on the Internet, making it appear that that node was responsible for the original traffic. There are, however, many other types of data that can be collected that are harder to hide. Even the patterns of how individual users browse and click through a website can be used to identify them.

This question of how *can* we identify and authenticate online activities is a different question from how *should* we. You may not have wanted your Social Security number to be revealed at a party. Or that dog might have preferred its identity remain secret, at least until the two of you had gone out on a few more online dates.

For the purposes of cybersecurity, the bottom line is that digital identity is a balance between protecting and sharing information. Limiting acquired information is not only good for privacy, it can prevent others from gaining information for more sophisticated authentication fraud. At the same time, each system has incentives to maximize the amount of data it collects, as well as use that data for its own goals.

What Do We Mean by "Security" Anyway?

There's an old joke in the security industry about how to secure any computer: Just unplug it.

The problem is not only that the joke is becoming outdated in an era of wireless and rechargeable devices, but once a machine is plugged in, there are practically an infinite number of ways its use might deviate from its intended purpose. This deviation is a malfunction. When the difference between the expected behavior and actual behavior is caused by an adversary (as opposed to simple error or accident), then the malfunction is a "security" problem.

Security isn't just the notion of being free from danger, as it is commonly conceived, but is associated with the presence of an adversary. In that way, it's a lot like war or sex; you need at least two sides to make it real. Things may break and mistakes may be made, but a cyber problem only becomes a cybersecurity issue if an adversary seeks to gain something from the activity, whether to obtain private information, undermine the system, or prevent its legitimate use.

To illustrate, in 2011 the Federal Aviation Administration ordered nearly half of US airspace shut down and more than 600 planes grounded. It seemed like a repeat of how American airspace was shut down after the 9/11 attacks. But this incident wasn't a security issue, as there was no one behind it. The cause was a software glitch in a single computer at the Atlanta headquarters building. Take the same situation and change the glitch to a hack: that's a security issue.

The canonical goals of security in an information environment result from this notion of a threat. Traditionally, there are three goals: Confidentiality, Integrity, Availability, sometimes called the "CIA triad."

Confidentiality refers to keeping data private. Privacy is not just some social or political goal. In a digital world, information has value. Protecting that information is thus of paramount importance. Not only must internal secrets and sensitive personal data be safeguarded, but transactional data can reveal important details about the relationships of firms or individuals. Confidentiality is supported by technical tools such as encryption and access control as well as legal protections.

Integrity is the most subtle but maybe the most important part of the classic information security triumvirate. Integrity means that the system and the data in it have not been improperly altered or changed without authorization. It is not just a matter of trust. There must be confidence that the system will be both available and behave as expected.

Integrity's subtlety is what makes it a frequent target for the most sophisticated attackers. They will often first subvert the mechanisms that try to detect attacks, in the same way that complex diseases like HIV-AIDS go after the human body's natural defenses. For instance, the Stuxnet attack (which we explore later in Part II) was so jarring because the compromised computers were telling their Iranian operators that they were functioning normally, even as the Stuxnet virus was sabotaging them. How can we know whether a system is functioning normally if we depend on that system to tell us about its current function?

Availability means being able to use the system as anticipated. Here again, it's not merely the system going down that makes availability a security concern; software errors and "blue screens of

death" happen to our computers all the time. It becomes a security issue when and if someone tries to exploit the lack of availability in some way. An attacker could do this either by depriving users of a system that they depend on (such as how the loss of GPS would hamper military units in a conflict) or by merely threatening the loss of a system, known as a "ransomware" attack. Examples of such ransoms range from small-scale hacks on individual bank accounts all the way to global blackmail attempts against gambling websites before major sporting events like the World Cup and Super Bowl.

Beyond this classic CIA triangle of security, we believe it is important to add another property: resilience. Resilience is what allows a system to endure security threats instead of critically failing. A key to resilience is accepting the inevitability of threats and even limited failures in your defenses. It is about remaining operational with the understanding that attacks and incidents happen on a continuous basis. Here again, there is a parallel to the human body. Your body still figures out a way to continue functioning even if your external layer of defense—your skin—is penetrated by a cut or even bypassed by an attack like a viral infection. Just as in the body, in the event of a cyber incident, the objective should be to prioritize resources and operations, protect key assets and systems from attacks, and ultimately restore normal operations.

All of these aspects of security are not just technical issues: they are organizational, legal, economic, and social as well. But most importantly, when we think of security we need to recognize its limits. Any gain in security always involves some sort of trade-off. Security costs money, but it also costs time, convenience, capabilities, liberties, and so on. Similarly, as we explore later on, the different threats to confidentiality, availability, integrity, and resiliency each require different responses. Short of pulling the plug, there's no such thing as absolute security.

What Are the Threats?

It sounds odd that reporters took a passenger jet to Idaho just to watch a cyberattack, but that's exactly what happened in 2011.

Worried that the public did not understand the magnitude of growing cyberthreats, the Department of Homeland Security

flew journalists from around the country to the Idaho National Laboratory. Only four years earlier, the INL, an incredibly secure and secretive facility that houses a Department of Energy's nuclear research facility, had conducted a top-secret test to destroy a large generator via cyberattack. In 2011, in an effort to raise awareness about cyberthreats, government experts not only declassified a video of the 2007 test, but held a public exercise for journalists to "watch" a faked cyberattack on a mock chemical plant. The government wanted to show that even their own experts couldn't prevent a team of hired hackers (known as a "red team") from overwhelming the defenses of a critical facility.

This episode is a good illustration of how those who professionally think and talk about cybersecurity worry that their discussions of threats are ignored or downplayed. Frustrated, the result is that they resort to turning the volume up to the proverbial 11 à la *Spinal Tap*, conducting outlandish exercises and only talking about the matter in the most extreme ways that then echo out into the media and public. Indeed, following a series of warnings by US government officials, by 2013 there were over half a million online references in the media to "cyber Pearl Harbor" and another quarter million to a feared "cyber 9/11."

The complacency these experts worry about stems in part from our political system's reluctance to address difficult, complex problems in general, and cybersecurity in particular. But this kind of tenor also feeds into a misunderstanding of the threats. For example, three US senators sponsored a large cybersecurity bill in the summer of 2011, and so wrote an op-ed in the *Washington Post* urging support for their legislation. They cited a series of recent, high-profile attacks, including those against the Citigroup and RSA companies and the Stuxnet worm's attack on Iranian nuclear research. The problem is that these three cases reflected wildly different threats. The Citigroup attack was about financial fraud. The RSA attack was industrial theft, and Stuxnet was a new form of warfare. They had little in common other than they involved computers.

When discussing cyber incidents or fears of potential incidents, it is important to separate the idea of vulnerability from threat. An unlocked door is a vulnerability but not a threat if no one wants to enter. Conversely, one vulnerability can lead to many threats: that unlocked door could lead to terrorists sneaking in a bomb,

competitors walking out with trade secrets, thieves purloining valuable goods, local hooligans vandalizing property, or even cats wandering in and distracting your staff by playing on the keyboards. The defining aspects of threats are the actor and the consequence.

The acknowledgment of an actor forces us to think strategically about threats. The adversary can pick and choose which vulnerability to exploit for any given goal. This implies that we must not only address a range of vulnerabilities with respect to any given threat, but also understand that the threat may evolve in response to our defensive actions.

There are many kinds of bad actors, but it is too easy to get lulled into using media clichés like "hackers" to lump them all together. An actor's objective is a good place to start when parceling them out. In the variety of attacks cited by the senators above, the Citigroup attackers wanted account details about bank customers with an ultimate goal of financial theft. In the attack on RSA, the attackers wanted key business secrets in order to spy on other companies. For Stuxnet (a case we'll explore further in Part II), the attackers wanted to disrupt industrial control processes involved in uranium enrichment, so as to sabotage the Iranian nuclear program.

Finally, it is useful to acknowledge when the danger comes from one of your own. As cases like Bradley Manning and WikiLeaks or Edward Snowden and the NSA scandal illustrate, the "insider threat" is particularly tough because the actor can search for vulnerabilities from within systems designed only to be used by trusted actors. Insiders can have much better perspectives on what is valuable and how best to leverage that value, whether they are trying to steal secrets or sabotage an operation.

It is also important to consider whether the threat actor wants to attack *you*, or just wants to attack. Some attacks target specific actors for particular reasons, while other adversaries go after a certain objective regardless of who may control it. Untargeted malicious code could, for example, infect a machine via e-mail, search for stored credit card details of anyone, and relay those details back to its master without any human involvement. The key difference in these automated attacks is one of cost, both from the attacker's and the defender's perspective. For the attacker, automation hugely reduces cost, as they don't have to invest in all the tasks needed, from selecting the victim to identifying the asset to coordinating the attack. Their attack costs roughly the same no matter how many victims they get. A targeted attack, on the other hand, can quickly scale up in costs as the number of victims rises. These same dynamics shape the expected returns. To be willing to invest in targeted attacks, an attacker must have a higher expected return value with each victim. By contrast, automated attacks can have much lower profit margins.

The good news is that there are only three things you can do to a computer: steal its data, misuse credentials, and hijack resources. Unfortunately, our dependence on information systems means that a skilled actor could wreak a lot of damage by doing any one of those. Stolen data can reveal the strategic plans of a country or undermine the competitiveness of an entire industry. Stolen credentials can give the ability to change or destroy code and data, changing payrolls or opening up dams, as well as the ability to cover tracks. Hijacking resources can prevent a company from reaching customers or deny an army the ability to communicate.

In the end, there are many things that can happen, but they have to be caused by someone. Threats should be assessed by understanding potential bad actors, what they are trying to do, and why.

And you shouldn't need to fly all the way to Idaho to learn that.

One Phish, Two Phish, Red Phish, Cyber Phish: What Are Vulnerabilities?

In 2011, London police confronted a mysterious and unusual spike in car thefts. The odd thing wasn't just that so many cars were being stolen, over 300 in all, but that the cars were all of a particular brand, new BMWs. And, second, the thieves were somehow stealing hundreds of cars equipped with some of the most advanced car security systems in the world, without activating the alarms.

What the police soon figured out by watching hidden security camera footage of the thieves in action was that the robbers had figured out how to use the car's advanced technology against itself. First, they used radio frequency jammers to block the signal of a car's electronic key. Instead of the car owner locking the doors as they walked away, the doors would remain unlocked. Once in the car, the thief would then plug into the OBD-II connector (the electronic port that mechanics use to diagnose your car's problems) and

then use that to obtain the car's unique key fob digital ID. Next, the thief would reprogram a blank electronic key to correspond with the car's ID. Then they simply drove away, with the owner of the advanced luxury car none the wiser. It all took only a few minutes. These vulnerabilities led to so many thefts that police resorted to leaving paper leaflets on all BMWs parked in London warning them of the danger.

The case of the lost luxury cars is a good illustration of how building a complex system can create new openings and hidden vulnerabilities that bad guys can try to exploit. Different vulnerabilities allow an attacker to achieve different goals. In some cases, it might be the ability to read confidential data. Or the goal could be the ultimate prize—compromise of the entire system. When the attacker has such "root access," the ability to execute any command, the victim is completely vulnerable, or what hackers call "pwned" (An apocryphal story is that a hacker meant to type that a target was now "owned." But he typed too fast, mistakenly hit the *p* key right next to the *o*, and a cool term was born.)

Often the easiest way to gain control of the system is simply to ask. A time-honored tradition for breaking into systems from hacking's early days is to call up a low-level employee, claim to be from technical support, and ask for the person's password. This falls into the category of what is known as "social engineering," manipulating people into revealing confidential information and thereby helping the attacker. The manipulation can take many forms, often with the attacker trying to set up a scenario designed to encourage cooperation through psychological mechanisms. Fear is a powerful motivator. When a user's computer displays a message threatening to expose activities on a pornographic website, fear of exposure can motivate payment. More often, however, users just follow social cues. In our daily lives, we regularly encounter problems that need fixing, like a program that won't close until you just "click here," or people who need our help, like your Aunt Suzy who somehow got robbed in Iceland and needs you to wire her money via Bangkok.

A particularly common form of social engineering is the "phishing" attack. Phishing e-mails look like official e-mails from the victim's bank, employer, or some other trusted entity. They claim to require some action by the victim, perhaps to correct an account error or see a message on Facebook, and fool victims into visiting

a web page where they are asked to enter their credentials. If the victim enters his or her account details, the attacker can now do anything with that information, from transfer money to read confidential e-mails. The phony credentials web page may have a URL that looks similar to the authentic one. If you don't look closely, maybe www.paypai.com looks like www.paypal.com. In sophisticated phishing attacks, the fake page may also actually log the user into the real website to minimize the chance of detection.

One of the most challenging subsets of these "phishing" attacks is known as "spear phishing." These target not just networks but key individuals inside those networks. It's the difference between you, along with scores of others people, receiving an e-mail from that kind Nigerian prince who just needs your bank account information, versus receiving an e-mail that looks exactly like it's from your mother. This is a good illustration of the difference between the automated and targeted threats you read about in the last section. Such specialized attacks require prior intelligence gathering to figure out how to trick a particular person and are mostly reserved for prime targets.

Attackers also prey on systems that have ignored basic precautions, such as products that have default login names and passwords, which users often forget to change. Most home wireless routers have default passwords that a shocking number of users leave in place. Find the right one and it is easy to steal a neighbor's Wi-Fi and eavesdrop on their conversations. This kind of vulnerability can also be created by product manufacturers that don't prioritize security enough or fail to factor in the likely human errors and even laziness of their customers. For instance, Microsoft's database product MS-SQL 2005 shipped without an administrator password, so that any user could control the entire database until an admin password was set. Other situations involve systems that have features that may be convenient but represent real security vulnerabilities, like that of the BMW remote access keys.

Applications can also create vulnerabilities if they are misconfigured. In one study, researchers at Dartmouth searched peer-to-peer file-sharing services, where users share specific files from their own computers with others, usually entertainment files like movies or TV shows. Because of misconfigured settings, in addition to users sharing episodes of *Game of Thrones*, a large number

also had unintentionally shared personal bank statements and tax documents. A similar study found that large financial institutions were unintentionally leaking highly sensitive internal documents through misconfigured applications.

Another vector is mistakes in the systems themselves—software vulnerabilities—that are exploited by more advanced attackers. It is practically impossible to build a modern IT system without some hidden vulnerabilities waiting to be discovered. Modern operating systems have millions of lines of code and have hundreds of subcomponents that interact. An attacker's goal is to find some chink in the armor of this code, where the system does not behave precisely as designed and exploit that weakness. An attack that exploits a previously unknown vulnerability is known as a "zero day." The term comes from the notion that the attacks take places on the zeroth day of the awareness that the rest of the world has of these vulnerability and thus before a patch to fix it can be implemented.

There are different types of vulnerabilities with different ways of exploiting them, but a common approach is to find some way of tricking the victim's computer into executing the attacker's commands rather than the intended program's. A key is that most computer systems treat data as both information to be processed and commands to be executed. This principle is foundational to the very idea of the modern computer, but also a major source of insecurity. A good illustration is a SQL (pronounced "sequel") injection, one of the most common ways a website is attacked. Many web applications are built on Structured Query Language (SQL), a type of programming language used to manage data. It's a highly effective system that dates back to the 1970s. But an attacker, instead of entering a name and address as requested, can enter specifically crafted commands that the database will read and interpret as program code, rather than just data to be stored. These commands can be used to learn about the database, read data, and create new accounts. In some cases, access can be used to discover and change security settings on the server, allowing the attacker to control the entire web system. As we explore later in the Part II section on hactivists, the Anonymous group used this kind of attack to penetrate the security firm HB Gary and share its embarrassing secrets with the world.

Beyond attacking applications, attackers can also exploit vulnerabilities in code at the system level. A common vulnerability is the

buffer overflow. Computers use memory to store data and instructions. If a program can be tricked into writing inputted data that is larger than expected, it can spill over the allocated "buffer," or storage area, and overwrite the space where the computer stores the next instruction to be executed. If that newly written memory space is then read and interpreted by the computer, the program can break or follow the attacker's instructions. Once the program executes arbitrary instructions, the attacker can effectively gain control of the system. In essence, it follows the same principle as the SQL attack, where the computer interprets data as instructions, but now it takes place at the system memory level.

Designing these types of attacks requires a great deal of skill and experience, but once the vulnerability has been exploited it is relatively easy to package. This "exploit" is a piece of software or set of commands that can take advantage of the vulnerability. And that is where cyber risk takes on a whole new level of concern, as it allows other, less sophisticated attackers in on the action. It's as if the master thief now writes a book about safecracking that comes with a handy-dandy set of tools.

Malicious software, or "malware," is a prepackaged exploitation of a vulnerability. There is often a "payload" of instructions detailing what the system should do after it has been compromised. Some types of malware contain instructions for reproduction, in order to spread the attack. "Worms" spread themselves automatically over the network. In some cases, this can be sufficient to cause drastic harm: many of the worms that attacked Microsoft Windows in the late 1990s and early 2000s had no direct malicious effect but still overwhelmed corporate networks because they tried to send out an exponentially large number of copies. One worm even sought to patch vulnerable computers, a "good worm," but still managed to cripple networks. Other vulnerabilities have been exploited to allow the attacker to capture valuable personal data or, in an anarchistic turn, just destroy data on the victim's computer.

Malware can also be spread over the Web via "drive-by" attacks, where the victim's only mistake is visiting the wrong website. Such attacks exploit vulnerabilities in the web browser or in the many components and add-ons that web browsers use to take advantage of sophisticated websites. The attacker first compromises the web server and then simply attempts to exploit vulnerabilities in any

browser that requests files from that website. Drive-by attackers often target groups by going after websites used by specific communities, a so-called "watering hole" attack (taken from the ideas that smart lions don't chase after their prey across the African savannah, but rather just wait for them all to come to the watering hole). For example, a group out to steal secrets from a US defense company indirectly targeted it by compromising the website of a popular aerospace magazine that many employees read. In one case, a watering hole attack infected five hundred accounts in a single day.

More recently, malware has been used not only to take control of a computer system but to keep control of that computer, in order to exploit its computational and network resources. By capturing victims' systems and coordinating their behavior, attackers can assemble armies of "zombie" computers. Millions of machines can be controlled by a single actor through a range of different command and control mechanisms. These are referred to as "botnets," and most computer users will never know if they are part of one.

Botnets are powerful resources for a host of nefarious activities. Regular access to the victims' machines allows monitoring to capture valuable data. Botnet controllers can also leverage the network connections of their victims' systems to send spam, host websites to sell illegal products, or defraud online advertisers. Perhaps most insidiously, botnets can launch a "distributed denial of service" (DDoS) attack.

DDoS attacks target the subsystems that handle connections to the Internet, such as web servers. Their vulnerabilities are based on the principle that responding to an incoming query consumes computational and bandwidth resources. If someone were to call your phone incessantly, you would first lose the ability to concentrate and then lose the ability to use your phone for any other purpose. Similarly, in the cyber world, if the attacker can overwhelm the connection link, the system is effectively removed from the Internet. It is fairly easy to defend against a single attacker from a fixed source: one just has to block the sender, just like blocking an annoying caller's number (but never your mother's. Never.). In a distributed denial-of-service attack, the attacker uses a botnet of thousands or even millions to overwhelm the victim. It's the equivalent of having thousands or even millions of people trying to call your phone. Not only would you get nothing done, but the calls you actually want to receive wouldn't easily get through.

This kind of power, and the fact that such attacks are fairly overt and obvious, means that DDoS attacks are often linked to some other goals. Criminal gangs may go to a website and threaten to take it offline unless they pay for "protection." ("That's a nice website you've got there. It'd be a real shame if anything were to... happen.") Or they may also be used as a diversion, to overwhelm the attention and defenses of the victim while going after data elsewhere. They are also increasingly common as a form of political protest or even suppression. During the initial stages of the crisis in Syria in 2011, supporters of the Syrian regime shared DDoS tools to attack critics of the government and news organizations that covered the growing violence.

The bottom line is that vulnerabilities exist on every type of information system in cyberspace. Despite how scary they sound, many of them are not new. For instance, the common buffer overflow attack was first developed in the 1970s and almost brought down the adolescent Internet in the 1980s. By 1996, a detailed how-to guide appeared in a hacker's magazine.

As threats evolve, so too must our responses to them. Some can be mitigated with small changes in behavior or tweaks in code, while whole classes of vulnerabilities can be prevented only by developing and implementing new technologies. Other vulnerabilities are simply a structural consequence of how we use systems. As we explore in Part III, how we navigate these challenges comes down to accepting that bad guys are out to exploit these vulnerabilities and then developing the best possible responses that allow us to keep benefiting from the good parts of the cyber age.

Or you could opt out, sell your advanced luxury car and ride the bus instead. Just make sure you don't check the bus schedule online.

How Do We Trust in Cyberspace?

There is perhaps no more important duty for a citizen than to vote, and no part of the voting process is more important than preserving the integrity of that vote. And yet the Founding Fathers of American democracy didn't imagine a world of computerized voting machines, nor one in which Pac-Man might chomp his way into an election.