Google IT Support Specialization

Course 5: IT security

05 Defense in depth

Table des matières

[1 Understanding IT security threat 2](#_Toc14378789)

[2 Pelcgbybtl cryptology 2](#_Toc14378790)

[3 AAA security not roadside assistance 2](#_Toc14378791)

[4 Securing your network 2](#_Toc14378792)

[5 Defense in depth 2](#_Toc14378793)

[5.1 System hardening 2](#_Toc14378794)

[5.1.1 Intro to defense in depth 2](#_Toc14378795)

[5.1.2 Disabling unnecessary components 2](#_Toc14378796)

[5.1.3 Host based firewall 3](#_Toc14378797)

[5.1.4 Logging and auditing 4](#_Toc14378798)

[5.1.5 Antimalware protection 5](#_Toc14378799)

[5.1.6 Disk encryption 6](#_Toc14378800)

[5.1.7 Heather self learning 8](#_Toc14378801)

[5.2 Application hardening 8](#_Toc14378802)

[5.2.1 Software patch management 8](#_Toc14378803)

[5.2.2 Application policies 9](#_Toc14378804)

[5.3 Case study the security model of chrome os 10](#_Toc14378805)

[5.3.1 The security principles of chrome os 10](#_Toc14378806)

[5.3.2 Chrome os verified boot 11](#_Toc14378807)

[5.3.3 Chrome os data encryption 11](#_Toc14378808)

# Understanding IT security threat

# Pelcgbybtl cryptology

# AAA security not roadside assistance

# Securing your network

# Defense in depth

## System hardening

### Intro to defense in depth

Defense in depth is the concept of having multiple overlapping systems of defense to protect IT systems. This ensures some amount of redundancy for defensive measures. It also helps avoid a catastrophic compromise in the event that a single system fails, or a vulnerability is discovered in one system. Think of this as having multiple lines of defense. If an attacker manages to bypass your firewall, you're still protected by strong authentication systems within the network. This would require an attacker to find more vulnerabilities in more systems before real damage can occur. These next lessons will focus on bringing together the different security systems and measures we've discussed so far into a comprehensive security design. It'll offer defense in depth from a variety of known and unknown threats. By the end of this course, you'll be able to implement the appropriate methods for system hardening and application hardening. You'll also be able to determine the policies to use for operating system security. All right, let's get started.

### Disabling unnecessary components

Think back to the beginning of this course when we talked about attacks and vulnerabilities. The special class of vulnerabilities we discussed called zero day vulnerabilities are unique since they're unknown until they're exploited in the wild. The potential for these unknown flaws is something you should think about when looking to secure your company's systems and networks. Even though it's an unknown risk, it can still be handled by taking measures to restrict and control access to systems. Our end goal overall is risk reduction. Two important terms to know when talking about security risks are attack vectors and attack surfaces. An attack vector is a method or mechanism by which an attacker or malware gains access to a network or system. Some attack vectors are email attachments, network protocols or services, network interfaces, and user input. These are different approaches or paths that an attacker could use to compromise a system if they're able to exploit it. An Attack Surface is the sum of all the different attack vectors in a given system. Think of this as the combination of all possible ways an attacker could interact with our system, regardless of known vulnerabilities. It's not possible to know of all vulnerabilities in the system. So, make sure to think of all avenues that an outside actor could interact with our systems as a potential Attack Surface. The main takeaway here is to keep our Attack Surfaces as small as possible. This reduces the chances of an attacker discovering an unknown flaw and compromising our systems. There are lots of approaches you can use as an IT support specialist to reduce Attack Surfaces. All of them boil down to simplifying systems and services. The less complex something is, the less likely there will be undetected flaws. So, make sure to disable any extra services or protocols. If they're not totally necessary, then get them out of there. Every additional surface that's operating represents additional Attack Surfaces, that could have an undiscovered vulnerability. That vulnerability could be exploited and lead to compromise. This concept also applies to access and ACLs. Only allow access when totally necessary. So, for example, it's probably not necessary for employees to be able to access printers directly from outside of the local network. You can just adjust firewall rules to prevent that type of access. Another way to keep things simple is to reduce your software deployments. Instead of having five different software solutions to accomplish five separate tasks, replace them with one unified solution, if you can. That one solution should require less complex code, which reduces the number of potential vulnerabilities. You should also make sure to disable unnecessary or unused components of software and systems deployed. By disabling features not in use, you're reducing even more tech services, even more. You're not only reducing the number of ways an attacker can get in, but you're also minimizing the amount of code that's active. It's important to take this approach at every level of systems and networks under your administration. It might seem obvious to take these measures on critical networking infrastructure and servers, but it's just as important to do this for desktop and laptop platforms that your employees use. Lots of consumer operating systems ship a bunch of default services and software enabled right out of the box, that you probably won't be using in an enterprise network or environment. For example, Telnet access for a managed switch has no business being enabled in a real world environment. You should disable it immediately if you find it on the device. Any vendor specific API access should also be disabled if you don't plan on using these services or tools. They might be harmless especially if you set up strong firewall rules and network ACLs. This one service might represent a fairly low risk, but why take any unnecessary risk at all? Remember, the defense in depth concept is all about risk mitigation and implementing layers of security. Now, let's think about the layered approach to security. What if our access control measures are bypassed or fail, in some unforeseen way? As an IT support specialist, this is exactly what you want to think about. How do we keep this component secure if the security systems above it have failed?

### Host based firewall

We briefly mentioned host based firewalls when we talked about network monitoring and intrusion detection systems. Host based firewalls are important to creating multiple layers of security. They protect individual hosts from being compromised when they're used in untrusted and potentially malicious environments. They also protect individual hosts from potentially compromised peers inside a trusted network. Our network based firewall has a duty to protect our internal network by filtering traffic in and out of it, while the host based firewall on each individual host protects that one machine. Like our network based firewall, we'd still want to start with an implicit deny rule. Then, we'd selectively enable specific services and ports that will be used. This let us start with a secured default and then only permits traffic that we know and trust. You can think of this as starting with a perfectly secure firewall configuration and then poking holes in it for the specific traffic we require. This may look very different from your network firewall configuration since it's unlikely that your employees would need remote SSH access to their laptops, for example. Remember that to secure systems you need to minimize attack surfaces or exposure. A host based firewall plays a big part in reducing what's accessible to an outside attacker. It provides flexibility while only permitting connections to selective services on a given host from specific networks or IP ranges. This ability to restrict connections from certain origins is usually used to implement a highly secure host to network. From there, access to critical or sensitive systems or infrastructure is permitted. These are called Bastion hosts or networks, and are specifically hardened and minimized to reduce what's permitted to run on them. Bastion hosts are usually exposed to the internet so you should pay special attention to hardening and locking them down to reduce the chances of compromise. But they can also be used as a sort of gateway or access portal into more sensitive services like core authentication servers or domain controllers. This would let you implement more secure authentication mechanisms and ackles on the Bastion hosts without making it inconvenient for your entire company. Monitoring and logging can be prioritized for these hosts more easily. Typically, these hosts or networks would also have severely limited network connectivity. It's usually just to the secure zone that they're designed to protect and not much else. Applications that are allowed to be installed and run on these hosts would also be restricted to those that are strictly necessary, since these machines have one specific purpose. Part of the host base firewall rules will likely also provide ackles that allow access from the VPN subnet. It's good practice to keep the network that VPN clients connected to separate using both subnetting and VLANs. This gives you more flexibility to enforce security on these VPN clients. It also lets you build additional layers of defenses, while a VPN host should be protected using other means, it's still a host that's operating in a potentially malicious environment. This host is then initiating a remote connection into your trusted internal network. These hosts represent another potential vector of attack and compromise. Your ability to separately monitor traffic coming and going from them is super useful. There's an important thing for you to consider when it comes to host based firewalls, especially for client systems like laptops. If the users of the system have administrator rights, then they have the ability to change firewall rules and configurations. This is something you should keep in mind and make sure to monitor with logging. If management tools allow it, you should also prevent the disabling of the host based firewall. This can be done with Microsoft Windows machines when administered using active directory as an example.

### Logging and auditing

A critical part of any security architecture is logging and alerting. It wouldn't do much good to have all these defenses in place, if we have no idea if they're working or not. We need visibility into the security systems in place to see what kind of traffic they're seeing. We also need to have that visibility into the logs of all of our infrastructure devices and equipment that we manage. But it's not enough to just have logs, we also need ways to safeguard logs and make them easy to analyze and review. If there is a dedicated security team at your company, they would be performing this analysis. But at a smaller company, this responsibility would likely fall to the IT team. So, let's make sure you're prepared with the skills you might need for incident investigation. Many investigative techniques can also be applied to troubleshooting. All systems and services running on hosts will create logs of some kind, with different levels of detail. It depends on what it's logging, and what events it's configured to log. So, an authentication server we'd log every authentication attempt, whether it's successful or not. A firewall would log traffic that matches rules with details like source and destination addresses, and ports being used. All this logged information gives us details about the traffic and activity that's happening on our network and systems. This can be used to detect compromise or attempts to attack the system. When there are a large number of systems located around your network, each with their own log format, it can be challenging to make meaningful sense of all this data. This is where security information and event management systems or SIEMS come in. A SIEM can be thought of as a centralized log server. It has some extra analysis features too. In the system administration and IT infrastructure course of this program, you learn ways that centralized logging can help you administer multiple machines at once. You can think of SIEM as a form of centralized logging for security administration purposes. A SIEM system gets logs from a bunch of other systems. It consolidates the logs from all different places and places it in one centralized location. This makes handling logs a lot easier. As an IT support specialist, an important step you'll take in logs analysis is normalization. This is the process of taking log data in different formats and converting it into a standardized format that's consistent with a defined log structure. As an IT support specialist, you might configure normalization for your log sources. For example, log entries from our firewall may have a timestamp using a year, month, and day format, while logs from our client machines may use day, month, year format. To normalize this data, you choose one standard date format, then you define what the fields are for the log types that need to be converted. When logs are received from these machines, the log entries are converted into the standard that we defined, and stored by the logging server. This lets you analyze and compare log data between different log types and systems in a much easier fashion. So what type of information should you be logging? Well, that's a great question. If you log too much info, it's difficult to analyze a data and find useful information. Plus, storage requirements for saving logs become expensive very quickly. But if you log too little, then the information won't provide any useful insights into your systems and network. Finding that middle ground can be difficult. It will vary depending on the unique characteristics of the systems being monitored, and the type of activity on the network. No matter what events are logged, all of them should have information that will help understand what happened and reconstruct the events. There are lots of important fields to capture in log entries like timestamp, the event or error code, the service or application being logged, the user or system account associated with the event, and the devices involved in the event. Timestamps are super important to understanding when an event occurred. Fields like source and destination addresses will tell us who was talking to who. For application logs, you can grab useful information from the logged in user associated with the event, and from what client they used. On top of the analysis assistance it provides, a centralized live server also has security benefits. By maintaining logs on a dedicated system, it's easier to secure the system from attack. Logs are usually targeted by attackers after a breach, so that they can cover their tracks. By having critical systems send logs to remote logging server that's locked down, the details of a breach should still be logged. A forensics team will be able to reconstruct the events that led to the compromise. Once we have logging configured and the relevant events recorded on a centralized log server, what do we do with all the data? Well, analyzing log details depends on what you're trying to achieve. Typically, when you look at aggregated logs as an IT support specialist, you should pay attention to patterns and connections between traffic. So, if you're seeing a large percentage of Windows hosts, all connecting to specific address outside your network, that might be worth investigating. It could signal a malware infection. Once logs are centralized and standardized, you can write automated alerting based on rules. Maybe you'll want to find an alert rule for repeated unsuccessful attempts to authenticate to a critical authentication server. Lots of SIEMS solutions also offer handing dashboards to help analysts visualize this data. Having data in a visual format can potentially provide more insight. You can also write some of your own monitoring and alert systems. Now, it doesn't matter if you're using a SIEM solution or writing your own, it can be useful to break down things like commonly used protocols in the network, quickly see the top talkers in the network, interview reported errors over time to reveal patterns. Speaking of top talkers, I have just one more thing to call out. But we'll take a break before the next video. Another important component to logging to keep in mind as an IT support specialist, is retention. Your log storage needs will vary based on the amount of systems being logged, the amount of detail logs, and the rate at which logs are created. How long you want or need to keep logs around will also really influence the storage requirements for a log server. Some examples of logging servers and SIEMS solutions are the open source rsylog, Splunk Enterprise Security, IBM Security Qradar, and RSA Security analytics. You can learn more about these solutions in the supplementary readings of this lesson. Okay. Breaktime. I'll see you at the top of the next video on anti malware protection.

### Antimalware protection

Anti malware defenses are a core part of any company's security model in this day and age. So it's important as an IT support specialist to know what's out there. Today, the internet is full of bots, viruses, worms, and other automated attacks. Lots of unprotected systems would be compromised in a matter of minutes if directly connected to the internet without any safeguards or protections in place. And they need to have critical system updates. While modern operating systems have reduced this threat vector by having basic firewalls enabled by default, there's still a huge amount of attack traffic on the Internet. Anti malware measures play a super important role in keeping this type of attack off your systems and helping to protect your users. Antivirus software has been around for a really long time but some security experts question the value it can provide to a company especially since more sophisticated malware and attacks have been spun up in recent years. Antivirus software is signature based. This means that it has a database of signatures that identify known malware like the unique file hash of a malicious binary or the file associated with an infection. Or it could be that network traffic characteristics that malware uses to communicate with a command and control server. Antivirus software will monitor and analyze things like new files being created or being modified on the system in order to watch for any behavior that matches a known malware signature. If it detects activity that matches the signature, depending on the signature type, it will attempt to block the malware from harming the system. But some signatures might only be able to detect the malware after the infection has occurred. In that case, it may attempt to quarantine the infected files. If that's not possible, it will just log and alert the detection event. At a high level, this is how all antivirus products work. There are two issues with antivirus software though. The first is that they depend on antivirus signatures distributed by the antivirus software vendor. The second is that they depend on the antivirus vendor discovering new malware and writing new signatures for newly discovered threats. Until the vendor is able to write new signatures and publish and disseminate them, your antivirus software can't protect you from these emerging threats. Boo. Antivirus, which is designed to protect systems, actually represents an additional attack surface that attackers can exploit. You might be thinking, wait, our own antivirus tools can be another threat to our system? What's the deal with that? Well, this is because of the very nature of one antivirus engine must do. It takes arbitrary and potentially malicious binaries as input and performs various operations on them. Because of this, there are a lot of complex code where very serious bugs could exist. Exactly this kind of vulnerability was found in the Sophos Antivirus engine back in 2012. You can read more about this event in the supplementary readings. So, it sounds like antivirus software isn't ideal and has some pretty large drawbacks. Then why are we still recommending it as a core piece of security design? The short answer is this. It protects against the most common attacks out there on the internet. The really obvious stuff that still poses a threat to your systems still needs to be defended against. Antivirus is an easy solution to provide that protection. It doesn't matter how much you user education you instill in your employees. There will still be some folks who will click on an e mail that has an infected attachment. A good way to think about antivirus in today's very noisy external threat environment is like a filter for the attack noise on the internet today. It lets you remove the background noise and focus on the more important targeted or specific threats. Remember, our defense in depth concept involves multiple layers of protection. Antivirus software is just one piece of our anti malware defenses. If antivirus can't protect us from the threats we don't know about, how do we protect against the unknown threats out there? While antivirus operates on a blacklist model, checking against a list of known bad things and blocking what gets matched, there's a class of anti malware software that does the opposite. Binary whitelisting software operates off a white list. It's a list of known good and trusted software and only things that are on the list are permitted to run. Everything else is blocked. You can think of this as applying the implicit deny ACL rule to software execution. By default, everything is blocked. Only things explicitly allowed to execute are able to. I should call out that this typically only applies to executable binaries, not arbitrary files like PDF documents or text files. This would naturally defend against any unknown threats but at the cost of convenience. Think about how frequently you download and install new software on your machine. Now imagine if you had to get approval before you could download and install any new software. That would be really annoying, don't you think? Now, imagine that every system update had to be whitelisted before it could be applied. Obviously, not trusting everything wouldn't be very sustainable. It's for this reason that binary whitelisting software can trust software using a couple of different mechanisms. The first is using the unique cryptographic hash of binaries which are used to identify unique binaries. This is used to whitelist individual executables. The other trust mechanism is a software signing certificate. Remember back when we discussed public key cryptography and signatures using public and private key pairs? Software signing or code signing is the same idea but applied to software. A software vendor can cryptographically sign binaries they distribute using a private key. The signature can be verified at execution time by checking the signature using the public key embedded in the certificate and verifying the trust chain of the public key. If the hash matches and the public key is trusted, then the software can be verified that it came from someone with the software vendor's code signing private key. Binary whitelisting systems can be configured to trust specific vendors' code signing certificates. They permit all binary sign with that certificate to run. This is helpful for automatically trusting content like system updates along with software in common use that comes from reputable and trusted vendors. But can you guess the downside here? Each new code signing certificate that's trusted represents an increase in attack surface. An attacker can compromise the code signing certificate of a software vendor that your company trusts and use that to sign malware that targets your company. That would bypass any binary whitelisting defenses in place. Not good. This exact scenario happened back in 2013 to Bit9, a binary whitelisting software company. Hackers managed to breach their internal network and found an unsecured virtual machine. It had a copy of the code signing certificates private key. They stole that key and used it to sign malware that would have been trusted by all Bit9 software installations by default.

### Disk encryption

We briefly discussed disk encryption earlier when we talked about encryption at a high level. Now, it's time to dive deeper. Full disk encryption, or FDE, is an important factor in a defense in depth security model. It provides protection from some physical forms of attack. As an IT support specialist, you likely assist with implementing an FDE solution. If one doesn't exist already, help with migrating between FDE solutions and troubleshoot issues with FDE systems, like helping with forgotten passwords. So FDE is key. Systems with their entire hard drives encrypted are resilient against data theft. They'll prevent an attacker from stealing potentially confidential information from a hard drive that's been stolen or lost. Without also knowing the encryption password or having access to the encryption key, the data on the hard drive is just meaningless gibberish. This is a very important security mechanism to deploy for more mobile devices like laptops, cell phones, and tablets. But it's also recommended for desktops and servers too. Since disk encryption not only provides confidentiality but also integrity. This means that an attacker with physical access to a system can't replace system files with malicious ones or install malware. Having the disk fully encrypted protects from data theft and unauthorized tampering even if an attacker has physical access to the disk. But in order for a system to boot if it has an FDE setup, there are some critical files that must be accessible. They need to be available before the primary disk can be unlocked and the boot process can continue. Because of this, all FDE setups have an unencrypted partition on the disk, which holds these critical boot files. Examples include things like the kernel and bootloader, that are critical to the operating system. These files are actually vulnerable to being replaced with modified potentially malicious files by an attacker with physical access. While it's possible to compromise a machine this way, it would take a sophisticated and determined attacker to do it. There's also protection against this attack in the form of the secure boot protocol, which is part of the UEFI specification. Secure boot uses public key cryptography to secure these encrypted elements of the boot process. It does this by integrated code signing and verification of the boot files. Initially, secure boot is configured with what's called a platform key, which is the public key corresponding to the private key used to sign the boot files. This platform key is written to firmware and is used at boot time to verify the signature of the boot files. Only files correctly signed and trusted will be allowed to execute. This way, a secure boot protects against physical tampering with the unencrypted boot partition. There are first party full disk encryption solutions from Microsoft and Apple called Bit Locker and FileVault 2 respectively. There are also a bunch of third party and open source solutions. On Linux, the dm crypt package is super popular. There are also solutions from PGP, TrueCrypt, VeraCrypt, and lots of others. Check out the supplementary readings for a detailed list of FDE tools. Just pick your poison or antidote, I should say. Full disk encryption schemes rely on the secret key for actual encryption and decryption operations. They typically password protect access to this key. And in some cases, the actual encryption key is used to derive a user key, which is then used to encrypt the master key. If the encryption key needs to be changed, the user key can be swapped out, without requiring a full decryption and re encryption of the data being protected. This would be necessary if the master encryption key needs to be changed. Password protecting the key works by requiring the user entry passphrase to unlock the encryption key. It can then be used to access the protected contents on the disk. In many cases, this might be the same as the user account password to keep things simple and to reduce the number of passwords to memorize. When you implement a full disk encryption solution at scale, it's super important to think about how to handle cases where passwords are forgotten. This is another convenience tradeoff when using FDE. If the passphrase is forgotten, then the contents of the disk aren't recoverable. Yikes! This is why lots of enterprise disk encryption solutions have a key escrow functionality. Key escrow allows encryption key to be securely stored for later retrieval by an authorized party. So if someone forgets the passphrase to unlock their encrypted disk for their laptop, the systems administrators are able to retrieve the escrow key or recovery passphrase to unlock the disk. It's usually a separate key passphrase that can unlock the disk in addition to the user to find one. This allows for recovery if a password is forgotten. The recovery key is used to unlock the disk and boot the system fully. You should compare full disk encryption against file based encryption. That's where only some files or folders are encrypted and not the entire disk. This is usually implemented as home directory encryption. It serves a slightly different purpose compared to FDE. Home directory or file based encryption only guarantees confidentiality and integrity of files protected by encryption. These setups usually don't encrypt system files because there are often compromises between security and usability. When the whole disk isn't encrypted, it's possible to remotely reboot a machine without being locked out. If you reboot a full disk encrypted machine, the disk unlock password must be entered before the machine finishes booting and is reachable over the network again. So while file based encryption is a little more convenient, it's less protected against physical attacks. An attacker can modify or replace core system files and compromise the machine to gain access to the encrypted data. This is a good example of why understanding threats and the risks these threats represent is an important part in designing a security architecture and choosing the right defenses. In our next lesson, we'll cover application hardening. I'll see you there.

### Heather self learning

I think self learning is the key to success in this particular field. Technology changes all the time. And you have to have a drive to learn about the new things coming out. And you're only going to get that if you're constantly keeping on top of what's going on. And the more you do that, the more of an expert you become in lots of different areas. And by becoming an expert in them, it propels you forward because if you can be the expert in the room, you can teach others. And you can also be the person that people rely on, and it also builds your confidence over time. There are people who specialize deeply in security, and that's all they do all day long. But in fact, every person who works at Google is a security person. I think the IT support role is an incredibly pivotal role within Google from a security standpoint. And in many cases, it's the IT support people who see Google being hacked for the first time. Had I not had an IT support experience early in my career, I probably wouldn't be able to bring the perspective that I do to the table in terms of how we make security better for everyone.

## Application hardening

### Software patch management

While some parts of software features are exposed, a lot of attacks depend on exploiting bugs in software. This triggers obscure and unintended behavior which can lead to a compromise of the system running the vulnerable software. These types of vulnerabilities can be fixed through software patches and updates which correct the bugs that the attackers exploit. As an IT support specialist, it's critical that you make sure that you install software updates and security patches in a timely way in order to defend your company's systems and networks. Software updates don't just improve software products by adding new features and improving performance and stability, they also address security vulnerabilities. There are some software bugs that are present in the core functionality of the software in question. This means, that the vulnerability can't be mitigated by disabling the vulnerable service, not good. An example of this was the Heartbleed vulnerability. A bug in the open source TLS library Open SSL. This was discovered in widely publicized in April of 2014. The bug showed up in how the library handled TLS heartbeat messages. They're special messages that allow one party in a TLS session to signal to the other party that they like the session to be kept alive. This works by sending a TLS heartbeat request message, a packet that has a text string and the length of the string. The receiving end is supposed to reply with the same tech string and response. So, if the heartbeat requests message contains the text "I am still alive" and the length of 15, the receiving end would reply back with the same text, "I am still alive." But, the bug in the Open SSL library was that the replying side would allocate memory space according to the value in the received packet. This was based on the specified length of the string like it's defined in the packet. Not based on the actual length of the string. The value was not verified. This meant that an attacker can send a malformed heartbeat request message with a much larger length specified than what was allowed. The reply would contain the original text message but would also include bits of memory from the replying system. So, an attacker can send a malformed heartbeat request message containing the text, "I'm still alive", but with a length of 500. Because the length value wasn't verified. this means that the response back would be, "I'm still alive", followed by the next 485 characters in memory. So it was possible for an attacker to read up the 64 kilobytes of a target's memory. This memory was likely used before by Open SSL library, so it might contain sensitive information regarding other TLS sessions. This bug meant that it was feasible for an attacker to recover the private keys used to protect TLS sessions. This would allow them to decrypt TLS protected sessions and recover details like log in credentials. This is a great example of a mistake in the code leading to a very high profile software vulnerability. It could only be fixed through a software update or switching to a different TLS library entirely. While the heartbeat functionality is enabled by default, it's possible to disable it in the Open SSL library but it wasn't a simple argument to pass to an application. Disabling this functionality required compiling the library with a flag that was specified to disable heartbeats. Then, you had to replace the installed version with the custom compiled one. That's not something most people would do. This was also a library widely used by both server applications and client applications. This means, that it may not be possible to replace the Open SSL library with a customized version or a different library. The only way to address the vulnerability in client software that implemented Open SSL, was to wait for a patch from the software vendor. What a mess! Here's the bad news, with software continuing to grow more complex over time, these types of bugs are likely to become more commonplace. Attackers will be looking for exactly this type of vulnerability. The best protection is to have a good system and policy in place for your company. The system should be checking for, distributing and verifying software updates for software deployment. This is a complex problem when considering a large organization with many machines to manage that run a variety of software products. This is where management tools can help make this task more approachable for you. Solutions like Microsoft's SCCM or Puppet Labs puppet in fact and tools allow administrators to get an overview of what software is installed across their fleet of many systems. This lets a security team analyze what specific software and versions are installed, to better understand the risk of vulnerable software in the fleet. When updates are released and pushed to the fleet, these reporting tools can help make sure that the updates have been applied. SCCM even has the ability to force install updates after a specified deadline has passed. Patching isn't just necessary for software, but also operating systems and firmware that run on infrastructure devices. Every device has code running on it that might have software bugs that could lead to security vulnerabilities, from routers, switches, phones even printers. Operating system vendors usually push security related patches pretty quickly when an issue is discovered. They'll usually release security fixes out of cycle from typical OS upgrades to ensure a timely fix because of the security implications. But, for embedded devices like networking equipment or printers, this might not be typical. Critical infrastructure devices should be approached carefully when you apply updates. There's always the risk that a software update will introduce a new bug that may affect the functionality of a device, or if the update process itself would go wrong and cause an outage. I hope you can see the importance of applying software patches and firmer updates in a timely fashion. It would be pretty embarrassing if you wind up being compromised by a vulnerability that could have been easily fixed with a software update.

### Application policies

As you can see, application software can represent a pretty large attack surface. This is especially true when it comes to a large fleet of systems used throughout an organization. So, it's important to have some kind of application policies in place. These policies serve two purposes. Not only do they defined boundaries of what applications are permitted or not, but they also help educate folks on how to use software more securely. We've seen the risks that software can pose because of security vulnerabilities. It makes sense to have a policy around applying software updates in a timely way. A common recommendation or even a requirement is to only support or require the latest version of a piece of software. From the I.T. support perspective, this is important because software updates will often fix issues that someone may be encountering. But from the security side of things, making sure the latest version of the software will ensure that all security patches have been applied and the most secure version is in use. This should be clearly called out in a policy. People tend to be pretty lazy about applying updates to software that they use a lot. Lots of times, applying an update requires restarting the application, which can feel inconvenient and disruptive to users. It's generally a good idea to disallow risky classes of software by policy. Things like file sharing software and piracy related software tend to be closely associated with malware infections. They usually don't have a business use either. Let's not even talk about the legal implications of this type of software. Understanding what your users need to do their jobs will help shape your approach to software policies and guidelines. If there's a common use case for a certain type of software, it would be helpful to select a specific software implementation and require the use of that solution. This lets to reevaluate the most secure solution and benefit from a more uniform software installation. Remember, the name of the game is to minimize attack surfaces. Each piece of software that accomplishes the same thing represents a different set of potential attack surfaces that could have a vulnerability lurking inside. Helping your users accomplish tasks by recommending or supporting specific software makes for a more secure environment. It also helps users by giving them clear solutions to accomplish tasks. If you want to employ a binary white listing solution, it's also important to define a policy around what type of software can be whitelisted. So it's probably unnecessary to have video games whitelisted, unless your company is a video game studio, of course. These policies usually require some kind of business use case or justification to avoid a lot of one off personal software requests. Another class of software that you might want to have policies defined for are browser extensions or add ons. Since a lot of workflows live exclusively within the web browser now, they represent a potential vector for malware that often gets overlooked. Extensions that require full access to web sites visited can be risky since the extension developer has the power to modify pages visited. Some extensions may even send user input to a remote server. This could potentially leak confidential information. Clearly defining classifications of risky extensions and add ons will help protect your systems and provide guidance to your users. But, policies are usually not enough to arm users with the information they need to make informed choices. Their decisions can impact the security of your organization. That's where education and training comes into play which, we'll discuss in the next module. We went over a lot of really dense information on security in these lessons. Take time to review some of the videos so that it really sinks in. Okay, awesome work. Now, it's time for a project that will test what you learned about the system hardening, then if you can believe it, you'll move on to the last lesson of the last course of this program. Woohoo.

## Case study the security model of chrome os

### The security principles of chrome os

Let's look at the Chrome OS Security Model, as an example of of a system that applies the principles of defense and depth, that we've been studying. This OS has lots of different security layers designed to protect against attacks. So that if one layer is compromised the others are still in effect. As we call that on the first course Chrome OS runs Linux under the hood and features a simple user interface. Most of the system interaction happens through the Chrome browser. The goal of the OS is to be a secure and simple way for the user to interact with the web. The user doesn't have administrative privileges or permission to modify the system, since the Chrome browser is running with user permissions managed by the administrator. This means that even if the browser is exploited, the system can't be tampered with. Chrome OS defaults to securing the operating system. This means that the system settings are already configured for security and the user doesn't need to take any additional action. Chrome OS turns on hardening options that are available to all Linus operating systems. But not all Linux distributions enable these options. Let's go over some other specific security features developed for Chrome OS. As we've said before, applying security updates is critical for keeping a secure infrastructure. Chrome OS tackles this problem by implementing automatic updates. As soon as an update is available, the OS downloads and installs it in the background without any user interaction. Users will be prompted to restart when updates are available. The update then goes live as soon as the machine is restarted. One of the security features of Chrome OS is called sandboxing. Each tab in a Chrome browser, as well as each of the system services runs in a separate process completely independent of the others. So, if the user visits a malicious website, most of damage that the website can do is limited to that single tab. Only the session data stored on the computer hard drive such as cookies, caches, and bookmarks is accessible to all tabs. And of course, session data created by one website is not accessible to other websites. Another important Chrome OS security feature is the automatic and easy to use recovery mode. When the system detects a problem either because of data corruption or an attacker tampering with the installed operating system. It will automatically enter recovery mode and help the user return the system to a working, reliable state. Powerwash is another Chrome OS security feature. It allows the user to quickly reset the machine to its factory default settings. Since all the users personal data is stored in the Cloud, this feature will delete any downloaded files, cookies, and caches stored in the hard drive of the computer. Being able to delete everything from local drives improves security, when travelling or entering other unsecure or unknown environments. But that's not all, in the next two videos, we'll cover two central security measures in Chrome OS. Verified boot, and data encryption.

### Chrome os verified boot

Chrome OS features an integrity checked rollback protected group process called Verified boot. As with secure boot, this process verifies the integrity of the operating system being used. If an attacker tries to tamper with the machine, the system will detect it upon reboot and refuse to operate. Verified boot also stops known vulnerable versions of the operating system from booting even if they're signed. The verified boot process has four main components. Here's how it works. The first of these components, is the read only firmware. Chrome OS machine ship with this firmware, which can't be written to without physically modifying the machine, so remote attackers can't tamper with it. This read only firmware contains the cryptographic keys necessary to verify that the next component has been signed by a trusted source and that it has the minimum amount of code needed to execute the next component in the chain. The read write firmware. This is firmware that can get updated automatically when necessary. Once the read only firmware has verified that the contents come from a trusted source, the code from this second piece of the firmware will be executed. If instead the read only firmware isn't able to verify the contents, the the system will attempt to verify the latest backup of the read/write firmware. If the read/ write backup verifies it will use that copy instead. If the backup doesn't verify either, it will automatically invoke the recovery mode which is read only and can't be tampered with. So, we know that if the read/write firmware is executed it's because the system has verified that it comes from a trusted source. The read/write firmware is then in charge of verifying the actual operating system in a similar fashion and it will verify that the kernel comes from a trusted source and has not been tampered with. When the kernel is executed, it will verify the integrity of the root file system it's using. The root file system contains the actual operating system installation and the locally stored user data. So, if everything goes fine, the read only firmware verifies and executes the read/write firmware. The read/write firmware verifies and executes the kernel and the kernel mounts and verifies the root file system. Chrome OS stores the kernel and the root file system on separate hard drive partitions and there are two for each of them. The machine is currently using partitions that are read only so the system that's running can't modify itself. The other two can be modified by the software that performs the automatic updates. When an update is downloaded, it's deployed into the partitions that aren't currently being used. When the machine reboots, it will detect the update, verify it and boot from the updated partitions. When it reboots, if the machine detects that one of the kernels or the file system partitions is corrupted or was tampered with it, will boot from the other one instead. If both are in bad shape, it will automatically enter recovery mode.

### Chrome os data encryption

As we've called out before, Chrome OS stores most user data in the cloud, but some information is stored locally to help performance and offline use. This information includes downloaded files, bookmarks, cookies and caches among other things. Chrome OS ensures that only the user can access this information by encrypting it with both the user's password and the Trusted Platform Module or TPM. As a reminder, the TPM is a cryptographic device that can be used by the operating system to ensure the integrity of the system and to store cryptographic keys, so attackers can't access them. In Chrome OS, the TPM is used only for storing cryptographic keys. In newer generations of Chrome OS, the cryptographic device is called H1. It serves a similar purpose as the TPM, and may be used to enable even more security functions in the future. When a user logs in for the first time, Chrome OS uses a key generated by the cryptographic device and the user's password to create an encrypted directory for that specific user called a vault. In order to decrypt the directory data, that specific machine must successfully boot Chrome OS and the user must login with their password. If an attacker tried to extract the disk and mount it onto a different machine, they wouldn't be able to access any of the data, and since the data is encrypted with each user's password, other people using machine will not be able to access it. This gives users both security and privacy. If the Powerwash feature is activated, the key stored in the TPM are white, along with the deleted data, making it impossible for an attacker to recover the information from a hard drive. Now this doesn't mean that Chrome OS is invulnerable. All operating systems have vulnerabilities, so we should always keep important security practices in mind. In the case of Chrome OS, we should stay alert to avoid installing malicious Chrome extensions that can steal our passwords, for example, but thanks to its automatic updates, sandboxing, verified boot, data encryption, Powerwash and easy to use recovery mode, Chrome OS provides a layered security model that makes it the operating system of choice for lots of security experts. Wow. You've made it to the end of the module. Great work. If you can believe it, the next module is the final module for the entire program. Woohoo.