



## Cybersecurity: Crash Course Computer Science #31

Crash Course: Computer Science

<https://youtube.com/watch?v=bPVaOIJ6In0>

<https://nerdfighteria.info/v/bPVaOIJ6In0>

Hi, I'm Carrie Anne, and welcome to Crash Course Computer Science.

Over the last three episodes, we've talked about how computers have become interconnected, allowing us to communicate near instantly across the globe. But not everyone who uses these networks is going to play by the rules or have our best interests at heart.

Just as how we have physical security like locks, fences, and police officers to minimize crime in the real world, we need Cybersecurity to minimize crime and harm in the virtual world.

Computers don't have ethics. Give them a formally specified problem and they'll happily pump out an answer at lightning speed. Running code that takes down a hospital's computer systems is no different to a computer than code that keeps a patient's heart beating.

Like the Force, computers can be pulled to the light side or the dark side. Cybersecurity's like the Jedi Order, trying to bring peace and justice to Cyberspace.

[Intro Music]

The scope of Cybersecurity evolves as fast as the capabilities of computing. We can think of it as a set of techniques to protect the secrecy, integrity, and availability of computer systems and data against threats. Let's unpack those three goals.

Secrecy, or confidentiality, means that only authorized people would be able to access or read specific computer systems and data. Data breaches where hackers reveal people's credit card information is an attack on secrecy.

Integrity means that only authorized people should have the ability to use or modify systems and data. Hackers who learn your password and send emails masquerading as you is an integrity attack.

And availability means that authorized people should always have access to their systems and data. Think of Denial of Service attacks, where hackers overload a website with fake requests to make it slow or unreachable for others. That's attacking the service's availability.

To achieve these three goals, security experts start with the specification of who your enemy is at an abstract level called a Threat Model. This profiles attackers: their capabilities, goals, and probable means of attack. What's called, awesomely enough, an Attack Vector.

Threat Models let you prepare against specific threats rather than being overwhelmed by all the ways hackers could get to your systems and data. And there are many many ways.

Let's say you want to secure physical access to your laptop. Your Threat Model is a nosy roommate. To preserve the secrecy, integrity, and availability of your laptop, you could keep it hidden in your dirty laundry hamper.

But if your Threat Model is a mischievous younger sibling who knows your hiding spots, then you'll need to do more. Maybe lock it in a safe.

In other words, how a system is secured depends heavily on who it is being secured against. Of course, Threat Models are typically a bit more formally defined than just "nosy roommate". Often, you'll see Threat Models specified in terms of technical capabilities.

For example, "someone who has physical access to your laptop along with unlimited time."

With a given Threat Model, security architects need to come up with a solution that keeps a system secure. As long as certain assumptions are met, like no one reveals their password to the attacker.

There are many methods for protecting computer systems, networks, and data. A lot of security boils down to two questions: who are you?, and what should you have access to?

Clearly, access to should be given to the right people, but refused to the wrong people. Like bank employees should be able to open ATM's to restock them, but not me. Because I'd take it all. All of it. That ceramic cat collection doesn't buy itself!

So to differentiate between right and wrong people, we use authentication, the process by which a computer understands who it's interacting with. Generally there are three types, each with their own pros and cons. 'What You Know', 'What You Have', and 'What You Are'.

'What You Know' authentication is based on a knowledge of a secret that should be known only to the real user and the computer. For example, username and password. This is the most widely used today because it's the easiest to implement.

But it can be compromised if hackers guess or otherwise come to know your secret. Some passwords are easy for humans to figure out like '123456' or 'qwerty', but there are also ones that are easy for computers.

Consider the pin '2580'. This seems pretty difficult to guess, and it is for a human, but there are only 10,000 possible combinations of 4-digit pins. A computer can try entering 0000 then try 0001 and then 0002 all the way up to 9999 in a fraction of a second.

This is called a Brute Force Attack because it just tries everything. There's nothing clever to the algorithm. Some computer systems lock you out or have you wait a little after, say, three wrong attempts. That's a common and reasonable strategy and it does make it harder for less sophisticated attackers.

But think about what happens if attackers have already taken over tens of thousands of computers, forming a bot-net. Using all of these computers, the same pin, '2580', can be tried on many of tens of thousands of bank accounts simultaneously.

Even with just a single attempt per account, they'll very likely get into one or more that just happen to use that pin. In fact, we've probably guessed the pin of someone watching this video.

Increasing the length of pins and passwords can help, but even 8-digit pins are pretty easy to crack. This is why so many websites now require you to use a mix of upper- and lower-case letters and special symbols. It explodes the number of possible password combinations.

An 8-digit numerical pin only has 100 million combinations. Computers eat that for breakfast. But an 8 character password with all those funky things mixed in has more than 600 trillion combinations.

Of course, these passwords are hard for us mere humans to remember, so a better approach for websites is to let us pick something more memorable. Like three words joined together.

"Green brothers rock" or "pizza tasty yum".



## Cybersecurity: Crash Course Computer Science #31

Crash Course: Computer Science

<https://youtube.com/watch?v=bPVaOIJ6In0>

<https://nerdfighteria.info/v/bPVaOIJ6In0>

English has about 100 thousand words in use, so putting three together would give you roughly one quadrillion possible passwords. Good luck trying to guess that!

I should also note here that using non-dictionary words is even better against more sophisticated kinds of attacks, but we don't have time to get into that here. Computerphile has a great video on choosing a password, link in the doobly-doo.

'What you Have' authentication on, on the other hand, is based on possession of a secret token that only the real user has. An example is a physical key and lock. You can only unlock the door if you have the key.

This escapes this problem of being guessable and they typically require physical presence, so it's much harder for remote attackers to gain access. Someone in another country can't gain access to your front door in Florida without getting to Florida first.

But 'What You Have' authentication can be compromised if an attacker is physically close. Keys can be copied, smart phones stolen, and locks picked.

Finally, 'What You Are' authentication is based on you. You authenticate by presenting yourself to the computer. Biometric authenticators like fingerprint readers and iris scanners are classic examples. These can be very secure, but the best technologies are still quite expensive.

Furthermore, data from sensors varies over time. 'What You Know' and 'What You Have' authentication have the nice property of being deterministic-either correct or incorrect. If you know the secret or have the key, you're granted access 100% of the time. If you don't, you get access 0% of the time.

Biometric authentication is probabilistic. There's some chance the system won't recognize you. Maybe you're wearing a hat or the lighting is bad. Worse, there's some chance the system will recognize the wrong person is you, like your evil twin. Of course, in production systems, these chances are low, but not zero.

Another issue of biometric authentication is that it can't be reset. You only have so many fingers, so what happens if an attacker compromises your fingerprint data? This could be a big problem for life.

And recently, researchers showed it's possible to forge your iris just by capturing a photo of you, so that's not promising either. Basically, all forms of authentication have strengths and weaknesses and all can be compromised in one way or another.

So security experts suggest using two or more forms of authentication for important accounts. This is known as two-factor or multi-factor authentication. An attacker maybe able to guess your password or steal your phone: but it's much harder to do both.

After authentication comes Access Control. Once the system knows who you are, it needs to know what you should be able to access. And for that, there's a specification of who should be able to see, modify, and use what.

This is done through Permissions or Access Control Lists, which describe what access each user has for every file, folder and program on a computer.

Read Permission allows a user to see the contents of a file. Write Permission allows a user to modify the content and Execute Permission allows a user to run a file like a program.

For organizations with users at different levels of access privilege like a spy agency, it's especially important for Access Control Lists to be configured correctly to ensure secrecy, integrity, and availability.

Let's say we have three layers of access: public, security, and top secret.

The first general rule of thumb is that people shouldn't be able to 'read up'. If a user is only cleared to read secret files, they shouldn't be able to read top secret files, but should be able to access secret and public ones.

The second general rule of thumb is that people shouldn't be able to 'write down'. If a member has top secret clearance then they should be able to write or modify top secret files, but not secret or public files. It may seem weird that, even with the highest clearance, you can't modify less secret files, but it guarantees that there's no accidental leakage of top secret information into secret or public files.

This no 'read up', no 'write down' approach is called the Bell-LaPadula Model. It was formulated for the U.S. Department of Defense's multilevel security policy.

There are many other models for access control like the Chinese Wall Model and Biba Model. Which model is best depends on your use case.

Authentication and Access Control help computers determine who you are and what you should access. But depend on being able to trust the hardware and software that run the Authentication and Access Control programs. That's a big dependence.

If an attacker installs malicious software called malware, compromising the host computer's operating system, how can we be sure security programs don't have a back door that let attackers in? The short answer is: we can't.

We still have no way to guarantee the security of a program or computing system. That's because even while security software may be security in theory, implementation bugs can still result in vulnerabilities.

But we do have techniques to reduce the likelihood of bugs like quickly finding and patching bugs when they do occur and mitigating damage when a program is compromised. Most security errors come from implementation error. To reduce implementation error, reduce implementation.

One of the Holy Grails of system level security is a security kernel or a trusted computing base: a minimal set of operating system software that's close to provably secure.

A challenge in constructing these security kernels is deciding what should go into it. Remember the less code-the better

Even after minimizing code blow, it would be great to guarantee the code that's written is secure. Formally verifying the security of code is an active area of research. The best we have right now is a process called Independent Verification and Validation.

This works by having code audited by a crowd of security-minded developers. This is why security code is almost always open-source. It's often difficult for people who wrote the original code to find bugs. But external developers with fresh eyes and different expertise can spot problems.

There are also conferences where like-minded hackers and security



## Cybersecurity: Crash Course Computer Science #31

Crash Course: Computer Science

<https://youtube.com/watch?v=bPVaOIJ6ln0>

<https://nerdfighteria.info/v/bPVaOIJ6ln0>

---

experts can mingle and share ideas. The biggest of which is Defcon, held annually in Los Vegas.

Finally, even after reducing code and auditing it, clever attackers are bound to find tricks that let them in. With this in mind, good developers should take the approach that not if, but when their programs are compromised, the damages should be limited and contained. And not let it compromise other things running on the computer.

This principle is called isolation. To achieve isolation, we can sandbox applications. This is like placing an angry kid in a sandbox. When a kid goes ballistic, they only destroy the sand castle in their own box. But other kids in the playground continue having fun.

Operating systems attempt to sandbox applications by giving each their own block of memory the other programs can't touch.

It's also possible for a single computer to run multiple Virtual Machines. Essentially simulated computers that live in their own sandbox. If a program goes awry, worst case is it crashes or compromises only the virtual machine on which it's running.

All other virtual machines running on the computer are isolated and unaffected.

Okay, that's a broad overview of some key computer security topics. And I didn't even get to network security like firewalls.

Next episode, we'll discuss some methods hackers use to get into computer systems. And after that, we'll touch on encryption. Until then, make your passwords stronger, turn on two-factor authentication, and never click links in unsolicited emails.

I'll see you next week!

[Outro Music]

Crash Course Computer Science is produced in association with PBS Digital Studios. At their channel you can check out a playlist of shows like Physics Girl, Deep Look, and PBS Spacetime.

This episode was filmed at the Chad and Stacey Emigholz Studio in Indianapolis, Indiana. And it was made with the help of all of these nice people and our wonderful graphics team, Thought Cafe.

That's where we're going to have to halt and catch fire.

See you next week!