Mastering EOS

Release 1.0

Sean Fisk and Ira Woodring

May 22, 2014

CONTENTS

Mastering EOS is an initiative to produce better documentation for the Grand Valley State University School of Computer and Information Systems Exploratory Operating System Labs (GVSU CIS EOS Labs). It is intended to augment and eventually supplant the original EOS documentation located on the School of CIS website.

Mastering EOS consists of two parts:

- A poster of lesser-known tips and tricks that can be used in the EOS Lab.
- A full manual documenting both basic and advanced tasks.

Mastering EOS is written by:

- Sean Fisk, BS and MS in Computer Science from GVSU
- Ira Woodring, GVSU EOS Lab system administrator

Contents:

CHAPTER

ONE

INTRODUCTION

The Exploratory Operating Systems Labs (EOS Labs) are a collection of computer labs maintained by the GVSU School of Computing and Information Systems. Some are for general CIS student use, and others are specific to certain courses, labs, or applications. The labs within the EOS Lab umbrella include:

Name	Location	Purpose
Exploratory Operating Systems Lab (EOS Lab)	MAK A-1-171	studying operating systems; general CIS computing use
Architecture Lab (Arch Lab)	MAK A-1-101	studying computer architectures; general CIS computing use
Data Communications Lab (Datacom Lab)	MAK 1-1-167	studying networking
Hardware Lab	MAK 1-1-105	studying hardware; multi-purpose

When addressing this collections of labs, we often refer to it simply as *EOS*. Though the EOS Lab is only one of the CIS department's computing labs, it was the original lab and other labs are largely based upon it. It is also the one most often used by the greatest variety of students.

1.1 Physical Access (Keycards)

The EOS Labs contain equipment specific to the CIS majors and are therefore closed to the general public. To access the labs, you are required to obtain a keycard. To be eligible for a keycard, you must be currently enrolled in a course which utiltizes the lab you'd like to access. After registration, visit the computing office in MAK C-2-100 to receive your card. A $25 deposit paid from your student account is charged upon receipt of the card. When the card is returned in acceptable condition, the deposit will be refunded.

This keycard grants access to the EOS and Architecture Labs. Simply swipe the card at the reader next to the door to open it. In addition, the card grants 24-hour access to Mackinac Hall. The courtyard door which is closest to the Architecture Lab possesses a reader which will open when the key is swiped. No other building doors are equipped with readers.

It is important that you do not allow anyone else to use your card. In addition, do not keep the card in your wallet, as the cards contain tiny hair-like wires that break easily when the card is flexed.

1.2 Computer Access (Credentials)

The EOS system uses separate user accounts from the general university computing infrastructure. EOS uses the same network ID given by GVSU's IT department, but authenticates against the CIS department's own LDAP server. The accounts are not interchangeable and administrators for one account cannot reset passwords for the other.

If you are registered for one or more eligible CIS courses [#eligible_cis_courses], you qualify for an EOS account. On the Friday before the semester starts, an account will be automatically created for you and a temporary password sent to your GVSU email address. To activate your account, you need to log in to the system once, either at a physical machine or *remotely using SSH*, and follow these steps:

- Enter your username.
- Enter the temporary password you have been given.
- If the temporary was entered correctly, you will be asked for the Current Password again. Enter the temporary password again.
- If the two temporary passwords match, you will be asked to create a new password.

Note the following rules when creating a new password:

- Your password must be at least 7 characters.
- Yoru password must not be based on a dictionary word.
- Your password should not be all numbers. The system will accept such a password, but it is incredibly insecure. Additionally, the system will often prevent login with such a password.

Please take the time to memorize your password! Password resets are available by contacting Ira Woodring, but it often takes a day to get to it. Professors are also able to reset passwords via an SSH reset mechanism, though some are unaware of this mechanism.

CHAPTER

TWO

RULES AND PROCEDURES

It is important that some basic rules and procedures be established to help maintain the lab and aid in its shared use. To those ends, please be aware of the following guidelines:

2.1 Disk Space

You are given 10 gigabytes (GB) of disk space. Once you exceed this limit, you can no longer store data on the system. This often leads to being unable to login via a graphical session, as the desktop manager must be able to write to disk. You must then log in via a text-based console and delete files to make space.

2.2 Copyrighted Material

Any files may be stored in your home directory. This includes games, movies, and music. However, allowing others to transfer copyrighted material may constitute a copyright infringement.

These incidents are usually caused by unintentional permissions issues or deliberate misuse of the system. Whatever the cause, copyright infringement is a violation of school policy.

These cases are taken very seriously. Your user account will be terminated immediatly upon discovering the infringement. Additionally, this offense is against our school's honor code, so you may be expelled or even face criminal proceedings.

Suffice to say, please be very careful when dealing with copyrighted materials within the EOS system. Be familiar with how permissions work, and take the time to set them correctly.

2.3 Food and Drink

You are encouraged to eat and drink in the labs — these labs were made for you and we want them to be as comfortable and useful as possible. Many restaraunts in the area deliver to the labs, including Papa John's and Jimmy John's. However, it is your responsibility to clean up after yourself. If the lab becomes too messy, policies to limit food and drinks will be instituted.

2.4 Overnight Parking

Even if you have a parking pass, it is still neccessary to obtain a special permit to park overnight. These permits are granted by Campus Safety. Parking overnight without one of these permits can result in a ticket or towing.

2.5 Living in the Lab

It should go without saying (but hasn't in the past) that you cannot live in the lab. People have been found living in the lab for brief periods of time between leases, etc. We must note that this is not only a huge safety violation but is illegal. If caught living in the lab, you will be removed and Campus Safety notified.

2.6 Malicious Activity

The infrastructure provided by the EOS Labs includes very powerful tools that can be used to secure network infrastructure. Unfortunately, these tools may also be used for malicious purposes. We provide these tools for you to learn to defend future systems it may be your job to secure. Under no circumstances should these tools be used to attack other students, machines, or entities. We do not provide these resources without reasonable oversight as to their use, and those using them illegally will be noticed and face strong consequences — possibly including removal from the university and criminal charges.

2.7 Games

We encourage playing games in the lab. However, if you are playing games and all machines are currently in use, please be polite and yield your machine to students needing to complete coursework.

CHAPTER

THREE

REMOTE ACCESS (SSH/VNC)

When not sitting at a physical machine in the EOS or Arch lab, EOS can be accessed from home using a protocol called Secure Shell (SSH). SSH allows the running of remote commands on any EOS machine.

SSH is a command-line-only technology. However, graphical remote access is available through a protocol called Virtual Network Computing (VNC). VNC allows access to a graphical desktop as if sitting at an EOS machine. Because the VNC protocol has no security of its own, our lab setup requires tunnelling VNC traffic through the SSH protocol. Each respective guide describes how to do this, but remember that you will first need to successfully set up SSH before attempting to use VNC.

The hostnames for the EOS machines are organized as follows: `eos`*`XX`*`.cis.gvsu.edu` where *XX* is 01 through 24 and `arch`*`XX`*`.cis.gvsu.edu` where *XX* is 01 through 10. Use these names to connect to a specific EOS machine.

Your SSH and VNC clients of choice depend on your machine's operating system.

3.1 Microsoft Windows

3.1.1 SSH

The most popular SSH client for Windows is called PuTTY. It can be installed by visiting the PuTTY download page. We recommend installing via the Windows installer, labeled *A Windows installer for everything except PuTTYtel*.

Logging In

The first we will do is create a saved session for our EOS connection configuration. This will save time for future logins.

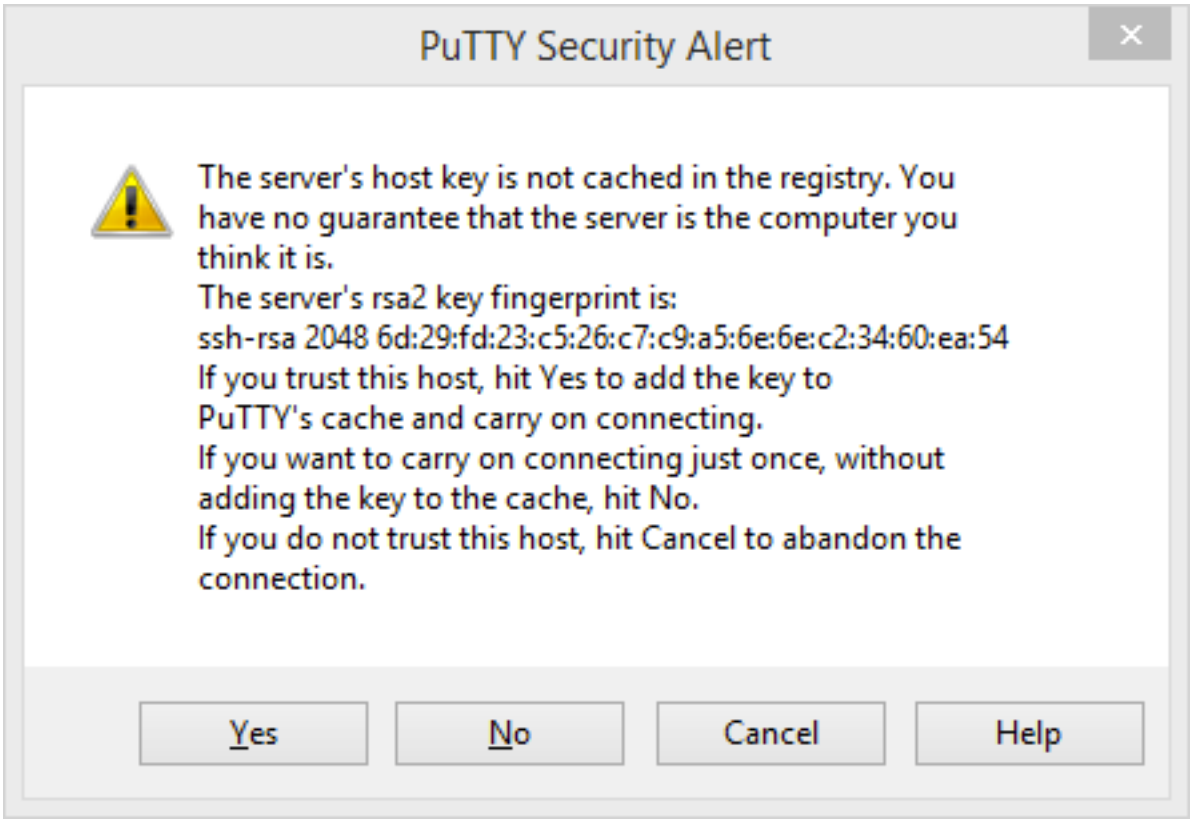
Open PuTTY and enter your username and the hostname of the EOS machine into the *Host Name* field. This will take the form *user@host*, for example, `doej@eos`*`XX`*`.cis.gvsu.edu`, where `XX` is the number of the chosen machine.

In the field under *Saved Sessions*, type `EOS` (this name is not strictly required, but is assumed in the next section). Click *Save*.

To log in to EOS, click *Open*. For future logins, simply select the session you created and click *Open* to connect.

Checking Host Fingerprints

When logging in to an EOS machine for the first time, you will see a dialog like this:



This is your SSH client requesting you to validate the identity of the machine to which you are connecting.

Each EOS machine has a so-called fingerprint, a series of characters which is used to verify its identity. To ensure that an attacker between your client and the actual EOS machine is not pretending to be an EOS machine, you must check that the machine's fingerprint matches the table below. Please report any mismatches to Ira Woodring immediately.

Host	Fingerprint
arch01	7e:df:14:7b:92:65:5e:29:59:99:4a:17:d1:79:1f:96
arch02	b7:b0:fb:63:e5:be:25:b0:d9:2c:15:78:1b:cc:ca:6f
arch03	99:d3:7d:15:72:2b:80:4c:cb:56:8c:f9:d6:15:1e:38
arch04	fa:ae:f8:97:89:59:eb:91:fa:d0:d9:37:c1:f2:45:44
arch05	67:b0:39:03:b2:c2:1c:1a:72:80:dd:49:23:d8:19:c4
arch06	6e:fc:21:fb:35:f5:b3:87:52:37:d3:5f:0b:d3:b3:b5
arch07	7b:0d:0e:e0:51:c4:f7:bb:64:8a:14:e7:ab:66:de:34
arch08	e5:80:ac:98:22:6e:b5:41:26:18:27:c8:a5:22:52:42
eos01	6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54
eos02	05:d1:d5:61:a4:c4:4e:ba:07:85:e2:18:8e:4b:7b:59
eos03	98:e4:c6:10:9a:63:e0:1d:63:ae:a8:27:c3:39:e4:ba
eos04	7c:b7:a2:65:74:c0:4f:52:ee:a2:2d:b8:12:38:9a:be
eos05	db:70:3d:04:44:d3:7f:a4:83:72:13:71:0c:be:ab:97
eos06	80:a9:e8:ea:b8:54:1c:4d:3b:9d:e2:c2:a7:d0:37:46
eos07	f4:15:32:2a:b5:18:e9:e6:51:8a:58:63:71:64:ba:8b
eos08	0d:0b:31:06:0a:6e:e6:b4:fb:48:0f:11:d6:7b:98:8a
eos09	6e:c0:20:4b:7d:f2:88:3d:df:71:ee:2e:63:2b:3c:33
eos10	be:fe:47:76:1d:5a:4b:96:fc:b9:1e:f2:42:fd:ed:41
eos11	be:47:a2:1c:6a:30:e7:98:ab:9a:1c:d3:4b:33:f2:57
eos12	15:21:3f:d2:b5:0b:04:06:29:2f:10:a7:04:3e:83:b7
eos13	31:5e:6f:ab:4a:7c:2f:3e:36:06:bd:e4:e0:a4:a3:22

Continued on next page

Table 3.1 – continued from previous page

Host	Fingerprint
eos14	87:af:45:1a:59:48:3d:57:66:a1:d6:6f:6e:a8:05:59
eos15	c2:b2:34:fa:8e:16:65:49:c5:ea:f4:70:c2:54:50:bc
eos16	57:71:e3:80:36:ff:25:d3:d1:c9:96:3d:f0:2d:42:a4
eos17	fd:3e:66:2a:39:08:b1:8f:c2:d7:e9:8b:59:40:a1:d9
eos18	a8:c5:e3:8f:1c:32:8e:72:c6:f4:dd:52:ad:23:63:31
eos19	d7:65:50:f2:23:b9:d7:58:76:9a:e2:5c:f3:1b:c0:11
eos20	a0:76:77:7a:2f:8f:cf:a6:f7:50:38:a4:36:12:4b:2d
eos21	23:3a:2b:42:b4:cf:f4:9e:c8:14:57:2f:73:8f:65:46
eos22	a5:fc:1e:65:bb:29:9b:ee:1c:4a:e9:46:41:a1:8e:cd
eos23	86:bc:19:fa:64:05:73:ef:81:78:cb:24:b5:f7:8c:fd
eos24	ec:5c:2f:a7:e9:86:9f:36:e6:85:a1:bc:54:04:ce:a5

Password-less Logins

It is often handy to be able to SSH into a host without having to type a password, for instance as part of a script. First, we need to generate your public/private key pair. Open PuTTYgen from the PuTTY distribution to begin the generation process. Click *Generate* and do the mouse nonsense to generate your keys.

Copy the text from the field labelled *Public key for pasting into OpenSSH authorized_keys file*. Open Notepad and paste this text. Save it to the desktop as `id_rsa.pub`. Now open Windows PowerShell from the Start Menu and run the following command. If your EOS saved session is named something other than "EOS", you will need to change it in the command below.

```
Get-Content "$env:USERPROFILE\Desktop\id_rsa.pub" | & "$env:SYSTEMDRIVE\Program Files (x86)\PuTTY\pl
```

Your public key has now been uploaded to EOS. The file `id_rsa.pub` may be deleted now.

However, we still need to be able to tell PuTTY to use your private key to log in to EOS. Back in PuTTYgen, click *Save private key* and save the resulting PPK file to a location of your choosing. We recommend your home directory. Answer *Yes* when you are warned about saving the private key without a passphrase.

Note: If you would like to use a passphrase for your key, see the PuTTY Guide to Pageant after completing this guide. Setting up an SSH agent is out of the scope of this guide.

Now start up PuTTY, click your saved session, then click *Load*. In the configuration tree to the left, expand *Connection* → *SSH* and click on *Auth*. Click *Browse...* to the right of the field labelled *Private key file for authentication*. Select the PPK file you saved earlier.

Go back to *Session* and click *Save*. PuTTY is now configured to use this private key to connect to EOS. Click *Open* to log in, which you should be able to do without a password.

As is obvious from these instructions, SSH key management is not a simple process. We recommend reading the PuTTY Guide to SSH Keys, which is the source for much of this information. If you would like to use a passphrase with your key, please see the PuTTY Guide to Pageant, PuTTY's SSH agent.

Alternatives

Though PuTTY is the recommended client, OpenSSH is also available for Windows. The recommended way of running OpenSSH on Windows is through Cygwin.

3.1.2 VNC

Viewers:

- UltraVNC
- RealVNC Viewer
- RealVNC Viewer for Google Chrome
- TightVNC
- TigerVNC

3.2 Mac OS X

3.2.1 SSH

Mac OS X comes preinstalled with OpenSSH, the most popular implementation of the SSH protocol. The client can be run from the command-line and is simply called `ssh`.

Logging in

First, open your terminal emulator to start a shell. To connect to a specific machine, run:

```
ssh smithj@eosXX.cis.gvsu.edu
```

You should see a prompt open to *Bash*, the default shell on EOS. From here, you can run commands as you would inside a normal terminal emulator in an EOS desktop session. The commands will be run on the EOS machine to which you have connected.

Checking Host Fingerprints

When logging in to an EOS machine for the first time, you will see a message like this:

```
The authenticity of host 'eos01.cis.gvsu.edu (148.61.162.101)' can't be established.
RSA key fingerprint is 6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54.
Are you sure you want to continue connecting (yes/no)?
```

This is your SSH client requesting you to validate the identity of the machine to which you are connecting.

Each EOS machine has a so-called fingerprint, a series of characters which is used to verify its identity. To ensure that an attacker between your client and the actual EOS machine is not pretending to be an EOS machine, you must check that the machine's fingerprint matches the table below. Please report any mismatches to Ira Woodring immediately.

Host	Fingerprint
arch01	`7e:df:14:7b:92:65:5e:29:59:99:4a:17:d1:79:1f:96`
arch02	`b7:b0:fb:63:e5:be:25:b0:d9:2c:15:78:1b:cc:ca:6f`
arch03	`99:d3:7d:15:72:2b:80:4c:cb:56:8c:f9:d6:15:1e:38`
arch04	`fa:ae:f8:97:89:59:eb:91:fa:d0:d9:37:c1:f2:45:44`
arch05	`67:b0:39:03:b2:c2:1c:1a:72:80:dd:49:23:d8:19:c4`
arch06	`6e:fc:21:fb:35:f5:b3:87:52:37:d3:5f:0b:d3:b3:b5`
arch07	`7b:0d:0e:e0:51:c4:f7:bb:64:8a:14:e7:ab:66:de:34`
arch08	`e5:80:ac:98:22:6e:b5:41:26:18:27:c8:a5:22:52:42`

Continued on next page

Table 3.2 – continued from previous page

Host	Fingerprint
eos01	6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54
eos02	05:d1:d5:61:a4:c4:4e:ba:07:85:e2:18:8e:4b:7b:59
eos03	98:e4:c6:10:9a:63:e0:1d:63:ae:a8:27:c3:39:e4:ba
eos04	7c:b7:a2:65:74:c0:4f:52:ee:a2:2d:b8:12:38:9a:be
eos05	db:70:3d:04:44:d3:7f:a4:83:72:13:71:0c:be:ab:97
eos06	80:a9:e8:ea:b8:54:1c:4d:3b:9d:e2:c2:a7:d0:37:46
eos07	f4:15:32:2a:b5:18:e9:e6:51:8a:58:63:71:64:ba:8b
eos08	0d:0b:31:06:0a:6e:e6:b4:fb:48:0f:11:d6:7b:98:8a
eos09	6e:c0:20:4b:7d:f2:88:3d:df:71:ee:2e:63:2b:3c:33
eos10	be:fe:47:76:1d:5a:4b:96:fc:b9:1e:f2:42:fd:ed:41
eos11	be:47:a2:1c:6a:30:e7:98:ab:9a:1c:d3:4b:33:f2:57
eos12	15:21:3f:d2:b5:0b:04:06:29:2f:10:a7:04:3e:83:b7
eos13	31:5e:6f:ab:4a:7c:2f:3e:36:06:bd:e4:e0:a4:a3:22
eos14	87:af:45:1a:59:48:3d:57:66:a1:d6:6f:6e:a8:05:59
eos15	c2:b2:34:fa:8e:16:65:49:c5:ea:f4:70:c2:54:50:bc
eos16	57:71:e3:80:36:ff:25:d3:d1:c9:96:3d:f0:2d:42:a4
eos17	fd:3e:66:2a:39:08:b1:8f:c2:d7:e9:8b:59:40:a1:d9
eos18	a8:c5:e3:8f:1c:32:8e:72:c6:f4:dd:52:ad:23:63:31
eos19	d7:65:50:f2:23:b9:d7:58:76:9a:e2:5c:f3:1b:c0:11
eos20	a0:76:77:7a:2f:8f:cf:a6:f7:50:38:a4:36:12:4b:2d
eos21	23:3a:2b:42:b4:cf:f4:9e:c8:14:57:2f:73:8f:65:46
eos22	a5:fc:1e:65:bb:29:9b:ee:1c:4a:e9:46:41:a1:8e:cd
eos23	86:bc:19:fa:64:05:73:ef:81:78:cb:24:b5:f7:8c:fd
eos24	ec:5c:2f:a7:e9:86:9f:36:e6:85:a1:bc:54:04:ce:a5

Password-less Logins

It is often handy to be able to SSH into a host without having to type a password, for instance as part of a script. First, generate your public/private key pair with:

```
ssh-keygen
```

Accept the default values by pressing `Enter` at each prompt unless you know what you are doing. Once the keys have been generated, you can copy the public key over to the remote system by entering:

```
ssh smithj@eos01.cis.gvsu.edu 'umask u=rwx,go= && mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys' <
```

When you SSH into EOS now, you should be able to do so without having to provide a password:

```
ssh smithj@eos01.cis.gvsu.edu
```

Note: In this setup, we created our public/private key pair without a passphrase, which is less secure. If you would like to use a passphrase, please see Mark Hershberger's excellent guide to ssh-agent and Github's guide to SSH passphrases.

Tunnelling / Port Forwarding

The SSH protocol possesses a special feature which allows it to tunnel other protocols within itself. This is called tunnelling or port forwarding. SSH can forward local ports (allowing the local machine access to resources on the remote machine) and remote ports (allowing the remote machine access to resources on the local machine). Port

forwarding can be accomplished by passing arguments to the OpenSSH command-line client or editing the OpenSSH client configuration file.

Local port forwarding is the more used feature, and is explained in the following sections. Remote port forwarding is similar but is outside the scope of this guide.

Forwarding on the Command Line

Forwarding local ports on the command-line can be accomplished using the following syntax:

```
ssh -L local_port:remote_host:remote_port user@host
```

For example, to access a web server running on port 8000 on `eos01.cis.gvsu.edu` from your machine on port 5555, use the following command line:

```
ssh -L 5555:eos01.cis.gvsu.edu:8000 smithj@eos01.cis.gvsu.edu
```

You can actually test this by running this in the SSH prompt:

```
python -m SimpleHTTPServer
```

and opening http://localhost:5555/ in your local web browser.

The remote host which is hosting the resource need not be the EOS machine to which you are connecting with SSH. For example, to access the CIS web server through your SSH tunnel, you can run:

```
ssh -L 5678:cis.gvsu.edu:80 smithj@eos01.cis.gvsu.edu
```

and visit http://localhost:5678/ in your local web browser. The CIS home page should appear!

Forwarding in the Config File

The command-line works well for one-off tunnels, but for frequently established tunnels, it pays to alter the OpenSSH client configuration file. The OpenSSH client configuration resides on your local machine in the file `~/.ssh/config`. This is a file inside a hidden directory inside your home directory. The easiest way to open this file, creating it if it doesn't exist, is to run:

```
umask u=rwx,go= && mkdir -p ~/.ssh && touch ~/.ssh/config && open -t ~/.ssh/config
```

To establish the CIS web server forwarding shown in the last section, one could use the following configuration:

```
Host eoscisweb
HostName eos01.cis.gvsu.edu
User smithj
LocalForward 5678 cis.gvsu.edu:80
```

To use this host from the command line, simply type:

```
ssh eoscisweb
```

3.2.2 VNC

First, we need to create a tunnel in order to forward VNC through our SSH connection. The remote port to which we must connect depends on the desired resolution of the remote desktop. Select a desired resolution from the following table, and note the port to which it corresponds.

Port	Geometry
5900	1280x1024
5901	1024x768
5902	800x600
5903	640x480
5904	1440x900
5905	1280x800
5906	1152x864
5907	1680x1050
5908	1920x1200
5909	1280x768
5910	1360x768
5911	1400x1050
5912	1440x1000
5913	1024x600
5914	1600x900
5915	1920x1080
5916	1680x950

Replace `REMOTE_PORT` with the port that you have selected in the following command or file.

To create the tunnel, use the following command line:

```
ssh -L 5900:eosXX.cis.gvsu.edu:REMOTE_PORT smithj@eosXX.cis.gvsu.edu
```

Or the following configuration file:

```
Host eosvnc
HostName eosXX.cis.gvsu.edu
User smithj
LocalForward 5900 eosXX.cis.gvsu.edu:REMOTE_PORT
```

If you used the configuration file, run the following to create the tunnel:

```
ssh eosvnc
```

You are now ready to tunnel your VNC session.

The recommended VNC client for OS X is Chicken, which is free and open-source software. Visit the website to download, install as you would any other Mac OS X application, and open.

You will be prompted to create a new server. If not prompted, click *Connection → Open Connection...* from the menu bar. Double click "New Server" in the list on the left and rename it to "EOS", or create a new session with the + button if none exist. Leave the *Host* field as `localhost` or fill it in if missing. Leave the *Display or port* field at 0 or fill it in if missing.

Chicken has had some problems with the ZRLE encoding with our server. As this can cause a premature end to your session, our recommendation is to manually disable this encoding. To do this, first click the drop-down menu next to *Profile*, and click *Edit Conneciton Profiles....* The *Profile Manager* configuration window will open. In the bottom left, enter "EOS" into the field and click the + button. Now click the checkbox next to the ZRLE encoding to disable it for EOS sessions. Close the *Profile Manager* window.

Click *Connect* to begin your VNC session with EOS. To connect in the future, select *Connection → Open Connection...* from the menu, select your EOS configuration, and click *Connect*.

Note: Although Chicken offers an option to tunnel directly through SSH, we have not had luck using this option with our setup. We recommend sticking with the traditional SSH tunnel, as it is tested and works well.

Alternatives

Chicken is not the only VNC viewer available for Mac OS X. Some alternatives are:

- RealVNC — free and paid versions available
- RealVNC Viewer for Google Chrome — free Google Chrome extension
- JollysFastVNC — trial available
- TigerVNC — free and open source, command-line connection interface only
- Chicken of the VNC — older version of Chicken, not recommended

3.2.3 Advanced OpenSSH

OpenSSH can do much more than simply allow the user to establish connections with remote servers. If you use OpenSSH, there are a great many neat tricks available to you.

This section is based in large part on the Smylers SSH Productivity Tips blog post. Please visit this post for *even more SSH awesomeness!*

Hostname Aliases

It's useful not to have to type out the entire full-qualified domain names to EOS machines. What you might normally type would be something like this:

```
ssh smithj@eos02.cis.gvsu.edu
# or
ssh smithj@arch04.cis.gvsu.edu
```

By adding a section to the config file, this becomes easier. Add this to your `~/.ssh/config` as mentioned earlier:

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
```

With this, now you need only type:

```
ssh eos02
# or
ssh arch04
```

Shared Connections

There is a about a minute timeout between allowed successive connnections to each individual EOS machine. This can prove very annoying when establishing multiple SSH connections to use **scp** to copy files or opening multiple terminals (but see terminal multiplexing). One way to mitigate this annoyance is by using GVSU's VPN. Another way is to used SSH shared connections. These solutions are also not mutually exclusive.

Shared connections are established by creating a socket which multiplexes multiple connections. This socket is controlled by the `ControlMaster` and `ControlPath` keywords. The first connection is called the "master" and creates the socket. Subsequent connections use the already-created socket. This behavior can be automated by setting `ControlMaster` to the value `auto`. `ControlPath` specifies the path to the socket, with variables substituted as necessary. The following config amend the previous config to add connection sharing to EOS machines.

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
ControlMaster auto
#                        Host
#                          |Port
#                          |   | Username
#                          V   V   V
ControlPath /tmp/ssh_mux_%h_%p_%r
```

Connection sharing may be useful to enable for most hosts. However, it needs to be done with care because it typically conflicts with X forwarding and port forwarding.

Persistent Connections

It is often useful to keep connections open in the background even after the terminal has actually been closed. This is useful as it allows OpenSSH to reconnect to the server without re-establishing a connection. Turning this behavior on is trivially simple. Add the following line under the host for which you would like connections to persist:

```
# Persist connections for 2 hours.
ControlPersist 2h
```

For GitHub users, this is especially useful when using Git over SSH. Within this period, OpenSSH does not need to re-establish a connection to the Git server, which makes pushes and pulls much faster.

Multi-Hop Connections

Oftentimes a machine is only available when SSH'ing into another machine. For example, this is the case with the DEN's Okami server, used in CIS 677 High-Performance Computing. In addition, Okami's SSH server is only available on a non-standard port. This typically results in the user going through this process:

```
local$ ssh smithj@eos01.cis.gvsu.edu
eos01$ ssh -p 43022 okami
okami$ # Finally here!
```

This is annoying and unnecessary. By using the `ProxyCommand` keyword in our config file, we can automate this process:

```
# DEN Okami
Host okami
User smithj
Port 43022
ProxyCommand ssh eos01 -W %h:%p
```

The `-W` flag allows us to hop through the first host to the host and port specified by the variables (`okami:43022`). Note that the use of `eos01` here requires presence of the aliases set up in Hostname Aliases.

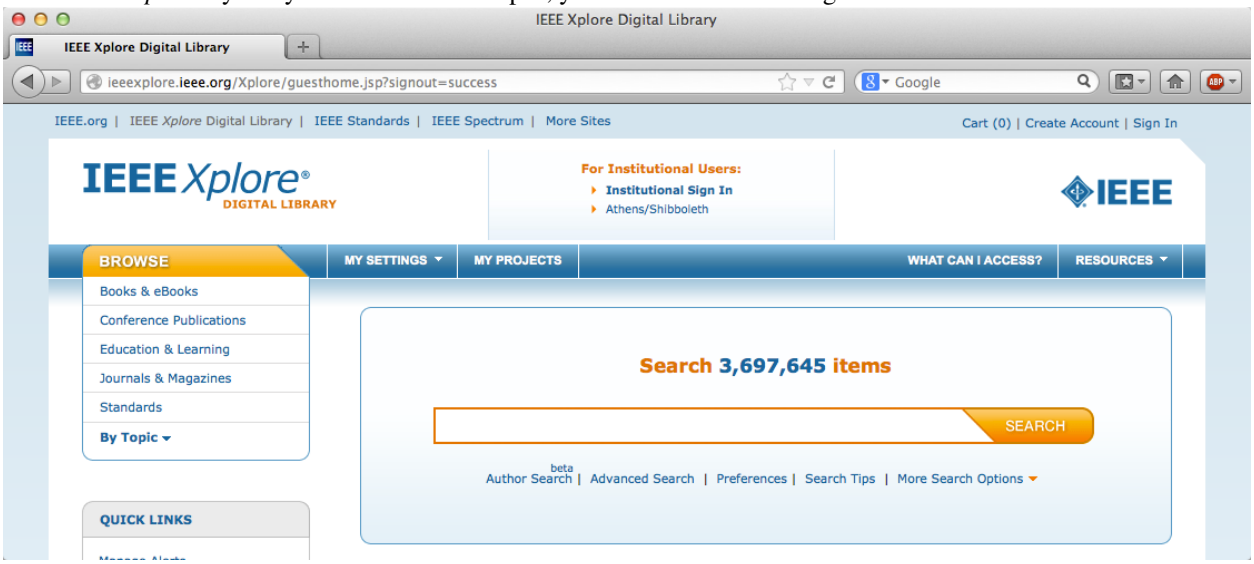
The process has now been simplified to:

```
local$ ssh okami
okami$ # Yay! Easy!
```

Using SSH as a Proxy

It is also possible to use SSH as a proxy for all network traffic. This can be useful if there are resources available from the SSH server that are not available from the local machine.

An example of such a resource is the IEEE Xplore Digital Library, which contains technical articles targeted at computer scientists and engineers. GVSU subscribes to this library, but access to the subscription is only available while *on campus*. If you try to access it off campus, you will see the following:



By using a proxy through the EOS machines, we can transparently access the IEEE library as if we were on campus.

OpenSSH support the SOCKS protocol for proxying. Activating the SOCKS feature is accomplished with the `-D` flag like so:

```
ssh -D 5555 eos01
```

This establishes a SOCKS proxy with EOS01 served up on the local machine on port 5555. Now we must configure our operating system or browser to use this proxy.

On Mac OS X, configuring your system to use our SOCKS proxy is quite simple. First, open *System Preferences*. From here, choose *Network* → *Advanced...* → *Proxies* → *SOCKS Proxy*. Under the label *SOCKS Proxy Server*, enter `localhost` and `5555` to match the port passed to the `-D` flag. This can be any port as long as these numbers match. Check the box next to *SOCKS Proxy*, then click *OK* and *Apply* to turn on the proxy.

If you again try to access the IEEE Xplore Digital Library, you should see the following:

You have now successfully used OpenSSH to establish a SOCKS proxy!

Warning: By using a SOCKS proxy, *all* your network traffic is sent through the proxy. This has two implications:

- Network access will likely be slower.
- GVSU will be able to monitor your traffic as they do when you are on campus.

Please keep this in mind when using the proxy feature.

Example

For an example OpenSSH configuration file, see Sean's SSH config.

3.3 GNU/Linux

3.3.1 SSH

The most popular implementation of the SSH protocol on GNU/Linux is OpenSSH. The SSH client can be run from the command-line and is simply called **ssh**.

Many GNU/Linux distributions come with OpenSSH pre-installed. If your GNU/Linux distribution does not have it installed by default, please install it with your package manager. You should not install this software from scratch or on your own.

On Debian-based systems (Ubuntu, Linux Mint, and friends), run the following command:

```
sudo apt-get install openssh-client
```

On Red Hat-based systems (Fedora, CentOS, RHEL, and friends), run the following command:

```
sudo yum install openssh-clients
```

With other distributions (Arch, etc.), you are on your own.

Logging in

First, open your terminal emulator to start a shell. To connect to a specific machine, run:

```
ssh smithj@eosXX.cis.gvsu.edu
```

You should see a prompt open to *Bash*, the default shell on EOS. From here, you can run commands as you would inside a normal terminal emulator in an EOS desktop session. The commands will be run on the EOS machine to which you have connected.

Checking Host Fingerprints

When logging in to an EOS machine for the first time, you will see a message like this:

```
The authenticity of host 'eos01.cis.gvsu.edu (148.61.162.101)' can't be established.
RSA key fingerprint is 6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54.
Are you sure you want to continue connecting (yes/no)?
```

This is your SSH client requesting you to validate the identity of the machine to which you are connecting.

Each EOS machine has a so-called fingerprint, a series of characters which is used to verify its identity. To ensure that an attacker between your client and the actual EOS machine is not pretending to be an EOS machine, you must check that the machine's fingerprint matches the table below. Please report any mismatches to Ira Woodring immediately.

Host	Fingerprint
arch01	7e:df:14:7b:92:65:5e:29:59:99:4a:17:d1:79:1f:96
arch02	b7:b0:fb:63:e5:be:25:b0:d9:2c:15:78:1b:cc:ca:6f
arch03	99:d3:7d:15:72:2b:80:4c:cb:56:8c:f9:d6:15:1e:38
arch04	fa:ae:f8:97:89:59:eb:91:fa:d0:d9:37:c1:f2:45:44
arch05	67:b0:39:03:b2:c2:1c:1a:72:80:dd:49:23:d8:19:c4
arch06	6e:fc:21:fb:35:f5:b3:87:52:37:d3:5f:0b:d3:b3:b5
arch07	7b:0d:0e:e0:51:c4:f7:bb:64:8a:14:e7:ab:66:de:34
arch08	e5:80:ac:98:22:6e:b5:41:26:18:27:c8:a5:22:52:42
eos01	6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54
eos02	05:d1:d5:61:a4:c4:4e:ba:07:85:e2:18:8e:4b:7b:59
eos03	98:e4:c6:10:9a:63:e0:1d:63:ae:a8:27:c3:39:e4:ba
eos04	7c:b7:a2:65:74:c0:4f:52:ee:a2:2d:b8:12:38:9a:be
eos05	db:70:3d:04:44:d3:7f:a4:83:72:13:71:0c:be:ab:97
eos06	80:a9:e8:ea:b8:54:1c:4d:3b:9d:e2:c2:a7:d0:37:46
eos07	f4:15:32:2a:b5:18:e9:e6:51:8a:58:63:71:64:ba:8b
eos08	0d:0b:31:06:0a:6e:e6:b4:fb:48:0f:11:d6:7b:98:8a
eos09	6e:c0:20:4b:7d:f2:88:3d:df:71:ee:2e:63:2b:3c:33
eos10	be:fe:47:76:1d:5a:4b:96:fc:b9:1e:f2:42:fd:ed:41
eos11	be:47:a2:1c:6a:30:e7:98:ab:9a:1c:d3:4b:33:f2:57
eos12	15:21:3f:d2:b5:0b:04:06:29:2f:10:a7:04:3e:83:b7
eos13	31:5e:6f:ab:4a:7c:2f:3e:36:06:bd:e4:e0:a4:a3:22
eos14	87:af:45:1a:59:48:3d:57:66:a1:d6:6f:6e:a8:05:59
eos15	c2:b2:34:fa:8e:16:65:49:c5:ea:f4:70:c2:54:50:bc
eos16	57:71:e3:80:36:ff:25:d3:d1:c9:96:3d:f0:2d:42:a4
eos17	fd:3e:66:2a:39:08:b1:8f:c2:d7:e9:8b:59:40:a1:d9
eos18	a8:c5:e3:8f:1c:32:8e:72:c6:f4:dd:52:ad:23:63:31
eos19	d7:65:50:f2:23:b9:d7:58:76:9a:e2:5c:f3:1b:c0:11
eos20	a0:76:77:7a:2f:8f:cf:a6:f7:50:38:a4:36:12:4b:2d
eos21	23:3a:2b:42:b4:cf:f4:9e:c8:14:57:2f:73:8f:65:46

Continued on next page

Table 3.3 – continued from previous page

Host	Fingerprint
eos22	a5:fc:1e:65:bb:29:9b:ee:1c:4a:e9:46:41:a1:8e:cd
eos23	86:bc:19:fa:64:05:73:ef:81:78:cb:24:b5:f7:8c:fd
eos24	ec:5c:2f:a7:e9:86:9f:36:e6:85:a1:bc:54:04:ce:a5

Password-less Logins

It is often handy to be able to SSH into a host without having to type a password, for instance as part of a script. First, generate your public/private key pair with:

```
ssh-keygen
```

Accept the default values by pressing `Enter` at each prompt unless you know what you are doing. Once the keys have been generated, you can copy the public key over to the remote system by entering:

```
ssh-copy-id smithj@eos01.cis.gvsu.edu
```

When you SSH into EOS now, you should be able to do so without having to provide a password:

```
ssh smithj@eos01.cis.gvsu.edu
```

Note: In this setup, we created our public/private key pair without a passphrase, which is less secure. If you would like to use a passphrase, please see Mark Hershberger's excellent guide to ssh-agent and Github's guide to SSH passphrases.

Tunnelling / Port Forwarding

The SSH protocol possesses a special feature which allows it to tunnel other protocols within itself. This is called tunnelling or port forwarding. SSH can forward local ports (allowing the local machine access to resources on the remote machine) and remote ports (allowing the remote machine access to resources on the local machine). Port forwarding can be accomplished by passing arguments to the OpenSSH command-line client or editing the OpenSSH client configuration file.

Local port forwarding is the more used feature, and is explained in the following sections. Remote port forwarding is similar but is outside the scope of this guide.

Forwarding on the Command Line

Forwarding local ports on the command-line can be accomplished using the following syntax:

```
ssh -L local_port:remote_host:remote_port user@host
```

For example, to access a web server running on port 8000 on `eos01.cis.gvsu.edu` from your machine on port 5555, use the following command line:

```
ssh -L 5555:eos01.cis.gvsu.edu:8000 smithj@eos01.cis.gvsu.edu
```

You can actually test this by running this in the SSH prompt:

```
python -m SimpleHTTPServer
```

and opening http://localhost:5555/ in your local web browser.

The remote host which is hosting the resource need not be the EOS machine to which you are connecting with SSH. For example, to access the CIS web server through your SSH tunnel, you can run:

```
ssh -L 5678:cis.gvsu.edu:80 smithj@eos01.cis.gvsu.edu
```

and visit http://localhost:5678/ in your local web browser. The CIS home page should appear!

Forwarding in the Config File

The command-line works well for one-off tunnels, but for frequently established tunnels, it pays to alter the OpenSSH client configuration file. The OpenSSH client configuration resides on your local machine in the file `~/.ssh/config`. This is a file inside a hidden directory inside your home directory. The easiest way to open this file, creating it if it doesn't exist, is to run:

```
umask u=rwx,go= && mkdir -p ~/.ssh && touch ~/.ssh/config && gedit ~/.ssh/config
```

To establish the CIS web server forwarding shown in the last section, one could use the following configuration:

```
Host eoscisweb
HostName eos01.cis.gvsu.edu
User smithj
LocalForward 5678 cis.gvsu.edu:80
```

To use this host from the command line, simply type:

```
ssh eoscisweb
```

3.3.2 VNC

First, we need to create a tunnel in order to forward VNC through our SSH connection. The remote port to which we must connect depends on the desired resolution of the remote desktop. Select a desired resolution from the following table, and note the port to which it corresponds.

Port	Geometry
5900	1280x1024
5901	1024x768
5902	800x600
5903	640x480
5904	1440x900
5905	1280x800
5906	1152x864
5907	1680x1050
5908	1920x1200
5909	1280x768
5910	1360x768
5911	1400x1050
5912	1440x1000
5913	1024x600
5914	1600x900
5915	1920x1080
5916	1680x950

Replace `REMOTE_PORT` with the port that you have selected in the following command or file.

To create the tunnel, use the following command line:

```
ssh -L 5900:eosXX.cis.gvsu.edu:REMOTE_PORT smithj@eosXX.cis.gvsu.edu
```

Or the following configuration file:

```
Host eosvnc
HostName eosXX.cis.gvsu.edu
User smithj
LocalForward 5900 eosXX.cis.gvsu.edu:REMOTE_PORT
```

If you used the configuration file, run the following to create the tunnel:

```
ssh eosvnc
```

You are now ready to tunnel your VNC session. There are a number of capable VNC clients for GNU/Linux. Which one you choose depends on desktop environment and personal preference. For KDE, KRDC (KDE Remote Desktop Client) is the standard application. For GNOME, the standard remote desktop viewer is called Vinagre. If you would also like to use *RDP with Winserv*, you may want to try Reminna, a third-party free and open-source remote desktop viewer. You should install these from your package manager.

TigerVNC and RealVNC Viewer are also available for GNU/Linux. RealVNC also offers RealVNC Viewer for Google Chrome, a free viewer which runs inside the Google Chrome browser.

Operation of each of these applications is similar. For the host, enter in the hostname of the EOS machine to which you have SSH'ed. If a display is requested, enter `0`. If a port is requested, enter `5900` (these mean the same thing). If the viewer offers support for multiple protocols, make sure you select "VNC".

3.3.3 Advanced OpenSSH

OpenSSH can do much more than simply allow the user to establish connections with remote servers. If you use OpenSSH, there are a great many neat tricks available to you.

This section is based in large part on the Smylers SSH Productivity Tips blog post. Please visit this post for *even more SSH awesomeness!*

Hostname Aliases

It's useful not to have to type out the entire full-qualified domain names to EOS machines. What you might normally type would be something like this:

```
ssh smithj@eos02.cis.gvsu.edu
# or
ssh smithj@arch04.cis.gvsu.edu
```

By adding a section to the config file, this becomes easier. Add this to your `~/.ssh/config` as mentioned earlier:

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
```

With this, now you need only type:

```
ssh eos02
# or
ssh arch04
```

Shared Connections

There is a about a minute timeout between allowed successive connnections to each individual EOS machine. This can prove very annoying when establishing multiple SSH connections to use **scp** to copy files or opening multiple terminals (but see terminal multiplexing). One way to mitigate this annoyance is by using GVSU's VPN. Another way is to used SSH shared connections. These solutions are also not mutually exclusive.

Shared connections are established by creating a socket which multiplexes multiple connections. This socket is controlled by the `ControlMaster` and `ControlPath` keywords. The first connection is called the "master" and creates the socket. Subsequent connections use the already-created socket. This behavior can be automated by setting `ControlMaster` to the value `auto`. `ControlPath` specifies the path to the socket, with variables substituted as necessary. The following config amend the previous config to add connection sharing to EOS machines.

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
ControlMaster auto
#                          Host
#                            |Port
#                            |   | Username
#                            V   V   V
ControlPath /tmp/ssh_mux_%h_%p_%r
```

Connection sharing may be useful to enable for most hosts. However, it needs to be done with care because it typically conflicts with X forwarding and port forwarding.

Persistent Connections

It is often useful to keep connections open in the background even after the terminal has actually been closed. This is useful as it allows OpenSSH to reconnect to the server without re-establishing a connection. Turning this behavior on is trivially simple. Add the following line under the host for which you would like connections to persist:

```
# Persist connections for 2 hours.
ControlPersist 2h
```

For GitHub users, this is especially useful when using Git over SSH. Within this period, OpenSSH does not need to re-establish a connection to the Git server, which makes pushes and pulls much faster.

Multi-Hop Connections

Oftentimes a machine is only available when SSH'ing into another machine. For example, this is the case with the DEN's Okami server, used in CIS 677 High-Performance Computing. In addition, Okami's SSH server is only available on a non-standard port. This typically results in the user going through this process:

```
local$ ssh smithj@eos01.cis.gvsu.edu
eos01$ ssh -p 43022 okami
okami$ # Finally here!
```

This is annoying and unnecessary. By using the `ProxyCommand` keyword in our config file, we can automate this process:

```
# DEN Okami
Host okami
User smithj
```

```
Port 43022
ProxyCommand ssh eos01 -W %h:%p
```

The `-W` flag allows us to hop through the first host to the host and port specified by the variables (`okami:43022`). Note that the use of `eos01` here requires presence of the aliases set up in Hostname Aliases.

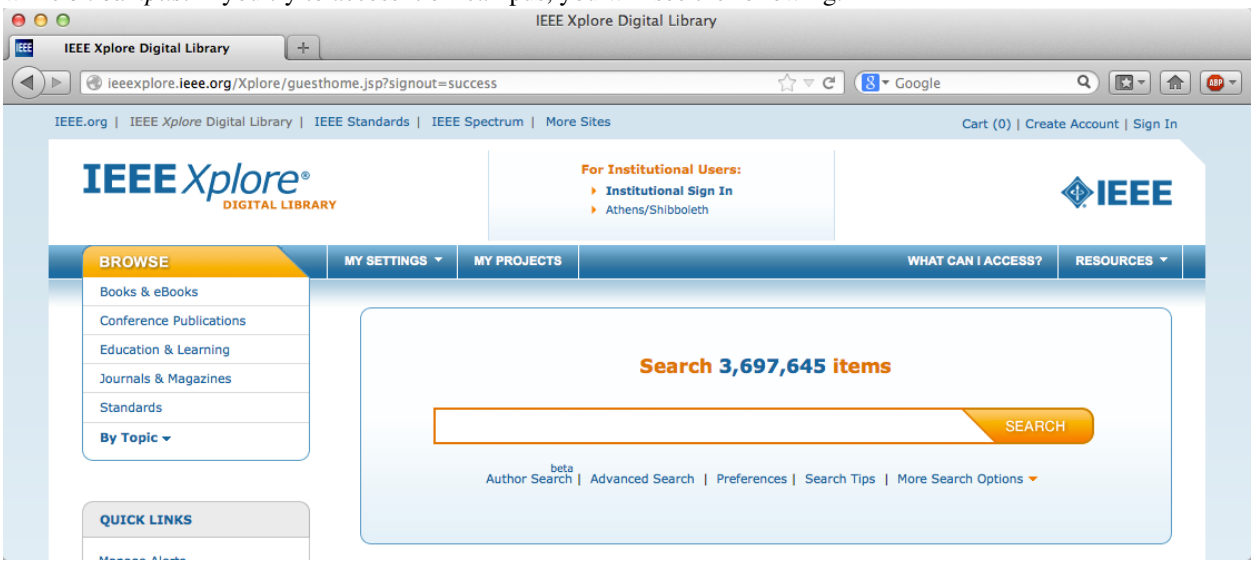
The process has now been simplified to:

```
local$ ssh okami
okami$ # Yay! Easy!
```

Using SSH as a Proxy

It is also possible to use SSH as a proxy for all network traffic. This can be useful if there are resources available from the SSH server that are not available from the local machine.

An example of such a resource is the IEEE Xplore Digital Library, which contains technical articles targeted at computer scientists and engineers. GVSU subscribes to this library, but access to the subscription is only available while *on campus*. If you try to access it off campus, you will see the following:



By using a proxy through the EOS machines, we can transparently access the IEEE library as if we were on campus.

OpenSSH support the SOCKS protocol for proxying. Activating the SOCKS feature is accomplished with the `-D` flag like so:

```
ssh -D 5555 eos01
```

This establishes a SOCKS proxy with EOS01 served up on the local machine on port 5555. Now we must configure our operating system or browser to use this proxy.

TODO: Add configuration for GNU/Linux.

If you again try to access the IEEE Xplore Digital Library, you should see the following:

You have now successfully used OpenSSH to establish a SOCKS proxy!

Warning: By using a SOCKS proxy, *all* your network traffic is sent through the proxy. This has two implications:

- Network access will likely be slower.
- GVSU will be able to monitor your traffic as they do when you are on campus.

Please keep this in mind when using the proxy feature.

Example

For an example OpenSSH configuration file, see Sean's SSH config.

CHAPTER

FOUR

SHELLS

A shell is a textual user interface that allows you to control the operating system and run programs. The EOS Labs have a variety of different shell options from which you can choose. The default is Bash, but you may use any of the following provided shells.

4.1 Bash

GNU Bash (Bourne Again SHell) is setup for you automatically. It is the default shell for the GNU Project and is widely the standard shell for most distributions of GNU/Linux.

4.2 Tcsh

Tcsh (TENEX C SHell) is a C style shell that adds a few extra features such as command line completion.

4.3 Kornshell

Kornshell is a shell developed at AT&T Bell Laboratories. It has features of both the Bash and Korn shells, as well as additional features.

CHAPTER

FIVE

WINSERV

Part of the department infrastructure is a Windows Server installation, called Winserv. Accounts for this machine are given as necessary — for example, when taking a course that involves projects which require the Windows platform. This server is named `winserv.cis.gvsu` and can be accessed via the Remote Desktop Protocol (RDP). The following are methods for accessing Winserv from various RDP clients.

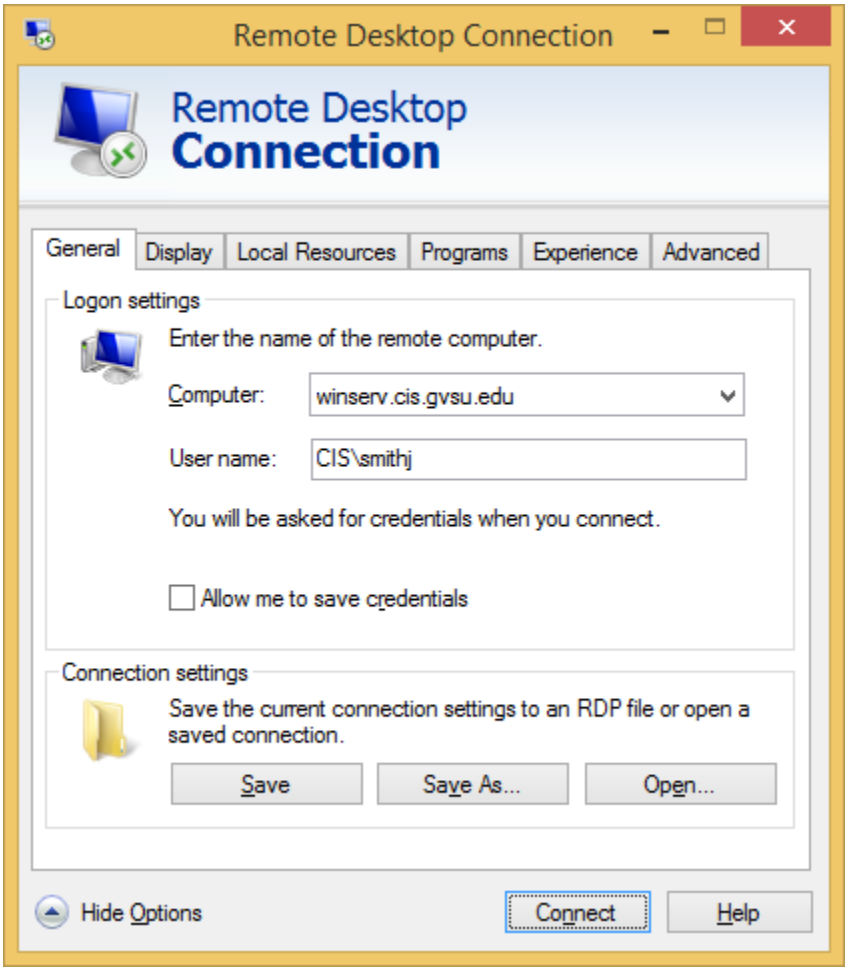
5.1 Common Settings

Regardless of the RDP client or platform you use, please remember a few things:

- The machine's IP address may change; use the DNS name instead.
- If outside of the EOS network, you will need to use the fully qualified domain name, `winserv.cis.gvsu.edu`. While on the EOS network, you can simply use `winserv`.
- You must login to the CIS domain. For instance, if your username is `smithj`, your login would be `CIS\smithj`. These may be specified together with the backslash or separately depending on your client.
- Our certificate is self-signed. You may want to instruct your client to save this information, or you will have to accept a security warning each time you login.

5.2 Microsoft Windows

All versions of Windows have the built-in Microsoft Terminal Services Client. You can find this program by searching for **mstsc** or *Remote Desktop Connection* in the Start Menu. Here is a sample configuration for Winserv:

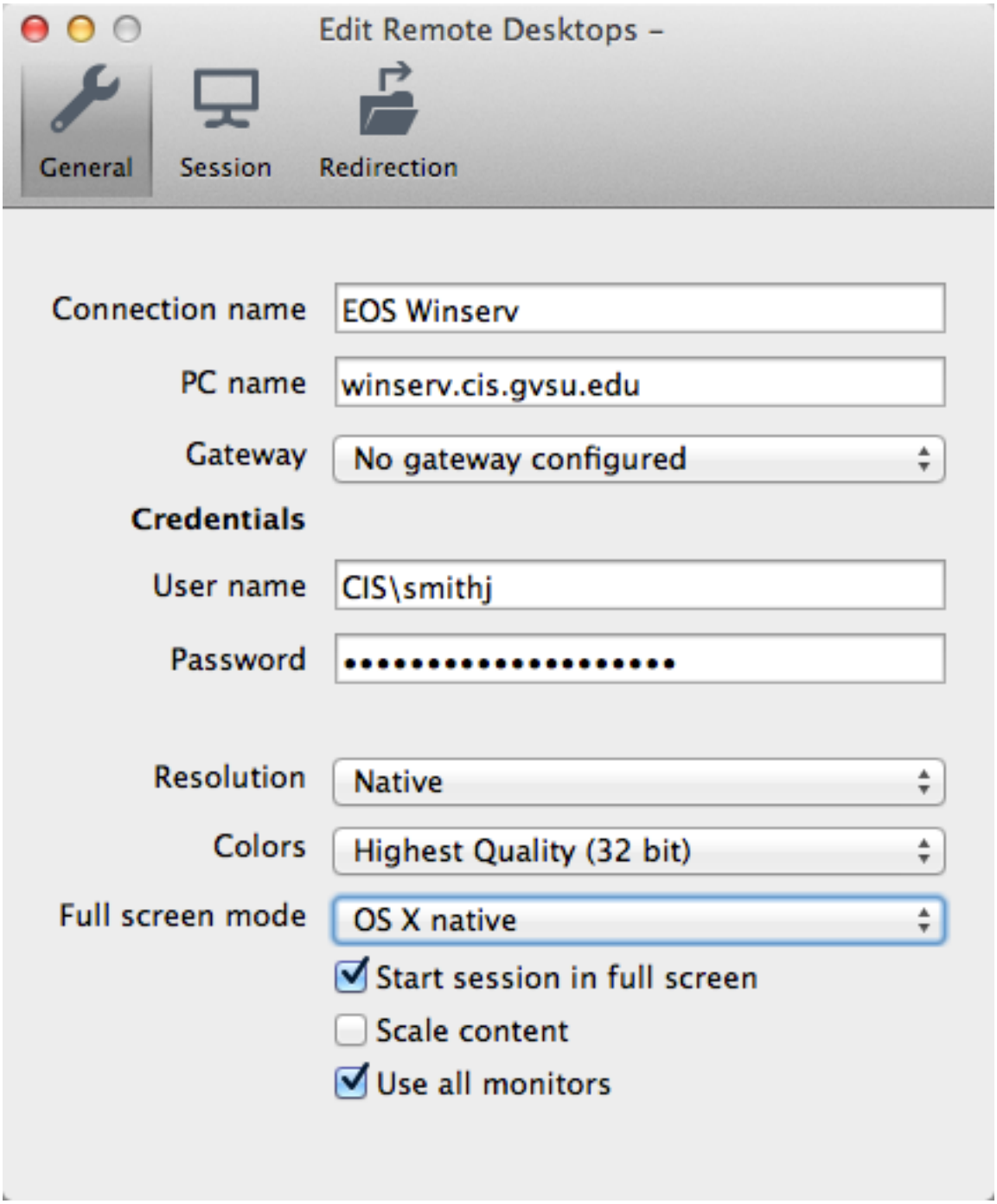


5.3 Mac OS X

Multiple options exist for Mac OS X, including the official Microsoft client and an open-source client called CoRD. Both work well and provide a similar set of features, so it is up to you which one you'd like to use. See here for a comparison of three different options including the two just mentioned.

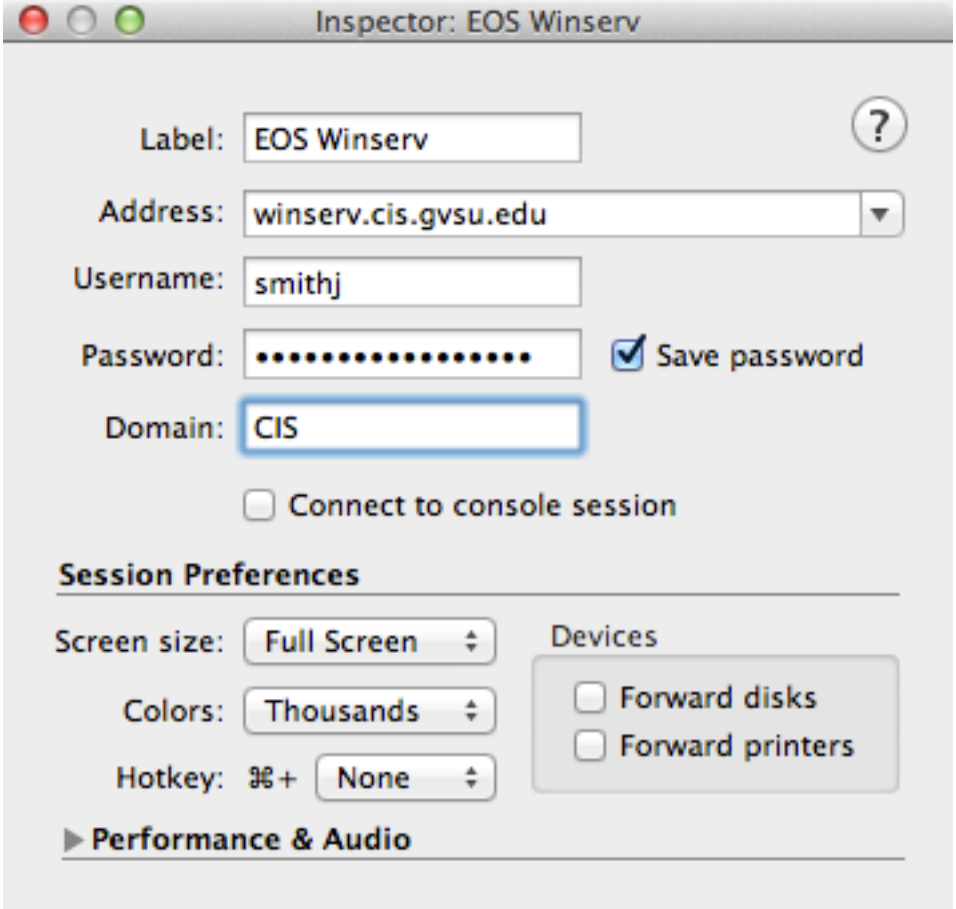
5.3.1 Microsoft Remote Desktop

This application is available as Microsoft Remote Desktop in the Mac App Store. *Do not* attempt to download Microsoft Remote Desktop Connection Client for Mac 2.1.1, as it does not work for more recent versions of Mac OS X. You do not need to configure a gateway in order to use this client with Winserv. Here is a sample configuration:



5.3.2 CoRD

CoRD is an open-source rdesktop-based RDP implementation for Mac OS X. Although their website states that CoRD "doesn't seem to work on 10.9 Mavericks", we have had no issues with the latest version. Here is a sample configuration for Winserv:



5.4 GNU/Linux

GNU/Linux systems usually have the rdesktop command-line tool in their repositories. If not, it can most likely be built from source. rdesktop provides a geometry flag that accepts both screen percentages or resolutions from the command-line to help adjust the client to an appropriate size. For instance, to allow the client to take up 90% of your screen:

```
rdekstop winserv.cis.gvsu.edu -g 90%
```

Alternatively, to force a resolution of 1024x768 pixels:

```
rdekstop winserv.cis.gvsu.edu -g 1024x768
```

For a graphical RDP client that can also be used for *VNC*, check out Remmina.

CHAPTER

SIX

DATABASES

Learning to setup and maintain a database is essential to any Computer Science curriculum. The EOS infrastructure supports multiple database variants depending on the your needs. To save space and setup time database accounts are only given to those who need them for a course or a specific project. Contact your professor and Ira Woodring if you need a database account for any reason.

6.1 MySQL

MySQL is a powerful open-source database. To access it via the command-line, login to an EOS/Arch machine and enter:

```
mysql -u smithj -p -h cis.gvsu.edu
```

This attempts to log you in with the provided username, using a password, to the host cis.gvsu.edu (our MySQL server).

6.2 Oracle

Oracle is a very powerful and complex enterprise quality system. Once you have been granted access, you can access it with the command:

```
sqlplus smithj@orcl
```

Please note that when you change your password for Oracle that you must not use the `@` character. Oracle will accept this but you will be unable to login.

6.3 Oracle APEX

Oracle also provides the APEX system for web based database development. An APEX account is separate from the normal Oracle account; a password for one will not work for the other. You may login to APEX once you have been granted access by opening your web browser to the URL

http://dbserv.cis.gvsu.edu:5560/apex

You will need to provide a workspace name, username, and password. If your username is `smithj`, a sample login would be

Workspace smithj_ws

Username smithj

Password **********

Do not attempt to use the *Reset Password* feature on the APEX homepage; it has never worked properly. If you attempt to use it you will be unable to login until a system administrator can delete and recreate your account.

6.4 MSSQL

Microsoft also provides an enterprise quality database server that we provide. Microsoft's database is called MSSQL. We host MSSQL on the Winserv machine, and accounts are granted when needed.

6.5 SQLite

The SQLite system is a relational database that can exist within your home directory. SQLite is different from the above mentioned databases in that it does not operate as a client/server set of processes, but instead can be linked to the application being programmed. As many databases as need be created (within storage limits) can be created by you, as each database is merely a separate file on the filesystem.

Outside of a programming context, SQLite can be accessed from the commandline with:

```
sqlite3
```

This will provide you with a command-line interface from which you can work using SQL statements.

6.6 Remote Database Connections

It is often advantageous for programs to connect their programs to databases to do work. There are a variety of ways to accomplish this task, and many are language specific. It is of note though that our databases are not accessible from outside of our network due to firewall restrictions. However, programs running from within the EOS infrastructure can make connections to databases.

CHAPTER

SEVEN

INTEGRATED DEVELOPMENT ENVIRONMENTS

Programming is often aided by the use of an integrated development environment (IDE). The EOS system has several IDEs to assist in programming tasks.

7.1 BlueJ

BlueJ is an IDE designed with learning in mind. While it is not the best choice if you are an advanced user or have a large project, it is one of the best IDEs for learning to program in Java. BlueJ's object inspector makes it quite easy for you to examine objects as they progress through a software instance's life cycle. BlueJ may be accessed by typing **bluej** from the command line.

7.2 Eclipse

Eclipse is a powerful, extensible, free and open-source IDE that can be used with a large number of languages. By default, Eclipse on EOS supports Java, PHP, Python, C, C++, and Android. Support for other languages or projects can be easily added if needed. Eclipse may be accessed by typing **eclipse** from the command line.

7.3 IntelliJ

For those that don't like Eclipse, we also provide the IntelliJ Java IDE. IntelliJ is a professional quality IDE and rather expensive to purchase. IntelliJ may be accessed by typing **idea.sh** from the command line.

CHAPTER

EIGHT

CONTRIBUTING TO MASTERING EOS

This manual is a living document. It is maintained by its authors, but students (e.g., you!) are encouraged to contribute to the guide. The manual is written by GVSU CIS students for GVSU CIS students.

The source code for Mastering EOS is hosted on GitHub and is available to the general public. GitHub is a popular site for hosting code repositories using Git, a popular version control system.

The easiest way to to contribute is by reporting an issue or requesting a section by filing an issue with the GitHub issue tracker. One of the authors should respond to your issue and give feedback.

The Mastering EOS manual is written using Sphinx, an excellent multi-target documentation generator. The documentation is written in the markup language reStructuredText. The poster is written in LaTeX using Beamer and beamerposter, although contribution to that is not necessarily as useful for obvious reasons. All of our tooling is written in Python. Intimate familiarity with these tools is not necessary for contribution.

The source code for Mastering EOS is hosted on GitHub. If you are familiar with the GitHub contribution process, contributing to Mastering EOS is exactly the same. If not, read on.

For those unfamiliar with Git and GitHub, contributing to Mastering EOS may seem rather involved. However, by following the steps outlined, it should be rather straightforward. We request that you first give it a try on your own, but failing that, contact the authors. We want to see your contributions!

To contribute to Mastering EOS, follow these steps:

- Read GitHub's guide to Contributing to a Project.
- Set Up Git if you have not already done so.
- Fork the Mastering EOS repository and make your changes.
- Create a pull request to the original repo containing your changes.

An author should respond to your pull request, and with any luck, your changes should go live! We look forward to seeing your contributions!

8.1 Writing Style

When writing technical documentation, it is important to follow a consistent writing style. Within this manual, we attempt to follow the OpenStack writing conventions. Their documentation presents a great summary of how to write coherent technical documentation.

Please also see the following links for writing technical documentation:

- ACS Distance Education Guidelines for Technical Writing
- Novell Open Source Documentation Style Quick Start
- BlueBream Documentation Guidelines

- Description of Imperative Mood on Wikipedia