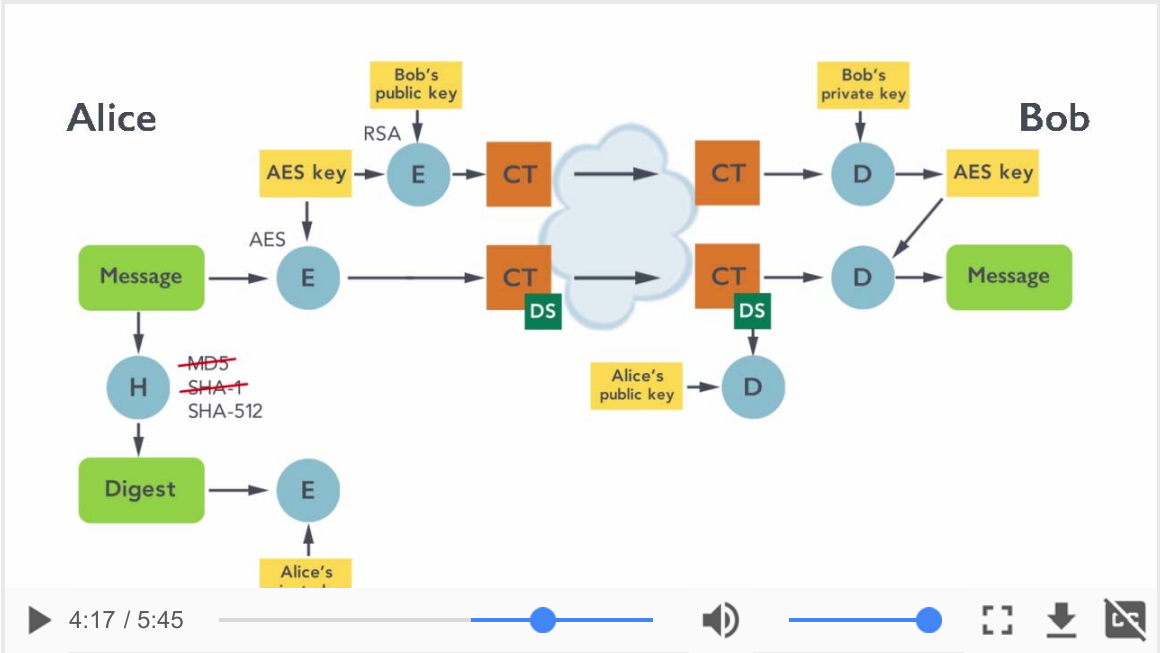
Operate and Implement Cryptographic Systems Overview

The previous modules explained the characteristics, strengths, and uses of cryptographic algorithms. As a security practitioner, we will use these algorithms in many ways including secure email, VPNs (Virtual Private Networks), e-commerce, and wireless security. Most cryptographic systems use a combination of symmetric and asymmetric algorithms, hashing, and digital signatures. Since a picture is often worth a thousand words, there will be several videos used to demonstrate the ways that cryptography is commonly used.

Let's first of all look at some of the ways cryptography can be implemented at each layer of the TCP/IP model. Once we have looked at this at a high level, we will look at the actual process in more detail.

Implementation of Secure Communications—Email

The first implementation of cryptography we are going to examine is the process of secure email as provided through S/MIME. This combines almost everything we have looked at in cryptography so far and is based on the scenario where Alice has to send a confidential email to Bob, but she also needs to provide proof of origin and assurance of message integrity.



▼ Transcript

Let's bring all of the parts of cryptography together in an example of how to send a secure email. We have Alice. Alice has a message she wants to send to Bob. We know that we can send that message effectively through an insecure channel using cryptography. The way to do that is to encrypt that message. We know that asymmetric cryptography is very slow. So when we use and select the crypto system to encrypt the message, we'll almost undoubtly use symmetric such as, well, the advanced encryption standard as the actual algorithm to use for that encryption. If we're going to use AES, we also then will have to select. What is that symmetric key that will be used in encrypting the message? The AES key. We now have ciphertext of that message that can be sent through an insecure channel to our friend Bob. When Bob receives

ciphertext, he needs to have the key. If he is going to be able to decrypt that message, so we have to have a way to send the key, the AES key, securely to Bob. The way to do that is to take that AES key and encrypt it. We'll encrypt the AES key with Bob's public key. Using asymmetric encryption such as, for example RSA. When we do that, we create ciphertext of the AES key which we can send through that insecure channel as well. When Bob receives the ciphertext of the AES key, he will be able to decrypt that using Bob's private key. That will give him the AES key he needs to decrypt the message. This is what we would call a fairly simple hybrid implementation. We're using, as you can see, symmetric key cryptography to encrypt the message and send it using the speed and confidentiality it provides and we're using asymmetric for key distribution to distribute the AES key or the symmetric key securely to the far end. But when you send an email, you might also want to provide proof of origin and integrity of that email. For that, we will take that original message and we will hash it using a hashing algorithm. In the old days, we used MD5 or SHA1. Today, you're probably using SHA-512 or SHA-2. When we hash a message, we create a digest of that message. A fingerprint, if you wilL, of the message itself. This proofs integrity of the message. We can now encrypt the digest and we will encrypt the digest with Alice's private key. When I encrypt a digest with a asymmetric algorithm such as Alice's private key, we create something known as a digital signature. We can append that to the message and send it along. When that's received by Bob he now sees the digital signature and he will decrypt the digital signature using Alice's public key. When we decrypt the digital signature, we will get the digest that Alice had created. Bob does the same thing. He takes the message he received from Alice and runs it through the same hasing algorithm to get the digest of the received message. If that digest of the received message is the same as the digest that was signed by Alice, Bob knows that he has an exact copy of that message that Alice had actually then signed. The digital signature provides integrity through the hash, proof of origin by signing it with a private key. We also have confidentiality of the message and we have a way to confidentially distribute the symmetric key. This is what happens if you use a product like PGP for example and you say to both encrypt and to sign the message. You've signed it with a digital signature, you've encrypted it using both symmetric and asymmetric encryption. Providing proof of origin, message confidentiality, secure symmetric key, distribution, as well as of course integrity of the message.

You will notice that in this video, we encrypted the message before we added the digital signature to the end of the encrypted message. Not all vendors operate that way. Some will encrypt the message and the digital signature together, which may actually be a more secure method of communication,

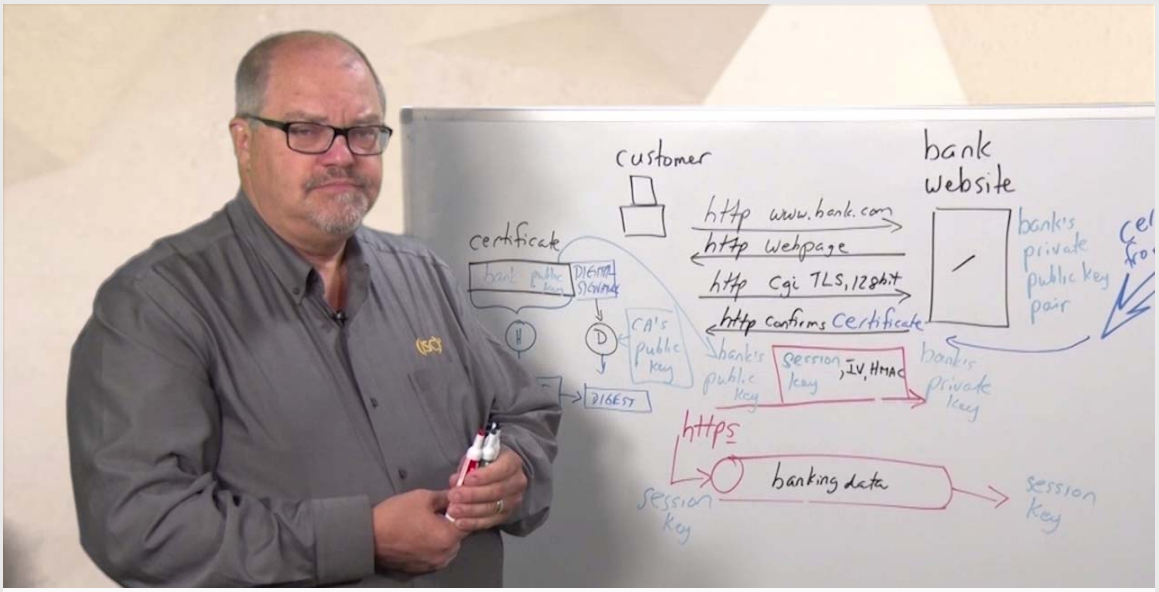
but the purpose of this demonstration was to show the interaction of the various steps in the process regardless of slight variations in deployment.

SSL/TLS

One of the most commonly used implementations of cryptography is to support e-commerce transactions. This is often done through the use of SSL (Secure Socket Layer) or TLS (Transport Layer Security). Most organizations today have moved away from SSL due to some of the weaknesses in the SSL protocol that have been identified and exploited over the past years. PCI-DSS does not permit the use of SSL any longer for secure communications.

There are two steps to this video. The scenario is that of a client wanting to do some online banking. In order to support that service, the first action of the bank is to create a private/public key pair and then obtain a certificate from a CA that provides assurance that this certificate identifies the bank and that the public key in the certificate is the bank's public key.

Once the bank has a certificate and launches a web application, the client is able to establish a secure connection to the bank. The first part of the process of connecting to the bank is shown here using an asymmetric algorithm such as RSA or ECC. It would also be possible to use D-H for key negotiation if the bank chose to do so but that is a little less common. Also note that the initial communication between the client and the bank's web application is shown to be in cleartext in this video. Some banks immediately go to an encrypted connection, while others follow this process and only establish the secure connection once the client indicates that they want to sign in. Some banks even do both depending on what country you are in!



9:28 / 9:28

▼ Transcript

Let's look at the steps involved in setting up secure online banking. We have a bank. That bank has it's website. But associated with that website, we need to have an asymmetric key pair. We have to have the bank's private key. And from that, we've calculated the bank's public key. In order to do secure online banking, the bank will provide a copy of it's public key to a certificate authority. A certificate authority is a trusted third party that generates, creates, issues, and revokes certificates. A certificate binds a public key to it's owner. And it usually follows the X.509v3 standard. The certificate authority creates a certificate. The X.509v3 standard specifies the various fields within the certificate: The serial number, the effective dates, as well as, of course, the name of the bank, and the bank's public key. Once that has been done, the certificate authority generates a hash of that certificate, quite often using something like SHA-256 these days. A hash, as we know, is used for integrity, so we can prove that this certificate will not be tampered with. And we generate a digest by running the certificate through a hashing algorithm. That digest can now be encrypted and the certificate authority encrypts that digest with the certificate authority's private key. This generates a digital signature. And it's the digital signature of that certificate by the CA. So we know that this certificate was issued by that certificate authority and that it has not been tampered with or changed. That certificate is now provided to the bank. The advantage of this is now when the bank sets up to do online banking, we have a certificate that can help us to know that we truly, as the customer, have the public key of that bank. Not the public key of someone else pretending to be that bank. Let's look at how we can practically use cryptography to setup secure online banking. We know that a bank generates a public-private key pair. They've passed that public key to a

certificate authority, so they're able to generate a certificate that can be used to link the bank's public key to the bank. Now we have a customer that would like to connect to the bank and set up a secure online session. Now there are different ways of doing this, and different banks do it differently, but we're going to use sort of a simple example here, of a way that many banks do do this process. The customer initially goes to the bank and sends a http request that says, "Hey, I would like to see your webpage." And the bank replies back, also, http, with their webpage. Now this is where it is a little different for different banks. Some banks will actually go immediately to encryption, and some don't. We're gonna show you the steps of the process here. So now the customer has the ability to see the bank's webpage and they decide they would like to sign in. When they sign in, the click on that little button that says sign in or log in, and that sends a request back to the bank that says, "Hey, we would like to sign in." And quite often, that'll be executed by executing a common gateway interface script up at the bank's website. In addition to sending that request, or run that script, they also send information about what type of encryption they will support. TLS and 128-bit. And a random number actually as well. It's used for indexing. The bank now replies back. It confirms that it will accept TLS with 128-bit, but it also sends back it's certificate it had received from the CA. The customer's browser receives the certificate. And as we know, a certificate has two main parts. We have the certificate itself and at the end of the certificate, we have the digital signature of the CA. The customer's browser hashes the certificate to generate a digest of the received certificate. It then decrypts the digital signature using the CA's public key. When we decrypt a digital signature, we will retrieve from the digital signature the digest that the CA had signed. If the digest of the receiver's certificate is the same as the one initially signed by the certificate authority, we have confidence that this certificate is the same as the one generated by the CA. It has not been tampered with or modified in transit and that it truly did come from that CA. So we have a level of confidence that the information inside the certificate, that is, the name of the bank and the public key of the bank, are correct. Having the bank's public key, we can now use that. We can send an encrypted communication back to the bank. And that encrypted communication will be encrypted with the bank's public key that we retrieved from the certificate. Within this encrypted communication, we will send a symmetric key. A symmetric key, which we will use for this online banking, because symmetric is so much faster than asymmetric, but because we're only gonna use it for this one banking session, and then we're gonna throw it out. So the customer's browser creates, or selects, a session key, a symmetric key, quite possibly an AES key. It also generates an initialization vector, and the value that will be used for hashed macking, or keyed hashing, to ensure integrity of our

communications. All of these were generated by the customer's browser, encrypted with the bank's public key, and sent back to the bank. The bank can open that using the bank's private key. And the bank now has all of the cryptographic information necessary for it to be able to setup a secure banking session. Now, we set up a secure pipe. And through that secure pipe, we can send all of our banking data. And, of course, usually the first thing we're going to send will be your account number and password. All of that traffic will be encrypted and can be decrypted using this symmetric key that we have now sent to the bank. So we have, in this case, moved from simple http traffic, to now, what we call https. The S standing for, in the old days, SSL. But we're sending http traffic over an SSL, or TLS, connection. This allows us to have secure banking with the speed of symmetric cryptography, but using the benefit of asymmetric for key distribution of the symmetric key. Hence, we have a secure banking session.

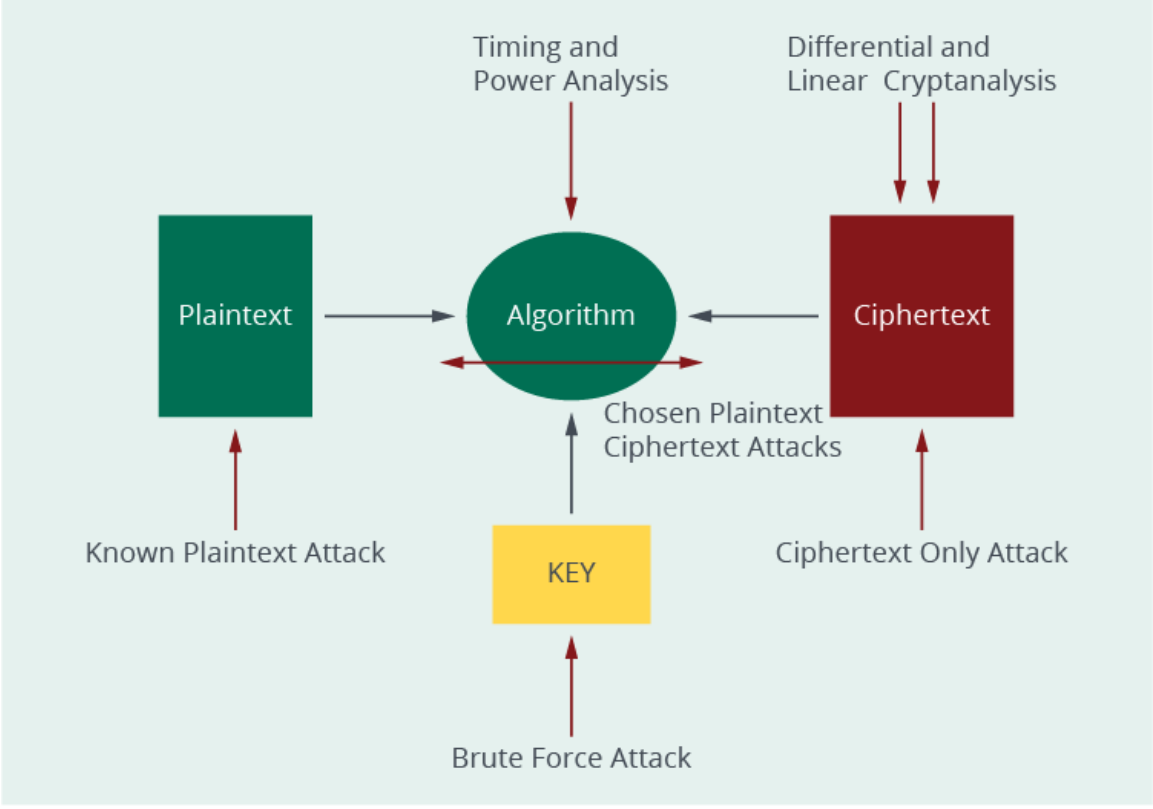
Steganography

An interesting area in the field of cryptography has been the art of steganography. This refers to the ability to send a hidden message so that no one even knows about the presence of the message. This has been done for centuries through null ciphers, invisible ink, microdots, and other secret means of communication. Steganography today often uses videos, pictures, and music to carry a hidden message that can be buried in the least significant bit of each byte of the picture. This makes the very presence of the hidden message difficult to detect or read without possessing the correct steganography tool.

Cryptanalysis

This last part of the section on cryptography will examine some of the ways that the protection of files and communications has been compromised through the science and art of cryptanalysis. There are many avenues of attack (attack vectors) that can be used against cryptosystems: to attack the key, the algorithm, the ciphertext, the implementation, or the people, but the weakest link in cryptography has to be the people. The easiest attack is through social engineering and convincing someone to disclose their password or other sensitive information.

This is a picture of some of the common attack methods:



Predictability is the enemy of cryptography and using predictable keys or patterns makes cryptanalysis much simpler. Many messages that an organization can send may be similar in format and content. If a cryptanalyst can examine several pieces of similar ciphertext, they may be able to find similarities and differences between them that could uncover some of the operations of the cryptosystem. For this reason, many cryptosystems will use

an initialization vector (IV) during the encryption process. The IV is simply a random value added to the plaintext message before it is encrypted. Since this value is random, it will ensure that even if similar documents are encrypted using the same key, the resulting ciphertext will be substantially different. One of the examples of a weak implementation was the use of the IV in WEP (wired equivalent privacy). In the implementation of the streaming cipher RC4 in WEP the IV was too short and, therefore, it was not random enough to break up patterns in the encrypted traffic, leading to a compromise of the WEB-encrypted transmissions.

Password Attacks

There are three common types of attacks on passwords (besides the ever-popular social engineering). These are brute-force, dictionary, and Rainbow table attacks. Most systems will store a password as a hash value. A hash is computed on the password entered by the users and stored in the password file (for example, a SAM file). The next time the user tries to log in, they must enter their password, what they enter is then hashed to see if the password just entered has the same hash as the hash value stored for the correct password. If the hash is not the same, then access is denied.

Clipping Levels

Many systems have a clipping level and will allow a user to try to log in several times before locking out the user account. If the user has entered the incorrect password too frequently, then the account is locked and must be reset. Some systems will only allow an account to be reset by an administrator; other systems will allow the user to reset their own password once they have answered some secret questions that validate the user. Allowing several incorrect attempts before locking the account will allow for normal human error, but locking the account after several incorrect tries will try to prevent a brute force attack.

An unauthorized user sitting at the workstation of an authorized user will only have a few tries to log in before they are locked out, but if the unauthorized user can obtain a copy of the password file then they can launch an offline attack. In this case, the attacker will attempt to hash various possible passwords to see if they can guess the correct password for a user. Many people choose passwords based on common words, and some of the Dictionary-based password cracking tools will try to use common words and adaptations of common words (substituting 0 and O, 3 and E, A and @, for example) to try to learn the passwords being used. Here is where the term

work factor becomes important. Work factor is the amount of time or resources that would be required to defeat cryptographic protection. The more difficult the password is to guess, the more time would be required by the attacker.

An attack that would take even more time is a brute force attack. In a brute force attack, the attacker would try every possible password combination to gain access.

Over the past few years, rainbow tables have been created. These are tables of all possible password hash values and the plaintext value that is used to generate that hash. If a password is based on a simple hash of the password, then an attacker can look up the hash value and immediately know the plaintext value the user uses as a password. To get around these types of attacks, the system will often add a SALT value into the password hashing process. This makes the use of rainbow tables less effective. The SALT (sometimes called a SEED) is just a value chosen by the administrator during the setup of the system to avoid password cracking.

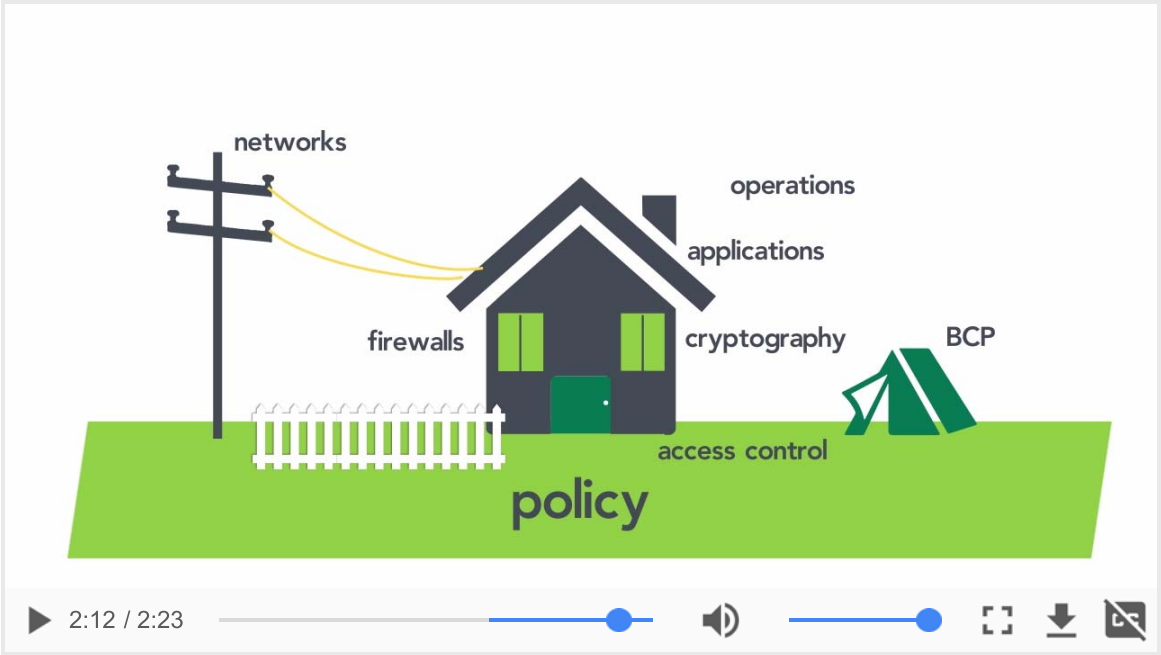
System Architecture / Interoperability of Systems Overview

One of the greatest challenges in information technology is the pace of change and the need to act without enough planning. Most LAN and System Administrators are overwhelmed by the number of tasks that need to be done immediately and the lack of time or resources to do all those tasks properly.

Over the past few decades, information technology has evolved from a small group that supported a few business processes, such as finance, to an integral platform that is the foundation for nearly all business processes. Over this time, systems have been developed that were initially independent and operated as a stand-alone system to an integrated network of systems that share data and are dependent on one another. At the same time, networks moved from small internal networks where a few users shared the same printer to global networks based on the Internet where data can be accessed from anywhere at any time. This evolution also saw many mergers and acquisitions where the applications and networks used by different companies were forced together even though they used different equipment, different record formats, and different cultures.

On top of all these challenges to the IT department, there is the added focus on the need for information security. As security practitioners, we are often stepping into a process that is already overworked and underfunded, so it is not a surprise that the IT managers are not always excited about adding security responsibilities into their operations.

Securing our systems, networks, and applications requires an integrated solution that is interoperable, easy to maintain, and easy to monitor. It is not enough to secure an application if the underlying system it is running on is insecure.



▼ Transcript

Let's take a bit of a humorous look at architecture and how architecture supports our information security program. Our information security program is based on a foundation of policy. We know that policy sets out management's intent and expectations, it gives us the authority and foundation then to build a good security program. Our security program also needs to have clear boundaries. What is within our area of control and what's outside of our area of control. And these can be things like firewalls or gateways. Every building needs a roof. The roof for us, we could call the applications. And we know it's the applications that face the weather and protect this building. But even the smallest hole in an application can lead to a compromise of everything below that. We have operations, the day to day operations of our systems, making sure they are accomplishing the desired result and supporting business needs. We need to have access control. Access control allows us to control who gains access to our buildings, networks, systems, applications, and more over our data. We have to provide a certain level of viability to our data. But in addition to visibility, we need to control access to that data as well and this is where we use cryptography. We know an important part of any structure is physical security and we put in physical security to try and protect our buildings and systems. None of our systems operate entirely as an island and instead we have networks and communications. But in case everything goes wrong it's always good to have a business continuity plan. So we can see how architecture supports and brings together all of the parts of our information security program.

Security of any one element is linked to the security of many other elements as well. As we have seen, many breaches of financial systems (for example)

have been made possible because of a flaw in email or the breach of database was related to a flaw in application design. We know that the old concept of approaching security as an “egg” where the security focus was on maintaining a hard outer shell and keeping the enemies outside of our internal systems does not apply anymore. Security must be built into each and every part of the integrated system.

As anyone who lives in an old house knows, it is much more difficult to renovate an old house than it is to build new. In the old house the walls are not straight, and it can be hard to upgrade to some of the features that would be desired. This is how it is with IT; old systems may not really be upgradeable and nothing integrates really well, we are constrained by the legacy equipment we have, the limitations of budget, and the need to support older business practices.

Architectural Vulnerabilities

There are many potential vulnerabilities in IT architecture just as there are in physical architecture. A building may have problems with its foundation, plumbing, electrical, the roof or just be in an unstable environment (earthquake). An IT system may be vulnerable in hardware, network, application, utilities, drivers, databases, or of course at the user level. A flaw or vulnerability at any level can affect the rest of the system. As security practitioners, we need to be diligent to find the vulnerabilities—only when we have identified the vulnerability will we be able to repair it. Later in the course, we will examine the processes of vulnerability assessment, penetration testing, and audit for that reason.

The most obvious problem is the lack of a security perimeter on most systems today. We commonly used to use the term “Trusted Computing Base” to refer to all of the software, hardware, and communications elements that were responsible for enforcing security on our systems and the area we had control over or responsibility for. Around these elements was the imaginary wall or fence known as the security perimeter. Ideally, traffic would only be allowed through the perimeter at carefully controlled access points or gateways, where malicious traffic could be identified and stopped before it even entered our internal systems in behind the fence. Such a concept was only applicable when there were a few controlled access points such as a firewall, however, there are far too many ways to gain access in behind the firewalls and the perimeter today. Portable media such as USB drives, smartphones, and wireless access points present new potential avenues of attack and enable an attack to bypass the security perimeter. For this reason, we need to secure every device and system and not simply trust that the externally facing firewall will protect us.

As we can see, architectural security is based on the principles of layered defense, or defense in depth. This means that systems are protected by multiple defensive levels, and access to sensitive data would require the attacker to overcome a series of controls. This also addresses the frequent problem of a single point of failure. A single point of failure is a vulnerable point that could affect more than one system if it was breached or where

there is only one control in place. Examples of these vulnerabilities could be a power supply that supports multiple pieces of equipment or a system administrator that is the only person that knows a certain system. In either case, a problem with the single point of failure could result in a serious impact on business mission.

The security practitioner should be alert to identifying single points of failure so that they can be addressed.

Architectural Design

It is possible to be busy without being productive and to expend a lot of energy but still not make progress. This is common in IT where a lot of time is taken up with maintaining and fixing systems but not enough time is put into planning and design. Architecture is based on learning the needs of the business and then designing a solution that can meet those needs. There are many parts to an information security strategy, and planning is essential to align all of those parts and integrate the pieces together into a functioning unit.

Summary

The art of discovering user needs and designing a solution before just getting to work building can save a lot of re-work and disappointment. Too many projects fail to deliver according to user expectations, and critical elements like security are easily forgotten. It is critical that security requirements are documented at the same time as the functional requirements for the system. Security that is built into the project is almost certainly more effective and efficient than security that is added on at the end of the project.

Software Overview

Many of the security breaches that organizations face today are the result of vulnerabilities in software. Software comes in many forms, ranging from operating systems to utilities and from database management systems to web applications. The problem is often that software is written to perform a function—a business purpose and security requirements are not addressed in the design and development of the software. The result is software that has built-in flaws and requires regular patching to fix those vulnerabilities. In fact, one problem might be that consumers of software products have become accepting of a “find and fix” culture for software where initial quality is not expected.

Our challenge as security practitioners is to work to integrate security into all business processes and functions, including software. We know that building security functionality into a product is much more effective and much less expensive than trying to ‘bolt’ security onto the product once it has been released.

To do this effectively we need two things: the need to understand security requirements, and the need to understand business requirements. A secure system that does not support the business is a failure. Part of the challenge is that the business struggles to describe its needs clearly, and many software development projects fail because of changing requirements or unrealistic expectations. Into the middle of that confused environment, we are attempting to introduce the added element of security.

Types of Software

The world of software is changing continuously as new languages are adopted and new delivery methods available (such as the Cloud). Early software languages such as Assembler and COBOL required substantially different programming approaches than newer languages in use today. Each language has its own characteristics and flaws that can be exploited by hackers (such as buffer overflows), and these flaws are inherent in the language and will exist in the written software code if the software developer does not knowingly remove them. If the developer does not know to remove these flaws, then they will exist in the completed program. Many of the software related flaws in web applications that are exploited by hackers are avoidable through better programming practice, but often the problem is that the developer is not aware of the need to remove these flaws. A large part of the duties of a security practitioner is education—to highlight the need for security requirements to be addressed in software and the need to educate development teams on the common problems experienced in software programs.

Originally, most software programs were custom programs written for an organization by an internal development team or by a third-party contractor. The advantage of a custom program is that it is designed to meet the needs of the business. The disadvantage is the time and expertise needed to write and maintain the software program. Older custom programs were often written with minimal attention to security, in part because they were written to operate in very different operational environment than they operate in today. In some cases, it is not even possible to build security into some older programs. Developers that are approaching retirement age wrote older programs, and there is frequently a lack of documentation or other staff with the knowledge needed to maintain these programs. This is the security risk where a developer may be a single point of failure for the organization and impact the availability of the software.

If an organization chooses to have custom software built today, they may use an internal team or outsource the development to external contractors. If a software project is outsourced, there are security concerns related to this.

These concerns include the ownership of the software, the jurisdiction in case of disputes, and privacy laws that apply to testing the software.

Code Ownership

The source code for software is often considered to be intellectual property and must be protected. Ownership of the finished product must be established by contract, whether the software source code is the property of the contractors that write the program or the company that outsourced the development.

Jurisdiction

Every contract should stipulate the jurisdiction for any disputes or legal matters that apply to the contract terms or interpretation. An organization may want to insist that the contract is enforced under their local laws and not according to the laws of the country that the outsourcing firm operates in.

Privacy

Testing a software program requires test data that can simulate all possible inputs and operational actions related to the software. This requires the creation of test data and testing scenarios. It may also require the creation of databases that will be accessed by the program. The creation of test data can take a lot of time; so many organizations may use copies of production data and databases to test the software. (Using copies of production data is preferable since a problem with test in production could cause system outages or corruption of production data). Production data may also contain sensitive information about customers, such as medical or credit card data. If such data is provided to a development team, then the developers may see confidential data they are not authorized to see. For this reason, the test data should be sanitized or masked to ensure it is not available to unauthorized personnel.

Commercial Off-the-Shelf Software (COTS)

Commercial Off-the-Shelf Software is software that has been pre-written and is sold by a vendor, usually to multiple clients. Many organizations choose to purchase this type of software for various applications: financial bookkeeping systems, point-of-sale software, and office support. This software has the advantage of being readily-available and including some level of vendor support. COTS software also has standard interfaces and formats that facilitate the exchange of data between organizations. The disadvantage is that the purchaser gets a 'boilerplate' or standard process flows that may not match the organization's processes very well. The purchaser also receives an executable and rarely has access to the source code. This makes customization and ongoing support of the software difficult.

Escrow

When an organization purchases software from a vendor, then it may feel vulnerable to a failure of the vendor to support the software, perhaps due to vendor bankruptcy or the vendor closing down support for the product. In order to protect its interests, the purchaser may request a copy of the source code so that they can address that vulnerability; however, since source code represents the intellectual property of the vendor, the vendor may be unwilling to provide access to the source code. A compromise that addresses the concerns of both parties is to keep a copy of the source code with a trusted third party—in escrow. The escrow copy would only be made available to the purchaser if the vendor was unable to meet contract terms or to provide support.

Whether the organization purchases a COTS product or develops their own software, they face the challenge of upgrading the software or fixing any vulnerabilities that are found. If the vendor does not patch the vulnerabilities, then the purchasing organization can become a victim of a breach.

Licensing

Theft of software is a crime in many countries, and a purchaser must be aware of the limitations and terms for software use and distribution. The security practitioner should watch that they only use the software according to the number of licenses they have purchased. Failure to abide by software licenses may be an act of piracy and lead to fines and embarrassment for the organization. Some software is also sold to an organization with hardcoded controls that limit the devices it can be run on, and this can affect the usability of backups or disable software in the case of changes to the hardware.

Hybrid

In some cases, an organization may purchase a software package from a vendor and then modify it to meet their requirements. This hybrid solution may be a good compromise between buying a proven product and being able to mold that product as if it were a custom software program. Purchasing a template for a program and then being able to build a customized software solution can also do this.

Open Source Software

For many years, the debate has raged in the software development community about what software is best: proprietary software or open source software. While there is no authoritative answer to this, it is an opportunity to compare the advantages and disadvantages of both.

Discussion: Open Source versus Proprietary Software

What is better open source or proprietary software?

What are the advantages and disadvantages of each one?

Software-as-a-Service (SaaS)

There are a lot of advantages to deploying a Cloud-based Software-as-a-Service solution. For many organizations, this has proven to be an excellent software delivery model in that the software provider manages the support, access to, and processing support for the software. The Cloud will be examined in more detail in this course.

The SaaS solution may be hosted on a platform owned by the software vendor directly or on a Cloud Service Provider's platform. The primary concerns of the security practitioner are the ensured availability of the software and the protection of the sensitive data that may be processed and stored on the application. A failure either in the network or on the Service Provider's platform may result in an interruption of a critical business process. There may also be legal issues if the Service Provider is transmitting to, accessing from, or storing the data in another country.

Data Security

Data must be protected according to legal requirements and the value of the data to the organization. Data is one of the most important assets an organization owns, and the loss, theft, or corruption of data may result in significant financial penalties or business interruption. Since applications and operating systems manage access to, storage, and processing of data, it is essential to ensure that data protection is built into these software elements.

Applications store data in multiple ways—in workspace memory while processing the data, and writing it to long-term memory for later access. Workspace memory is often shared between various applications. When an application begins to execute, it is assigned memory areas t it can use to store variables and perform calculations. Failure to restrict the way an application can accept and process data can lead to buffer overflows and the compromise of the system, including the compromise of the entire system. If the application allows input data that is larger than the allocated memory area, then the data can overflow into memory areas allocated to system processes, executable code, or other variables. This is known as a buffer overflow. One way to make the exploitation of a buffer overflow more difficult is to use Address Space Layout Randomization (ASLR) that randomly allocates memory to an application as opposed to the allocation of memory in sequential blocks. This randomization makes it harder for the attacker crafting a buffer overflow attack from being able to access memory allocated to pointers or executable code.

Input Data Validation

This is probably one of the most important security requirements for applications. All data that is received from user input or from other processes should be considered untrusted. The application should validate that all data is within its correct bounds (size), correct range (e.g., days of the month), and allowable values (special characters). Performing input data validation can avert many system breaches, but failure to protect against invalid input can lead to arbitrary code execution (e.g., installation of a rootkit) and compromise of the system.

Protection of Displayed Data

Data is displayed on monitors (screens) and reports. Displayed data must abide by the rule of 'need to know.' This means that if a person does not have a business requirement to view the data, then the data should not be shown to them. This includes both employees and customers. An employee working in a call center may need to update a customer's account, and this can require the ability to see payment card data; however, that data should only be visible when required and not visible otherwise. The protection of displayed data can be accomplished through masking the data (e.g., only showing the last four digits of a credit card). A report or receipt should also mask sensitive data (personally identifiable information, private health information, or payment card data).

Screen filters are a good way to protect data from unauthorized access by preventing anyone except the authorized user of the system from being able to observe the data on the user's screen.

Good practice is also to have a "clear screen" and "clean desk" policy that requires all users to lock their system and lock away any confidential documents before leaving their desk. This prevents another person from observing the data or accessing the system when the user is not present.

Swiping Attacks

A common attack against Automated Teller Machines (ATMs) or Point of Sale (POS) devices is to capture data as it is being entered into a system. This is known as 'shoulder surfing.' It is common to find that cameras are surreptitiously installed at an ATM to record PIN numbers as the client enters them in. This type of attack is known as 'swiping.'

Overt and Covert Channels

Data must be protected from compromise through either overt (obvious), or covert (hidden) channels. Both overt and covert channels may be accidental or intentional. The effect of either is the release if information in violation of law or policy. There are two types of covert channels: timing and storage.

A covert storage channel refers to the improper storage of data (perhaps on a report, USB drive, or in an insecure area) where unauthorized personnel can access it.

A covert timing channel refers to the release of information through the manipulation of resources. An example of this is to detect that a process has started by observing the change in CPU (Central Processing Unit) usage.

Shared Data Storage Security

An application that is assigned a memory area for processing may find that the memory area still contains data from a previous operation. This can lead to the corruption of the data, or the ability to see confidential data processed by the previous operation. This was a common problem with older programming languages that required the re-initialization of all variables before they were reused. Failure to re-initialize the field could leave data from the previous transaction in the field that would corrupt the next transaction.

A current example of this is the sharing of USB drives or other portable media that may have residual data from a previous use. Even if the user deleted the data on the USB, it may be possible for the next user to recover the deleted data.

Data Deletion

The deletion of data does not remove the data from the storage device, it only indicates that the memory area used to store the data is available for reuse. Data that is shown as deleted can often be recovered with minimal effort.

Overwriting data means that the area of memory used by the deleted data is now overwritten with random values. This makes it very difficult to recover the deleted data without the use of specialized equipment. When data is deleted or overwritten, there still may be some residual data left on the media, and some methods used to delete the data may not be working as well as expected. The security practitioner should test the media following overwriting to ensure that no data can be accessed.

Degaussing is the process of applying a strong magnetic force to magnetic media. This has the effect of re-randomizing the magnetic charges used to store data on the magnetic media and destroying the data itself. It will also usually damage the electronics that operate the drive. Even degaussing may leave some data remnants that can be used to recover the data using sophisticated equipment.

The problem is that all of the above methods to delete data only really work on standard magnetic media and will not work on an optical disk or a solid-state drive. To securely delete data from these types of media requires physical destruction. Physical destruction (e.g., shredding) is considered to be the most secure method of deleting data.

Old Equipment

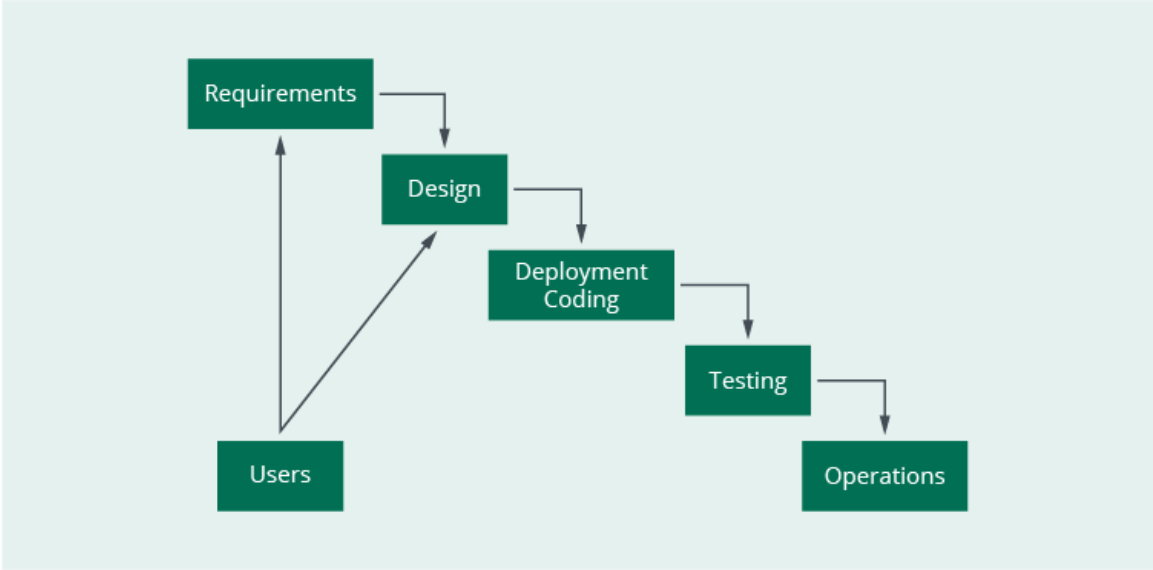
The disposal of old equipment has a security risk if the equipment had previously contained sensitive data. The equipment should be purged to ensure the removal of any data. A lot of old equipment, such as printers, scanners and fax machines, contain memory chips that contain data from previous transactions. Before such equipment leaves the organization, its memory should be purged of all data.

The Software Development Lifecycle (SDLC)

Note: the acronym SDLC can be used interchangeably between Systems and Software Development Life Cycle.

The sad fact of IT systems and software development projects is that most of these projects would have to be considered a failure from a project management perspective. IT systems projects are almost always late, over budget, and fail to deliver on user requirements. There are many reasons for this, ranging from unrealistic expectations to changes to the project requirements, and organizations have struggled for years to overcome these problems. Various SDLC methodologies have been employed such as the Waterfall, Rapid Application Development, to Extreme Programming and Agile. The reality is that none of these work—or perhaps it could be said that all of these work, but only if they are used correctly with diligence to ensure that the staff learns the process and follows it. The security practitioner is not responsible for the choice of which methodology to use, nor are they responsible for ensuring the success of the project, but they are responsible for working to integrate security concepts into the design, development, and deployment of the IT system or software. For this reason, we need to understand the SDLC methodology being used and learn how to weave security into the project.

Even though there are many SDLC methodologies, they all revolve around the same basic steps:



As can be seen, there are several problems that emerge when examining this process:

- A lack of user involvement in all phases of the project
- Difficulty with adapting to changing requirements later in the project
- Poor utilization of resources, such as programmers, during early phases
- Problems or mistakes in one phase will spill over into later phases

Even the Agile methodology (a currently lauded methodology) is really just a series of small projects based on the above diagram.

The areas of concern for the security practitioner, regardless of the methodology being used are:

- Lack of documentation making the project hard to support and maintain
- Security requirements not addressed in the Requirements phase
- Lack of time to do adequate testing because of pressure to implement
- Retention of knowledge of the development once the project has been implemented and staff is reassigned
- Security controls being disabled to increase performance

The core concept that must be remembered by the security practitioner is that the applications must be secure in themselves, it is not acceptable to depend on network security or security at any other level to protect applications. The principle of security architecture today is that the security perimeter is only a small part of the security solution, and each component and element of a system must be secure in its own right and not dependent on the security provided by other components. This creates layers of defense that make the task of the attacker difficult.

Development/Acquisition/Provisioning

When an organization builds their own applications, then they must ensure that the security requirements are built into the delivered product. When an organization purchases an application, then they have to ensure that the delivered product meets the security requirements. This requires the organization to:

- Specify the security requirements in any Request for Proposal (RFP)
- Ensure the security requirements are addressed in the vendor response
- Ensure the security requirements are in the contract
- Enable the security functionality in the delivered product
- Maintain the product's security functions and patch the product as necessary
- Monitor for vendor compliance with Service Level Agreements (SLAs) or Maintenance Level Agreements (MLAs)

Summary

Software is one of the most important pieces in the security puzzle. Software enables the operation and access to our information, our networks, and information systems, and a vulnerability in software can be compared to a leak in the roof of a building—no matter how good the building is on lower levels, even a small hole in the roof can compromise everything below it. Software must be designed to be secure and built to withstand the attacks it will face. As written in one document, all systems today must be built with an expectation that they will be attacked.

Identify and Analyze Malicious Code and Activity Overview

The term “malicious code” refers to the many types of malware in use today. In many cases, people use the term “virus” incorrectly to include all types of malware. In fact, a virus is only one form of malware.

Malware is the contraction of the two terms “Software” and “Malicious.” It is the term often used to discuss the various forms of malicious software code that have been written to cause damage or perform unauthorized activity on a system. Malware is not used to describe a software bug or logic flaw in a system because those are not written to intentionally perform unauthorized actions. There are many forms of malware in use (or as is often said, “in the wild”) today, and over the years malware has evolved as malware authors have had to discover ways to compromise a system and to achieve their goals.

Some malware is overt and obvious doing extensive damage to systems and data, while other malware is hidden and can lie dormant on a system for months or years undetected, just waiting to respond to a call from the implementer of the malware.

Early versions of malware were either a virus or a worm and often spread by passing floppy disks from person to person (Brain) or exploiting a network connection (Morris Worm). The infected floppy disk would contain a [boot sector] virus that overwrote the boot sector on the disk. When the disk was inserted into a system, the system would read the boot sector to determine what data was on the disk and load the virus sitting in the boot sector. Understandably it took years for such a virus to spread around the world. Other virus types included the macro virus that would exploit the macro language used in some office productivity products, or the various forms of malware that would spread as email attachments or through links in an email.

Types of Malware

A virus is a self-replicating piece of code that spreads without the consent of the user. It infects a host system. Many virus infections cause extensive damage while others are humorous or just for fun. A virus will often infect other programs and spread to other systems.

A virus often is constructed of several parts: the infection mechanism that searches for new victims or programs to infect; the trigger that launches (activates) the virus; and the payload which then executes the intended purpose of the virus. An example of a well-known virus is Melissa.

A worm is another form of malware, but it is not the same as a virus. A worm will also self-replicate and usually will spread through a vulnerability in a system, but unlike a virus, it does not infect a host or program. Well-known worms include Kournikova and SQL Slammer.

A Trojan Horse is a type of malware (usually a virus) that deceives the victim. This exploit is commonly based on social engineering where the victim is manipulated to do something they should not have done. Trojan Horse malware appears to be good but has malicious content hidden inside. For example, a user may download a game or picture not realizing the game contains code that will compromise their system, perhaps installing a keystroke logger to record all keyboard entries from the user, or stealing data.

Ransomware is one of the most common malware attacks in use today. Ransomware will take over a victim's machine, encrypt their files, and only allow the victim to regain access if they pay a ransom to the attacker.

Logic Bombs are malware that will trigger their payload on a specified date or action. (The trigger in a virus is actually a type of logic bomb.) Examples of logic bombs have included several attacks by an ex-employee of an organization that installed code on the organization's system to do serious damage to the systems and data of the organization if the former employee's name was no longer in the payroll file.

Rootkits, backdoors, trapdoors, and wormholes are some of the ways to gain privileged access to a system. In the case of a Remote Access Trojan, it allows an unauthorized person can take control of another person's machine. A rootkit may be installed through a worm or buffer overflow and, once installed, can be incredibly difficult to detect or eradicate without rebuilding the entire system.

Phishing attacks are a type of social engineering attack that attempt to convince a victim to provide confidential personal information to the attacker. Phishing attacks are often based on emails or instant messaging applications that direct the victim to a malicious website in order to enter data. Often these sites impersonate a bank or other trusted entity. Phishing emails are often distributed as spam to a large group of potential victims.

Spear phishing attacks are a targeted form of phishing that contain personal information about the intended victim to appear more legitimate. Like phishing attacks, they are intended to convince a victim to disclose personal information.

Whaling attacks are a type of spear phishing aimed at executives of an organization. A whaling attack often is sent to the executive as a demand for immediate payment of a fake "outstanding" invoice with a threat of litigation. This may result in the executive approving the invoice for payment without checking carefully on whether the invoice is legitimate. Another form of whaling is done by sending an email to the finance department of an organization appearing to be from the CEO authorizing an immediate wire transfer to a "vendor," but it is an entirely fictitious email. These types of attacks have defrauded organizations of large amounts of money.

Zombies are computers that are controlled and can execute commands when instructed by a third party. Such victim computers may have been infected with a program that allows remote manipulation of the victim. This may allow the person controlling the zombie to use it to send spam, launch a denial of service attack, or participate in some other unauthorized activity. When several infected machines are combined into a single network, this is known as a robotically controlled network or "botnet."

Botnets may include thousands of compromised machines that are under the control of a "botherder" who can manage and manipulate the machines to execute attacks. Modern botnets may include many devices besides traditional computers including networking devices (e.g., wireless home routers), IP cameras, or any other Internet of Things (IoT) device that has network access and is insecurely installed. Botnets have been used to launch massive Distributed Denial of Service (DDoS) attacks in recent years that can

take the website of an organization, a government agency, or even an entire country out of service.

Spyware is code written to steal information from a victim. The victim may lose banking data, passwords, or other confidential information due to spyware. Spyware comes in several forms such as a keystroke logger that captures each keystroke as the victim enters it (gleaning passwords), screen scrapes that record the data on the user's screen, and software that enables the victim's camera to take pictures of the victim without the victim's knowledge.

Scareware is a form of social engineering that attempts to extort money from a victim by deceiving them into thinking their computer has been compromised or that they owe money to a government agency that must be paid immediately.

Attackers

There are many types of people involved in malware. The malware author may just write malware that they sell for other people to use. In many countries it is not illegal to write malware, it is only illegal to use it, so the malware author is relatively protected from prosecution. The thief is a person that attacks a victim for financial gain, such as the compromise of a credit card or bank account. The “script kiddie” was a derogatory term used for the immature hacker in the 1990s that would deface a webpage or do damage just to show off for a “friend” or prove their skill. An attacker could be motivated by revenge or ideology. Such attackers could be ex-employees, customers that want to exact some revenge against the victim, or an activist that launches an attack against an organization for ideological reasons such as political or social causes. This is often known as “hacktivism.”

Advanced Persistent Threats (APTs)

Advanced Persistent Threats are the polar opposite of the “script kiddies” mentioned earlier. As their name suggests, APTs are highly skilled (advanced), determined and tenacious (persistent) and a real threat. They are often very adept at hiding their activity and breaking into systems without the knowledge of the victim. APTs usually consist of a team of skilled professional attackers sponsored by a government, military, or organized crime syndicate that are employed to conduct industrial espionage, theft, or compromise of military or government systems of another country, or launch major financially motivated attacks against people, financial institutions, or organizations.

The argument has been made that it is nearly impossible to prevent a breach by an APT, therefore, security is an idealistic dream. This may be true, however, it misses the core point: most APTs are successful because organizations do not practice basic and known security fundamental principles, not because of the APTs skill or motivation. In fact, we know that many reports indicate that the majority of security breaches would have been avoidable through following basic security principles. The security

practitioner should know that it is important to build a solid security framework of good practices and thereby avoid a majority of breaches.

Zero-Day Exploits

A common misconception in the news media is the perception that most security breaches are the result of zero-day exploits. A zero-day exploit is an attack that is launched at the same time as a vulnerability is discovered. The result is that organizations do not have time to deploy a patch or take countermeasures to prevent the exploitation of the vulnerability. In other words, the time from the discovery of the vulnerability until the attack is zero time.

The launch of a zero-day attack would require the attacker to have discovered the vulnerability and kept its presence secret until an exploit had been developed. In actual fact, this is rare. Most vulnerabilities are found and become public knowledge weeks before any attacks have been developed that can exploit those vulnerabilities. The attackers write an attack to exploit a known vulnerability, counting on the fact that many organizations are slow to roll out patches or deploy countermeasures that would fix the known vulnerability. Such organizations, or people, become a victim of the attack. This is an example of completely avoidable attacks. (But patching is not easy, and fixing this problem is not simple, as will be addressed in this course.)

Web Application Based Vulnerabilities

Security was much simpler when the only way to gain access to an organization's data was through physical presence on the organization's network or in-house applications. When the employees went home at the end of the day and vacated the office, then the security practitioner could relax knowing that the systems and data were relatively safe from unauthorized access until the next business day. This all changed with the introduction of the Internet. As organizations created websites and web applications, they opened up access to their systems to a worldwide audience that could access systems at any time, day or night. Web applications were built for function—to support business—and they rarely had adequate levels of security. Software development often moved from a professional team of developers with formal education to an innovative and creative group of web designers that were self-taught or had not been educated about the security risks associated with deploying applications on the Internet (to be fair, very few people really knew what risks the Internet would present).

The rapid deployment of web applications was often driven by a "push to market" mentality that forgot the painful lessons of the past related to software development. Lessons were forgotten such as "Test thoroughly," "Document," and "Follow an SDLC – Do NOT make changes in Production Systems." Once again, organizations had to learn the hard way about the value of following such software development principles.

The result is that many web applications are vulnerable to well-known attacks. These attacks do not just affect web applications; they provide access to the data and other underlying functionality of the IT systems that can be accessed through those applications. They also commonly exploit problems in the architecture of the system hosting the web application, for example, placing a database in the Demilitarized Zone (DMZ) that hosts the application. This allows an attacker to gain access through the application to a database containing customer data such as credit cards.

OWASP

OWASP (owasp.org) is the Open Web Application Security Project, which is a not-for-profit organization that encourages the development of secure web applications. One of the products available from the OWASP website is a list of the Ten Most Critical Web Application Security Risks. It is an excellent document that every web application developer should be familiar with. It describes the commonly found web application problems. It also explains how such vulnerabilities can be exploited and how to test for the presence of the vulnerability. Nearly every week brings news of another organization that has been compromised because of one of these well-known, documented, and preventable vulnerabilities.

Some of the common web application vulnerabilities include:

- Injection attacks (SQL, LDAP, OS Command)
- Cross Site Scripting (XSS)
- Cross Site Request Forgery (CSRF)
- Broken Authentication
- Insecure Direct Object Reference

Discussion: Vulnerable Web Applications

How can an organization avoid the trap of implementing a vulnerable web application?

Detecting and Preventing Malware

Some malware infections can be extremely difficult to detect. Malware authors continually adjust malware to avoid detection, and many rootkits are so buried in the core of the system that they cannot be detected by most anti-virus products. However, it is possible to detect many malware infections by monitoring system and network activity. Malware will often consume CPU and network resources, run unauthorized processes on the system, congest bandwidth, or slow down system performance/response time. Malware may also cause unexplained changes to a system including default applications or browser settings.

Every organization should also deploy security tools such as firewalls, anti-virus and anti-spam solutions. Real-time traffic should be examined on the network to detect malicious activity, and disk drives and end-point devices should be scanned on a regular basis to discover any malware code that may have been installed.

Anti-Virus Products

The use of anti-virus products is strongly encouraged as a security best practice and is a requirement for PCI-DSS compliance. There are several anti-virus (AV) products available and many can be deployed as part of an enterprise solution that integrates with several other security products.

AV systems try to identify malware based on the signature of known malware or by detecting abnormal activity on a system. These will be examined in more detail at the same time as IDS (Intrusion Detection Systems) and IPS (Intrusion Prevention Systems).

There are two problems with AV systems: the first is that a signature-based system is only accurate if the signatures are up to date; and that an anomaly-based system is only effective if it knows what “normal” activity is.

A common problem with many security products is the lack of time that security practitioners have to operate, maintain, and learn about the

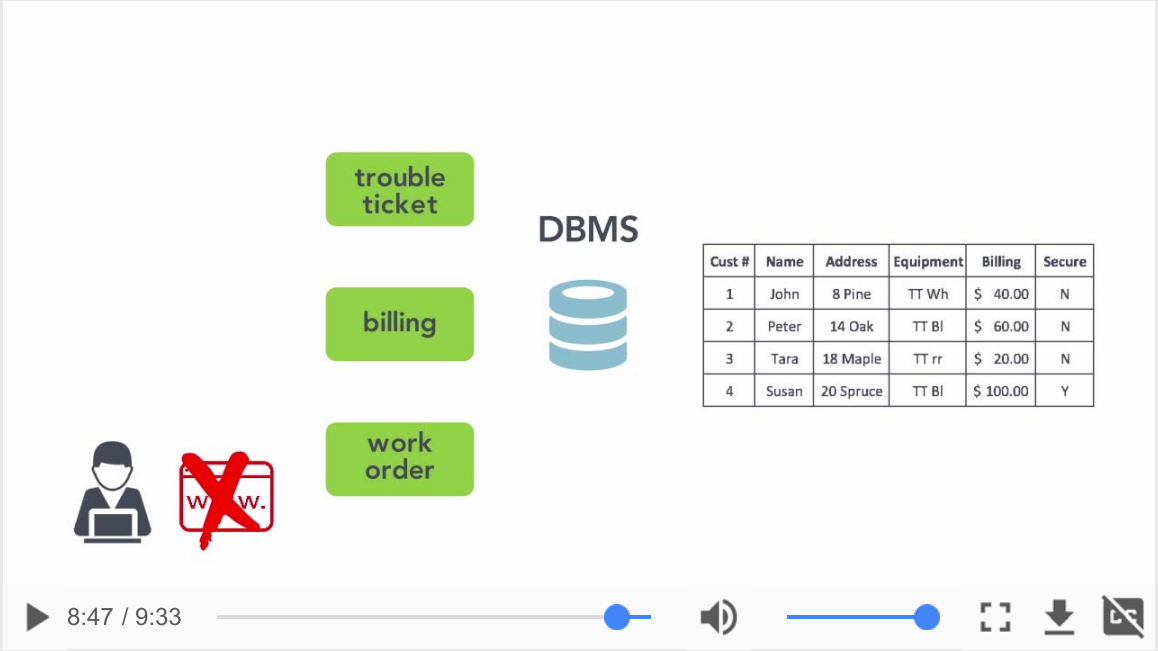
product. This means that many security products fail to deliver up to their full potential.

Countermeasures

Without a doubt, the most effective countermeasure to malware is education. The time spent in educating staff and customers about the risk of malware and how to detect and prevent malware is often the best investment an organization can make. We often hear that security is a combination of People, Processes, and Technology; this means that educating people, having good technology, and processes, such as patch management and change control, provide a robust and resilient security solution.

System hardening is where all protocols, services, and functionality not required for the system's operations are disabled. This reduces the attack surface available to the attacker. The fewer points of entry the attacker has, the easier it will be to keep the attacker out.

Video: Filing and Limiting Access



▼ Transcript

You've all met people whose office is a complete, cluttered mess. They will always tell you they know where everything is. There's piles of paper on their desks, on their chairs, on their floor, and everything they need is right there at hand. However, for us, we realize it's not very well organized. Some pieces of paper they might need are in one pile or another and finding those is often a case of shuffling through many different documents. The idea was developed many years ago of developing a filing system based on a filing cabinet. We have a filing cabinet with a drawer and file folders in it that we can organize everything according to project or according to however they should normally be grouped together, something we call today, normalization. But there's a problem with the filing cabinet. A filing cabinet has the limitation that it's a physical device and if you're not physically there you can't retrieve a document from that filing cabinet. You also have the problem that if you're working on a project with somebody that's located in a different part of the country, they will have a copy of all the documents and so will you. And anytime we have duplication of documents, we also have, of course, the introduction of inconsistencies and errors. Today we use

electronic filing cabinets, databases, and databases have been developed specifically to help us organize all of our data into very logical groupings that can be accessed from anywhere in the world, especially with the cloud today. Databases have evolved a lot over the years from old hierarchical databases to network databases to, of course, a lot of the databases we use today are relational databases, a relational or a table and an example of that we show it here. We have a relation, a relational database model, which we can see has four different customers in it and the various attribute or descriptions of those four customers. The database is based on a primary key, the index we can use to find individual records. Even though we could have customers with the same name, each one would have their own customer identifier to ensure we can keep one John separate from another John. The idea of a database is that a database is a way to organize our data that's easily accessible by many different applications through the librarian or the Database Management System. The Database Management System manages the access or retrieval of information and updating of information within that database. And various applications can access it, so let's say, for example, we have a customer service representative who gets a phone call from a customer. That customer calls in and says, I have a problem with my phone. They access a trouble ticket application, which then looks up that customer's record within the database through the Database Management System. But someone who's dealing with trouble tickets probably does not need to see billing information, so therefore, we'll often use something called a database view. A view is where we determine a subset of the data that will be accessible or visible to that application. For example, a person logging in using the trouble ticket application would usually only be able to see the information that was relevant for them. The name, address, and type of equipment, who would not need to see billing or other information that might also be in the same database. That view-based access control helps provide, should we say, enforcement of the security concept of need to know. However, let's look at a very real example. We have a condition where a customer called in to a customer service representative and said, I have a problem with my phone. Could you please send Leo or Claude out to fix that phone? Well, why would a customer ever ask for a specific repairman? Because that customer happened to have a high security flag on their account. They had an account which was marked with a flag saying this is a secure account and the person that was handling that request, the customer service representative, was not entitled or authorized to see the information about that customer. So that customer knew that if someone was gonna come to fix their phone, they had to specify the repairman that did know where they were located. This is a form of access control we call content dependent. And content dependent access control is where we make a

determination usually associated with a database of what information to show based on the specific content in one field within a record. This is one of the most granular forms of access control we have. It means that the data has been requested from a trouble ticket application through the Database Management Systems, the database. The information was returned and now is filtered by the application to say, oh wait a second, if this is a high security account, please don't show it. In this case, what actually happened was Claude went out to fix that customer's phone. When Claude went out, he realized he did not have the right equipment on his truck, so he had to change the type of equipment. When he calls back into the customer service representative he said, would you please now create a work order to update the equipment in the database. What do we have now? We have something we call multiple paths to information. A database frequently is accessible by multiple, different applications or multiple paths to that info. And what happened in this case, is that the content dependent access control was enforced by the trouble ticket application, but could be bypassed by the same customer service representative using the work order application instead of using the trouble ticket application. Obviously, a breach of the security intent of limiting this person's access according to need to know. We had another very good example of this with a phone company in the United States, where a customer accessed through the internet, through the company's website, and said could I please have you come fix my phone? The website called to trouble ticket application. The information was retrieved and returned back to the customer's browser. The customer's browser then took a look at that information and said, oh wait a second, is this a secure account? If it is, then I should not show the data, but you can understand the problem with this. The data has already been returned to the browser and simply looking at viewing the page source, the customer was able to see the data of secure accounts that they should not have had access to. So there's several concepts that we've discussed here with relation to databases, the benefits and preventing duplication, allowing for access, reducing the number of errors that we could have. But also, through managing access through view-based access control, content dependent access control, but also the risk we have with multiple paths to information, or where the controls were enforced in the wrong place within the process.

Summary

Protection against malware is an important responsibility of the security practitioner. The security practitioner should recognize the types of malware and know how to prevent and detect a malware infection.

Mobile Device Management (MDM) Overview

Control over software has become even more difficult with the use mobile devices throughout the organization. As employees migrate to the use of mobile devices for corporate work, there are new security concerns. When sensitive data is on a device not directly controlled by the organization, it can be more difficult to ensure the protection of the data and to prove compliance with privacy laws and regulations.

Remote Working—Telework

Organizations can save a lot of money and possibly increase employee satisfaction by allowing employees to work remotely. If an employee can work from home instead of going into an office, they may be more productive and enjoy flexible working hours without losing several hours a day commuting in traffic. The organization also benefits since it does not need to lease/acquire office space for remote employees.

There are many security concerns with supporting remote employees:

- Is the remote location physically secure and safe?
- Are all data backups and network communications encrypted (VPN)?
- What data and equipment is stored on-site?
- Is there secure disposal for paper and equipment?
- Is the wireless configured correctly?
- At the termination of employment can backups and equipment be recovered?

Employee Owned or Managed Equipment

The organization may provide teleworkers with the equipment required to work remotely such as laptops, shredders, remote access tokens, mobile phones, Internet access, and desks. This is sometimes called Customer Owned – Personally Provisioned Equipment (COPE).

They may also allow the teleworkers to acquire equipment of their choosing and use personally owned equipment for business (and personal) use. This is sometimes called Bring Your Own Device (BYOD). Another option is to present the teleworker or employee with a list of approved equipment and require the employee to select (choose) one of the products on the trusted list (Choose Your Own Device/CYOD).

Letting employees choose the device that is most comfortable for them is good for the employee but presents additional challenges for the security practitioner. The organization loses some control over standardization and privacy. If the employee is allowed to use their phone and laptop for both

personal and business use, then it can create a challenge if the device has to be examined for a forensics audit. It can be hard to ensure that the device is configured securely and does not have any backdoors that could be used to access organizational data or systems.

The first step to addressing security challenges is almost always policy and that is certainly true in this case. Before allowing access to the networks, systems, and data of the organization, policies should be in place to mandate the conditions the employee must follow in order to use a personal device or work remotely.

Virtual Machines

One of the ways that organizations control the use of mobile devices is Mobile Device Management software. This software can operate in multiple ways, but the common purpose is to exert control over the data on the mobile device. Some of the benefits of MDM include:

- Remote device locking (remote locking or wiping of a lost or stolen device)
- Secure email
- VPN (Virtual Private Network) configuration
- Secure browsing
- Segregation of data

By ensuring that all organizational systems run in a virtual machine (VM) or a sandbox, it is possible to limit the ability of outside applications from gaining access to sensitive data or systems.

Organizations may also limit which non-organizational applications can be run on the machine. Knowing that many mobile applications contain backdoors or do not have an adequate level of security, the organization may restrict access to untrusted applications.

Early computer systems were hardware-based. The owner purchased hardware and then installed an operating system that acted as the interface and managed the use of the hardware resources on behalf of the user. The core of the system was the hardware, and the operating system just made the hardware work. However that direct reliance on hardware has changed in many areas of information technology over the past few decades as there are many systems now where the software manages all the operations and the hardware is just an underlying platform. This can be seen with software-defined storage, software-defined networks, and the use of virtual machines. Now through the use of virtual machines, one piece of hardware can run multiple operating systems. Hardware has become a platform, and software manages that platform. A virtual machine creates a software copy of the system that looks and acts like a real system, but it is only a software-managed interface/environment for the user to work in. One piece of

hardware can be running multiple Virtual Machines at the same time with different operating systems. The user can be working in a VM and running applications, but the VM is isolated from other VMs and from the underlying hardware. This protects the hardware and the other VMs since a problem in one VM should not spill over and affect other VMs. A VM is often used as a sandbox, which is an isolated environment that can be used to run untrusted or unproven software. An anti-virus analyst will often execute suspicious code in a sandbox so that they can examine the code without damaging their system. If a VM is compromised then it is usually just shut down and restarted or rebuilt thereby protecting the underlying hardware.

Hypervisor

Many virtual machines operate through a hypervisor that acts as the interface between either the underlying hardware and operating system or just the underlying hardware.

A bare-metal hypervisor runs directly on the hardware. This is often known as a Type-1 hypervisor. Multiple VMs running on the hypervisor can be running various operating systems. This type of system has been in use since the mainframe days.

A Type-2 hypervisor runs on top of the operating system that is running on the hardware platform. This is often known as a hosted hypervisor.

The host hardware is known as the host machine, the virtual machines are known as the guest machines.

A VM is an asset that should be controlled through Asset Management. This will ensure that there is a secure baseline configuration for the VMs that can be used to launch the VM.

The security issues related to VMs are to ensure control over the configuration and use of VMs to ensure secure operation and to ensure that the hypervisor is patched since it presents a new attack surface for a person wishing to compromise the system.

The use of VMs is a core component of cloud computing. This course will examine the benefits, characteristics, and risk of cloud computing.

Summary

The needs of the business are changing rapidly as organizations evolve to meet new business processes. Teleworking and Virtual Machines are two of the most important changes in business practices in the past decade, and they present new security challenges for the security practitioner.