



Hackers & Cyber Attacks: Crash Course Computer Science #32

Crash Course: Computer Science

https://youtube.com/watch?v=_GzE99AmAQU

https://nerdfighteria.info/v/_GzE99AmAQU

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

Last episode, we talked about the basics of computer security, principles and techniques used to keep computer systems safe and sound. But despite our best efforts, the news is full of stories of individuals, companies, and governments getting cyber-attacked by hackers. People who, with their technical knowledge, break into computer systems.

Not all hackers are bad though. There are hackers who hunt for bugs and try to close security holes in software to make systems safe or more resilient. They're often hired by companies and governments to perform security evaluations.

These hackers are called White Hats; they're the good guys. On the flip side, there are Black Hats, malicious hackers who have intentions to steal, exploit, and sell computer vulnerabilities and data.

Hackers' motivations also vary wildly. Some hack for amusement and curiosity, while cyber criminals hack most often for monetary gain. And then there are hacktivists, who use their skills to promote a social or political goal.

And that's just the tip of the iceberg. Basically, the stereotypical view of the hacker as some unpopular kid sitting in a dark room full of discarded pizza boxes probably better describes John Green in college than it does hackers.

Today, we're not going to teach you how to be a hacker. Instead, we'll discuss some classic examples of how hackers break into computer systems to give you an idea of how it's done.

===== Intro (1:14-1:21) =====

The most common way hackers get into computer systems isn't by hacking at all, it's by tricking users into letting them in. This is called social engineering, where a person is manipulated into divulging confidential information or configuring a computer system so that it permits entry by attackers.

The most common type of attack is phishing, which you'll most often encounter as an email asking you to login to an account on a website - say your bank. You'll be asked to click a link in the email which takes you to a site that looks legit to the casual observer, but it's really an evil clone.

When you enter your username and password, that information goes straight to the hackers, who then can log into the real website as you. Bad news.

Even with the 1/10 or 1% success rate, a million phishing emails might yield a thousand compromised accounts.

Another social engineering attack is pre-texting, where attackers call up, let's say a company, and then confidently pretend to be from their IT Department. Often attackers will call a first number and then be asked to transfer to a second so that the phone number appears to be internal to the company.

Then the attacker can instruct an unwitting user to configure their computer in a compromising way or get them to reveal confidential details like passwords or network configurations.

S-sorry, one sec.

Oh hey, it's Susan from IT, we're having some network issues down

here, can you go ahead and check a setting for me?

And it begins.

Attackers can be very convincing, especially with a little bit of research beforehand to find things like key employee's names. It may take 10 phone calls to find a victim, but you only need one to get in.

Emails are also a common delivery mechanism for Trojan Horses, programs that masquerade as harmless attachments like a photo or invoice, but actually contain malicious software called malware.

Malware can take many forms. Some might steal your data like your banking credentials, others might encrypt your files and demand a ransom - what's known as ransomware.

If they can't run malware or get a user to let them in, attackers have to force their way in through other means. One method, which we briefly discussed last episode, is to brute force a password - try every combination of password until you get your entry.

Most modern systems defend against this kind of attack by having you wait incrementally longer periods of time following each failed attempt, or even lock you out entirely after a certain number of tries.

One recent hack to get around this is called NAND Mirroring, where if you have physical access to the computer, you can attach wires to the device's memory chip and make a perfect copy of its contents.

With this setup, you can try a series of passwords until the device starts making you wait. When this happens, you just refresh the memory with the original copy you made, essentially resetting it, allowing you to try more passwords immediately with no waiting.

This technique was shown to be successful on an iPhone 5c, but many newer devices include mechanisms to thwart this type of attack.

If you don't have physical access to a device, you have to find a way to hack it remotely, like over the Internet. In general, this requires an attacker to find and take advantage of a bug in the system. And successfully utilizing a bug to gain capabilities or access is called an exploit.

One common type of exploit is a buffer overflow. Buffers are a general term for a block of memory reserved for storing data. We talked about video buffers for storing pixel data in episode 23.

As a simple example, we can imagine an operating system's login prompt, which has fields for a username and password. Behind the scenes, this operating system uses buffers for storing text values that are entered.

For illustration, let's say these buffers were specified to be a size 10. In memory, the two text buffers would look something like this. Of course, the operating system is keeping track of a lot more than just the username and password, so there's going to be data stored both before and after in memory.

When a user enters a username and password, the values are copied into the buffers where they can be verified. A buffer overflow attack does exactly what the name suggests: overflows the buffer. In this case, any password longer than 10 characters will overwrite adjacent data in memory.

Sometimes, this will just cause a program or operating system to crash, because important values are overwritten with



Hackers & Cyber Attacks: Crash Course Computer Science #32

Crash Course: Computer Science

https://youtube.com/watch?v=_GzE99AmAQU

https://nerdfighteria.info/v/_GzE99AmAQU

gobbledygook. Crashing a system is bad and maybe that's all the mischievous hacker wants to do - be a nuisance.

But attackers can also exploit this bug more cleverly by injecting purposeful new values into a program's memory. For example, setting an `is:admin` variable to `true`.

With the ability to arbitrarily manipulate a program's memory, hackers can bypass things like login prompts and sometimes even use that program to hijack the whole system.

There are many methods to combat buffer overflow attacks. The easiest is to always test the length of input before copying it into a buffer, called bounds checking. Many modern programming languages implement bounds checking automatically.

Programs can also randomize the memory location of variables, like our hypothetical `is:admin` flag, so that hackers don't know what memory location to overwrite and are more likely to crash the program than gain access.

Programs can also leave unused space after buffers and keep an eye on all those values to see if they change. If they do, they know an attacker is monkeying around with memory. These regions are called canaries, named after the small birds miners used to take underground to warn them of dangerous conditions.

Another classic hack is code injection. It's most commonly used to attack websites that use databases. Which pretty much all big websites do. We won't be covering databases in this series, so here's a simple example to illustrate this type of attack. We'll use structured query language, SQL, also called sequel, a popular database API.

Let's imagine our login prompt is now running on a webpage. When a user clicks login, the text values are sent to a server, which executes code that checks if that username exists. And if it does, verifies the password matches.

To do this, the server will execute code known as a SQL query that looks something like this. First, it needs to specify what data we're retrieving from the database. In this case, we want to fetch the password.

The server also needs to specify from what place in the database to retrieve the value from. In this case, let's imagine all the user's data is stored in the data structure called a table labelled 'users'.

Finally, the server doesn't want to get back a giant list of passwords for every user in the database, so it specifies that it only wants data for the account whose username equals a certain value.

That value is copied into the SQL query by the server based on what the user typed in. So the actual command that's sent to the SQL query database would look something like this 'where username = 'philbin'.

Know also that SQL commands end with a semicolon. So how does someone hack this? By sending in a malicious username with embedded SQL commands. Like we could send the server this funky username.

When the server copies this text into the sequel query, it ends up looking like this. As I mentioned before, semicolons are used to separate commands. So the first command that gets executed is this.

If there is a user named 'whatever' the database will return the password. Of course, we have no idea what 'whatever's' password

is, so we'll get it wrong and the server will reject us.

If there's no user named 'whatever', the database will return no password or provide an error and the server will again reject us. Either way, we don't care, because it's the next sequel command we're interested in: `DROP TABLE users;`, a command that we injected by manipulating the username field.

This command instructs a SQL database to delete the table containing all user data. Wiped clean. Which would cause a lot of headaches at a place like a bank or really anywhere.

And notice that we didn't even break into the system. It's not like we correctly guessed the username or password. Even with no formal access, we were able to create mayhem by exploiting a bug.

This is a very simple example of code injection, which almost all servers today have defenses against. With more sophisticated attacks, it's possible to add records to the database, like a new administrator account.

Or even get the database to reveal data, allowing hackers to steal things like credit card numbers, social security numbers, and all sorts of nefarious goodies. But we're not going to teach you how to do that.

As with buffer overflows, programmers should always assume input coming from the outside to be potentially dangerous and examine it carefully. Most username and password forms on the web don't let you include special symbols like semicolons or quotes as a first level defence.

Good servers also sanitize input by removing or modifying special characters before running database queries. Working exploits are often sold or shared online. The more prevalent the bug or the more damaging the exploit, the higher the price or prestige it commands.

Even governments sometimes buy exploits which allow them to compromise computers for purposes like spying.

When a new exploitable bug is discovered that the software creators weren't aware of, it's called a zero-day vulnerability. Blackhat hackers rush to use the exploit for maximum benefit before white hat hackers release a patch for the bug.

This is why it's so important to keep your computer's software up to date. A lot of those downloads are security patches. If bugs are left open on enough systems, it allows hackers to write a program that jumps from computer to computer automatically, which are called worms.

If a hacker can take over a large number of computers, they can be used together to form what's called a botnet. This can have many purposes, like sending huge volumes of spam, mining Bitcoins using other peoples' computing power and electricity, and launching distributed denial of service - or DDoS attacks against servers.

DDoS is where all the computers in the botnet send a flood of dummy messages. This can knock services offline, either to force owners to pay a ransom of just to be evil.

Despite all of the hard-working white hats, exploits documented online and software engineering best practices, cyber attacks happen on a daily basis. They cost the global economy roughly half a trillion dollars annually, and that figure will only increase as we become more reliant on computing systems. This is especially worrying to governments, as infrastructure is increasingly computer-driven, like power plants, the electrical grid, traffic lights, water treatment plants, oil refinery, air traffic control, and lots of other



Hackers & Cyber Attacks: Crash Course Computer Science #32

Crash Course: Computer Science

https://youtube.com/watch?v=_GzE99AmAQU

https://nerdfighteria.info/v/_GzE99AmAQU

key systems.

Many experts predict that the next war will be fought in cyberspace, where nations are brought to their knees not by physical attack, but rather crippled economically and infra-structurally through cyber warfare.

There may not be any bullets fired, but the potential for lives lost is still very high. Maybe even higher than conventional warfare. SO we should all adopt good cyber security practices. and, as a community interconnected over the internet, we should ensure our computers are secured against those who wish to use their great potential for harm.

So, maybe stop ignoring that update notification? See you next week.

Crash Course Computer Science is produced in association with PBS digital studios. At their channel, you can check out a playlist of shows like BrainCraft, Comaniddy, and PBS Infinite Series.

This episode was filmed at the Chad and Stacy Emmigholz Studio in Indianapolis, Indiana.

It was made with the help of all of these nice people, and our wonderful graphics team: Thought Cafe

Thanks for watching! I'll see you later

(?~[10:08](#))