Song
Spring 2015

CS 161
Computer Security

Project 3

Due: May 8 11:59pm, 11:59PM

Background

Your unsung-but-heroic efforts a month ago lead to the miraculous defeat of the evil Calnet! Freedom reigns across Caltopia ...

... Or does it?

Walking home from a spirited late-night victory celebration with your Birkland colleagues, you find your mind uneasy. In the wake of Calnet's destruction, Governor Sylvester Stalloon has called for an election to determine Caltopia's future. Surely this is the right step forward, yet ... Something is out of place. Something is wrong. Your security spider-sense tingles!

Your phone vibrates with an incoming text. But whoa, you have zero bars—how did that happen? Making a mental note to dump AT&T as soon as your contract expires, you check out the short message: `Vital u act or with this election we lose all. Stalloony's gonna steal it. Access and leverage his comm w/ gov-of-caltopia.info. Cripple repressive sw + steal email cred. + expose fraud = profit. --Neo`

Your head spins with questions and possibilities. Could such a heinous plot really be under way? Couldn't you just have a break from all this hassle until the end of the semester?

And how should you even begin? Can you really go up against the very top of the government all alone by yourself?

Your phone vibrates again. `Also, I suggest you work in team of 2. --Neo`

Getting Started

Haunted by the prospect of a stolen election, and daunted by the fate of Caltopia resting entirely in your hands, you find yourself sleepless and your mind racing. Throughout the night, texts come in, building out the picture and providing snippets of guidance.

First, Neo points you at an obscure file on a Berkeley server that turns out to supply you with a VM tailored for enabling you to complete each step of the task ahead.

Software Setup

You can run and investigate the VM on your own computer. You will need the following software:

- VirtualBox[1], the virtualization server.
- Your favorite SSH client.[2]

Start VirtualBox and go to File → Preferences → Network. Make sure there is a network adapter listed under "Host-only Networks" named `vboxnet0`. [3] If the adapter list is empty, click the plus on the right side to add a new interface. Confirm with OK.

Neo placed the VM image at http://cs.berkeley.edu/~cthompson/cs161s14/nethack_sp14.ova.[4] Download it and import it via File → Import Appliance.

Once you have imported the VM, you'll need to point the CDROM at a Linux LiveCD, which you can find at http://preview.tinyurl.com/xubuntutorrent (Torrent) or http://preview.tinyurl.com/xubuntuiso (HTTP). To set up the CDROM open VirtualBox and select the VM. Then click on Settings → Storage. Under Controller: IDE select Empty. Now click on the little CD icon on the right. Select Choose a virtual disk and point it to the Linux image you've downloaded. NOTE: If you are using Windows you may need to disable the Serial Port. To do this click on Settings → Ports and uncheck "Enable Serial Port." You may need to do this for Port 1 and Port 2.

Pro Tip: It may be useful to make snapshots of your virtual machine in case you make mistakes or break anything. Before you turn on your VM for the first time, select it and then click Snapshots, and then press the camera icon to create a new snapshot (you can name this one "pre login"). For example, if you enter your logins incorrectly, this will save you from having to re-import the VM.

All of the network programs will run inside of this VM. The image is a bare-bones Xubuntu Linux desktop installation on a 32-bit Intel architecture. The first time you boot the image, you have to enter your class accounts in the format `cs161-`x_1x_2`,cs161-`x_3x_4, where $x_1, \ldots, x_4$ are the letters of your class accounts. You need to list the accounts in alphabetical order, *with no spaces in between*. For example, if a student with class account `cs161-we` teams with a student with class account `cs161-vv`, then you would enter the string "`cs161-vv,cs161-we`".[5]

[1] VirtualBox is available at https://www.virtualbox.org, or from your package manager in Linux. Neo says the VM has successfully worked with versions 4.1.18, 4.2.6, and the latest 4.3.10.

[2] On Windows, Neo recommends PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/.

[3] On Windows, the interface may have a different, much longer name.

[4] The SHA1 hash of the file is a5e3d191ef877fb36062a49d30b0362bc37d9317. You can calculate the SHA1 hash of a file using `shasum` or `sha1sum` on most *nix machines.

[5] If you want to do some initial exploration by yourself before you've finalized your team, you can start off using just your class account for this configuration step. Once you have your team in place, you'll need to start again with a clean VM image configured as mentioned here. Any solution secrets you've learned for your private VM image will be different from your team's final secrets. This reconfiguration process should go quickly once you've developed solutions the first time.

Memory management. The Virtual Machine comes pre-configured with a 1GB memory limit. This is on the low side, and we recommend increasing it to at least 2GB, or as much memory as you can afford. To do this, go to Settings → System and adjust the Base memory slider. You can experience unexpected crashes if the VM runs out of memory. To conserve memory, when operating within your VM:

- don't surf the web (except where necessary);
- don't browse "heavy" websites (such as `nytimes.com`);
- and definitely don't watch any videos.

Save early, save often.

Instructional computing. (NOTE: If you use your own computer to run the project, skip this section.) Certain computers in the instructional labs have VirtualBox pre-installed and pre-configured. We've tested this on the Hive machines. The virtual machines for this project are somewhat large, so you'll need to make a a temp directory. You need to set this up before importing your VM, using the following steps:

1. SSH to `cory.eecs.berkeley.edu` and log in using your instructional account.
2. Run `/share/b/bin/mkhometmpdir` to create your personal directory. This creates a folder `/home/tmp/<accountname>`. Logout.
3. Start VirtualBox on one of the Hive machines.
4. Go File → Preferences → General. Change the option Default Machine Folder to `/home/tmp/cs161-XX` (your account name).

You can get more information about tmp folders in `/home/tmp/README.txt`.

SSL Tools

The OpenSSL Project develops the `libcrypto` and `libssl` libraries, and the accompanying `openssl` tool, which together provide full support for the latest SSL/TLS protocols. Important tools you should familiarize yourself with include:

- `openssl genrsa`: This tool allows you to generate public/private keypairs. This keypair can be used for both authentication and encryption. As indicated by its name, this tool generates keys for use with the RSA algorithm. While OpenSSL also supports other public-key algorithms, Neo indicates you shouldn't need those.
- `openssl rsa`: This tool allows you to inspect keys generated with the previous tool.
- `openssl req`: This tool allows you to generate and inspect Certificate Signing Requests (CSR's). A CSR is what you hand to a Certificate Authority (CA) for them to sign,

and the CA will copy information from the CSR into a signed Certificate that they'll give you back.

- `openssl x509`: X.509 is the standard used for formatting certificates. This tool allows you to inspect and manipulate such certificates.
- `openssl s_client`: The netcat of SSL connections. You can use this to connect to SSL-enabled servers and send and receive data and view certificates and other connection metadata.

For more information about each tool, check out their man pages (e.g. `man genrsa`).

To inspect SSL-related files, you can use the following syntax:

```
  # RSA Private Key
openssl rsa -in filename -text

  # RSA Public key. You may need to use -pubin instead
  # of -RSAPublicKey_in depending on the input file.
openssl rsa -in filename -text -RSAPublicKey_in

  # Certificate Signing Request
openssl req -in filename -text

  # X.509 certificate
openssl x509 -in filename -text
```

Ethics

This project requires that you communicate and interact with **real** machines on the Internet. You **must not** attack the machines in any way other than via intercepting and altering communication that originates from your VM as part of this project. Do not ever attack the servers directly for any reason! Any such attack or attempted unauthorized access constitutes a violation of the **Berkeley Campus Code of Student Conduct** and could have direct consequences for you as a student. In addition, it may expose you to **legal jeopardy**. Finally, attacks on our servers or other university infrastructure may make it impossible for us to do a project like this in the future. Please don't ruin that opportunity for other students.

You should conduct all of your analysis and exploration of the VM environment from within the VM environment. Inspecting the VM externally (such as mounting it as a disk image from outside VirtualBox) is not allowed and as such constitutes cheating.

If you have any questions about whether an attack or other action is in scope, do not hesitate to email an instructor or post a private note on Piazza.

The Task

When the VM boots up (after you've entered your login), you will be able to select which question you want to work on. While it's possible to do each question independently, we strongly encouraged you to go in order, as each question builds conceptually upon the prior.

NOTE: **You must log out of the VM via the graphical interface to switch questions. You cannot do Question 2 after selecting Question 1 from the menu, for example!**

Once you select which question you want to work on, you may interact with the VM via the provided graphical desktop. You can open a terminal in the graphical interface (from Menu → Accessories → Terminal Emulator), or via SSH. For question 1, the *username:password* is `q1-student:q1-student`, and so on.

Question 0 ***Can You Hack It?*** **(20 points)**

NOTE: For this particular question you use the same environment as for Question 1, so the SSH credentials you should use are `q1-student:q1-student`.

Neo is certain that he has obtained a packet capture of a secret conversation—but alas, one encrypted with TLS. `~/q0/q0.pcap` holds a copy.

Normally, the content of the conversation would be completely unknowable . . . but in this case, the server's private key showed up on Pastebin, and Neo has provided a copy in `~/q0/q0_privkey.priv`. You must decrypt the conversation and obtain the secret within.

Neo has emphasized to you the importance of familiarizing yourself with tools for capturing and analyzing network traffic. The VM comes with two of these already installed: the graphical *Wireshark* utility (start via Menu → Internet → Wireshark), and the command-line tool *tcpdump*. *Wireshark* has the ability to decrypt TLS traffic if you know the private key. *tcpdump* does not.

After opening wireshark, you can use the Files → Open option in Wireshark's main interface to load the packet capture. This will show the encrypted packets from the captured traffic. Look over the packets and the format of the connection between the client and the server.

To decrypt SSL traffic, you must install the private key into Wireshark. `http://wiki.wireshark.org/SSL` has detailed documentation about working with SSL in Wireshark, but in short, you'll need to:

- Open the SSL protocol preferences: Edit → Preferences → Protocols → SSL
- Add a new RSA private key, next to "RSA keys list" click Edit... → New, and fill in:
  - **IP address:** the server's IP, or the string `any`
  - **Port:** 443

- **Protocol:** http
- **Key File:** *browse to find private key*
- **Password:** *leave empty*

You'll now be able to view the decrypted traffic from the packet capture.

Submit the following files:

`q0/secret`
: This file contains the secret from the conversation. This is a long random string that appears in the (English) plaintext.

`q0/explanation.txt`
: This file includes:

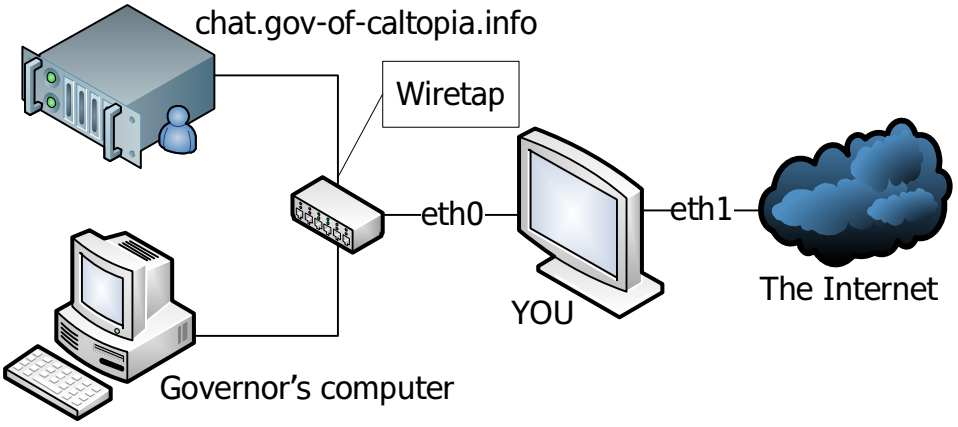
a) A description of the procedure you used to obtain the secret.

b) A list of cryptographic algorithms that were used for this TLS connection. For each algorithm, describe in one sentence its use in the protocol.

c) A discussion of whether there was any technical way by which you could have been prevented from decrypting the conversation even though you have obtained the private key. If so, briefly describe the technical approach. If not, explain why no such approach is feasible.

Question 1 ***Rumors Coming True*** **(60 points)**

Neo informs you that there's a little surprise tucked away inside Gov. Stalloon's headquarters: a nifty device that captures the Governor's network traffic. The device tunnels the traffic to your Virtual Machine, where it shows up at the Ethernet device `eth0`:



The network diagram for Neo's traffic-capture hack.

The buzz on the street is that Gov. Stalloon regularly checks his chat logs at `chat.gov-of-caltopia.info`. If only you could somehow figure out how to get a peak at those logs ... But, alas, they're encrypted using TLS.

You find a copy of the server's public key at `~/q1/server_pubkey.pub`. But how is that helpful?

A late-night text from Neo indicates he managed to recover the code for the tool that the server at `chat.gov-of-caltopia.info` used to generate the public-private keypair, `~/q1/generate_rsa_pair.c`, and a `Makefile` that compiles it in the same directory. Neo only had about 20 seconds to look it over before passing it along to you but it "`has luser stamped all over it`" ...

You must somehow leverage the code's poor quality to recover the contents of the encrypted conversation.

Neo senses that `chat.gov-of-caltopia.info` is set up to only accept traffic from Gov. Stalloon's computer. So don't bother trying to access it directly.

Submit the following files:

`q1/secret`
: This file contains the secret from the conversation.

`q1/chat.priv`
: This file contains the private key for `chat.gov-of-caltopia.info` in PEM format.

(continued on next page)

`q1/generate_rsa_pair.c`

This file contains the code you used to generate the private key. The autograder will compile this file using the same `Makefile` that shipped with your VM.

`q1/run`

A script that runs `generate_rsa_pair`, the compiled version of the source code you submitted, and prints the private key in PEM format to standard output.

NOTE: **Do not hard code your solution.** Your solution should take a public key from `~/q1/server_pubkey.pub` and print the corresponding private key. The public key is guaranteed to have been generated by the original `generate_rsa_pair.c`.

`q1/explanation.txt`

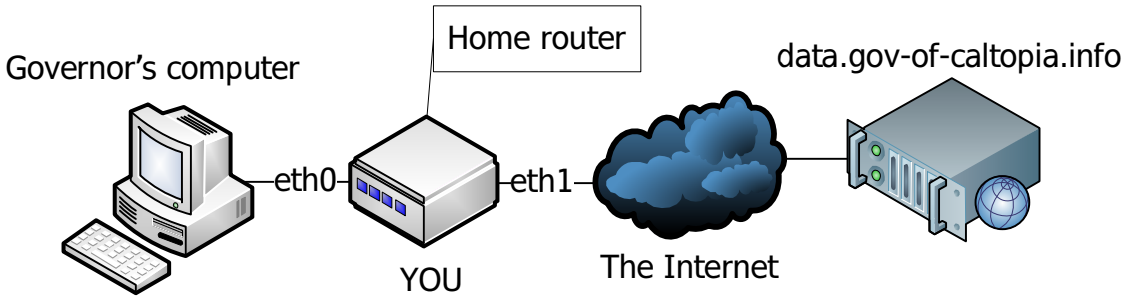
This file includes:

a) A description of the procedure you used to obtain the secret.

b) The line number of the line in the original `generate_rsa_pair.c` that doesn't do what the comment above the line states. Discuss whether or not (and **why**) fixing this line so it does what the comment says it should will make this key generation scheme be secure.

Question 2 ***Dead Man Walking*** **(60 points)**

Gov. Stalloon's home router used a default password, which Neo leveraged to "upgrade" its firmware: hello, MITM! The hacked code again tunnels all of the traffic through your VM, with the home network attached to `eth0` and the Internet on `eth1`:



The network diagram for MITM fun.

Something related to the previous chat logs appears to connect to `data.gov-of-caltopia.info` every 30 seconds. You should use Wireshark to verify this. You need to **intercept and rewrite** these communications to foil Gov. Stalloon's plan.

Neo has obtained a snippet of code this software uses to verify the certificates. You can find it at `~/client_fragment.c`. Neo also spotted that `data.gov-of-caltopia.info`'s TLS certificate is signed by the *Budget Certs R Us* Certification Authority.[6] Interestingly, this is the very same CA that as a joke Neo uses for one of his sites, `https://neocal.info`.

Courtesy of Neo, you can request a signed certificate for *any* domain ending in `.neocal.info` at `http://www.budget-certs-r-us.biz`. When providing Budget Certs R Us with a certificate signing request (CSR), you need to provide the contents of `~/IDENTIFICATION_SECRET` to identify yourself.[7] To create a new CSR, you'll want to use the `openssl req` tool. Neo has also provided you with an additional tool, `~/rewrite_cn`, that given a CSR will rewrite the common name with **exactly** what you've given it in a text file.[8] This will come in handy.

The `sslsniff` tool (provided) is very useful for performing MITM attacks on TLS connections. Given a certificate/private key pair, it intercepts incoming connections and presents them with your certificate. It also connects as a client to the original incoming connection target, and subsequently relays any data between the two connections.

When doing so, sslsniff has access to the plaintext data (because of the use of your certificate, plus that it creates the actual connection to the original server). It logs the plaintext to the file `~/sslsniff/sslsniff.log`.

[6] While the software in this question may trust *Budget Certs R Us* , your computer's operating system is more prudent, and does not. Expect certificate warnings if you attempt to visit `https://data.gov-of-caltopia.info` directly.

[7] **You remembered to log out and select question 2, right?**

[8] A word to the wise: it's prudent to inspect the text file given to rewrite_cn with `hexdump`.

You can install the certificate/private key pair using `~/sslsniff/sslsniff_install_cert`, and start sslsniff with `~/sslsniff/sslsniff`. Neo has modified sslsniff in this VM to pass all HTTP requests to the simple Ruby script `~/sslsniff/rewriter.rb` for modification. You can modify `rewriter.rb` to change these requests on-the-fly. (**Warning:** be careful when altering application level protocols. You must adhere to the application specifications. Failure to do so can result in client crashes, or the server appearing to hang. For HTTP, the `content-length` header must *exactly* match the length of the request body!) Your computer **must** have a working Internet connection to solve this problem!

Neo has flagged for you that **there are two distinct approaches to this problem**, and it's vital that you develop successful attacks for both of them. Doing so ensures that even if one of the vulnerabilities gets fixed, the other will still work for future intelligence-gathering.

IMPORTANT: one of the approaches *requires* `rewrite_cn` (i.e., there is no way to carry it out the attack using the standard `openssl` tool). If you believe you have two different approaches neither of which requires using `rewrite_cn`, contact us privately before you develop them further so we can potentially help you avoid unnecessary work.

ALSO IMPORTANT: it is crucial that you check the success of your attack, and also know how to undo its effects. In order to do that, Neo has hacked into the data server, and, with his usual humor, set up a way for you to *break the 4th wall*[9] by going to `http://data.gov-of-caltopia.info`[10].

Submission *Important:* Leave `data.gov-of-caltopia.info` in the right state between your submission and the submission deadline (don't reset via the 4th wall after you carry out the attack for the final time).

Submit the following files:

`q2/secret`
: This file contains the secret from the communication.

`q2/rewriter.rb`
: This file contains your modified code that rewrites the network traffic.

`q2/data0.priv`

`q2/data0.req`

`q2/data0.x509`
: These files contain the private key, certificate signing request, and certificate respectively for the first approach you use to MITM the connection. The files must be in PEM format.

`q2/data1.priv`

[9] http://en.wikipedia.org/wiki/Fourth_wall

[10] Neo laughs, "bet these lusers wouldn't even get the reference!"

`q2/data1.req`

`q2/data1.x509`

These files contain the private key, certificate signing request, and certificate respectively for the second approach you use to MITM the connection. The files must be in PEM format.

`q2/explanation.txt`

This file includes:

a) A description of the procedure you used to obtain the secret.

b) A discussion of what, if anything, `gov-of-caltopia.info` can do to protect against these attacks.

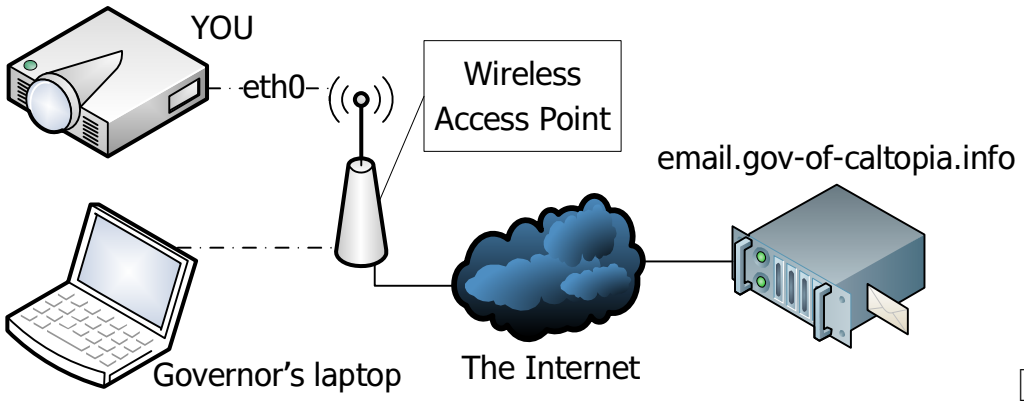
c) A discussion of what, if anything, Governor Sylvester Stalloon can do to protect against these attacks.

d) A discussion of what, if anything, `budget-certs-r-us.biz` can do to protect against these attacks.

Question 3 ***Anything You Say Can And Will Be Used Against You*** **(60 points)**

The events of the last question have left Governor Sylvester Stalloon feverishly attempting to log into his webmail, `https://email.gov-of-caltopia.info`, to delete any evidence. He can't believe his appalling luck when he finds the email server is currently undergoing maintenance! He can log in, but can't reach his email. In a haze of desperation he repeatedly attempts to log in over and over.

After suspecting Gov. Stalloon's malicious intentions, Neo arranged for a spy camera to appear in the Governor's office to obtain evidence, connected to the office wireless network, just like the Governor's laptop. The camera tunnels traffic to and from your Virtual Machine—you can view the Governor's browser in your VM using a handy desktop shortcut. The wireless network is attached to `eth0`:



[width=.75]

The network diagram for the Governor's "closeup".

You have to somehow get Gov. Stalloon's webmail credentials in order to expose his nefarious freedom-squashing plot. Unfortunately, though, one of the Governor's lackeys set things up so he only ever logs into his webmail by opening a fully-patched Chromium web browser,[11] typing `https://email.gov-of-caltopia.info`, hitting Enter, then entering his credentials into a webform ... and, in addition, as someone used to working from scripts, he'll only do this if everything *exactly matches* what he's expecting.

To carry out this attack, you need to somehow redirect the Governor to your own webserver—but while having his *fully-up-to-date* version of Chromium still saying he's visiting `https://email.gov-of-caltopia.info` **with no warnings**. If any warnings appear, the Governor will figure Wait A Sec That's Not Right, refrain from entering any information, restart Chromium, and begin again.

Neo has provided some tools to go after this vital task. `~/digipwntar` contains some very interesting files from a Certificate Authority called DigiPwntar. DigiPwntar is trusted by the Governor's browser[12]. Neo has also given you a skeleton for using a python packet capture library called `scapy`[13]. The skeleton can be found in

[11] Chromium is the open source browser that forms the basis for Google Chrome.

[12] And sensibly not trusted by your computer. Expect certificate warnings if visiting directly.

[13] `http://www.secdev.org/projects/scapy/doc/usage.html` has extensive documentation on how to use `scapy`. One hint: to access layers of a packet `pkt`, you can e.g. use `pkt.haslayer(TCP)` and `pkt[TCP]`.

`~/pcap_tool/pcap_tool.py`.

Lastly, Neo has provided you with your own webserver at `~/local_webserver/` that will do everything you need to do—should you somehow manage to get Gov. Stalloon's browser to visit it without any warnings. As before, your computer **must** have a working Internet connection to solve this problem.

NOTE: The attack needs to be as stealthy as possible! The attack should not disrupt communications to any other site Gov. Stalloon visits.

Submit the following files:

`q3/secret`
: This file contains the Governor's password.

`q3/pcap_tool.py`
: This file contains your modified code that carries out the attack.

`q3/email.priv`

`q3/email.req`

`q3/email.x509`
: These files contain the private key, certificate signing request, and certificate respectively that you used for your local webserver. The files must be in PEM format.

`q3/explanation.txt`
: This file includes:

  a) A description of the procedure you used to obtain the secret.

  b) A discussion of whether there are any mechanisms or protocols Governor Sylvester Stalloon could have used to defend himself against your attack. If so, explain why your attack wouldn't work when using these. If not, discuss the implications of this attack for the use of TLS in the Internet today.

Feedback

Since this is still a relatively new project, it would be especially valuable to have feedback on it. Please include a file `feedback.txt` if you are so inclined.

Submission Summary

In summary, you must submit the following directory tree:

```
q0/secret
q0/explanation.txt
q1/secret
q1/chat.priv
q1/generate_rsa_pair.c
q1/run
q1/explanation.txt
q2/secret
q2/rewriter.rb
q2/data0.priv
q2/data0.req
q2/data0.x509
q2/data1.priv
q2/data1.req
q2/data1.x509
q2/explanation.txt
q3/secret
q3/pcap_tool.py
q3/email.priv
q3/email.req
q3/email.x509
q3/explanation.txt
feedback.txt (optional)
```