4

Internet Security

The Internet was originally designed to be a shared resource between trusted users. But as the Internet opened up for public use in the ’90s, it didn’t take long for people to use it for malicious purposes.

Unlike system failures, which are often “internal” and can be solved with more reliable infrastructure, attacks on the Internet stem from bad actors preying on unsuspecting users or unprotected systems. This leads to the need for network security.

This chapter explores the basics of network security. We’ll first look at the many aspects of security before moving on to cryptography. Finally, we’ll explore common types of cyberattacks and how to defend against them.

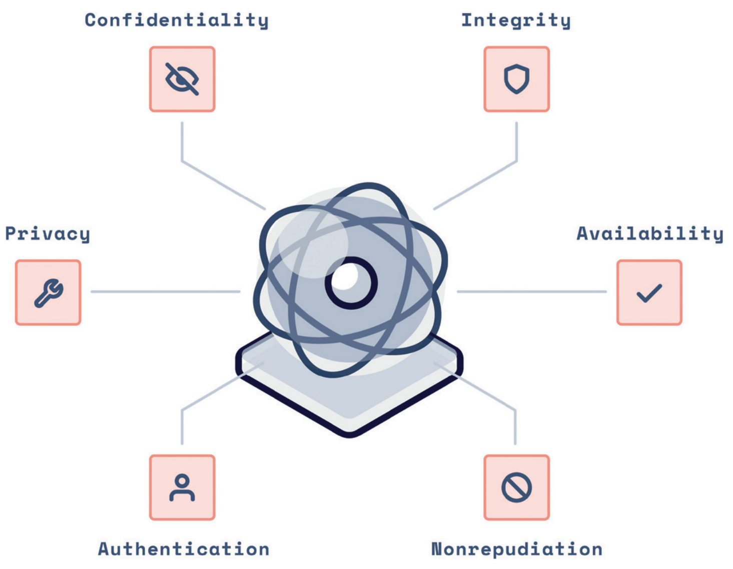
The concepts explored in this chapter are important for any modern Internet user and will be fundamental to understanding the content of the next chapter, on blockchain.

Six aspects of security

When most people think about Internet security, they generally think about firewalls, viruses, and hackers. While these are important components in the field of security, they don't provide a definition for it.

Simply put, network security is the field concerned with protecting the confidentiality, integrity, and availability of networks. In addition to these three core properties, endpoint authentication, nonrepudiation of actions, and privacy are also highly valuable attributes.

Six aspects of security



Confidentiality: Only the sender and intended recipient of a message should be able to access the contents of the message.

Integrity: Messages sent over the network are safe from intentional or accidental modification while in transit.

Availability: The network can continually function as intended.

Nonrepudiation: Users cannot deny actions conducted over the network.

(Endpoint) Authentication: Users can verify each other's identity.

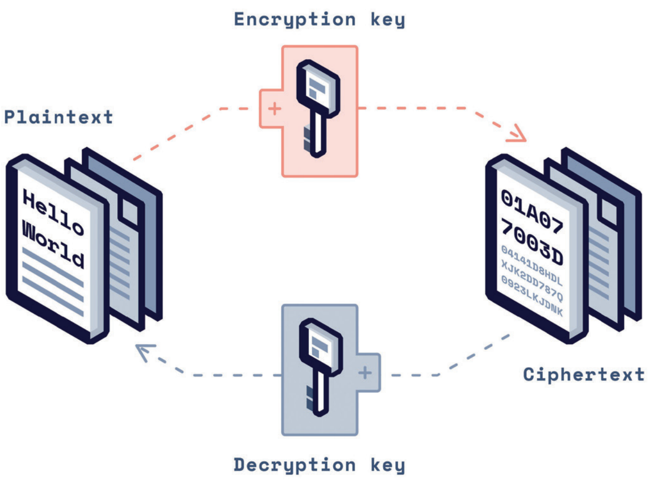
Privacy: Users have control over their online information. This is often confused with anonymity, where users can hide their true identity. A museum's donor wall maintains privacy but not anonymity—it highlights the names of the donors without disclosing the amounts each donor gave. On the flip side, cryptocurrencies often focus on anonymity but not privacy—you might be able to transact without tying your account to your physical identity, but every transaction is recorded for all to see.

Cryptography and the basics of encryption

Cryptography is an essential part of computer and network security. Broadly defined, it is the field of disguising (or translating) messages so they are protected from unintended recipients.

In practice, cryptography involves taking **plaintext** and applying an **encryption key** to it. This produces **ciphertext**. Once the encrypted ciphertext is produced, it can be sent over an unsecure channel (like the Internet) to its intended recipient. To recover the plaintext, the recipient applies the correct **decryption key** to "reverse" the encryption.

How encryption and decryption work

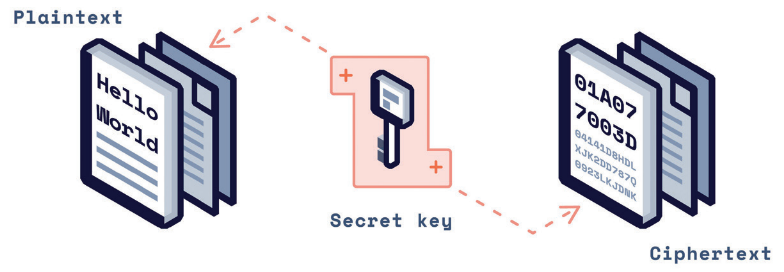


Encryption keys are generated by **encryption algorithms**. While encryption keys should be carefully protected, modern encryption algorithms are actually widely known. Public lockers make a useful analogy: Everyone knows the same type of lock (i.e., encryption algorithm) is used for each locker, but you can't open an individual locker if you don't know the number combination (i.e., the encryption/decryption key).

There are two main types of encryption:

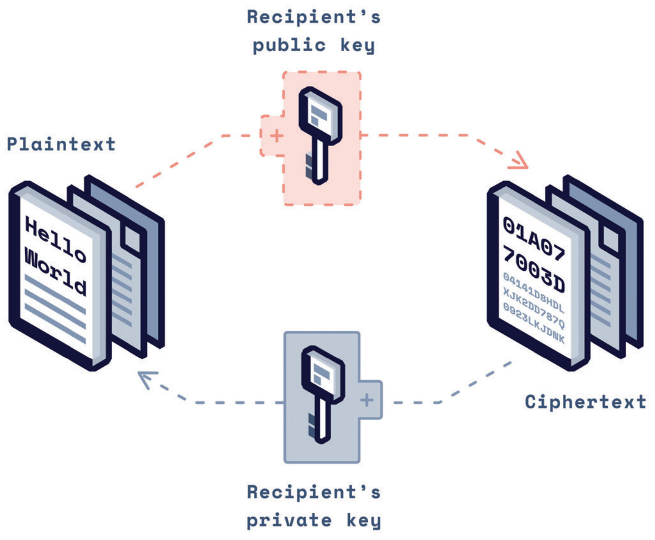
In **symmetric key encryption**, the encryption and decryption keys are the same. Because there is only one key used in symmetric key cryptography, it must be kept secret at all times and is therefore called the **secret key**. DES (Data Encryption Standard) and AES (Advanced Encryption Standard) are symmetric key algorithms.

How symmetric key encryption works



In **public key encryption**, the encryption key can be made public. The decryption key is only known to the recipient. When using public key encryption, the sender encrypts the message with the recipient's public key before sending it. The recipient applies their private key to the ciphertext to decrypt the message. RSA (Rivest–Shamir–Adleman) is a well-known public key algorithm.

How public key encryption works



Before we move on to cryptographic applications, it's important to stress that encryption is not a silver bullet for every security problem. Encryption algorithms are very hard to break, but attackers have been able to crack poorly implemented encryption systems by looking at other vulnerabilities. During the Cold War, Soviet intelligence was able to decrypt over 3,000 messages from Western countries by simply reusing old keys. In effect, modern cryptography has transformed encryption problems into key-management problems.

Used properly, encryption keeps your data and files safe. But keep in mind it can also be used for nefarious purposes. Attackers can just as easily use encryption to stop you from accessing your files or devices. Locky (released in 2016) and WannaCry (2017) are two examples of ransomware used to encrypt a victim's files and extort payment.

Cryptographic applications for message integrity

In the previous section, we looked at the basics of cryptography and how encryption can keep messages safe from prying eyes, even when the messages are traveling over an unsecure medium like the Internet.

Now, we delve into another important use for cryptography: maintaining the integrity of messages.

Cryptographic hash functions

A **hash** is a fixed-length "fingerprint" of a message. For any message *m*, a hash algorithm can compute a unique hash ***h(m)***. Even the smallest changes to the message will produce a very different hash. This makes hashes very useful when checking whether a message has been modified or tampered with.

Unlike encryption, which is a “two-way” cryptographic process, hashing is a one-way operation: It is impossible to reverse-engineer a hashed value to reveal the original message. This is because the set of all possible messages (including messages of any length) is so much bigger than the set of fixed-length, hashed messages. It’s therefore impossible to come up with a function that “reverses” hashes to give the original input.

A well-known hashing algorithm is SHA-256. With SHA-256, the output is always 256 bits (or 32 bytes) in length.

Message authentication codes

A hash can ensure a message isn’t tampered with or modified, but it cannot guarantee that a particular message is what the original sender intended to send. This is where **message authentication codes (MAC)** come into play.

To use a MAC, two communicating parties need a shared secret key, *k*. The sender generates the MAC by calculating the hash of the message and the key, *h(m+k)*. The original message and the MAC, *m* and *h(m+k)*, are sent to the recipient.

When the recipient receives *m* and *h(m+k)*, they can use *k* and *m* to calculate *h(m+k)*. If the calculated hash is equal to the MAC, the recipient can be sure the message was sent by the real sender (assuming they are the only ones who know the secret key).

Digital signature

As the name suggests, **digital signatures** can be used to verify the identity of a person online. Unlike hashes and MACs, digital signatures use public key encryption to prove, in an unforgeable way, that someone is who they claim to be.

To digitally sign a document, the signer produces a hash of the document and applies their private key to it. To verify the signature, anyone can apply the signer’s public key and recover the hashed document. They can then calculate the hash of the document and compare it with the hash of the document that was signed. If the two values match, then the identity of the signer (as well as the ownership of the document) is verified. All of this hinges on the assumption that the signer’s private key was not stolen or used against their will.

Digital certificate

Digital certificates are used to bind names to public keys. They certify that an entity owns the public key they claim to own. This binding is done by a **Certification Authority (CA)**, which verifies an entity is who they claim to be before issuing a certificate that binds the name and public key of the entity.

Generally, a certificate contains:

- the public key for the entity,
- unique information about the owner of the public key,
- a serial number for the certificate,
- issue and expiry dates for the certificate, and
- the digital signature of the CA.

Digital certificates are useful because they allow us to verify the public keys we see online. As long as you trust the CA that issued the certificate, you can be sure the public key for an entity actually belongs to said entity.

As you might expect, trust in digital certificates boils down to trust in the CAs. Therefore, a CA has extra incentive to protect its private key (which is used to digitally sign any certificates it issues).

The system for issuing, managing, and revoking digital certificates is (unhelpfully) called **public key infrastructure (PKI)**.

Email security: PGP

One of the most popular encryption programs around is PGP, which stands for **Pretty Good Privacy**. PGP can be used to sign and encrypt things like files and emails.

To get started, find and download a trusted version of PGP. Good examples can be found below:

- Windows: gpg4win.org
- Mac: gpgtools.org
- Android: guardianproject.info/code/gnupg
- iOS: ipgmail.com
- Linux: wiki.gnome.org/Apps/Seahorse and enigmail.net

The PGP software will help you create a public key and a private key. The public key can be freely shared, but you'll want to keep the private key to yourself. The private key is protected by a password, so be careful when choosing and maintaining your password. If you lose the password, you'll lose access to your private key too.

What can PGP do? Well, you can sign a file by applying your private key. Others can verify your signature by applying your public key to the signature. Or you can encrypt a file by applying a friend's public key so only they can read it (by applying their private key).

As useful as PGP is, some caution is needed. Since anyone can upload and share keys, you'll have to make sure a public key you're planning to use actually belongs to that person. PGP also received flak when vulnerabilities in email clients led to messages being decrypted. To defend against flaws like this, update your version of PGP regularly and make sure you haven't accidentally enabled any features you don't need (or disabled any that you do).

Cyberattacks

After exploring the basics of cryptography, encryption, and ways to validate messages and identity, we now look at some of the most common attacks on the Internet.

As we'll see, although there are a multitude of individual attacks on the Internet, many of them operate on the same underlying logic. We start with one of the most common: malware attacks.

Malware

Malware, as the name suggests, is malicious software that infects a host and performs unauthorized actions. There is a wide range of malware on the Internet, from **keyloggers** that record keyboard input to **ransomware** that locks your device (by encrypting all of its data) and demands payment to its creator. Here are the most common types of malware:

Virus

A **virus** is a piece of self-replicating code that activates or spreads with help from the host. Typically, a user needs to perform some action (e.g., clicking a link) to download the virus onto their device. Once a virus has entered a device, it inserts itself into other programs.

Worm

A **worm** is similar to a virus but can enter a host without user interaction. Worms do not need to attach themselves to another program. Rather, they self-replicate and spread through network vulnerabilities.

Trojan horse

A **Trojan horse** is malware that disguises itself as legitimate software. A Trojan horse does not replicate itself, but it is used to sneak malicious software onto a victim's device.

Defending against malware

There are several ways to defend against malware. A **firewall** can block unauthorized traffic and filter out packets that are deemed dangerous. We'll cover firewalls in greater detail later. **Antivirus software** can scan your device and remove malware. Since malware is always evolving, it's a good idea to periodically scan your devices and update your antivirus software whenever an update is available. Finally, an **intrusion detection system** performs more thorough packet inspection than a regular firewall and is often needed for more complex types of malware.

While all of these tools provide useful defenses against malware, common sense is also essential. Don't open suspicious links, don't visit sketchy websites, don't download anything from a source you don't trust, but do keep your software up to date. You can inspect a link by hovering your cursor over it. If you're on a mobile device, tap and hold (but don't let go too soon!) or copy the link into a text editor. Carefully examine the link for typos or unexpected characters before following it.

Denial of Service attacks

Denial of Service (DoS) attacks work by overwhelming a resource with bogus traffic, rendering the resource unavailable for legitimate users. When a DoS attack is carried out by a multitude of malicious hosts, it becomes a **Distributed Denial of Service (DDoS)** attack.

DDoS attacks are usually carried out by a **botnet**, which consists of a group of malware-infected hosts. Botnets are directed by a bot controller.

DNS flooding attack

An example of an application-layer DoS attack is a **DNS flooding** attack. Here, the attacker sends a large amount of bogus DNS queries, overwhelming the DNS servers and preventing legitimate DNS requests from being serviced.

Defending against DNS flooding

If all of the bogus DNS traffic is coming from the same source, an upstream router can block the traffic from the attacker and keep the DNS server available for legitimate DNS requests. These days, most servers also limit the amount of resources available to each visitor to prevent abuse from bandwidth-flooding.

SYN flooding attack

A **SYN flooding** attack follows the same principle as a DNS flooding attack, but instead of sending DNS queries, the attacker initiates a deluge of TCP handshakes by flooding the target server with SYN segments and not completing the handshake process. The hope is that all of the server's resources are exhausted by these bogus SYN messages, shutting out SYN requests from legitimate users.

Defending against SYN flooding attacks

SYN cookies help servers defend against SYN flooding. When a server receives a SYN message, it sends a SYN-ACK packet with a hashed sequence number (the "cookie") to the client. The server then forgets the cookie and doesn't allocate resources to the SYN request. A legitimate client will return an ACK segment with an acknowledgment value that is one plus the hashed sequence number. Once the server receives and verifies this ACK segment, it can then safely open a connection with the client.

Ping flooding attack

On the network layer, a **ping flooding** attack happens when an attacker floods the victim's network with ICMP (Internet Control Message Protocol) echo, or ping, requests.

Defending against ping flooding attacks

To defend against a ping flooding attack, you can configure your firewall to block pings from outside your network. Note that this does not stop internal attacks and may leave you unable to diagnose issues in your network. Alternatively, you can disable ICMP functionality on your router. This can be quite troublesome to perform and will also prevent you from running certain network diagnostic functions.

DDoS attacks

So far, we've assumed the attack comes from a single attacker. Things get trickier in a Distributed Denial of Service attack, where multiple attackers (often a botnet) overwhelm the victim with enormous amounts of traffic. Since a botnet usually includes thousands of hosts sending traffic at the same time, it is very difficult to identify and block each host in the botnet.

While there is no bulletproof way to defend against botnets, here are a few possible approaches:

- Firewalls and packet filters can be used to block suspicious traffic patterns.
- Scale up the number of available servers. This tends to be very expensive.
- The cache in intermediate servers can be used to satisfy legitimate queries and lessen the impact of the DDoS attack felt by end users.

- There are services that can divert traffic from a target's servers and "scrub" the traffic, so only legitimate traffic reaches its intended destination.
- You can try to identify the bot controller and block its IP address. However, it's not difficult for a blocked bot controller to move to a new IP if needed.

In general, it's hard to stop a DDoS attack at the source. To avoid falling victim to one, server or site owners should make the attacker commit more resources, to the point that it's not worth the effort to launch an attack.

Malicious code attacks

Before the Internet, developers only had themselves to blame for bad code. Nowadays, network administrators and site owners have to deal with malicious code from end users, who can wreak havoc with a few lines of input.

Cross-site scripting attack

Cross-site scripting (XSS) is one of the most common attacks on the Web. An XSS attack happens when an attacker submits malicious code to a site, which is executed during another user's session. For example, a blog visitor can post HTML and JavaScript code in the comments section that, when executed, lets them access the website's cookies and steal user credentials.

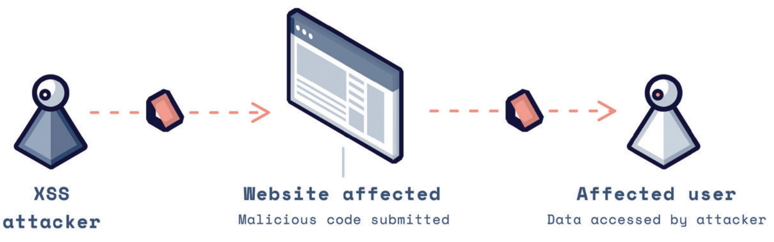
Defending against cross-site scripting

One of the simplest ways to prevent cross-site scripting attacks is to **validate** user input. Do not allow unnecessary characters or input that may be used to form malicious code. Even after input is submitted, employ double-check validation on the server before the information is stored in a database.

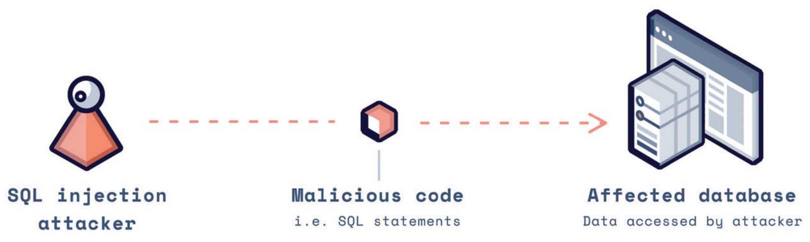
Another important step is to **sanitize** user input before it leaves the database. In practice, this means using a function like "htmlentities" to remove HTML markup, converting the code into a harmless string. The process of converting user input into a string of safe characters is known as "escaping" user input.

Finally, it's a good idea to limit where user-generated content goes. This means only inserting content from trusted users onto approved parts of the site's template files. Content from untrusted sources should be displayed only as text.

Cross-site scripting (XSS) attack



SQL injection attack



SQL injection attack

Like a cross-site scripting attack, an **SQL injection** attack takes advantage of poor input filtering to introduce malicious code into a site's servers. But where an XSS attack primarily uses HTML and JavaScript to attack other web users, an SQL injection attack uses SQL (the standard database query language) to steal information from a database or destroy it.

Defending against SQL injection

Due to the similarities between SQL injection and XSS attacks, the same principles of **validating** and **sanitizing** user input apply when defending against SQL injections. The good news is most SQL frameworks already include these functions.

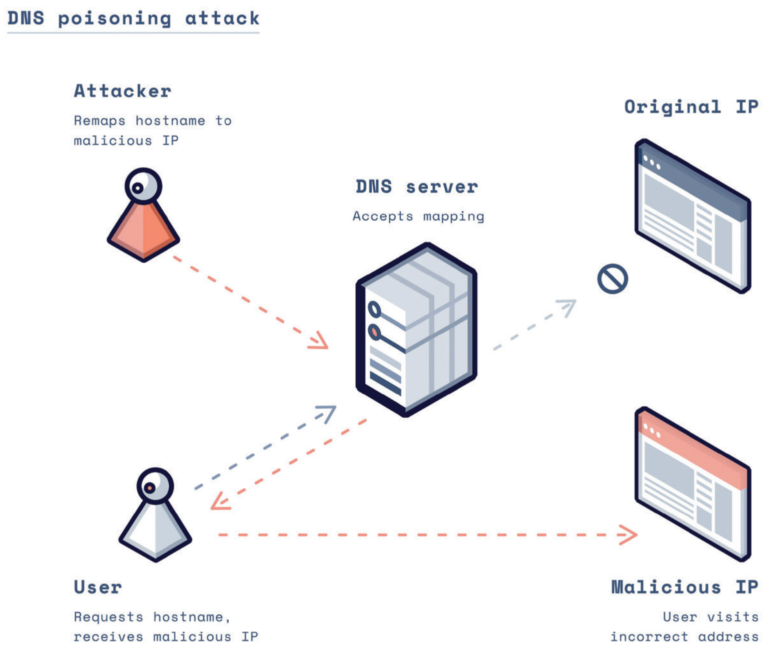
Another approach, unique to dealing with SQL injections, is to use **parameterized queries**. A parameterized query is a query (i.e., a data access request) in which placeholders are used for parameters. The placeholders are only replaced by actual values at execution. This eliminates attempts to sneak in malicious commands through user input.

Spoofing attacks

Spoofing attacks happen when an attacker pretends to be a normal network user or device and uses false information to attack legitimate entities.

DNS poisoning attack

In a **DNS poisoning** attack, an attacker asks the DNS resolver to resolve a hostname that isn't stored on the server. The attacker then sends spoofed replies that map the hostname to a malicious IP address. If the server accepts the mapping, any legitimate users requesting that hostname will be fed the attacker's malicious IP address.


DNS poisoning attack
Attacker
Remaps hostname to malicious IP
DNS server
Accepts mapping
Original IP
User
Requests hostname, receives malicious IP
Malicious IP
User visits incorrect address

Defending against DNS poisoning

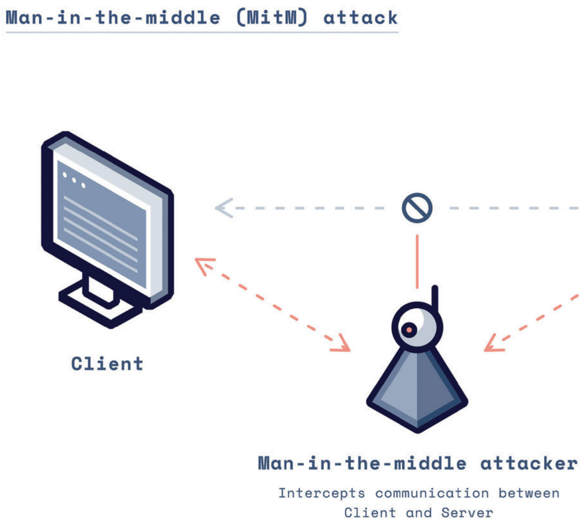
Fortunately, DNS poisoning attacks tend to be hard to execute. They depend on factors such as the attacker successfully guessing port numbers and often require a DoS attack to the DNS server to prevent legitimate replies from arriving before the attacker's spoofed replies.

If you are running your own DNS server, one easy approach is to simply block Internet DNS queries (over port 53) or to make your resolver private and available only to hosts in your network.

Man-in-the-middle attack

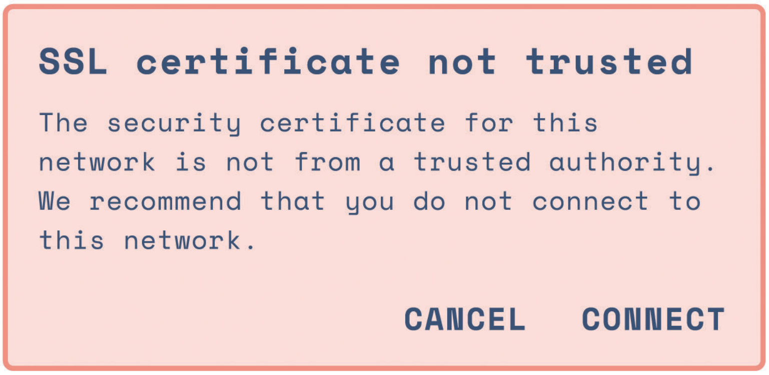
A **man-in-the-middle (MitM)** attack happens when a client is tricked into opening a transport-layer session with an attacker, who then opens a session with the server.

This "man in the middle" sits between the client and the server, who each believe they're directly communicating with the other. The MitM attacker can then intercept or modify communications, including any confidential information, between the two endpoints.


Man-in-the-middle (MitM) attack
Client
Server
Man-in-the-middle attacker
Intercepts communication between Client and Server

Defending against MitM attacks

To protect yourself against MitM attacks, avoid connecting to the Internet via public Wi-Fi hotspots. This is especially true if the network you're trying to use does not have a trusted SSL certificate. You'll usually be notified of suspicious certificates by a dialog box like this:



SSL certificate not trusted

The security certificate for this network is not from a trusted authority. We recommend that you do not connect to this network.

In addition, avoid sites that run only on HTTP (instead of the secured variant, HTTPS) and make sure you enter URLs correctly to avoid being taken to a fake site. Finally, you can encrypt your device traffic by using a trusted and secure consumer VPN service, which we'll go into later. This is a good idea if you really need to connect to public Wi-Fi or visit a non-HTTPS site.

IP spoofing attack

IP spoofing attacks happen when an attacker sends IP packets with a spoofed source address. By hiding the true source of the packets, an attacker can trick the victim into accepting these spoofed packets.

IP spoofing is often used in DoS attacks. The attacker can directly overwhelm the victim with spoofed packets, or use the victim's IP address as the source address. In this second scenario, the attacker will send spoofed packets (with the victim's IP address as the source) to all manner of recipients, leading them to flood the victim with response packets.

Defending against IP spoofing

It's hard to defend against IP spoofing in the IPv4 space, as there is no reliable protection mechanism for the source and destination fields in the packet headers. IPv6 offers encryption capabilities (using IPsec) that would, in theory, eliminate IP spoofing. That said, IPv6 adoption has remained slow.

Until things pick up on that front, other measures include using a packet filter to block suspicious traffic and setting up an access control list to regulate who can interact with your network. Of course, these mechanisms won't stop attackers from faking the source address of packets, but they allow you to react and protect yourself if you're receiving a lot of strange traffic.

ARP spoofing attack

ARP maps IP addresses to MAC addresses. In an **ARP spoofing** attack, an attacker sends fake ARP messages that link a victim's IP address with a MAC address controlled by the attacker. Once these messages are accepted, data intended for the victim is sent to the attacker instead.

Defending against ARP spoofing

You can check if you're suffering from ARP spoofing by reviewing your IP-MAC address mapping. If you see a MAC you don't recognize or if an IP is "flip-flopping" between two MAC addresses, you might be dealing with an ARP spoofing attack.

A modern switch can be configured to prevent (or at least detect) ARP spoofing, but this feature is not commonly available on home switches. There are two other things you can do to defend yourself against ARP spoofing. Each comes with its own caveats.

First, you can use a VPN to encrypt your device traffic. It's important to note a VPN doesn't actually stop ARP spoofing—it simply encrypts packets so they can't be read by an attacker who captures packets from your network.

To "properly" defend against ARP spoofing, you can create static ARP entries for every machine on your network. This leads them to ignore ARP replies, including bogus ones. Unfortunately, static ARP entries only work against simpler forms of ARP spoofing and tend to be a pain to set up.

Social engineering

Easily the least technical of cyberattacks, **social engineering** simply refers to scams executed through human interaction. **Phishing** is a form of social engineering where the perpetrator tricks the victim into giving up confidential information. Scam emails and fake websites are often used by fraudsters to phish information out of unsuspecting victims.

While you can configure your email service to block spam emails, the best way to defend against social engineering attempts is to use common sense. It's impossible to win a contest you didn't enter, and if the heir to some multibillion-dollar estate needed someone to hold onto their money, you're likely not the first person they'd think of.

Also remember that no legitimate bank or service provider will ask you to submit your account credentials over an email. If in doubt, call the party in question (make sure you call their actual number!) or use a trusted search engine to visit their official website. If you're typing a URL, make sure you don't misspell it to avoid winding up on a counterfeit site.

Other security measures

In the previous section, we looked at some of the most prevalent forms of cyberattacks. You may have noticed a few defensive mechanisms repeatedly mentioned as countermeasures. Here, we consider how some of these tools work.

While it's impossible to find a silver-bullet solution that can stop every possible form of cyberattack, a general defensive principle is to create a setup that requires the attacker to spend more resources than what they would gain from executing the attack.

TLS (and SSL)

By default, TCP and UDP don't offer any encryption. This means any data sent over the Internet can be intercepted and modified. An obvious solution is to introduce encryption on top of the transport layer.

Transport Layer Security (TLS) (along with its predecessor, **Secure Sockets Layer (SSL)**) is a security protocol that's implemented in the application layer, on top of TCP.

In addition to a handshake protocol, TLS features a cipher suite to ensure messages are protected with approved security parameters. The cipher suite also has an alert protocol that notifies the receiver when a connection will be closed or if an error has occurred.

When used with TLS, unsecure HTTP web traffic becomes encrypted HTTPS traffic. TLS can be used by any application that runs over TCP, but each application has to "ask" for security and may have to undergo code changes to enjoy TLS encryption.

IPsec

As its name suggests, **IPsec** offers security at the network layer. Unlike TLS, IPsec automatically encrypts all data from the network layer and above. Individual applications do not have to "ask" for security.

There are two flavors of IPsec protocols: The **Authentication Header (HA)** and **Encapsulation Security Payload (ESP)** protocols. Both provide source authenticity and data integrity, but ESP also looks after confidentiality.

Firewall

A **firewall** is a combination of hardware and software that controls what traffic can enter or leave your network. There are two main types of firewalls:

Packet filters examine the content of data packets before deciding whether they can be permitted to pass. Typically, a packet filter will examine fields like the source and destination IP address, protocol type (e.g., TCP, UDP, ICMP, etc.), and source and destination port. **Stateful packet filters** go beyond examining individual packets and can track each stage of a TCP handshake. This makes them a more nuanced solution than traditional packet filters.

Application gateways are the other major type of firewall. Operating at the application layer, an application gateway looks beyond IP/TCP/UDP headers and examines data for a specific application. This is known as deep packet inspection and offers more “tailored” security coverage than a packet filter. Each gateway is application specific, so you will need one for each application you want to examine.

IDS and IPS

An **intrusion detection system (IDS)** performs IP/TCP/UDP header inspection (like a packet filter) as well as deep packet inspection (like an application gateway). Alerts are raised if there are anomalies in the observed traffic (compared with a “normal” traffic profile) or if certain known attack signatures are detected.

An **intrusion prevention system (IPS)** performs the same function as an IDS but also offers the ability to filter out and block suspicious traffic.

VPN

A **virtual private network (VPN)** is a private network that's configured over a wider network (like the Internet). It is a "virtual" private network because the private network actually uses a public network as its backbone. That said, VPNs are private because they use encryption to create a secure "tunnel" between you and the network you want to access. When you use a properly configured VPN, no one can see the data you send over the tunnel.

Depending on the VPN protocol used, the VPN itself operates on different levels of the protocol stack. PPTP and L2TP (which offer almost no protection) work on the data link layer, and IPsec on the network layer; OpenVPN can be used on both layers but typically operates on the network layer.

Corporate VPNs

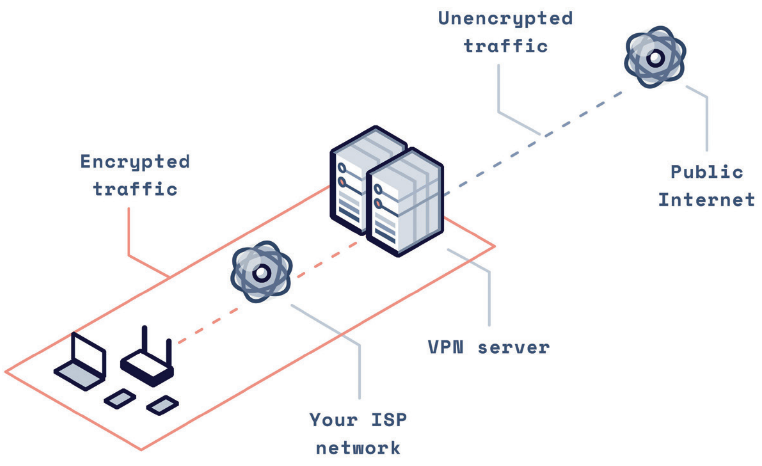
Historically, the term "VPN" refers to corporate VPNs, which connect an employee to a company's network (or a branch office to headquarters). With a corporate VPN, an employee is privately and securely connected to the company's network and can access company resources as if they were using an office machine.

Consumer VPNs

Consumer VPNs have become popular over the last decade or more and can be used by anyone looking for online security and privacy. With a consumer VPN, the consumer first connects to a VPN server (via their ISP). The traffic between the client and the VPN server is encrypted, so the ISP can't inspect the traffic that's sent to the VPN server.

When the traffic leaves the VPN server and travels across the Internet, it is decrypted and can be seen by others. This is done because the destination for your traffic cannot decrypt the VPN server's encryption, so measures like HTTPS are still necessary even if you're using a VPN. To the receiver, your (decrypted) traffic appears to have originated from the VPN server, so your real IP address is hidden.

Connecting to a consumer VPN



Does a consumer VPN make you completely "invisible" on the Internet? Hardly. Your traffic is hidden from your ISP and potential attackers, but service providers like YouTube can still identify you if you sign in to your account. YouTube would not know where you're signing in from, but if you watch a video, it would be able to track your activity via its own analytics. Using a VPN also doesn't protect you from malware. The best defense against that is using a combination of firewalls, antivirus software, and your own common sense.