Chapter 1: Introduction & threat modeling

Suggested reading:

- Security Engineering, Chapter 1
- Tools & Jewels Chapter 1
- An introduction to approachable threat modeling
- Attack trees

Case study:

- Subway fare evasion, from the point of view of attacker and defender (the MTA)

Security is fundamentally a **contest between an attacker and defender.** Defenders have security policies they wish to enforce and attackers, for various reasons, aim to undermine those policies.

Security engineering is about designing mechanisms for defenders to prevent attackers from causing harm. Defenders also must balance competing requirements such as a budget, ease-of-use and compliance with the law.

The most challenging aspect of security engineering is that **attackers adapt and evolve over time**. Security engineers continually try to predict what attackers will do in response to a new system being deployed so they can build in security in advance.

This class will have a large emphasis on (recent) history. We'll look at many areas in class where attackers and defenders have been battling over time. In many cases we'll learn about attack strategies that used to work, but have been thwarted by modern technology. Why learn about an attack that hasn't been widely used in 10 years? One of the best ways to develop the ability to predict and model attackers is to look at the history of security systems and how they've evolved.

As the famous saying goes:

"Those who cannot remember the past are condemned to repeat it."

- George Santayana

Security vs. Safety: The challenge of evolving attackers

In English, there is a distinction between **security** and **safety**. Many languages do not distinguish between these concepts as clearly. For example, in Spanish *la seguridad* is the most common translation for both.

Both safety and security describe a system's resistance and resilience from threats. The key difference is that safety deals with all types of threats, including (typically non-human) **threats that do not adapt**. These are typically not called "attackers" but threats or dangers.

For example, airplanes are designed for safety in case of mechanical failures. Physical forces like friction and gravity will always be the same. Buildings are designed to withstand extreme events like hurricanes or earthquakes. These risks may change over time, but they do not intentionally adapt to evade what safety engineers are doing.

Security, on the other hand, focuses on **attackers that adapt to system design.** Airport security provides a long history of this. Early airport screening didn't check passengers' shoes. After the infamous "shoe bomber" (Richard Reid), security agents began checking shoes for explosive material. Attackers next attempted to use liquid explosives, leading to a ban on taking water bottles through security, and so on.

A few cases are borderline, such as wild animals. If you're designing bear-proof canisters for hikers, bears may have some ability to adapt to different designs. Pathogens like viruses and bacteria adapt to human medical interventions via evolution, eventually developing resistance to vaccines and antibiotics.

Security is sometimes viewed as a subset of safety. Safety is minimization of all risks, security just those that are active/adaptable. You can see why the terms are sometimes used interchangeably and many languages don't make a strong distinction.

A note on terminology & adversarial thinking

The terms *attacker* and *defender* are used neutrally by security engineers without making a value judgment about either party. Consider a government agency trying to build a system for blocking access to certain content on the Internet. Our sympathies might vary if the content is a banned political party's website in an autocracy vs. child sexual abuse images in a liberal democracy. But in either case we call the government the defender and people trying to access the content the attacker.

It's normal for security engineers to speak dispassionately about potential attack and defense strategy even in extreme cases: "one way to sneak a bomb onto a plane is...". This *adversarial thinking* is necessary to design a secure system. Security engineers consider it normal to have a discussion about optimal strategies for destroying an airliner and take it for granted that they don't want to actually do this. It's important to remember though that outside of a security engineering context, these conversations can be offensive or upsetting.

Defining security: perceptions vs. reality

Security is defined in many ways in different contexts and the different definitions illustrate the complexity of building secure systems.

There is a distinction between security as the actual **resistance of a system to potential harm** versus security as a **perception of protection from harm**. A classic example of the second notion is sleeping under a heavy blanket. Psychologically, this provides a nice feeling of security, even though the blanket is not protecting you from anything. Other security measures are useful but don't provide a warm, fuzzy feeling and are even quite annoying, like fraud detection analytics which might freeze your credit card while you're traveling.

The distinction between the two notions is not binary. Some security measures provide both. For example, locking your door when you arrive home provides some feeling of protection from the outside world. It also actually makes it more difficult for potential intruders to get inside (though in most cases probably far less than people believe).

Deceptive security mechanisms: security theater and deterrents

While almost everybody understands rationally that sleeping under a blanket provides no real security, sometimes elaborate security mechanisms are designed to appear effective to the general public although security engineers know they have little effect. This is called **security theater** (as coined by Bruce Schneier) or sometimes **placebo security**. Airport security is often cited as an example of security theater. An infamous example was the use of "puffer machines" (ETPs) in the US in the mid-2000s. They were consistently found to have no effectiveness at bomb detection under real operating conditions. It's been speculated they were kept in operation for years because the high-tech-looking machines helped ease public fear about flying after September 11th.

Deception about the effectiveness of a security mechanism can also be useful if potential attackers don't understand the true effectiveness and are scared off. This is called a **deterrent effect**. A common example is shops which mount fake security cameras to deter shoplifters. Or more classically, scarecrows or plastic owls can scare away unwanted birds.

Finally, the security industry is also littered with **snake oil**: products deceptively claiming to provide security which in fact are useless. These may fool the owner, but not potential attackers. The ADE 651 machine was widely sold in Iraq, though it literally did nothing except look like a bomb detector. Many argue that anti-virus software in the modern era is largely snake oil. Snake oil might offer some psychological benefit as security theater. The distinction is that with security theater the operator knows the mechanism is ineffective; with snake oil they've been deceived.

Attacker incentives

A critical aspect of security is incentives. **Attackers will give up** if security is strong enough that the potential gain from attacking is less than the cost. Nobody has seriously attempted to steal gold from Fort Knox in many decades, because the security is extremely strong. Again, safety threats are invariant: earthquakes don't stop happening in cities with robust building codes.

There are three common classes of attackers to worry about (the **three Ts**).

- **Thieves** seek financial gain
- **Trolls** seek to annoy or harass others, sometimes for attention or notoriety
- **Terrorists** seek to cause violence or destruction for political ends

This list isn't exhaustive but most attackers fall into one of these three groups. Note that we haven't distinguished between attackers' capabilities yet, only their incentives. You can have professional thieves, amateur terrorists and vice-versa.

Security and game theory

You'll often see the word *game* in security engineering, even if the "game" has life-or-death consequences. The language of **game theory** helps formalize a few basic concepts of security.

Security is a game between an attacker and defender, with each defined by a utility function. Security games are typically not zero-sum. The money spent by the defender on security mechanisms is not gained by the attacker. Attackers' gains might be the

direct loss of the defender in the case of thieves, but there's often what economists call a **deadweight loss:** a stolen item is usually more valuable to the original owner than to the thief who steals it. In the case of trolls or thieves, the utility gained by the attacker from causing damage can be very small relative to the potentially huge costs to the defender from successful attacks.

Attackers often (but not always) enjoy a significant edge in security games because the defender has to commit to a strategy first. This **last-mover advantage** means the attacker can observe the defender's strategy and then choose the best response.

Finally, security games are almost always **iterated**, meaning they are played many times. This allows both sides to evolve their strategy choices in response to observations about the opponent.

Policy vs. mechanisms

In game-theoretic terms, both sides choose a **strategy** or **strategy portfolio.** In traditional security engineering parlance, strategy is divided into **security policy** and **security mechanisms**.

Security policy specifies which properties a system should be designed to enforce:

- **Invariants**. Properties that should always be true. Example: passengers should always be able to exit from a subway platform to the street within 60 seconds.
- **Security goals**. Typically measurable outcomes that a defender would like to minimize/maximize. Example: fare evasion should be at most 1% of subway passengers.

After choosing security policies, defenders design **security mechanisms** which are the techniques used to enforce the desired policy. For example, if keeping fare evasion below 1% is the security policy, installing metal gates at all stations is a security mechanism that can have some effect towards this goal. For the security policy of passengers being able to exit within 60 seconds in an emergency, a suitable mechanism might be building emergency exits for every 200 feet of tunnel.

A useful design concept for defenders is **defense-in-depth**, or multiple layers of security mechanisms an attacker must bypass to violate security policy. Example: would-be airline hijackers must go through security checkpoints and then break an armored door. More advanced modeling can involve an **attack tree** drawing all of the

(potentially non-linear) dependencies between security mechanisms that an attacker must bypass to violate security policy.

Threat modeling

An important part of designing security policy and mechanisms is **threat modeling**: thinking through all potential attackers to the system and what attack strategies they might employ. For example, in a subway system, threats may range from simple fare evasion (riding without paying) to causing mass casualties by derailing a train.

The process of threat modeling can be as informal as having engineers brainstorming around a white board for an hour. It can also be a months-long process involving special tools and outside consultants. There is an entire textbook just on threat modeling. Engineers at Microsoft even developed a card game called *Elevation of Privilege* used to assist in threat modeling.

The most important thing in threat modeling is to be open-minded and imaginative. It's worth thinking through exotic or farcical threat ideas. The US Department of Homeland Security took an interesting approach in 2009: they hired a group of science-fiction writers to help create a threat model for airport security. The idea is that science-fiction writers can help think up new threats because they are used to thinking creatively (excluding the writing staff from *Star Wars Episode IX*).

The essential skill of threat modeling is to **think like an attacker**. This idea is quite old, going back at least to Sun Tzu in the 6th century BCE:

> *To know your enemy, you must become your enemy.*

The modern interpretation is that to be good at threat modeling, it helps to practice always thinking of how you might exploit any given system. Or more directly, it's common for companies to hire experienced outside hackers directly to try and test their systems (usually called **red teams** or **penetration testers**).

Evolution of secure systems

Once defenders have chosen security policies and mechanisms, attackers get to observe their choices and choose their own attack strategies. It may not be obvious at

first what the defender's choices are, requiring the attacker to first **reverse engineer** the design of a system's security mechanisms. We'll come back to this point later, but defenders should assume the attacker will eventually learn how the security mechanisms work.

After observing attackers, defenders can also adapt (or **patch**) their system. This may be quick for some systems (updating a website) or take many years in other systems (manufacturing new hardware to replace old devices). The ability to quickly create and deploy patches can itself be an important security mechanism.

There can be many iterations of attack/patch between the attacker and a defender. The speed of iterations can vary greatly by context. Web sites can be patched almost as quickly as defenders can write new code. Traditionally, manufactured objects like cars or door locks were impossible to patch without complete replacement. The trend is towards everything being patchable as more objects run software and have an Internet connection to receive updates (the **Internet of things**).

There are a few possible outcomes to the iterated game:

- **Attackers win** if the defenders run out of effective strategies to prevent attack and stop updating the security mechanism. Usually defenders eventually end up changing security policy if it is not enforceable. For example, the US eventually gave up on alcohol prohibition after failing to stop widespread bootlegging.
- **Defenders win** if attackers no longer invest the effort to attack. Instead they often focus their energy on attacking other systems. For example, fewer attackers are sending mass email spam anymore due to improved filtering and relatively little effort is going into designing new anti-filter strategies. Instead attackers are now more interested in social media spam/influence hacking.
- **Arms races** occur when no clear winner emerges over time and both sides continue to update their strategy. For example, counterfeiting currency has gone back and forth for centuries, with many generations of new banknotes issued.

It can be argued that all security games look like an arms race until they don't because a winner emerges. Some arms races (such as counterfeiting money) have been running for many millennia now. But there's always a chance that technology will eventually produce a winner. In the case of counterfeiting money, practical quantum money could lead to defenders winning. Unfortunately it doesn't exist yet.

Finally, it's important to realize that **winning is not always an optimal outcome for the defender**. Winning may represent wasteful over-spending on security. For example, the

US spent over $600 billion on homeland security in the decade after the September 11th attacks. The country did not suffer an airline hijacking or major act of foreign terrorism on domestic soil in that time period. Arguably the country "won" the security game vs. terrorists, at least in the context of airline security. But it's not as clear that this expenditure was worthwhile.

Policy and politics

Another distinction between policy and mechanism in security is that policy is set at a higher level within an organization. Within a company, policy may be set by upper management or even the CSO/CISO or CEO based on a company's financial goals. Within a country, policy is set by politicians (both words share the Greek root *polis*).

A classic blunder of security design is to **set security mechanisms as policy**. For example, a policy of "there should be an armed guard at every subway station" is likely a mistake. Having an armed guard is not a useful end unto itself. It may be a useful mechanism to achieve some security policy, but should not be encoded as a policy itself. If you find yourself working in a policy-setting role, always try to leave flexibility for security engineers to choose the best mechanism possible.

Security engineers usually only get to choose mechanisms, with policy dictated from above. If you're lucky as an engineer, you might voice some feedback ("this policy is impossible to enforce"), but typically you will have to do your best to design mechanisms based on a policy that's given to you.