MITOCW | ocw-6.033-lec21

0:00:00 Just one brief announcement. HKN reviews are going to be 0:00:03.313 done in class next Monday so you guys should make sure you come, 0:00:07.108 give us your feedback, let us know what you like and 0:00:10.18 what you don't like. And, with that, 0:00:12.289 we'll start talking about protection. 0:00:15 0:00:20 Protection is like fault-tolerance and 0:00:22.8 recoverability. One of these properties of 0:00:25.902 systems, or building secure and protected systems has 0:00:29.837 implications for the entire design of the system. 0:00:34 And it's going to be sort of a set of cross-cutting issues that 0:00:37.351 is going to affect the way that, for example, 0:00:39.729 the networking protocols are designed or that the sort of 0:00:42.756 modules that make up your biggest computer system are 0:00:45.567 designs. So it's going to be a whole set 0:00:47.675 of usually that we're going to look at through the course of 0:00:50.864 this discussion about protection that are going to affect the 0:00:54.108 system at all levels. In 6.033, we use the work 0:00:56.594 protection and security essentially synonymous. 0:01:00 Often times we'll talk about a secure system or a system that 0:01:04.47 has security or certain security goals that we have, 0:01:08.27 so we're going to use those words interchangeably. 0:01:11.921 Security is one of these topicals that you guys are 0:01:15.647 familiar with to some extent already. 0:01:18.329 You've heard about various things on the Internet going on 0:01:22.576 where people's information has been stolen on laptops or a 0:01:26.823 website has been cracked into or some worm or new virus, 0:01:30.921 the new I love you virus is spreading around and confection 0:01:35.243 people's computers. So you guys are sort of 0:01:39.481 familiar with it on a collegial sort of way, I'm sure. 0:01:42.896 You also are familiar with many of the tools that we're going to 0:01:46.956 talk about, so the applied versions of the many of the 0:01:50.371 tools that we're going to talk about. 0:01:52.69 You've all used a password to lock into a computer before or 0:01:56.492 you've used a website that using SSL to encrypt the communication 0:02:00.615 with some other website. So you're going to be familiar 0:02:04.682 with some of many of the high letter instances of the tools 0:02:07.934 that we'll talk through in this session, but what we're going to 0:02:11.467 try to delve down into in 6.033 is how those systems are 0:02:14.551 actually put together, what the design principals are 0:02:17.467 behind building these secure systems. 0:02:19.485 As I said, you guys are presumably very familiar with, 0:02:22.457 you've heard about these various kinds of attacks that 0:02:25.429 are going on. And one of the things that's 0:02:29.321 happened in the last few years, as the Internet has become more 0:02:34.462 and more commercial and larger and larger, is that it's meant 0:02:39.437 that security of computers has become much, much more of a 0:02:44.163 problem. So the growth of the Internet 0:02:47.231 has spawned additional attacks. If you go look at a website, 0:02:52.123 for example, there are several security 0:02:55.273 websites that track recent security breaches 0:03:00 This is one example. It's called Security Focus dot 0:03:03.098 com. I don't know if you can see 0:03:05.019 these, but this is just a list of things that have happened in 0:03:08.799 the last just few days on the Internet. 0:03:11.153 Somebody is reporting that web server hacks are up by 0:03:14.376 one-third, some IT conference was hacked, Windows says 0:03:17.66 "trusted Windows" is still coming. 0:03:19.705 [LAUGHTER] So it just goes on and on with these are things 0:03:23.237 that have happened in the last few days. 0:03:25.653 There is this huge number of things. 0:03:29 You may have heard recently about how there have been 0:03:32.018 several large companies recently that have had big privacy 0:03:35.327 problems where databases of customer information have been 0:03:38.636 stolen. AmeriTrade just had this 0:03:40.436 happen. The University of California at 0:03:42.642 Berkeley had something like several hundred thousand 0:03:45.602 applications and graduate student records were on a laptop 0:03:48.911 that was stolen. These are the kinds of things, 0:03:51.582 the kinds of attacks that happen in the world, 0:03:54.194 and these are the kinds of things that we're going to talk 0:03:57.503 about how you mitigate. The objective, 0:04:01.272 really, of security, one simple way to look at an 0:04:05.636 objective of security is that we want to sort of protect our 0:04:11 computer from bad guys. The definition of bad guy 0:04:15.363 depends on what you mean. It could be the 16 year old kid 0:04:20.454 in his dorm room hacking into people's computers. 0:04:24.818 It could be somebody out to sleep hundreds of thousands of 0:04:30 dollars from a corporation, but let's assume that there are 0:04:35.272 some bad people out there who want to sort of take over your 0:04:40.636 computer. But, at the same time, 0:04:44.483 the objective of security is also to allow access to the good 0:04:48.53 guys. So one way to make sure that 0:04:50.756 the bad guys don't get at your data is simply to turn your 0:04:54.601 computer off, right? 0:04:55.882 But that's not really a good option. 0:04:58.243 We want the data to be available to the people who need 0:05:01.885 the data and have the rights to access the data. 0:05:06 Often times we can sort of frame our discussion, 0:05:09.441 we can say that we're trying to protect we have some set of 0:05:13.688 information that we want to keep private. 0:05:16.617 So sort of a goal of a secure system, in some sense, 0:05:20.352 is providing privacy. So we have some set of data 0:05:23.867 that's on our computer system or that's being transmitted over 0:05:28.334 the network that we want to keep private, we want to keep other 0:05:32.874 people from being able to have access to or tamper with. 0:05:38 And throughout the sort of notion of what it means for a 0:05:41.104 computer system to be secure is sort of application dependent. 0:05:44.548 It depends very much on what the computer system we're 0:05:47.54 talking about is. It may be the case that in your 0:05:50.25 file system, you have a set of files that you want the entire 0:05:53.637 world to have access to, stuff that's on your webpage 0:05:56.572 they're willing to make public. You don't have any real 0:06:00.362 security concerns about who can read it. 0:06:02.673 But, at the same time, you might have banking data 0:06:05.576 that you really don't want people in the outside world to 0:06:08.894 be able to access. So you have a set of policies 0:06:11.678 that define, in your head, some sort of set of policies or 0:06:15.055 rules about what it is that you would like users to be able to 0:06:18.669 access. So almost any system has some 0:06:20.802 policies associated with data should be accessed by other 0:06:24.12 people. Some notion of what data they 0:06:26.253 want to keep private that can be sort of translated into this set 0:06:30.044 of policies. So in 6.033 it's sort of hard 0:06:33.816 to study in a systematic way different types of policy. 0:06:37.199 Policy is something that we're not going to have. 0:06:40.205

we could sit around and have an informal discussion about 0:06:43.713 different possible policies that you might want. 0:06:46.657 But instead, in 6.033, what we're going to 0:06:49.225 do, as we've done in much of the rest of the class is talk about 0:06:53.172 mechanisms that we can use to enforce these different security 0:06:56.993 policies that someone might have. 0:07:00 So we're going to talk about the tools that we use to protect 0:07:03.957 data. In thinking about security and 0:07:06.266 in thinking about these mechanisms, it's useful to start 0:07:09.894 off maybe by thinking about what we mean by security and 0:07:13.522 protection in the real world and sort of compare the mechanisms 0:07:17.612 that we have in the real world for protecting data from, 0:07:21.24 say, mechanisms that we have on the computer. 0:07:25 0:07:30 From the point of view of what we might want to accomplish with 0:07:34.428 a secure computer system, some of the goals and 0:07:37.714 objectives are similar to what we have in the real world. 0:07:41.714 Clearly, we have this same objective which is to protect 0:07:45.642 data. We can say, just like in the 0:07:48 real world, we have a lock on a door that protects somebody from 0:07:52.5 getting access to something we don't want them to have access 0:07:56.785 to. In the world of computers, 0:07:59.587 we can encrypt data in order to make so somebody who we don't 0:08:03.25 want to have access to our data doesn't have access to that 0:08:06.791 data. Similarly, there are also, 0:08:08.684 say, for example, in the real world a set of laws 0:08:11.614 that regulate who can access what data and what's legal and 0:08:15.155 what's not legal. It's not legal for me to break 0:08:18.025 into your house and take something from it. 0:08:20.589 Similarly, it's not legal for somebody to hack into a computer 0:08:24.313 and steel a bunch of files. There are also some 0:08:28.291 differences. The obvious one is one that has 0:08:31.215 been true of almost all of our comparisons between the real 0:08:35.158 world and, say, normal engineering disciplines 0:08:38.218 that involve building bridges and buildings and computer 0:08:41.957 systems. And that's this issue that 0:08:44.269 computer systems have a very high dtech over dt. 0:08:47.464 So the computer systems change quickly. 0:08:51 And that means there is always both new ways in which computers 0:08:54.018 systems are connected to the world, there are new and faster 0:08:56.89 computers that are capable of breaking encryption algorithms 0:08:59.762 that maybe we didn't think could be broken before, 0:09:02.148 there are new algorithms being developed both for protecting 0:09:05.02 data, and there are new strategies that people are 0:09:07.405 adopting to sort of attack computers. 0:09:09.158 In the recent years we've seen this thing where people are 0:09:11.933 doing what they call phishing, P-H-I-S-H-I-N-G. 0:09:15 Where people are putting up fake websites that look like 0:09:17.59 real websites. So all these emails that you 0:09:19.567 get from Bank One or whoever it is saying come to our website, 0:09:22.44 click on it and give us your social security number and your 0:09:25.218 credit card number, I hope you guys aren't 0:09:27.149 responding to those. Those are fake websites and 0:09:29.362 people are trying to steal your information. 0:09:32 We see new attacks emerging over time. 0:09:34.732 The other kinds of things that we see that are different, 0:09:38.869 clearly computer systems are, a tax in computer systems can 0:09:43.153 be both very fast and they can be cheap. 0:09:46.034 So, unlike in the real world, in a computer system you don't 0:09:50.392 have to physically break into something. 0:09:53.272 You can do this very quickly over a computer system. 0:09:58 So you see, for example, with some of these warms and 0:10:01.268 viruses, these things spread literally across tens of 0:10:04.537 thousands of computers in a matter of seconds. 0:10:07.365 So these are very efficient and effective attacks that can take 0:10:11.262 over a huge number of computers in a very short period of time. 0:10:15.159 And that leads to us wanting to have a different set of sort of 0:10:19.057 mechanisms for dealing with these kinds of problems. 0:10:22.262 And then, finally, it is also the case there are 0:10:25.217 some differences in between laws and computer systems and in the 0:10:29.177 real world. In particular, 0:10:31.773 because the computer systems change so fast because new 0:10:34.758 technologies develop so fast, the legal system tends to lag 0:10:37.963 significantly behind the state of the art and technology. 0:10:41.058 So the legal system often doesn't have regulations or 0:10:43.932 rules that specifically govern whether or not something is OK 0:10:47.248 or is not OK. And that means sometimes it's 0:10:49.569 unclear whether it's legal to do something. 0:10:51.891 So, for example, right now it's not clear 0:10:54.101 whether it's legal for you to take your laptop out in the City 0:10:57.473 of Cambridge, open it up and try and connect 0:10:59.849 to somebody's open wireless network. 0:11:03 Certainly, you can do that, it's very easy to do, 0:11:05.642 probably many of us have done this, but from a legal 0:11:08.45 standpoint there is still some debate as to whether this is OK 0:11:11.809 or not. What this suggests, 0:11:13.241 the fact that laws are often unclear, ambiguous or simply 0:11:16.324 unspecified about a particular thing is that we're going to 0:11:19.517 need additional sort of sets of, if you really want to make sure 0:11:22.986 your data is secure, if you want to enforce a 0:11:25.409 particular security policy you're going to need to rely 0:11:28.382 more on sort of real mechanisms in the computer software to do 0:11:31.741 this rather than on the legal system to say, 0:11:34.108 for example, protect you from something that 0:11:36.476 might be happening in the outside world. 0:11:40 0:11:45 Designing computer systems is hard. 0:11:47.401 A secure computer system is hard, in particular. 0:11:50.721 And the reason for that is that security, often times the things 0:11:55.171 we want to do in secure systems, the things we want to enforce 0:11:59.479 are so-called negative goals. So what do I mean by that? 0:12:04 0:12:12 An example of a positive goal is, for example, 0:12:16.939 I might say Sam can access pile F. 0:12:20.56 That, presumably, is something that is relatively 0:12:25.829 easy to verify that that's true. I can log onto my computer 0:12:31.585 system. And if I can access this file F 0:12:33.902 then, well, great, I can access the file F. 0:12:36.463 We know that's true. And that was easy to check. 0:12:39.329 Furthermore, if I cannot access the file and 0:12:41.951 I think that I should have the rights to access it, 0:12:45 I'm going to email my system administrator and say, 0:12:48.048 hey, I think I should be able to access this file, 0:12:51.036 why can't I, will you please give me access 0:12:53.597 to it? An example of a negative goal 0:12:57.092 is that Sam shouldn't be able to access F. 0:13:00.05 At first it may seem that this is just the same problem as 0:13:04.162 saying, it's just the inverse of saying Sam cannot access F, 0:13:08.419 but it seems like it should be just as easy or hard to verify. 0:13:12.82 But it turns out that when you're thinking about computer 0:13:16.86 systems, this sort of a problem is very hard to verify because, 0:13:21.333 while it may be true that when I try and open the file and I

problem is very hard to verify because, 0:12:55 while it may be true that when I try and open the file and I 0:13:25.59 log into my machine and I connect to some remote machine 0:13:29.558 and try and access that file, it may be the case that the 0:13:33.598 file system denies me access to that file. 0:13:38 But I may have many other avenues for obtaining access to 0:13:40.654 that file. For example, 0:13:41.696 suppose I have a key to the room in which the server that 0:13:44.35 hosts that file is stored. I can walk into that room and 0:13:46.957 may be able to sit down in front of the consol on this machine 0:13:49.848 and obtain super user route access to that machine. 0:13:52.218 Or, I may be able to pull the hard drive off the machine and 0:13:55.014 put it into my computer and read files off of it. 0:13:58 Or, I may be able to bribe my system administrator and give 0:14:01.367 him a hundred dollars in exchange for him letting me have 0:14:04.617 access to this file. So there are lots and lots of 0:14:07.462 ways in which users can get unauthorized or unintended 0:14:10.539 access to files or other information in computer systems. 0:14:13.79 And verifying that none of those avenues are available to a 0:14:17.157 particular user is very hard. Worse, or similarly, 0:14:20.001 this is hard because it's very unlikely that a user is going to 0:14:23.601 complain about having access to some file that they shouldn't 0:14:27.084 have access to. I'm not going to call up my 0:14:30.207 system administrator and say, hey, I have access to this 0:14:32.666 file, I don't think I should have access to it. 0:14:34.723 Nobody is going to do that. So, even though I'm not a 0:14:37.048 malicious user, I don't really have any 0:14:38.747 incentive to go to my system administrator and tell them that 0:14:41.43 there's this problem with the way that things are configured 0:14:44.069 on the computer. And that extends also to people 0:14:46.17 who, in fact, are malicious users. 0:14:47.646 When somebody breaks into a computer system, 0:14:49.569 they don't typically, usually, send out an 0:14:51.402 advertisement to everybody in the world saying, 0:14:53.459 hey, by the way, I got access to this file that 0:14:55.516 I should have access to. It's possible for them to log 0:14:57.886 in, read the file, log out, and nobody would be 0:14:59.943 any of the wiser. Many of our security goals are 0:15:05.711 negative, and that means building secure computer systems 0:15:12.639 is hard. 0:15:14 0:15:24 What we're going to do in 6.033, in order to get at this, 0:15:29.333 get at sort of building secure systems in the face of these 0:15:34.857 negative goals, is look at a set of different 0:15:39.047 security functions that we can use to protect information and 0:15:44.761 access to computers in different sorts of ways. 0:15:50 And we're typically going to talk about, throughout this, 0:15:54.238 a client server sort of a system where you have some 0:15:58.098 client and some server that are separated, typically over the 0:16:02.639 Internet, that are sort of trying to exchange information 0:16:06.878 or obtain information from each other in a way such that that 0:16:11.419 information exchange is protected and secure and so on. 0:16:15.506 Suppose we have a client sending some information out 0:16:19.442 over the Internet to some server. 0:16:23 What are the kinds of things that we want to make sure, 0:16:27.287 what are the sorts of security goals that we might want to 0:16:31.813 enforce in this environment? One thing we might want to do 0:16:36.339 is authenticate the client. The server might like to know 0:16:40.785 for sure that the person who issued this request is, 0:16:44.834 in fact, the client. They'd like to have some way of 0:16:48.884 knowing that the client issued this request and that the 0:16:53.251 request that was sent is, in fact, what the client 0:16:57.141 intended to be sent. For example, 0:17:00.741 that somebody didn't intercept this message as it was being 0:17:04.324 transmitted over the network, change it a little bit and then 0:17:08.032 send it on making it look as though it came from the client. 0:17:11.677 There might be different kinds of attackers that are sitting in 0:17:15.508 this Internet. For example, 0:17:17.114 there might be say an intermediate router along the 0:17:20.203 path between the client and the server where the system 0:17:23.54 administrator of the router is malicious. 0:17:27 Oftentimes, in the security literature, there are these 0:17:30.053 funny names that are attached to the different people in 0:17:33.163 different places. Oftentimes the client is called 0:17:35.877 Alice and the server is called Bob, the person receiving the 0:17:39.213 request. And we talk about two different 0:17:41.418 attackers. We talk about Eve who is an 0:17:43.51 eaves dropper who listens to what's going on and tries to 0:17:46.676 acquire information that she's not authorized to have and we 0:17:50.012 talk about Lucifer who is the bad guy who not only is trying 0:17:53.348 to overhear information but may do arbitrarily bad things. 0:17:56.571 He's trying to take over the data and corrupt it in any way 0:17:59.851 he possibly can. One goal is that we want to 0:18:03.873 prevent, say, for example, 0:18:05.607 Lucifer from being able to interfere with packets coming 0:18:09.423 from Alice to Bob. We want to make sure that 0:18:12.406 packets the server receives actually originated from Alice 0:18:16.361 and were the original request that Alice sent, 0:18:19.483 so that's authentication. We also might want to 0:18:22.674 authorize, at the server, that Alice is, 0:18:25.38 in fact, allowed to access the things that she's trying to 0:18:29.334 access. If Alice tries to read a file, 0:18:32.538 we need some way of understanding whether Alice is 0:18:35.051 allowed to access this file or not. 0:18:37 0:18:45 We also need to keep some information confidential. 0:18:49.545 We may want it to be the case that Eve, who overhears a 0:18:54.454 packet, cannot tell what the contents of that packet are. 0:19:00 We might want to be able to protect the contents of that 0:19:02.5 thing so that Eve cannot even see what's going over the 0:19:04.954 network. So notice that this is a little 0:19:06.727 bit different than authenticating. 0:19:08.227 Authenticating says we just want to make sure the packet, 0:19:10.772 in fact, came from the client. But we're not saying anything 0:19:13.454 about whether or not somebody else can see the contents of 0:19:16.045 that packet. Keeping confidential says we 0:19:17.863 want to make sure that nobody, except for the intended 0:19:20.272 recipients can, in fact, see the contents of 0:19:22.227 this. There are a couple other 0:19:23.545 properties that we want as well. One thing we might want is 0:19:27.418 accountability. We're going to talk about this 0:19:29.874 a little bit more. This says we need to assume 0:19:32.33 that it's always possible that something could go wrong. 0:19:35.332 It's always possible that I might have bribed my assistant 0:19:38.443 administrator and he might have given me access to the computer. 0:19:41.881 And, in the end, there is not much you're going 0:19:44.391 to be able to do about that. What you want to do is make 0:19:47.393 sure that when situations like that occur that there's some log 0:19:50.776 of what happened, you have some way of 0:19:52.796 understanding what it was that happened, why it happened and 0:19:56.016 how it happened so you can try and prevent it later. 0:20:00 You want to make sure you do a counting, you keep track of 0:20:03.433 what's been going on. And, finally,

want to make sure you do accounting, you keep track of access. What's been going on? And, finally,
0:20:05.481 you might want availability. This is you might want to make 0:20:08.975 sure that Lucifer, who is sitting here between the 0:20:11.927 client and the server cannot, for example, 0:20:14.397 send a huge number of packets at the server and make it 0:20:17.65 unavailable, just swamp it with a denial of service attack. 0:20:21.144 Availability means that this system, in fact, 0:20:23.795 functions and provides the functionality that it was 0:20:26.867 intended to provide to the client. 0:20:30 0:20:35 We are going to spend a while in 6.033 especially focusing on 0:20:38.649 these first three techniques. Essentially, 0:20:41.143 the next three lectures are going to be talking about how we 0:20:44.732 guaranty, how we authenticate and authorize users and how we 0:20:48.321 keep information confidential and private. 0:20:50.815 But all of these goals together there is a set of technical 0:20:54.343 techniques that we can talk about for trying to provide each 0:20:57.931 one of these things. But when you think about 0:21:01.4 building a secure system, it's not enough to simply say 0:21:04.549 we're going to employ, you know, I employ 0:21:06.883 authentication to make sure that Alice is, in fact, 0:21:09.799 Alice when she talks to Bob. What you want to do, 0:21:12.599 when you build a secure system, is think about sort of how to, 0:21:16.158 you want to get your mindset around building sort of the 0:21:19.366 general ideas behind a secure system. 0:21:21.466 And so, in 6.033, we have this set of principles 0:21:24.208 that we advocate called the safety net approach. 0:21:28 And the idea is the safety net approach is a way to help you 0:21:31.487 sort of in general think about building a secure system as 0:21:34.857 opposed to these specific techniques that we're going to 0:21:38.108 see how to apply later on. 0:21:40 0:21:45 The safety net approach advocates sort of a set of ways 0:21:51.982 of thinking about your system. The first one is be paranoid. 0:22:00 This is sort of the Murphy's Law of security. 0:22:03.764 It says assume that anything that can go wrong will go wrong. 0:22:08.898 Don't just assume that because you're authenticating the 0:22:13.604 communication between Alice and Bob that there is no way that 0:22:18.737 somebody else will pretend to be Alice. 0:22:21.989 You should always have something, a safety net, 0:22:25.925 some backup to make sure that your system is really secure. 0:22:32 A good example of being paranoid and applying the safety 0:22:35.282 net approach are things like suppose your router has a 0:22:38.444 firewall on it, suppose your home router has a 0:22:41.13 firewall that's supposed to prevent unauthorized users from 0:22:44.591 being able to access your computer, does that mean that 0:22:47.813 you should then turn off all password protection on your 0:22:51.095 computer so that anybody can log in? 0:22:53.184 No, you're not going to do that. 0:22:55.034 You're going to continue to protect the information on your 0:22:58.495 computer with the password because you may have a laptop 0:23:01.777 and may not be using it from home. 0:23:05 Or, you may be worried about somebody breaking into your 0:23:08.329 computer from inside of your house, say, perhaps. 0:23:11.234 That's an example of sort of thinking about the safety net 0:23:14.685 approach. You have multiple layers of 0:23:16.864 protection for your information. There are some sort of 0:23:20.133 sub-approaches, there are some sort of 0:23:22.372 sub-techniques that we can talk about in the context of being 0:23:26.004 prepared. One of them is accept feedback 0:23:28.365 from users. If you were designing a big 0:23:31.815 computer system and somebody tells you that something is 0:23:35.512 secure or that there is a problem, be prepared to accept 0:23:39.21 that feedback, have a way to accept that 0:23:41.831 feedback and respond to that feedback. 0:23:44.319 Don't simply say oh, that's not a real security 0:23:47.411 problem, that's not a concern or don't, for example, 0:23:50.84 make it so the users don't have a way to give you information. 0:23:54.941 Defend in depth. This is have multiple security 0:23:58.033 interfaces like passwords plus a firewall. 0:24:02 And, finally, minimize what is trusted. 0:24:04.738 You want to make sure that the, and this is sort of a good 0:24:08.846 example, we've talked about this as an example of system design 0:24:13.315 before. We want to try and keep things 0:24:15.981 as simple as possible, but this was really important 0:24:19.657 in the context of security because you want to make sure, 0:24:23.693 for example, that you try and keep the 0:24:26.36 protocols that you use to interact with people from the 0:24:30.252 outside world as simple as possible. 0:24:34 The more interfaces that you have with the outside world that 0:24:37.861 users can connect to your computer over, 0:24:40.371 the more places you have where your computer system is 0:24:43.782 vulnerable. So you want to try and 0:24:45.905 minimize, make it so that your computer system has as few sort 0:24:49.831 of openings it can that you have to verify our securer as 0:24:53.435 possible. Other examples of the safety 0:24:55.816 net approach, you need to consider the 0:24:58.198 environment. 0:25:00 0:25:05 This just means it's not enough, you know, 0:25:07.228 suppose that I have this connection here, 0:25:09.402 this connection over the Internet between Alice and Bob, 0:25:12.391 I may assume that I'm only worried about attackers who are 0:25:15.489 coming in, say, for example, 0:25:16.956 over the Internet. But if you're a server, 0:25:19.184 the server may have other connections available to it. 0:25:22.065 So it may be the case that this is a server inside of some 0:25:25.163 corporate environment and this server has a dialup modem 0:25:28.152 connected to it. Especially, nowadays, 0:25:31.084 this is less common, but it used to be the case that 0:25:33.719 almost all companies had a way that you could dial in and get 0:25:36.819 access to a computer when you didn't have a wired Internet 0:25:39.763 connection available to you. And often times this dial-in 0:25:42.656 access was a separate place where people could connect. 0:25:45.446 So, for example, it didn't have the same 0:25:47.461 interface as the sort of main connection to the Internet. 0:25:50.354 And that meant that there was sort of side channel, 0:25:52.937 in the environment, through which people could use 0:25:55.468 to get access to the computer. Similarly, this means you 0:25:59.394 should think about all the people who have access to the 0:26:02.46 computer. Is it the case maybe that this 0:26:04.635 is a computer in your office and the janitor who works in your 0:26:08.037 office comes into the office every night, would he or she be 0:26:11.327 able to sit down in front of your computer and get access to 0:26:14.617 the system when they shouldn't be able to? 0:26:16.903 And this may sound paranoid. It is being paranoid, 0:26:19.635 but if you really want to build a secure computer system you 0:26:22.925 need to sort of keep all these things in mind. 0:26:25.434 Need to plan for iteration. This is, again, 0:26:29.046 just a good system design principle, but it's especially 0:26:32.883 true here. Assume that there will be 0:26:35.325 security violations in your computer system, 0:26:38.325 assume there will be security problems, and plan to be able to 0:26:42.581 address those problems and also have a way to verify that once

problems, and plan to be able to 0:26:42.264 address those problems and also have a way to verify that once 0:26:46.837 you've addressed those problems the other parts of your system 0:26:51.093 that are supposed to be secure continue to be secure. 0:26:54.72 And, finally, keep audit trails. 0:26:58 This gets at our goal of accountability. 0:27:00.421 This just means keep track of everything. 0:27:02.904 All of the authentication and authorization requests that you 0:27:06.629 made in your system, when a user logs in, 0:27:09.113 keep track of where they logged into, maybe even keep track of 0:27:12.9 what they did so that you can come back, if it turns out that 0:27:16.625 this person was unauthorized, you later discovered that they 0:27:20.288 were up to no good, you can come back and 0:27:22.771 understand what it is that they did. 0:27:26 What all this sort of discussion, especially this 0:27:28.423 discussion about the safety net approach illustrates or gets at 0:27:31.553 is that there are lots of these issues that we're talking about. 0:27:34.734 So, for example, the janitor breaking into our 0:27:37.006 computer or me bribing my system administrator. 0:27:39.328 There aren't really computer system issues, 0:27:41.449 right? These are human issues. 0:27:42.913 These are things that I'm bypassing any sort of 0:27:45.236 authentication I might have in the computer or any kind of 0:27:48.114 security that I might have built into the computer because, 0:27:51.042 for example, my system administrator is 0:27:52.96 authorized to access anything on the computer he wants to. 0:27:57 And so I have, essentially from the computer 0:27:59.923 systems point of view, made it look like I have rights 0:28:03.528 to get at anything I want to if I can bribe the system 0:28:07.132 administrator. So this suggests that sort of 0:28:10.056 in many computer systems humans are the weak link. 0:28:14 0:28:19 Not only are people bribable but people make mistakes. 0:28:22.407 People don't read dialog boxes. People do things hastily. 0:28:26.007 People don't pay attention. The reason that these phishing 0:28:29.648 attacks work where somebody pretends to be Washington Mutual 0:28:32.89 or eBay and sends you an account that asks you to log in and type 0:28:36.406 in your social security number is that people don't think. 0:28:39.538 They just see this email and say oh, I guess eBay wants me to 0:28:42.835 give them my social security number. 0:28:44.758 OK. People make mistakes. 0:28:46.076 And this is the way that many, many security vulnerabilities 0:28:49.318 or security problems happen is through people mistakes. 0:28:52.285 So we're going to talk in most of 6.033 about technical 0:28:55.252 solutions to security problems. But, when you're actually out 0:28:59.57 in the world building a system, you need to be thinking about 0:29:02.818 the people who are going to be using this system almost as much 0:29:06.175 as you're thinking about the sort of security cryptographic 0:29:09.315 protocols that you design. That means, for example, 0:29:12.022 you should think about the user interface in your system. 0:29:15.054 It does the user interface to promote sort of users thinking 0:29:18.248 securely. The classic example of sort of 0:29:20.359 a bad user interface is your web-browser popping up this 0:29:23.337 dialogue box every time you access a site that isn't SSL 0:29:26.315 encrypted saying this website is not SSL encrypted, 0:29:29.022 are you sure you wish to continue? 0:29:32 And, after it has done this about ten times and these are 0:29:35.947 sites that you know and believe are safe like almost any site on 0:29:40.389 the Internet, you just click the dialogue box 0:29:43.49 that says never show me this alert again. 0:29:46.31 There is almost no useful information that the system is 0:29:50.187 giving you by complaining in that way. 0:29:52.796 Other examples of things you want to make sure you do is have 0:29:57.026 good defaults. In particular, 0:29:59.795 don't default to a state where the system is open. 0:30:02.58 When an error occurs, don't leave the system in some 0:30:05.479 state where anybody can have access to anything that they 0:30:08.662 want to. Instead, default to a state 0:30:10.651 where people don't have access to things so that you're not 0:30:13.948 exposing the system to failures. Don't default to a password 0:30:17.301 that anybody can guess. When you create a new user for 0:30:20.314 your system don't make the default password PASSWORD, 0:30:23.269 that's a bad idea. Make it some random string that 0:30:26.055 you email to the user so that they have to come back and type 0:30:29.465 it in. 0:30:31 0:30:36 Finally, give users least privilege needed to do whatever 0:30:39.123 it is they need to do. This means don't, 0:30:41.298 by default, make users administrators of the system. 0:30:44.143 If the user doesn't need to be an administrator of the system, 0:30:47.545 they shouldn't have administrative access, 0:30:49.832 they shouldn't be able to change anything that they want. 0:30:52.956 A good example of least privilege being violated is many 0:30:56.023 versions of Microsoft Windows, by default, make the first user 0:30:59.426 who is created an administrator user who has access to 0:31:02.382 everything. And this is a problem because 0:31:05.518 now when somebody breaks into that users account they now have 0:31:08.712 access to everything on the machine, as opposed to simply 0:31:11.643 having access to just that user's files. 0:31:14 0:31:19 In general, what this just means is keep your systems 0:31:21.791 simple, keep them understandable, 0:31:23.51 keep the complexity down. And that's sort of the safety 0:31:26.409 net. There are these two principles 0:31:28.234 that you want to think about. One, sort of be paranoid. 0:31:31.134 Apply this notion of having a safety net in a computer system. 0:31:35 And, two, don't just think about the technical protocols 0:31:37.955 that you're going to use to enforce access to the computer 0:31:41.017 but think about sort of who is using this system, 0:31:43.597 think about humans who have access to it, 0:31:45.746 and think about how to prevent those humans from being able to 0:31:49.023 do stupid things that break your security goals for your system. 0:31:53 0:31:58 This was a very sort of high-level fuzzy discussion 0:32:01.64 about security. Now what we're going to do is 0:32:04.843 we're going to drill in on some of these more specific technical 0:32:09.43 protocols. And today we're going to look 0:32:12.269 at a way in which you can think of most secure systems as 0:32:16.346 consisting of a set of layers, and those layers are basically 0:32:20.715 as follows. We have, at the top, 0:32:22.972 some application which we want to secure. 0:32:25.884 And then underneath this application we can talk about 0:32:29.743 three layers. 0:32:32 0:32:37 So we have some kind of functionality that we want to 0:32:41.089 provide, we have some set of primitives that we're going to 0:32:45.651 use to provide that, and then, at the very bottom, 0:32:49.505 we have cryptography which is the set of mathematics and 0:32:53.831 algorithms that we're going to use that we generally, 0:32:57.921 in modern computer systems, use to make sure that the 0:33:02.011 computer system is secure. The application may have its 0:33:07.352 high-level functions that correspond to these things over 0:33:12.058 here, so we may want to be able to authenticate users or we may 0:33:17.268 want to be able to authorize that users have access to 0:33:21.722 something or we may want to

provide confidentiality. 0:33:26.008 So we may want to be able to make sure that nobody can read 0:33:30.882 information they don't have access to. 0:33:35 So you're going to have a set of primitives for doing these 0:33:39.264 different things. For authentication, 0:33:41.911 we're going to need to talk about something called an access 0:33:46.25 control list. For authorization, 0:33:48.529 we're going to talk about primitives called sign and 0:33:52.279 verify. And for confidentiality we will 0:33:55.073 talk about these primitives called encrypt and decrypt. 0:34:00 And these topics, this stuff that's in this 0:34:02.04 middle set of primitives, we're going to describe these 0:34:04.663 in more detail in later lectures. 0:34:06.218 What I want to do with the rest of the lecture today is to talk 0:34:09.23 about this bottom layer, this cryptography layer. 0:34:12 0:34:17 We have some set of cryptographic ciphers and 0:34:20.771 hashes. And so a cipher is just 0:34:23.342 something that takes in, say, a message that the user 0:34:27.8 wants to send that is not protected at all. 0:34:32 And it flips the bytes in that message around in order to 0:34:35.579 create something that is not understandable unless you have 0:34:39.287 some piece of information that allows you to decipher that 0:34:42.931 message. You can say ciphering and 0:34:45.041 deciphering is sort of like encrypting and decrypting, 0:34:48.429 words you may be familiar with. We also talk about hashes. 0:34:52.073 Hashes we're going to use to authenticate or to authorize a 0:34:55.78 particular message to make sure that a message is -- 0:35:00 I'm sorry. Just for your notes, 0:35:02.222 I got this backwards. This should be authenticate 0:35:05.777 with sign and verify and we authorize with ACL. 0:35:09.185 We'll talk about these things more in a minute, 0:35:12.592 but it was just confusion over two words that start with auth. 0:35:17.111 So we're going to use hashes to basically authenticate that a 0:35:21.555 user is, in fact, who they claimed that they 0:35:24.74 were. And, again, we'll see in more 0:35:27.259 detail how this works over the next couple days. 0:35:32 0:35:42 Early cryptographic systems relied on this idea, 0:35:44.928 or early cryptography relied on this idea that we're going to 0:35:48.667 try and keep the protocol that's used for encoding the 0:35:51.97 information secret. A simple example of an early 0:35:54.899 encryption method might be something that many of you 0:35:58.139 played with when you were a kid where you transpose all the 0:36:01.753 letters. You have some map where you A 0:36:04.651 maps to C and B maps to F and so on, some mapping like that, 0:36:08.012 and you use that to encrypt it. And there are these puzzles 0:36:11.316 where you get told a few of the letters and you try and guess 0:36:14.734 what the other letters and decode a message. 0:36:17.183 That's a simple example of a kind of encryption that you 0:36:20.316 might apply. And that encryption relies on 0:36:22.651 the fact that this transform is essentially secret. 0:36:25.5 If you know the transform obviously you can decrypt the 0:36:28.575 message. These schemes are often called 0:36:32.227 closed design crypto schemes. And the idea is that because 0:36:36.459 the attacker doesn't know what scheme was used to encode the 0:36:40.839 information it can be very, very hard for them to go about 0:36:45.071 decoding it. For example, 0:36:46.853 the architecture for this might look like message goes into some 0:36:51.531 encryption box which is secured from the outside world. 0:36:55.54 It was just hidden from the outside world. 0:36:58.584 Nobody else knows what that is. And this goes over the Internet 0:37:04.575 to some other decryption box which then comes out on the 0:37:09.146 other end as a message. This is a closed design. 0:37:13.051 And these designs, in general, sort of turn out 0:37:16.873 not to be a very good idea because the problem is if 0:37:21.111 somebody does discover what this function is now you're in 0:37:25.848 trouble. Now this whole system is no 0:37:28.756 longer secure. And, worse than that, 0:37:31.971 when you make these things secure, if you suppose now 0:37:34.377 you're going to put this system out in the world with a hidden 0:37:37.199 protocol that nobody knows in the world, now it's sort of you 0:37:39.976 against the whole world. Whereas, if you had told 0:37:42.197 everybody what the protocol was to begin with and said this is 0:37:45.019 the protocol and there is this little bit of information that 0:37:47.795 we keep secret within the protocol, the two parties in the 0:37:50.432 protocol keep secret, but here's the sort of 0:37:52.422 algorithm that's in the protocol, let's let the entire 0:37:54.874 world verify whether this protocol is, in fact, 0:37:57.002 secure or not, you'd have a much better chance 0:37:59.085 of developing something that was secure. 0:38:02 It is sort of accepted wisdom that these kinds of closed 0:38:04.965 design systems tend not to be widely used anymore. 0:38:07.606 The systems that we are going to talk about, 0:38:09.925 for the rest of this talk today, are going to be so-called 0:38:12.998 open design systems. Of course, there are many, 0:38:15.478 many systems that were designed with this and there are many, 0:38:18.713 many closed cryptography systems. 0:38:20.438 And, in some ways, they're sort of the most 0:38:22.702 natural ones and the things you'd think of first. 0:38:25.29 And they've been very effective throughout history. 0:38:27.986 It's just the case that modern cryptography typically doesn't 0:38:31.22 rely on it. 0:38:33 0:38:43 Open design systems have an architecture that typically 0:38:49.375 looks as follows. It is pretty similar to what we 0:38:55.041 showed before. We have m going into E. 0:39:00 But this time this protocol E, the world knows what the 0:39:04.677 algorithm E is and instead this E has some piece of secret 0:39:09.614 information coming into it, a key, k. 0:39:12.732 And this key is usually not known to the world. 0:39:16.716 And now we go through the Internet, for example, 0:39:20.787 and we come out to a decryption box which also has a key going 0:39:26.07 into it. And these keys may or may not 0:39:29.275 be the same on the two boxes. And we'll talk about the 0:39:35.103 difference between making them the same or not making them the 0:39:40.451 same. Now we have this message that 0:39:43.431 comes out. Message comes in, 0:39:45.798 gets encrypted, goes over the Internet, 0:39:49.129 goes into the decryption box and gets decrypted. 0:39:53.25 If k1 is equal to k2 we say this system is a shared secret 0:39:58.246 system. And if k1 is not equal to k2 we 0:40:02.792 say that this is a public key system. 0:40:06.036 In k1 is equal to k2 these two keys are the same. 0:40:10.36 And, say, Alice and Bob on the two ends of this have exchanged 0:40:15.855 information about what this key is before the protocol started. 0:40:21.441 Alice called up Bob on the phone or saw Bob in the hallway 0:40:26.576 and said hey, the key is X. 0:40:30 And they agreed on this beforehand. 0:40:31.767 And now that they've agreed on what this key is they can 0:40:34.626 exchange information. In a public key system, 0:40:36.914 we will see the design of how one public key system works in a 0:40:40.085 little bit more detail, but typically it's the case 0:40:42.685 that, for example, the person who is sending the 0:40:45.128 message has a private key that nobody else knows.

0:40:47.623 only they know, and then there's a public key 0:40:49.911 that everybody else in the world knows and is sort of distributed 0:40:53.238 publicly. And these two keys are not 0:40:55.058 equal but there is some mathematical operation that you 0:40:57.865 can apply that, given something that's been 0:41:00.048 encrypted with k1, you can later decrypt it with 0:41:02.492 k2. And we will look at one example 0:41:06.224 of one of those mathematical functions. 0:41:08.95 The point here is closed design says the algorithm itself is 0:41:13.183 unknown to the world. What shared secret simply says 0:41:16.843 is that there is some little bit of information, 0:41:20.215 a little key, like a number that we've 0:41:22.869 exchanged, but it's not the algorithm, it's not the 0:41:26.457 protocol, it's just this one little bit of, 0:41:29.47 say, several hundred bits of key information that we 0:41:33.13 established beforehand. It still is the case that, 0:41:39.086 for example, this protocol in a shared 0:41:43.478 secret system is published. And so people can go and try 0:41:50.007 and analyze the security of this thing. 0:41:54.517 The community can look at what the math is that is going on. 0:42:01.521 Let's look at a simple example of a shared key system. 0:42:09 0:42:14 This is an approach called a one-time pad. 0:42:19 0:42:22 One-time pad is a very simple example of an encryption 0:42:25.812 protocol that is essentially cryptographically unbreakable. 0:42:29.985 That is it may be possible to break it through other means but 0:42:34.374 it is sort of provably not possible to break this through 0:42:38.402 some sort of mathematical analysis attack. 0:42:42 One-time pad depends on the ability of a source of truly 0:42:44.917 random bits. I need some way to generate a 0:42:47.092 set of bits which are sort of completely random. 0:42:49.586 And doing that is tricky, but let's suppose that we can 0:42:52.45 do it. What one-time pad says, 0:42:53.989 we're going to call this sequence of bits k, 0:42:56.27 that's going to be our key, and this key and the one-time 0:42:59.241 pad approach is not going to be short. 0:43:02 This key is going to be very, very long. 0:43:04.669 It's going to be as long as all of the messages that we possibly 0:43:08.982 want to send over all of time. Suppose that Alice and Bob, 0:43:12.885 the way they generate this is Alice writes a set of random 0:43:16.787 bits onto a CD, writes 650 megabytes of random 0:43:19.867 bytes onto a CD and gives that to Bob and they agree that this 0:43:24.044 is the key that they are going to use over time. 0:43:27.261 We have m and k coming in. They are being combined 0:43:31.818 together. They are being transmitted over 0:43:34.979 the Internet. And then, on the other side, 0:43:38.22 they are being decoded also with k where these two ks are 0:43:42.647 equal in this case and we've got m coming out. 0:43:46.204 This plus operation here is an XOR. 0:43:48.891 This plus with a circle is XOR. If you were to remember what 0:43:53.555 XOR does, the definition of XOR is given two bytes, 0:43:57.507 zero, one, two things that were XORing together, 0:44:01.222 two bytes that are either zero or one, the XOR of two zeros is 0:44:06.044 zero, the XOR of zero and one is one and the XOR of two ones is 0:44:10.944 zero. XOR has this nice property. 0:44:14.846 This XOR is a byte-wise operation. 0:44:17.102 This says the first byte in k is XORed with the first bid in m 0:44:21.273 and the second byte in k is XORed with the second byte in m 0:44:25.239 and so on. And the same thing happens over 0:44:28.042 here. XOR has the nice property that 0:44:31.151 m XOR k XOR k is equal to m. You can verify that that's true 0:44:35.383 if you look at a simple example. What happens is when the stream 0:44:39.902 m comes in and gets XORed with k, the encrypted message that is 0:44:44.35 traveling over here essentially looks like a random byte string 0:44:48.797 because k is a random byte string. 0:44:51.164 And this m, no matter what m is, when it is XORed with a 0:44:55.109 random byte string you're going to get something that looks like 0:44:59.628 a random byte string coming out. And then on this other side, 0:45:04.512 though, we also have access to the same byte string that was 0:45:07.697 used, and now we can decrypt the message. 0:45:09.858 So only if somebody knows exactly what k is can they 0:45:12.612 decrypt the message. This approach is hard to make 0:45:15.257 work in practice because it requires the availability of a 0:45:18.335 large amount of random data. One thing that you can do is 0:45:21.36 use a random number generator that is seeded with some number 0:45:24.599 and then use that random number generator to begin to generate 0:45:27.893 the sequence of bytes. Of course, that has the problem 0:45:32.209 now which is somebody can discover the seed or somebody 0:45:36.058 can guess which seed you used because the random number 0:45:39.907 generator used is not very good. They may be able to break your 0:45:44.327 protocol. But on-time pad is a nice 0:45:46.75 example of something which is sort of a cryptographic protocol 0:45:51.098 that is known to work pretty well. 0:45:53.45 What I want to do, just with the last five 0:45:56.373 minutes, is talk about one specific cryptographic protocol 0:46:00.436 which is called the RSA protocol. 0:46:04 And we probably won't have time to go through the details of how 0:46:12.42 RSA actually works, but RSA is a public key 0:46:18.034 protocol. And let me just quickly show 0:46:22.98 you what RSA uses and then we will skip over DES here. 0:46:31 0:46:36 What RSA does is says we are going to take two numbers, 0:46:39.042 p and q which are prime. This protocol is in the book. 0:46:42.028 It is in appendix one of the chapter so you don't need to 0:46:45.183 copy it down word for word if you don't want to. 0:46:47.83 If we have p and 1 are primes, we pick two numbers p and q 0:46:51.042 which are prime, and then we generate some 0:46:53.352 number n which is equal to p times q and another number z 0:46:56.507 which is equal to p minus one times q minus one. 0:47:00 And then pick another number e which is relatively prime to z. 0:47:05.264 So that relatively prime just means that e doesn't divide z 0:47:10.27 evenly. So z is not divisible by e. 0:47:13.205 And we pick a number d such that e times d is equal to one, 0:47:18.211 as long as the modulus is z. 0:47:21.404 If you take e times d and you take the modulus with z the 0:47:26.238 value should be one. If you pick a set of numbers 0:47:30.725 that satisfy this property then we can define the public key to 0:47:34.546 be e, n and the private key to be d, n. 0:47:36.888 And these numbers have this magic property. 0:47:39.477 And then we are going to be able to, with this, 0:47:42.312 encrypt any message that is up to n in length. 0:47:45.085 So, in general, we are going to want n and p 0:47:47.735 and q to be some set of very large numbers. 0:47:50.324 They are going to be hundreds of bytes long. 0:47:52.974 We are going to be able to encrypt any message that is up 0:47:56.425 to n bytes long, it is up to size of n. 0:48:00 So we may have to break the messages up into chunks that are 0:48:03.873 size n or smaller. This has this magic property 0:48:06.892 now that if we encrypt the data in the following way, 0:48:10.306 to encrypt a message we take m to the power of e and then take 0:48:14.31 the whole thing module n. Now, transmit that message c

take it to the power e and then take it mod n. The whole thing modulo n. Now, transmit that message C 0:48:17.789 across the network. Now, to decrypt, 0:48:20.087 we take that encrypted thing and take it to the power d and 0:48:23.894 then take the whole thing modulo n, we get the original message 0:48:27.964 out at the end. The mathematics of 0:48:30.936 understanding why this work turned out to be fairly subtle 0:48:33.903 and sophisticated. There is a brief outline of it 0:48:36.402 in the paper, but if you really want to 0:48:38.38 understand this it's the kind of thing that requires an 0:48:41.19 additional course in cryptography. 0:48:42.908 We're not going to go into the details of the mathematics, 0:48:45.875 but the idea is suppose we pick p to be 47 and q to be 49, 0:48:48.842 two prime numbers, generate n to be 2773, 0:48:50.924 just the product of those, z then is 2668. 0:48:53.058 And now we pick two numbers e and d. 0:48:54.88 We pick these numbers just by using some searching for these 0:48:57.951 numbers somehow over the set of all possible numbers we could 0:49:01.074 have picked. So we have these two numbers e 0:49:05.363 and d which satisfy this property. 0:49:07.963 That is 2668 is not divisible by 17, and 17 times 157 modulo 0:49:12.612 z, modulo 2668 is equal to one. And you have to trust me that 0:49:17.339 it true. Now, suppose we have our 0:49:19.86 message is equal to 31. If we compute now C is equal to 0:49:24.115 this thing, 31 to the 17th modulo 2773, we get 587. 0:49:29 And then, sort of magically at the end, we reverse this thing 0:49:33.407 and out pops our message. What we see here is a public 0:49:37.3 key protocol. What we say here is the public 0:49:40.459 key is equal to this combination of e and n and the private key 0:49:45.014 is equal to this combination of d and n. 0:49:47.879 Suppose that only Alice knows the private key and that Bob 0:49:52.066 knows the public key and everybody else in the world, 0:49:55.886 for example, knows what the public key is. 0:50:00 Now, what we can do is Alice can encrypt the message with her 0:50:04.094 private key which nobody else knows. 0:50:06.482 And then using the public key everybody else can go ahead and 0:50:10.576 decrypt that message. And by decrypting that message 0:50:14.056 they can be assured that the only person that could have 0:50:17.809 actually created this message to begin with is somebody who had 0:50:22.039 access to Alice's private key. So they can authenticate that 0:50:26.065 this message came from Alice. This protocol also has a nice 0:50:30.98 side effect which is that it is reversible. 0:50:33.849 If Bob encrypts a message using Alice's public key, 0:50:37.264 Alice can decrypt that message, and only Alice can decrypt that 0:50:41.499 message using her private key. We will talk more about these 0:50:45.529 properties next time. And take care.