer on desktop Clients on laptops Students choose protection params
Experienced	Teams build Clients using high level language Existing libraries may be used Twenty VMs and Clients per team Authentication, encryption, integrity checking provided Scorer source code is distributed Scorer log is made public	University Scorer on server Clients on student computers VPN networked
Advanced	Teams must build Clients using assembly language Existing networking libraries disallowed Hundreds of VMs and Clients per team Client binaries may or may not be distributed Authentication, encryption, integrity checking supported Scorer binary is made public, source withheld Scorer log is withheld	Advanced Scorer and Clients run on Cyber Range

Table 5: Sample configurations and target audience with expected supporting computing environment

CO Skill / Motivating	Configurations			
	Basic	**Intermediate**	**Experienced**	**Advanced**
Low-level coding	No	No	No	Yes
Operating System coding/analysis	No	No	No	Yes
Networking	No	No	Slight	Yes
Advanced analysis	No	No	Slight	Yes
Advanced tools	No	No	Yes	Yes
Elementary analysis	No	Slight	Yes	Yes
Defend	Slight	Yes	Yes	Yes
Attack	Yes	Yes	Yes	Yes
Engrossing	Yes	Yes	Yes	Yes

Table 6: *Expected impact on CO skill development by sample configurations described in* Table 5

Appendix G - Configurator Parameter Descriptions

Web Server

Web server URL:
> The URL that a competitor uses from a browser to access the scoreboard and log file (if posted). This is also pasted into the `scoreboard.html` file to make scoreboard refreshes every 45 seconds reload the scoreboard. The default is `file:///home/httpd/html` which obviously needs to be changed.

Scoreboard title:
> The title the scoreboard takes. For example 'CDX Practice Contest' to indicate that a practice competition is in session. The default is just 'Contest' as shown in Figure 3.

Web server directory:
> The directory where the `scoreboard.html` file and `cdx.log` file should be placed so that they can be served to the web. The default is `/home/httpd/html.`

Database

player.db state:
> Possibilities are STATIC and DYNAMIC. In the case of DYNAMIC a competitor may create its own OS identities, as many as it would like. This state is most suitable for practice and development. A STATIC state is used for competitions. A competitor is assigned an identity as well as IP address by the CA and only by the CA.

Recover database:
> If YES, the old competitor database file is retained after the Scorer is killed and is reinstated when the Scorer is restarted. If NO, the database file scores are cleared (become 0) when the Scorer is restarted.

Log

Scorer log file:
> The name of the file which contains the competition log. The file is located in directory `config`. It's name is fixed at `cdx.log.`

Logging option:
> `1`: Events named below plus events that occur during file and direction creation, and deletion, and file reading
> `2`: Events named below plus events that occur during configuration
> `3`: Events named below plus other events that take place during a competition
> `4`: Events concerned with checking that services are up

Competition Dates and Times

Start:
> Select the time and day the competition will begin. Choose month, day, year, hour, and minute. This information is recorded in file `Parms`, a unique version of which is given to competitors, in both Unix time and human readable time, to enable a competitor to easily get ready for the competition either by developing a function that automatically starts its OS's services at the appropriate time or by joining a competition manually. `Parameters.txt` also gets this information for the benefit of the Scorer. Times selected are in local time as indicated by the timezone menu (see below).

End:
> Select the time and day the competition will end. Choose month, day, year, hour, and minute. This information is recorded in `Parms`, a unique version of which is given to competitors, in both Unix time and human readable time to enable a competitor to easily get ready for the competition end either by developing a function that automatically shuts its OS services at the appropriate time or by clicking a Stop button manually. `Parameters.txt` also gets this information. Times selected are in local time as indicated by the timezone menu (see below).

timezone:
Select the official timezone for the competition. Daylight savings is automatically accounted for. As stated above the start and end times are given in Unix time and human readable time. The Unix time includes the timezone information.

Save Dates:
Click to record the dates in `Parms` files and `Parameters.txt`.

OpenVPN Key & Certificate Maker

country, state, city, organization, organizational unit:
The address and organization that is hosting the competition. This information becomes part of the OpenVPN credentials that are given to the competitors. If OpenVPN is not used then these fields are irrelevant and may be left blank. Otherwise, no field is to be left blank. A default is provided and should be edited. It is OK for blank spaces to exist in any of the fields. Characters such as & and ! would cause key making to fail. If any field contains one or more such characters they will be converted to '.' to prevent failure.

email:
Email address of the Competition Administrator that is taking care of the competition. This is needed only if OpenVPN is used. Otherwise, this field may be left blank. A default is provided.

#keys:
The number of key/certificate pairs (credentials) to be made. If OpenVPN is to be used, these credentials will be distributed to competitors. The possibilities are 150, 100, 50, and 10.

Make Keys:
Begin the process of making the OpenVPN key/certificate pairs, the number of which is given by the #keys menu. This takes a very long time, especially to make 150 credentials. When completed, certificates `clientXX.crt` and `server.crt` appear in directory `contest/keys/issued`, `clientXX.key` and `server.key`, `ca.key` appear in directory `contest/keys/private`, and `ca.crt` and `dh2048.pem` appear in directory `contest/keys`. The XX in those file names is a number from 1 to 255. Nothing is distributed at this point and no credentials will be distributed if OpenVPN is not used, but the credentials can be made regardless. Credentials do not have to be remade for each competition. Any pre-existing files in the directories named above are erased when this button is clicked.

Stop Making Keys:
May be clicked while credentials are being made to stop the process. All credentials made up until that time remain but some things are incomplete such as `dh2048.pem`.

Save:
Click this button to save to file `game-id.txt` the organizational information and email written above the button. When the Configurator, is launched the `game-id.txt` file is read and its contents are shown in the organizational fields and email field of the OpenVPN Key & Certificate Maker section. The `game-id.txt` file is a simple text file that can be edited by a text editor.

Prepare Files

Prepare with OpenVPN keys:
Click this button to create directories for each competitor and to include in those directories `Parms` and OpenVPN credentials and configuration files, plus `email.txt` which contains the email address of the competitor who will receive the files. All competitor directories appear in directory `Competitors`. See Appendix H for a description of the files and directory structure in competitor directories. Directory `Paramters` is also created with its only contents as `Parameters.txt` which is competition information for the CA.

Prepare without OpenVPN keys:
    Click this button to create directories for each competitor containing only `Parms` and `email.txt`. These directories appear in directory `Contestants`. Directory `Parameters` is also created and contains file `Parameters.txt` for the CA.

Controls

Send Files and Quit:
    Click this button to archive all the competitor directories as `tar` files. All the competitor `tar` files appear in directory `cdest/Contestants`. These files are sent to the email addresses given for the competitors. In addition, if 'Prepare Files with OpenVPN' had been clicked, the directory `VPNServer` is created and contains an archive `server.tar` of all the information an OpenVPN server needs to set up a VPN for the competition. The CA must ensure that this archive is received by the OpenVPN server host.

Quit, Do Not Send Files:
    Exit the Configurator without sending any files to anyone. The directories `Contestants`, `Parameters`, and `VPNServer`, if they exist, remain.

Show Help:
    Click this button to see help windows for each of the menus and buttons displayed in the Configurator. Just move the mouse cursor over a button or menu and information about that widget appears as, for example, in Figure 8. Move the mouse cursor to another widget and the current Help window is replaced by a Help window for the widget the mouse is currently over. The 'Show Help' button becomes the 'Hide Help' button when it is clicked. Click the 'Hide Help' button to remove the currently displayed Help window and disable the Help function. This function has no effect on the CDX Control Panel.

Cancel:
    Click this button to exit from the Configurator without any configuration changes being saved.

msgs:
    Notifications are shown here

Appendix H - Structure of Files in Competitor Directories

Competitor directories are located in directory `Contestants` and have the form `<competitor-name>` where `<competitor-name>` is what is entered in the rightmost field of a line in the competitor database file except that blanks are replaced with underscore characters. The name of the competitor database file is determined from `GameParameters.java` and is `players.db` by default.

Parms:

This file is unique to all competitors and exists regardless of whether OpenVPN is used. It contains competition start time, end time, and IP address assigned to the competitor's OS by the CA. Here is a sample `Parms` file:

```
# Start Time
1599516000
07.09.2020  18:00:00 Eastern

# End Time
1599775200
10.09.2020  18:00:00 Eastern

# Client Location
10.8.0.228
```

The start and end times are translations to Unix time from the date and time selected and saved in the Configurator (see Figure 3). A human readable date and time is also displayed. The Client Location comes from the third field of a line of the competitor database file (named `players.db` by default). The Client Location, which is the IP address from which all the competitor OS services are accessible, can be entered using the 'Add' command in the Control Panel window as shown in Figure 2.

run.client:

This is a script in the package sent to competitors and may be used to connect a competitor's OS to an OpenVPN network. This script is included only if OpenVPN is used. Line 4 is where the socks-proxy is invoked, if needed. In the supplied `run.client` file this is commented, meaning no socks proxy is used. Uncomment Lines 4 and 5 iIf a socks-proxy is used and replace the address visitor@helios.ececs.uc.edu with the address of the correct competition socks-proxy server. Also uncomment the socks-proxy line in `client.conf` (see below). If competitor OSes are to have one IP address per host the CA may decide to uncomment lines 2 and 3 to make sure no other `openvpn` processes are running when Line 6 is reached – interference with other `openvpn` processes will likely prevent a connection with the `openvpn` server. All edits should be done for convenience before configuration so as not to have to change a file for each competitor.

```
#!/bin/bash
# sudo killall openvpn
# sleep 1
# ssh -N -f -T -D 8080 visitor@helios.uc.edu
# sleep 1
sudo ./run.vpn
```

run.vpn:

This is called from `run.client` and is included only if OpenVPN is used. It's lines are separated from `run.client` so that `stop.client` can be used effectively. The process number of the running `openvpn` is saved in file `vpn.pid`, to be used by `stop.client` later. Contents of `run.vpn` are:

```
#!/bin/sh
openvpn client.conf &
echo $! > vpn.pid
```

stop.client:

A script in the package sent to competitors that is executed by a competitor to kill the connection to the OpenVPN network if such a network is used. This script also terminates the connection to the proxy, if one is used and the lines 2-10 are uncommented. The competitor will have to edit line 2 and replace 'visitor' with the username associated with the proxy, and uncomment the commented lines if a socks proxy is used. If the local port is different from 8080 that number needs to be changed here and in `client.conf`. This script is included only if OpenVPN is used.

```
#!/bin/bash
# LINE=`pstree -paul | grep ssh | grep 8080 | grep visitor`
# IFS=',' read -ra N <<< $LINE
# PID=${N[1]}
# if [ -z "$PID" ]; then
#    echo no proxy running
# else
#    kill -9 $PID
#    echo proxy with pid $PID killed
# fi

if [ -e vpn.pid ]; then
    sudo kill -9 `cat vpn.pid`
    rm -f vpn.pid
fi
```

client.conf:

This is the OpenVPN configuration file that is sent to competitors if OpenVPN is used. The text below is the uncommented contents of a typical configuration file. All competitors will see the same uncommented contents except for the lines beginning with `cert`, and `key`. Keys and certificates will be custom created for each competitor and the names of the key and certificate files for each competitor must replace the names in those lines below.   This is done automatically by the 'Make Keys' command of The Configurator and Page 36 and 'Prepare with OpenVPN keys' on Page 36, and 'Send Files and Quit' on Page 37. The line beginning with `remote` shows an IP address which is set in the 'OpenVPN server location' field of the CDX Control Panel shown in Figure 2. If a socks-proxy is used, the semi-colon at the beginning of the line starting with `;socks-proxy` must be removed (the semi-colon at the beginning of a line means the line is commented out). The CA should not have to edit this file.

```
client
dev tap
proto tcp
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
remote-cert-eku "TLS Web Server Authentication"
cipher AES-128-CBC
comp-lzo
verb 4
;socks-proxy 127.0.0.1 8080   # uncomment to use a socks proxy
ca keys/ca.crt
cert keys/client0.crt
key keys/client0.key
remote localhost 1194         # the IP address of the OpenVPN server replaces localhost
```

Other files in a Competitor's directory

- `email.txt`: has the email address of the recipient of the competitor's `tar` file
- `JAVA.txt`: instructions for using Java
- `README.txt`: general usage information
- `ca.crt`: OpenVPN server's certificate, located in subdirectory `keys`
- `clientXX.crt`: competitor's OpenVPN certificate, located in subdirectory `keys`
- `clientXX.key`: competitor's OpenVPN key, located in subdirectory `keys`

Appendix I - Files in Parameters and VPNServer directories

Parameters.txt:

This file is intended for the administrator of the competition and is the only file in directory `Parameters`. The directory and file are created when 'Prepare with OpenVPN keys' or 'Prepare without OpenVPN keys' is invoked in the Configurator. Contents are, by example, as follows:

```
//Database Parameters
PLAYER_DATABASE_STATE 0

//Recover Parameters
RECOVERY_STATE 1

//Logging Parameters
LOG_FILE cdx.log
LOGGING_OPTION 0

//Scorecard Parameters
SCORECARD_FILE standings.html
WEB_URL http://example.edu/
SCORECARD_TITLE Contest
WEB_DIRECTORY /var/www/

//Extra
START_TIME_UNIX 1599516000
START_TIME_CONV 07.09.2020  18:00:00 Eastern
END_TIME_UNIX 1599775200
END_IIME_CONV 10.09.2020  18:00:00 Eastern
```

The meaning of the parameters is given in Appendix G.

server.conf:

This is the configuration file for the OpenVPN server and is in `server.tar` in subdirectory `VPNServer`. The following shows the uncommented lines from `server.conf`. The `ca`, `cert`, and `key` lines act as similar lines do for `client.conf` except that in this case the `key` and `certificate` belong to the OpenVPN server. The line beginning with '`server`' ensures an IP address exists for the server – in this case it is `10.8.0.1`. The lines

```
ifconfig-pool-persist ipp.txt 0
```

and

```
client-config-dir ccd
```

ensure that, for all keys and certificates that are distributed to competitors, there is a match of OpenVPN credentials with an IP address that is stated in files of subdirectory `ccd` and lines of file `ipp.txt`. More on this is given in the sections below that describe `ccd` and `ipp.txt`. The line beginning with '`local`' contains an address that comes from the 'OpenVPN server location' text field of the CDX Control Panel. This file is used by `openvpn` in `run.vpn`. Here is a sample `server.conf` file:

```
port 1194
proto tcp
dev tap
ca keys/ca.crt
cert keys/server.crt
key keys/server.key   # This file should be kept secret
dh keys/dh2048.pem
server 10.8.0.0 255.255.255.0
```

```
ifconfig-pool-persist ipp.txt 0
client-config-dir ccd
route 10.8.0.0 255.255.255.0
client-to-client
keepalive 10 120
cipher AES-128-CBC    # AES
comp-lzo
max-clients 100
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
log  openvpn.log
verb 4
local localhost
```

ipp.txt:

This file is in the `server` subdirectory and is referenced in `server.conf`. Each line of this file pairs a `key/certificate` surname matched with an IP address. This file is created when invoking 'Make Keys' in the Configurator. The first few lines of an example `ipp.txt` file follow:

```
client0,10.8.0.50
client1,10.8.0.51
client2,10.8.0.52
client3,10.8.0.53
client4,10.8.0.54
client5,10.8.0.55
client6,10.8.0.56
client7,10.8.0.57
...
client49,10.8.0.99
```

Thus, for this example, key and certificate named `client5.key` and `client5.crt` belong to the Client that will live on IP address `10.8.0.55` in the competition.

clientXX:

These files are placed in subdirectory `server/ccd`. The XX in each is a number from 0 up to 149. They enforce the matching between key and competition IP address. These are created when 'Make Keys' is invoked in the Configurator. An example for file `client40` is the following:

```
ifconfig-push 10.8.0.90 255.255.255.0
```

run.server:

Runs `run.vpn`. Uncomment the two commented lines to kill `openvpn` processes that may be running before `run.vpn` is invoked. Contents of this script is:

```
#!/bin/sh
#sudo killall openvpn
#sleep 2
sudo nohup ./run.vpn
sleep 2
```

run.vpn:

This is called from `run.server`. Its lines are separated from `run.server` so that `stop.server` can be used effectively. The process number of `openvpn` is saved in file `vpn.pid`, to be used by `stop.server` later. Contents of `run.vpn` are:

```sh
#!/bin/sh
openvpn server.conf &
echo $! > vpn.pid
```

stop.server:

Kills the connection to the OpenVPN network. This script also terminates the connection to the proxy, if one is used. The CA will have to edit line 2 and replace 'visitor' with the username associated with the proxy, and uncomment the commented lines. If the local port is different from 8080 that number needs to be changed in `server.conf`. This script looks like this:

```sh
#!/bin/sh
if [ -e vpn.pid ]; then
    sudo kill -9 `cat vpn.pid`
    rm -f vpn.pid
fi

rm -f nohup.out
rm -f *.log
```

keys:

A directory containing the following server credential files:
- `ca.crt`: the certification authority certificate
- `server.crt`: the server certificate
- `server.key`: the server key
- `dh2048.pem`: the Diffie-Hellman parameters

These are referenced in `server.conf`.

Appendix J - Files and subdirectories of contest directory

The following descriptions do not include files that are part of the `easyrsa` package. Files in this directory and subdirectories are used by a Competition Administrator to create archives that are distributed to competitors, and possibly an OpenVPN host.

vpnKeyIds.txt:

Contains a list of directory names, taken from the competitor database file, so that directories may be created for the competitor and Scorer packages when the 'Prepare with OpenVPN keys' or 'Prepare without OpenVPN keys' button and the 'Send Files and Quit' button in the Configurator are pressed. Below is an example:

```
Forthcoming_-_A
Stack_Smashers_-_A
Sleepers_-_A
```

game-id.txt:

Contains information saved by and for the key maker as shown in Figure 3 and Section 'The Configurator'. An example is shown below.

```
US
CA
Los Angeles
UCLA
Cyber Game Room
competitor@gmail.com
```

var, var1, var2:

The contents of `game-id.txt`, with edits, is sandwiched between `var1` and `var2` to produce `var` which is used by `easyrsa` to locate `openssl` and the `openssl-easyrsa` configuration file as well as set up parameters for certificates including expiration dates. Editable parameters include the following (defaults are preceded with #):

```
#set_var EASYRSA  "${0%/*}"
set_var EASYRSA_OPENSSL "./bin/openssl"
set_var EASYRSA_PKI   "$PWD/keys"
set_var EASYRSA_DN   "org"
set_var EASY_REQ_COUNTRY "US"
set_var EASY_REQ_PROVINCE "OH"
set_var EASY_REQ_CITY "Cincinnati"
set_var EASY_REQ_ORG "University of Cincinnati"
set_var EASY_REQ_EMAIL "franco@gauss.ececs.uc.edu"
set_var EASY_REQ_OU "Dept. Electrical Eng. and Computer Sci."
set_var EASYRSA_ALGO   rsa
#set_var EASYRSA_CURVE     secp384r1
set_var EASYRSA_CA_EXPIRE  365
set_var EASYRSA_CERT_EXPIRE   365
#set_var EASYRSA_CERT_RENEW   30
#set_var EASYRSA_CRL_DAYS  180
set_var EASYRSA_NS_SUPPORT "no"
#set_var EASYRSA_SSL_CONF  "$EASYRSA/openssl-easyrsa.cnf"
#set_var EASYRSA_DIGEST    "sha256"
#set_var EASYRSA_EXT_DIR   "$EASYRSA/x509-types"
```

openssl-easyrsa.cnf:

The configuration file for `openssl` that is used by `easyrsa` to establish certification authority locations, names, expiration dates etc., policy, request handling, Distinguished Name handling, and extension handling. This does not need to be edited.

bin:

A directory that contains statically compiled software for mailing archives, `openssl`, and archiving sets of files as follows:

- `tar`: archives files in subdirectory `client` for use by competitors and subdirectory `server` for the OpenVPN server
- `mutt`: mails the archives – must be used on a machine that legitimately supports mail transfer
- `openssl`: for creating OpenVPN keys and certificates

keys:

The directory in which OpenVPN keys, certificates, revocations, requests, etc. are placed, in some cases for distribution. See Section OpenVPN Key & Certificate Maker for a description of the keys and certificates that are made by 'Make Keys' of the Configurator.

examples:

A directory that contains examples of some of the files that get distributed or are used to create files for distribution including `players.db`, `Parameters.txt`, `vpnKeyIds.txt`, `game-id.txt`, and `Parms`.

client:

A directory that contains files and templates that are customized for competitors and distributed to competitors as a set of files that are archived. See Appendix H - Structure of Files in Competitor Directories for a description of these files and directories.

server:

A directory that contains files and directories that are customized for the OpenVPN server if OpenVPN is used. These files and directories are archived as tar files. See Appendix I - Files in Parameters and VPNServer directories for a description of those files and directories.

workaround:

A kludge script used to input parameters to `easyrsa` for `keypair` generation and request.

cmd1-cmd9:

Scripts used from within Java code to control the key making process. See Page 10 for more information about these scripts.

Appendix K – Files of `config` directory

These files contain competition specific information that is created and used by the Configurator and the Scorer. The first two files are intended to be fluid: they are likely to change more than once before and during the competition.

players.db and players.db.bak:

Contains current competitor location information, one competitor per line. The following is a sample. Columns, from left to right, are Current Score, unused, IP address (the negative number is just the way the unsigned character number that is actually used is displayed in a text editor - e.g. `10.8.0.-40` is actually `10.8.0.216`), email address of the competitor and Team name.

```
3843 0 10.8.0.-52 forth@forth.gmail.com Forthcoming - A
2717 0 10.8.0.-48 smashers@smasher.gmail.com Stack Smashers – A
2762 0 10.8.0.-40 sleepers@sleeper.gmail.com Sleepers – A
```

cdx.log and cdx.log.bak:

Contain the log of configuration and competition activity, current and past.

Parameters.txt:

This file is intended for the administrator of the competition and is the only file in directory `Parameters`. The directory and file are created when 'Prepare with OpenVPN keys' or 'Prepare without OpenVPN keys' is invoked in the Configurator. Contents are, by example, as follows:

```
//Database Parameters
PLAYER_DATABASE_STATE 0   # 0: Static  1: Dynamic

//Recover Parameters
RECOVERY_STATE 1          # 0: No  1: Yes

//Logging Parameters
LOG_FILE cdx.log          # hard coded for now
LOGGING_OPTION 0          # 0: all 1: config+services+cmds 2: services+cmds 3: services

//Scoreboard Parameters
SCORECARD_FILE standings.html
WEB_URL file:///home/httpd/html/
SCORECARD_TITLE Contest
WEB_DIRECTORY /home/httpd/html/ # scoreboard at http://gauss.ececs.uc.edu/standings.html

//Extra
START_TIME_UNIX 1701154800
START_TIME_CONV 28.11.2023  02:00:00 Eastern
END_TIME_UNIX 1701338400
END_IIME_CONV 31.11.2023  05:00:00 Eastern
```

The meaning of these is given in Appendix G.

Appendix L – The Text-Based Control Panel

The text-based Control Panel is intended for remote administration of a competition where commands need to be executed during the competition and the administrator does not wish to be glued to the competition without a break. It can be run on a remote Linux VM which is connected to the competition using `tmux` or `screen`. Then the CA can log into the remote VM via `ssh`, open a terminal, run `screen`, run the Control Panel in `screen`, perform necessary commands, then detach from the process, do anything, reattach later for more commands, and so on until the competition is over.

Sample session with `screen`

Assume a network of two computers: client at local IP address 192.168.1.107, and servers at remote IP address 129.137.4.132. The server host has the domain name `gauss.ececs.uc.edu`. The CA has privileges on both, not necessarily root privileges. The CA's account name is `franco` on both computers. Assume the text-based Control Panel application is installed on `gauss` in directory `cdest` with contents shown in Figure 15:

Figure 15: *Contents of directory* `cdest` *on* `gauss.ececs.uc.edu`

The directory `jdk-lin`, containing the Java Development Kit from which `cdest` was compiled, is in the directory above `cdest`. It is there if all of `ElemCourse` was copied to the `server` or perhaps just `jdk-lin` was. This is important to prevent failure in case the `server` does not have a Java Runtime Environment or has a Java Runtime Environment that is not compatible with `cdest`. The CA logs into `gauss` from the client as in Figure 16:

Figure 16: *The CA logs into the remote server host*

The CA runs `screen`. App `screen` begins with a copyright and license page ending as shown in Figure 17:

Figure 17: *Application* `screen` *has been started on the* `server`

After pressing space twice, a terminal prompt appears. The CA runs the Control Panel as shown in Error: Reference source not found:

Figure 18: *The Control Panel is started on the* `server`

47

The first thing a CA may do is run command Help to see what commands are possible as shown in Figure 19:

```
------------------------------
   Contest is already stopped
------------------------------
command> help
help
   Commands:
      Start -       Begin probing the services of each player
      Stop  -       Kill the check services threads
      Load -        Load players from database
      Save -        Save players to database
      Time -        Show days, hours, minutes, seconds to start or end
      List -        List players
      Add -         Add named player with email and named ip addresses
      Delete -      Delete named player
      Check -       Run check services thread on named player
      Interrupt -   Find and interrupt a named player
      Bump -        Increment score of named player
      WP -          Hit WordPress of named player
      HTTP -        Query named player w/ payload, port, phrase
      Hit -         Hit apache server of named player
      Reset -       Zero all scores
      Exit -        Leave
------------------------------
command>
```

Figure 19: *The Help command shows possible other commands*

After the CA does some work such as Adding players, Listing players, checking Time to start and so on the CA detaches from the Control Panel by typing Control-A then Control-D. The result is as shown in Figure 20:

```
[detached from 73401.pts-0.gauss]
franco@gauss:~/cdest$
```

Figure 20: *The CA has detached from the Control Panel - note process 73401*

The CA can now continue working on the server or log out of the server altogether and take a coffee break or do some work somewhere else then log back into the server and run `screen -r` to reattach to the Control Panel. The last thing displayed in the terminal before detaching now appears after reattaching except possibly updated as the Control Panel and possibly Scorer have been running in the background. The CA continues with the updated process. Note: since the Control Panel had been running in the background it may have started the contest on its own and, if so, the Scorer has been maintaining the scoreboard.

Appendix M – Setting up a jailed proxy server account

Assume the proxy server host OS is Ubuntu Linux. The host will likely have an internal IP address which is different from the external address. Therefore, in addressing the proxy server account from anywhere use the qualified domain name which will always be the same. In the following simplified example an OpenVPN network of two computers will be used to demonstrate setting up and testing the proxy server. The two computers are on a LAN with IP addresses 192.168.1.107 for the client and 192.168.1.125 for the proxy server and the OpenVPN server. Here are the steps - similar steps will apply to other OSes. Steps 1-8 are executed on 192.168.1.125.

1. If package `jailkit` is not installed, install it with the following:

    ```
    sudo apt install jailkit
    ```

    If package `openssh-server` is not installed, install it with the following:

    ```
    sudo apt install openssh-server
    ```

2. Create a user account that a competitor will access to acquire a connection to the proxy server. In this example the name of the account is `visitor`. Do the following:

    ```
    prompt$ sudo adduser visitor
    Adding user `visitor' ...
    Adding new group `visitor' (1001) ...
    Adding new user `visitor' (1001) with group `visitor' ...
    Creating home directory `/home/visitor' ...
    Copying files from `/etc/skel' ...
    New password:
    ```

    At the `New Password:` prompt type in the password – say it is `LuckyMe123` for this example

    ```
    Retype new password:
    passwd: password updated successfully
    Changing the user information for visitor
    Enter the new value, or press ENTER for the default
            Full Name []:
    ```

    Put anything as the `Full Name`, even just hit return – there is no reason why it is necessary to reveal information about this account to others. Continuing:

    ```
            Room Number []:
            Work Phone []:
            Home Phone []:
            Other []:
    Is the information correct? [Y/n] Y
    ```

    No info about the account is revealed in this example as all the fields are left blank. The home directory for this account is `/home/visitor`.

3. Create a directory called `jail` and populate it.

    ```
    prompt$ sudo mkdir /jail
    prompt$ sudo jk_init -v -j /jail jk_lsh
    prompt$ sudo jk_init -v -j /jail ssh
    prompt$ sudo jk_init -v -j /jail netutils
    prompt$ sudo jk_init -v -j /jail basicshell
    prompt$ sudo jk_jailuser -m -j /jail students
    ```

4. As root, edit `/etc/jailkit/jk_chrootsh.ini`: change `[test]` to `[visitor]` and uncomment that line plus uncomment the line `env=...` Here is what it will look like:

    ```
    ## example for a user
    [visitor]
    env= DISPLAY, XAUTHORITY
    #
    ## example for a group, there should be only 1 space in between the words!
    ```

5. In file `/etc/passwd` change

```
visitor:x:1001:1001:::/jail/./home/visitor:/usr/sbin/jk_chrootsh
```
to
```
visitor:x:1001:1001:::/home/visitor:/bin/bash
```

   Note: the 1001 is determined by the OS so it could be different when you try this example

6. Edit `/etc/ssh/sshd_conf` - add this at the end of the `sshd_conf` file:

```
Match group visitor
    ChrootDirectory /jail
```

   Allow password authentication – change `PasswordAuthentication no` to this:

```
PasswordAuthentication yes
```

   Allow access to the visitor account by anyone, anywhere (you may want this to be more strict – but this will get things working at least)

```
AllowUsers visitor@*
```

   Note: Once `AllowUsers` is used in `sshd_config` an attempt to log in by any <user>@<machine> pair that is not in the list will be denied so you may want to add more <user>@<machine> pairs to the `AllowUsers` list if you have just now used `AllowUsers` for the first time: use space as the separator. Then restart `sshd`:

```
prompt$ sudo service ssh restart
```

7. Check the contents of directories

```
ls /jail/bin/ : bash and sh → bash plus many others
ls /jail/dev: null tty urandom
sudo ls -al /jail/home/visitor:

  drwxr-x--- 3 visitor visitor  4096 Nov 27 10:58 .
  drwxr-xr-x 3 root    root     4096 Nov 27 10:58 ..
  -rw-r--r-- 1 visitor visitor   220 Nov 27 10:27 .bash_logout
  -rw-r--r-- 1 visitor visitor  3771 Nov 27 10:27 .bashrc
  drwxr-xr-x 2 visitor visitor  4096 Nov 27 10:58 .config
  -rw-r--r-- 1 visitor visitor 14965 Nov 27 10:27 .face
  -rw-r--r-- 1 visitor visitor 14965 Nov 27 10:27 .face.icon
  -rw-r--r-- 1 visitor visitor   807 Nov 27 10:27 .profile
ls /jail/usr/sbin: jk_lsh
```

8. Check the `PATH` variable from the `visitor` account

```
prompt$ su visitor (give password)
prompt$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:
/usr/local/games:/snap/bin
```

9. Test it on the local network with machines having (hypothetical) IP addresses 192.168.1.107 (client) and 192.168.1.125 (proxy server and OpenVPN server). The `visitor` account is on the 192.168.1.125 host. Do this from the 192.168.1.107 client:

```
prompt$ ssh -N -f -T -D 8080 visitor@192.168.1.125
prompt$ pstree -paul | grep visitor
  |   |   |   |   |   |   |            +-grep,523862 visitor
    +-ssh,523748,franco -AXY -N -f -T -D 8080 visitor@localhost (it's running)
  prompt$
```

   If you are ready for this, on the 192.168.1.125 host open the CDX Control Panel (the manual shows how to do this in multiple places – see e.g. Page 2), GUI version. **Important: before going any further, set the 'OpenVPN server location' to 129.168.1.125 in the CDX Control Panel** (that number becomes part of `server.conf` and if it is wrong a connection to the server is not possible). Then click 'Configure' to open the Configurator. Set the start date to some time in the past and the end date to a few days into the future. Set #keys to 10 and click 'Make Keys'. After keys are made click 'Prepare with

OpenVPN keys' then click 'Send files and Quit'. Change to `cdest/VPNServer`, untar `server.tar` (use, for example, `tar xf server.tar`). Enter subdirectory `server` and run `./run.server`. Check that there is a `tap0` network interface – for example, running `ifconfig` will result in something like this:

```
prompt$ ifconfig
  ...
tap0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.8.0.1  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::f8e1:35ff:fe9d:dbe7  prefixlen 64  scopeid 0x20<link>
        ether be:e0:56:d9:9c:08  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33  bytes 4722 (4.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note the IP address 10.8.0.1.

Enter directory `cdest/Contestants` on 192.168.1.125 and find `Scorer.tar` (If there is no such tar file add an entry to `cdest/config/players.db` and re-run the Configurator – do not make any keys this time - keys are still there from before and new keys will invalidate the OpenVPN server keys on 192.168.1.125 - just prepare files with OpenVPN keys and send them – there will then be a `cdest/Contestants/Scorer.tar` file). Send the `tar` file to 192.168.1.107 – use `ssh` or transfer the file using a flash drive.

Now, on machine 192.168.1.107, try to ping the OpenVPN server like this:

```
prompt$ ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
^C
--- 10.8.0.1 ping statistics ---
17 packets transmitted, 0 received, 100% packet loss, time 16363ms
```

So, the OpenVPN server network cannot be joined from 192.168.1.107. Now use the proxy server. Untar the `Scorer.tar` file. Enter directory `Scorer`. **Important: edit `client.conf`**. Remove the semi-colon from the line

```
;socks-proxy 127.0.0.1 8080  # important, remove the ;
```

Check that the last line in the file is this:

```
remote 192.168.1.125 1194    # important, ensure the IP address of the VPN server
```

Save file `client.conf`. Run `./run.client`. Still on 192.168.1.107 check for an OpenVPN network interface and IP address:

```
prompt$ ifconfig
tap0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.8.0.107  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::388e:55ff:fe68:a733  prefixlen 64  scopeid 0x20<link>
        ether be:e0:56:d9:9c:08  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33  bytes 4888 (4.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note the network interface is `tap0` and the IP address is 10.8.0.107. Now ping 10.8.0.1.

```
prompt$ ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=8.22 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=4.68 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=5.09 ms
64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=4.81 ms
```

It works.