CS 161 Computer Security
Fall 2005 Joseph/Tygar/Vazirani/Wagner **Notes 1**

1 The scope of this class

Our goal in this class is to teach you the some of the most important and useful ideas in computer security. By the end of this course, we hope you will have learned:

- *How to build secure systems.* You'll learn techniques for designing, implementing, and maintaining secure systems.
- *How to evaluate the security of systems.* Suppose someone hands you a system they built. How do you tell whether their system is any good? We'll teach you how systems have failed in the past, how attackers break into systems in real life, and how to tell whether a given system is likely to be secure.
- *How to communicate securely.* We'll teach you some selections from the science of cryptography, which studies how several parties can communicate securely over an insecure communications medium.

Computer security is a broad field, that touches on almost every aspect of computer science. We hope you'll enjoy the scenery along the way.

What is computer security? Computer security is about computing in the presence of an adversary. One might say that the defining characteristic of the field, the lead character in the play, is the adversary. Reliability, robustness, and fault tolerance are about how to deal with Mother Nature, with random failures; in contrast, security is about dealing with actions instigated by a knowledgeable attacker who is dedicated to causing you harm. Security is about surviving malice, and not just mischance. Whereever there is an adversary, there is a computer security problem.

Adversaries are all around us. The Code Red worm infected a quarter of a million computers in less than a week, and contained a time-bomb set to try to take down the White House web server on a specific date. Fortunately, the attack on the White House was diverted—but one research company is estimating the worm cost $2 billion in lost productivity and in cleaning up the mess caused by infected machines. One company estimated that viruses cost businesses over $50 billion in 2003. Hackers armed with zombie networks of tens of thousands of compromised machines sell their services brazenly, promising to take down a competitor's website for a few thousand dollars. It's been estimated that, as of 2005, at least a million computers worldwide have been penetrated and "owned" by malicious parties; many are used to send massive amounts of spam or make money through phishing and identity fraud. Studies suggest that something like half of all spam is sent by such zombie networks. It's a racket, and it pays well—the perpetrators are raking in money fast enough that they don't need a day job. How are we supposed to secure our machines when there are folks like this out there? That's the subject of this class.

2 It's all about the adversary

The early history of computer security is interwoven with military applications (probably because the military were one of the first big users of computers, and the first to worry seriously about the potential for misuse), so it should not be surprising that much of the terminology has military connotations. We speak of an *attacker* who is trying to *attack* computer systems, of *defenders* working to protect their system from these *threats*, and so on. Well, you get the idea.

It might be surprising that we are going to spend so much time studying attackers and thinking about how to break into systems. Aren't the attackers the bad guys? Why on earth would we want to spread knowledge that will help bad guys be more effective?

Part of the answer is that you have to know how your system is going to be attacked, if you want to defend it properly. Civil engineers need to learn what makes bridges fall down if they want to have any chance of building a bridge that will stay standing. Software engineering is no different; you need to know how systems fail in real life, if you want to have the best chance of building a system that will resist attack. This means you'd better know what kinds of attacks you are likely to face in the field. And, because attacks change and get better with time, you'd better learn to anticipate the attacks of the future.

While learning about recent history is certainly a good start, it's not enough to learn only about attacks that have been used in the past. Attackers are intelligent (or some of them are, anyway). If you deploy a new defense, they will respond. If you build a new system, they will try to find its weak points and attack there. Attackers adapt. This means that we have to find ways to anticipate what kinds of attacks might be mounted against us in the future.

Security is like a game of chess, only it is one where the attackers often get the last move. We design a system, and then it is very hard to change once it has been deployed. If attackers find a security hole in a widely deployed system, the consequences can be pretty serious. Therefore, we'd better learn to predict in advance what the attackers might do to us, so that we can eliminate all the security holes before the system is deployed. We have to practice thinking like an attacker, so that we will know in advance how secure the system is.

Thinking like an attacker is not always easy. Sometimes it can be a lot of fun to try to outwit the system, like a game. Other times, it can be disconcerting to think about what could go wrong and who could get hurt, and that's not fun at all.

What happens if you don't anticipate how you may be attacked? The cellphone industry knows the answer. In the 1980's, they designed and deployed an analog cellphone infrastructure with essentially no security measures; cellphones transmitted all their billing information in the clear, and security rested on the assumption that attackers wouldn't bother to put together the equipment to intercept it. That assumption held for a while, but sooner or later criminals were bound to catch on, and they did. Technically savvy attackers built "black boxes" that intercepted the radio communications and cloned phones, and criminals used these to make fraudulent calls en masse and to mount call-selling operations for profit. Cellphone operators were unprepared for this, and in the early 90's, it had gotten so bad that the US cellphone carriers were losing more than $1 billion per year. At one point I was told that 70% of the long-distance cellphone calls placed from downtown Oakland on a Friday night were fraudulent. By this point the cellphone service providers were already well aware that they had a serious problem, but because it takes 5–10 years and a great deal of capital to replace the deployed infrastructure of cellular base stations, they were in a difficult position. This illustrates how failing to anticipate how your system might be attacked—or underestimating the threat—can be a costly mistake.

It is for these reasons that security design requires the study of attacks. Security experts spend a lot of time trying to come up with new attacks. This might sound counter-productive (why help the attackers?), but it makes sense when you realize that it is better to learn about vulnerabilities before the system is deployed than after. If you know about the possible attacks in advance, you can design a system to resist those attacks; anything else is a toss of the dice.

3 A process for security evaluation

How do we think about the ways that an adversary might use to penetrate system security or otherwise cause mischief? In this lecture, we're going to develop a framework to help you think through these issues.

The first place to start, when analyzing a system, is its *security goals*. What properties do we want the system to have, even when it is under attack? What are we trying to protect from the attacker? Or, to look at it the other way around, what are we trying to prevent?

Some common security goals:

- *Confidentiality.* Often there is some private information that we want to keep secret from the adversary. Maybe it is a password, a bank account balance, or a diary entry that we don't want anyone else to be able to read. It could be anything. We want to prevent the adversary from learning our secrets.
- *Integrity.* If the system stores some information, we might want to prevent the adversary from tampering with or modifying that information.
- *Availability.* If the system performs some function, it should be operational when we need it. Consequently, we may need to prevent the adversary from taking the system out of service at an inconvenient time.

For example, consider the database of grade information that we use in this class. One obvious goal is to protect its integrity, so that you can't just give yourself an A+ merely by tampering with the grade database. University rules require us to protect its confidentiality, so that no one else can learn what grade you are getting. We probably also want some level of availability, so that when the end of the semester comes we can calculate the grades everyone will receive.

Security goals can be simple, or they can be detailed. Figuring out the set of security goals that must be preserved is an exercise in requirements analysis—they are the specification of it means for a system to be secure. The security goals are the goals we want to be met even when an adversary is trying to violate them. You can recognize which goals are security goals by asking yourself: if someone were to figure out how to violate this goal, would it be considered a security breach? If the answer is yes, you've found yourself a security goal.

Security goals are highly application-dependent, so it's hard to say much more. Instead, I'll leave you with a famous quote from Young, Boebert, and Kain: "An program that has not been specified cannot be incorrect; it can only be surprising." A system without security goals has not been specified, and cannot be wrong; it can only be surprising.

After you have a set of security goals, the next step is to perform a *threat assessment*, which asks several questions. What kind of threats might we face? What kind of capabilities might we expect adversaries to have? What are the limits on what the adversary might be able to do to us? The result is a *threat model*, a characterization of the threats the system must deal with.

When performing a threat assessment, we have to decide how much we can predict about what kind of adversaries we will be facing. Sometimes, we know very well who the adversary is, and we may even know their capabilities, motivations, and limitations. For instance, in the Cold War, the US military was oriented towards its main enemy, the Soviets, and a lot of effort was put into understanding the military capabilities of the USSR (how many battalions of infantry do they have? how effective are their tanks? how quickly can their navy respond to such-and-such threat?). When we know what adversary we will be facing, we can craft a threat model using that knowledge, so that our threat model reflects what that particular adversary is likely to do to us and nothing more.

However, all too often the adversary is not known. In this case, we need to reason more generically about unavoidable limitations that will be placed upon the adversary. As a light-hearted example, physics tells us that the adversary can't go faster than the speed of light—I don't care who they are, they can't violate the rules of physics. That might be useful to know. More usefully, we can usually look at the design of the system and identify what things an adversary might be in a position to do. For instance, if the system is designed so that secret information is never sent over a wireless network, then we don't need to worry about the threat of eavesdropping upon the wireless communications. If our system design is such that people might discuss our secrets by phone, we had better include in our threat model the possibility that an insider at the phone company might be able to eavesdrop on our phone calls, or re-route them to the wrong place, or fool people into thinking they are talking with someone legitimate when actually they are speaking with the attacker.

A good threat model also specifies what threats we do not care to defend against. For instance, if I want to analyze the security of my home against burglary, I am not going to worry about the threat that a team of burglars might fly a helicopter over my house and rappel down my chimney to get into the house, Mission Impossible style. There are far easier ways to break into my house, without going to all that trouble.

One can often classify adversaries according to their motivation. For instance, consider adversaries who are motivated by financial gain. It's a pretty safe bet that a financially-motivated adversary is not going to spend more money on the attack than they stand to gain from it. For instance, no burglar is likely to spend thousands of dollars to steal my car radio; my car radio is simply not worth that much. In general, motives are as varied as human nature, and it is a good idea to be prepared for all eventualities.

It's often very helpful to look at the incentives of the various parties. This is probably a familiar principle. Does the local fast food joint make more profit on soft drinks than on the food? Then one might expect some fast food places to take steps to boost sales of soft drinks, perhaps salting its french fries heavily. Do customer service representatives make a bonus if they handle more than a certain number of calls per hour? Then one might expect some representatives to be tempted to cut lengthy service calls short, or to transfer trouble customers to other departments when possible. Do spammers make money from everyone who responds to the spam, while losing nothing from those who didn't wish to receive the spam? Then one can expect that some spammers might be inclined to send their emails as widely as possible, no matter how unpopular it makes them. As a rule of thumb, organizations tend not to act against their own self-interest, at least not too often. Incentives influence behavior—not always, of course, but frequently enough to help illuminate the motivations of potential adversaries.

Incentives are particularly relevant when two parties have opposing interests. When incentives clash, conflict often follows. In this case it is worth looking deeply at the potential for attacks by one such party against the other.

If threat assessment sounds difficult, just remember the three W's: Who? How? Why? In other words: Who are the adversaries we might face? How might they try to attack us, and what are their capabilities? Why might they be motivated to attack us, and what are their incentives?

Finally, once we have the security goals and a threat model, the last step is to perform a security analysis. The goal of the security analysis is to see whether there is any attack encompassed within the threat model that can successfully violate the security goals. Security analysis is often highly technical and depends on the details of the particular system being analyzed. We will show you many methods for security analysis in the rest of the course, without saying more now.

An analogy: The security goals and threat model defines the game, and the security analysis amounts to figuring out who can win the game. The threat model defines the set of moves the adversary is allowed to make, and the design of the system defines how the defender will play the game. The security goals define the success condition: if the adversary violates any security goal, he wins; otherwise, the defender wins. The security analysis involves examining all moves and counter-moves to see who has a winning strategy.

Another analogy: Mystery writers like to talk about means, motive, and opportunity. That's not a bad way of thinking about what we do during a security evaluation. The threat assessment examines the means and motive; the security analysis examines what opportunity the adversary might have to do harm.

To sum up, evaluating the security of a system involves three steps:

- *Identify the security goals.* What are we trying to protect?
- *Perform a threat assessment.* What threats does the system need to protect against?
- *Do a security analysis.* Can we envision any feasible attack that would violate the security goals? This is the place where it can get pretty technical.

The same process can be used for designing new systems. The security goals and threat assessment is especially relevant to system design, since it is usually easier to ensure security when you know what security goals you are trying to achieve and what threats you must protect against. And, as we perform our security analysis, we can refine the system design to defend against each new attack we discover.

4 An example

Enough slogans. Let's do an example. Let's analyze the security of my home against intruders.

What are my security goals? I'd like to protect the assets in my home from theft or from being tampered with by unauthorized parties (integrity). I'd like my personal safety to be protected; for instance, if someone does break in to steal money, I'd much prefer to know, so that I don't surprise them and get shot. I'd like my house and its contents to remain in full working order whenever I want them (availability). My home is my castle, and I sometimes want a certain measure of privacy when I'm home (confidentiality). We could probably identify other security goals, but that's more than enough to get us started.

Time for the threat assessment. Who am I trying to protect against, and how might they be motivated? A burglar might be motivated by financial gain. A peeping Tom might be motivated by curiousity. Someone who holds a grudge against me might want to secretly get revenge. And so on.

What capabilities (tools, skills, knowledge, access, etc.) might an adversary have? What threats might I face? A burglar might have lockpicks and the know-how to use them, or a crowbar to smash in the door, or the capability to cut my phone lines. A repairman might have unaccompanied access to the house for a time. Peepers might have binoculars or a telescope. One of my neighbors might have line-of-sight to my living room window, while another might be blocked by trees. But there are probably some kinds of threats I'm willing to ignore. For instance, I'm not worried that the Navy ships here for Fleet Day are going to start shelling my house. My house has no effective way to defend against such an attack, but I doubt very much that any officer dislikes me enough to throw away his career over it. We could go on for pages, assessing which threats are realistic given the possible motivations of the adversary, and ending up with a detailed threat model.

Finally, the security analysis. What kind of attacks are possible? One can envision all sorts of crazy scenarios. An everyday burglar might smash a window while I'm gone, sneak in and grab some valuables, and run off before they're caught. If I secure the windows, a determined burglar might take a chainsaw to the walls and break in that way (believe it or not, it's happened). A slightly smarter burglar might look under the flowerpot, find the spare house key, and waltz right in. A really sneaky burglar might throw a pebble against my window at 3am each morning, setting off the burglar alarm and bringing the police running, for days at a row, until the police decide to stop paying attention to my obviously unreliable burglar alarm—and then the burglar can strike without threat of police response. The neighbor with line-of-sight to my house might use his telescope to peer in through my living window. Someone intent on revenge might be able to leave unpleasant items on my lawn, or—depending on my home's security system—might be able to smash my windows without my noticing. A unscrupulous competitor who knows I have an important business meeting with a potential client early tomorrow morning might cut the power to my house in the middle of the night, in hopes that my alarm clock will fail to ring, I will fail to awaken, I will miss my meeting, and I will lose the client. One can come up with some truly outlandish attack scenarios, but you probably get the idea by now.

I hope this gives you some feeling for how a security evaluation might go. If the example seems rather trivial, it is probably because home security is already at least somewhat familiar to you. However, when dealing with a complex computer system, it helps to have a framework to structure the security evaluation, and that's what the process outlined above is intended to help with.

5 Terminology

Lastly: Some terminology you might run into, and a refresher on the terms we've been using so far.

- An *attack* is an attempt to breach system security. Not all attacks are successful.
- A *threat* is a circumstance or scenario with the potential to cause harm to a system. An attack usually refers to a specific strategem, whereas threat refers to a broader class of ways that things could go wrong.
- A *vulnerability* is an aspect of the system that permits someone to mount a successful attack. Sometimes called a *security hole*. A *security weakness* is like a vulnerability, only it is less clear whether it could actually lead to any direct violation of the security goals. A weakness might represent a potential vulnerability whose risk is unclear; or, several weaknesses might combine to yield a full-fledged vulnerability.
- A *security goal* is a goal that is supposed to be achieved by the system; if it fails, the system will be considered insecure.
- A *threat assessment* is an attempt to assess the set of all possible threats. A *threat model* is a characterization of the possible threats, usually produced during a threat assessment.
- *24 by 7* refers to the window of time in which systems are most vulnerable to attack.
  (Ok, this last one was a joke.)