

Problem 2

Any reasonable attack tree and reasonable cost estimates would be accepted.
A few things to avoid

- Attack trees that reduce the problem to itself or a very similar problem.
- Leaves with cost that is difficult to estimate.
- Breakdowns into tasks that aren't totally independent.
- Arithmetic errors

Interestingly, the estimates on the cost to read `/etc/shadow` were almost all between \$100 and \$10000, with most around \$1000.

Some advantages of attack trees:

- Attack trees promote holistic thinking.
- By encouraging you to consider all possible avenues of attack, you can spend a limited security budget on security measures with the biggest payoff.
- Attack trees can be easily extended as new attack strategies are discovered.

Some disadvantages of attack trees:

- Attack trees can give a false sense of security since it is easy to overlook an avenue of attack.
- Attack tree results are very dependent on the original cost estimates, which are hard to make accurately.
- Attack trees do not consider secondary factors. For example, in some scenarios it may be sufficient to catch an intruder instead of preventing the intrusion.
- Attack trees should really be attack directed acyclic graphs (DAGs).
- It may be difficult to break down an attack into independent steps
- Attack trees do not weigh the fact that anyone in the world can launch a remote attack over the internet, but only a few people can perform a physical breakin.
- Attack trees are really designed to evaluate a targeted attack. Most computer breakins are not targeted, though.

Problem 3

Suppose $N = pq$, $ed = 1 \pmod{\phi(N)}$, and $e = 3$. Since $ed = k\phi(N) + 1$ for some $k \in \mathbb{Z}$, we know that $ed - 1 = k\phi(N)$. Since $1 < d < \phi(N)$, we must have $2 < ed - 1 < 3\phi(N)$. Thus $k = 1$ or $k = 2$. Since p and q are prime, $p = 1 \pmod{3}$ or $p = 2 \pmod{3}$ and $q = 1 \pmod{3}$ or $q = 2 \pmod{3}$. Since $\phi(N) \pmod{3} = (p-1)(q-1) \pmod{3}$, we can compute the following table:

$p \pmod{3}$	$q \pmod{3}$	$\phi(N) \pmod{3}$
1	1	0
1	2	0
2	1	0
2	2	1

If $k = 1$, then $ed - 1 = \phi(N)$, but $ed - 1 = 2 \pmod{3}$ and, from the above table, $\phi(N) \not\equiv 2 \pmod{3}$. Hence $k \neq 1$. So $k = 2$.

So, if we know N , e , and d , where $e = 3$, then we can compute $u = \phi(N) = (ed - 1)/2$. Since $\phi(N) = (p - 1)(q - 1) = pq - p - q + 1$, we can compute $v = p + q = N - u + 1$. We know that $p = v - q$, so $N = (v - q)q$. Thus $q^2 - vq + N = 0$. By using the quadratic formula, we can compute q as

$$q = \frac{v \pm \sqrt{v^2 - 4N}}{2}$$

Given q , we can compute $p = N/q$. Alternatively, we could observe that the quadratic formula given above has two solutions: p and q .

Finally, if we did not realize that k cannot be 1, we could do the same computation above, trying it for each possible value of k . Eventually, we would discover a factor of N .

Problem 4

There are a variety of solutions to this problem. Here's one:

Each file, f , has its own encryption key, K_f . If f has security label $l_f = (s, C_f)$, where $C_f = \{c_1, \dots, c_n\}$ is the set of compartments containing f , then we generate $n + 1$ values x_0, \dots, x_n such that $x_0 \oplus \dots \oplus x_n = K_f$. Note that we can easily generate these x_i s by picking x_1, \dots, x_n randomly and solving for x_0 . We then assign an encryption key to each security level and to each compartment. We store with each file an access control record $(E(K_s, x_0), E(K_{c_1}, x_1), \dots, E(K_{c_n}, x_n))$. We grant a domain access to all the encryption keys for security levels lower than its level and for all compartments to which it belongs. When a domain wishes to access a file,

it decrypts each entry in the access control record using the appropriate key, reconstructs K_f , and decrypts the file.

The main advantage of this scheme is that even if an attacker is able to get physical access to the disk, he will not be able to read the files on that disk.

There are many disadvantages. First, performance will probably be lower because of the encryption, although that may be dwarfed by disk access overhead. Second, a process can now leak its file access rights by leaking the keys. This approach can not be applied to the Biba model for ensuring integrity because it could only detect modification by unauthorized processes, but in practice you want to prevent such modification.