

Web Application and Network Security

Common web vulnerabilities: SQL injection, XSS, CSRF; Web application penetration testing methodology; Network sniffing and spoofing; Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)

What Is Web Application Penetration Testing?

Web application vulnerabilities involve a system flaw or weakness in a web-based application. They have been around for years, largely due to not validating or sanitizing form inputs, misconfigured web servers, and application design flaws, and they can be exploited to compromise the application's security.

These vulnerabilities are not the same as other common types of vulnerabilities, such as network or asset. They arise because web applications need to interact with multiple users across multiple networks, and that level of accessibility is easily taken advantage of by hackers.

There are web application security solutions designed specifically for applications, and as such it's important to look beyond traditional vulnerability scanners and support a broader vulnerability management strategy to identify and remediate application security gaps. To really understand your risks, learn more about common types cybersecurity attacks, and how web scanners can help increase the safety of your applications.

Common types of web application vulnerabilities

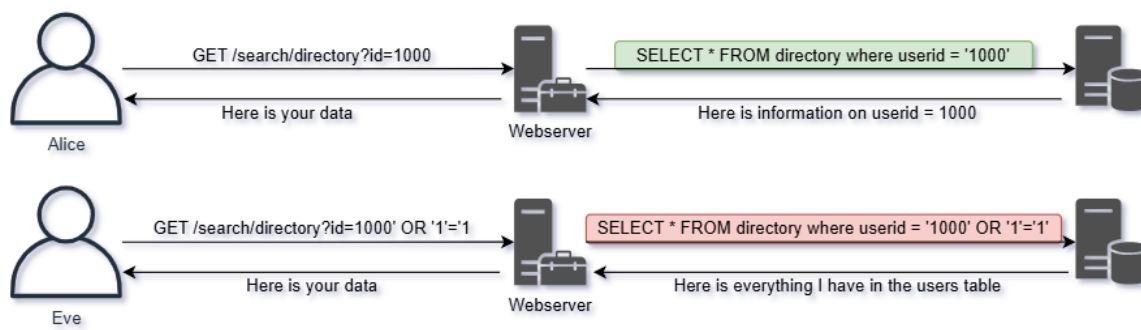
SQL injection attacks

Structured Query Language (SQL) is now so commonly used to manage and direct information on applications that hackers have come up with ways to slip their own SQL commands into the database.

These commands may change, steal or delete data, and they may also allow the hacker access to the root system. SQL (officially pronounced *ess-cue-el*, but commonly pronounced "sequel") stands for structured query language; it's a programming language used to communicate with databases. Many of the servers that store critical data for websites and services use SQL to manage the data in their databases.

An SQL injection attack specifically targets this kind of server, using malicious code to get the server to divulge information it normally wouldn't. This is especially problematic if the server stores private customer information from the website or web application, such as credit card numbers, usernames and passwords (credentials), or other personally identifiable information, which are tempting and lucrative targets for an attacker.

Successful SQL injection attacks typically occur because a vulnerable application doesn't properly sanitize inputs—something penetration testing is specifically designed to detect and validate. For example, an attacker might exploit a vulnerable search box by injecting SQL code that instructs the server to dump all stored usernames and passwords.



Types of SQL injection attacks

SQL injection attacks can be carried out in a number of ways. Attackers may observe a system's behavior before selecting a particular attack vector/method.

- **Unsanitized input**

Unsanitized input is a common type of SQLi attack in which the attacker provides user input that isn't properly sanitized for characters that should be escaped, and/or the input isn't validated to be the type that is correct/expected.

For example, a website used to pay bills online might request the user's account number in a web form and then send that to the database to pull up the associated account information. If the web application is building a SQL query string dynamically with the account number the user provided, it might look something like this:

*"SELECT * FROM customers WHERE account = " + userProvidedAccountNumber + " ;"*

While this works for users who are properly entering their account number, it leaves the door open for attackers. For example, if someone decided to provide an account number of `" or '1' = '1'`, that would result in a query string of:

*"SELECT * FROM customers WHERE account = " or '1' = '1' ;"*

Due to the `'1' = '1'` always evaluating to TRUE, sending this statement to the database will result in the data for *all* customers being returned instead of just a single customer.

- **Blind SQL injection**

Also referred to as Inferential SQL Injection, a Blind SQL injection attack doesn't reveal data directly from the database being targeted. Rather, the attacker closely examines indirect clues in behavior. Details within HTTP responses, blank web pages for certain user input, and how long it takes the database to respond to certain user input are all things that can be clues depending on the goal of the attacker. They could also point to another SQLi attack avenue for the attacker to try.

- **Out-of-band injection**

This attack is a bit more complex and may be used by an attacker when they cannot achieve their goal in a single, direct query-response attack. Typically, an attacker will craft SQL statements that, when presented to the database, will trigger the database system to

create a connection to an external server the attacker controls. In this fashion, the attacker can harvest data or potentially control behavior of the database.

A Second Order Injection is a type of Out-of-Band Injection attack. In this case, the attacker will provide an SQL injection that will get stored and executed by a separate behavior of the database system. When the secondary system behavior occurs (it could be something like a time-based job or something triggered by other typical admin or user use of the database) and the attacker's SQL injection is executed, that's when the "reach out" to a system the attacker controls happens.

SQL injection example

For this SQL injection example, let's use two database tables, Users and Contacts. The Users table may be as simple as having just three fields: ID, username, and password. The Contacts table has more information about the users, such as UserID, FirstName, LastName, Address1, Email, credit card number, and security code.

The Users table has information used for logins like:

1. jsmith,P@\$\$w0rd
2. sbrown,WinterIsComing!
3. kcharles,Sup3rSecur3Password\$

Note: Passwords should always be hashed and salted when stored in a database and never in cleartext.

When someone wants to log in, they'll go to the login page and enter their username and password. This information is then sent to the webserver, which will construct a SQL query and send that query to the database server. An example of what that query looks like might be:

Select ID from Users where username='jsmith' and password='P@\$\$w0rd'

The way SQL works is that it will then perform a true or false comparison for each row that the query requests. In our example, the query says to check the Users table and give back the ID value for every row where the username is jsmith and the password is P@\$\$w0rd. Often, the webserver will then see what is returned by the database server and if it is a number. In our case, the webserver would receive back a 1 and let the user past the login page.

But, what if we want to get malicious with this? Because the database server performs that true-or-false check, we can trick it into believing that we have successfully authenticated. We can do this by adding an OR to the password. If we log in with x' or 1=1 as our password, that will create a new SQL query that looks like:

Select ID from Users where username='jsmith' and password='x' or 1=1

This will work for us, because while x is not jsmith's password, the database server will then check the second condition. If x isn't jsmith's password, then does 1 equal 1? It does! The ID will be sent back to the application and the user will be successfully authenticated.

This doesn't have to be a 1=1 condition. Any two equal values will work, 2=2, 4726=4726 or even a=a.

If a web page can display data, it may also be possible to print additional data to the screen. To access the data, we can try to chain together two SQL requests. In addition to our ' or 1=1, we can add on to that a second statement like UNION SELECT LastName, credit card number, security code from Contacts. Extra clauses like this may take some extra work, but getting access to data is the ultimate goal of a SQL injection attack.

Another technique we can use for blind SQL injection, the one where no data is sent back to the screen is to inject other hints. Similar to our ' or 1=1 condition, we can tell the server to sleep. We could add: " ' or sleep(10) " and this will do what it seems like. It will tell the database server to take a 10-second nap and all responses will be delayed.

How to prevent SQL injection attacks

The following suggestions can help prevent an SQL injection attack from succeeding:

Don't use dynamic SQL

- Avoid placing user-provided input directly into SQL statements.
- Prefer prepared statements and parameterized queries, which are much safer.
- Stored procedures are also usually safer than dynamic SQL.

Sanitize user-provided inputs

- Properly escape those characters which should be escaped.
- Verify that the type of data submitted matches the type expected.

Don't leave sensitive data in plaintext

- Encrypt private/confidential data being stored in the database.
- Salt the encrypted hashes.
- This also provides another level of protection just in case an attacker successfully exfiltrates sensitive data.

Limit database permissions and privileges

- Set the capabilities of the database user to the bare minimum required by following the principle of least privilege access. This limits what an attacker can do if they gain access.
- This will limit what an attacker can do if they manage to gain access.

Avoid displaying database errors directly to the user. Attackers can use these error messages to gain information about the database.

Use a Web Application Firewall (WAF) for web applications that access databases

- This provides protection to web-facing applications.
- It can help identify SQL injection attempts.

- Based on the setup, it can also help prevent SQL injection attempts from reaching the application (and, therefore, the database).

Use web application security testing to routinely test web apps that interact with databases. Doing so can help catch new bugs or regressions that could allow SQL injection vulnerabilities.

Keep databases updated to the latest available patches as part of a strong vulnerability management program. This prevents attackers from exploiting known weaknesses/bugs present in older versions.

SQL injection is a popular attack method for adversaries, but by taking the proper precautions—such as encrypting sensitive data, protecting and testing your web applications, and staying up to date with patches—you can take meaningful steps toward keeping your systems secure.

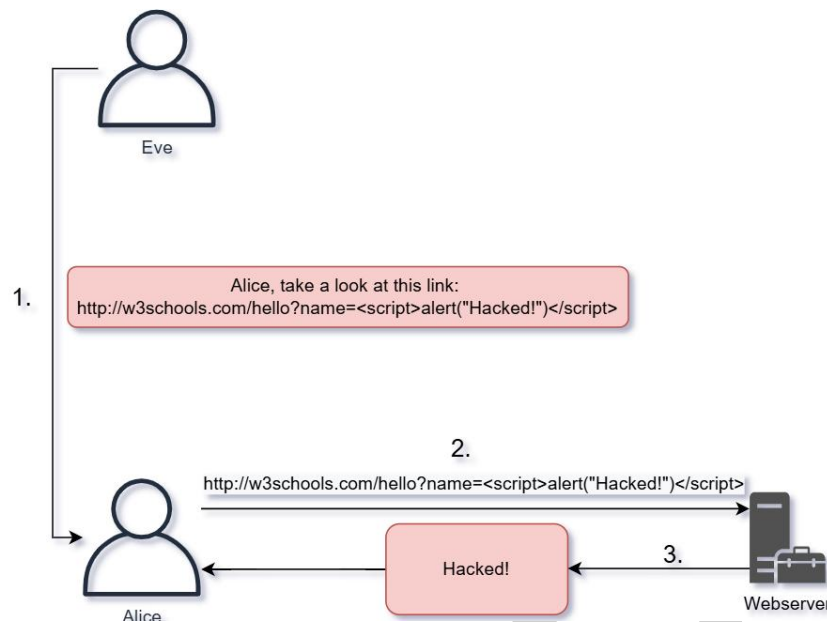
Even with strong prevention measures in place, organizations should have a clearly defined incident response plan to quickly contain and remediate SQL injection attacks if they occur.

Cross-site scripting (XSS)

In an SQL injection attack, an attacker goes after a vulnerable website to target its stored data, such as user credentials or sensitive financial data. But if the attacker would rather directly target a website's users, they may opt for a cross-site scripting attack. Similar to an SQL injection attack, this attack also involves injecting malicious code into a website or web-based app. However, in this case the malicious code the attacker has injected only runs in the user's browser when they visit the attacked website, and it goes after the visitor directly.

One of the most common ways an attacker can deploy a cross-site scripting attack is by injecting malicious code into an input field that would be automatically run when other visitors view the infected page. For example, they could embed a link to a malicious JavaScript in a comment on a blog.

Cross-site scripting attacks can significantly damage a web company's reputation by placing the users' information at risk without any indication that anything malicious even occurred. Any sensitive information a user sends to the site or the application—such as their credentials, credit card information, or other private data—can be hijacked via cross-site scripting without the owners realizing there was even a problem in the first place.



What are the impacts of cross-site scripting?

When a web page is compromised with cross-site scripting, a collection of issues can quickly emerge. Possible concerns include, but are not limited to:

- Sensitive user data being exposed
- Attackers seizing online accounts and impersonating users
- Vandalism of website content presentation
- Upload of malicious 'Trojan horse' programs
- Redirect of web pages to harmful locations

Cross-site scripting can be highly detrimental if it is not detected and remediated quickly. With businesses and clients both at risk of XSS attacks, reputations and professional relationships can be negatively impacted following a successful malware injection.

One high-profile example occurred during the 2018 Holiday Season with the rise of 'Magecart', a credit card-skimming malware that exploited client-side vulnerabilities by injecting malicious scripts into online checkout pages. This widespread campaign highlighted the dangers of script-based attacks and how easily user data can be exfiltrated.

Because XSS vulnerabilities can sometimes remain undiscovered for long periods, they are occasionally leveraged in a zero day attack—before developers are even aware of the flaw, let alone able to patch it.

Types of cross-site scripting attacks

Cross-site scripting attacks are typically categorized as one of the following types.

- Reflected XSS
- Persistent XSS
- Dom-Based XSS

Reflected XSS

A reflected XSS attack involves a vulnerable website accepting data (i.e. malicious script) sent by the target's own web browser to attack the target with. Because the malicious script is sent by the client itself and is not stored on the vulnerable server, this type of attack is also referred to as "non-persistent."

A simple example of a reflected XSS attack could involve an attacker crafting up a URL that passes a small, malicious script as a query parameter to a website that has a search page vulnerable to XSS:

http://vulnerable-website.com/search?search_term="<script>(bad things happen here)</script>"

The attacker then needs to have targets visit this URL from their web browsers. This could be accomplished by sending an email containing the URL (with plausible reason to trick the user into clicking it) or publishing the URL to a public, non-vulnerable website for targets to click.

When a target does click the link, the vulnerable site accepts the query parameter "search_term", expecting that the value is something the target is interested in searching the vulnerable-website.com site for, when in reality the value is the malicious script.

The search page then, as most website search pages will do when a user is searching for something, displays "Searching for <search_term>...", but because the vulnerable site didn't sanitize the search_term value, the malicious script is injected into the webpage that the target's browser is loading and is then executed by the target's browser.

Persistent XSS

As the name implies, a persistent XSS attack is stored/persisted on the vulnerable server itself. Unlike a reflected attack, where the malicious script is sent by the target, users of a vulnerable website or web app can be attacked during their usual interactions with the vulnerable site/app.

A simple example of a persistent XSS attack could involve an attacker posting a message to a forum hosted on a vulnerable website. Rather than a usual, innocuous forum post, this post content contains the attacker's malicious script. When a user visits this forum post, their web browser loads and executes the malicious script.

As you can see, a key differentiator between reflected and persistent XSS attacks is that persistent XSS attacks consider **all users** of a vulnerable site/app as targets for attack.

DOM-Based XSS

Another type of XSS attack is DOM-based, where the vulnerability exists in the client-side scripts that the site/app always provides to visitors. This attack differs from reflected and persistent XSS attacks in that the site/app doesn't directly serve up the malicious script to the target's browser. In a DOM-based XSS attack, the site/app has vulnerable client-side scripts which deliver the malicious script to the target's browser. Similar to a reflected attack, a DOM-based attack does not store the malicious script on the vulnerable server itself.

A simple example of a DOM-based XSS attack could involve the same setup for the reflected XSS example scenario above. The attacker creates a URL with a malicious script as the “search_term” and solicits it to potential targets.

Once a target clicks the URL, their browser loads the site search page and the vulnerable client-side processing scripts. While the “search_term” is still provided as a query parameter to the site back end for processing, the site itself does not generate the web page with the injected malicious script.

Instead, the site’s vulnerable client-side scripts are designed to locally (in the target’s browser) dynamically substitute in the search term value (i.e. the malicious script) in the target’s rendered search page, causing the target’s browser to load and execute the attacker’s script.

DOM-based XSS attacks highlight the fact that XSS vulnerabilities aren’t limited to server-side software.

How to prevent cross-site scripting attacks

With multiple variations of cross-site scripting attacks, organizations need to implement layered protections—from input validation to dynamic application security testing (DAST) tools—to prevent exploitation of web application vulnerabilities. As modern applications evolve through fast-paced CI/CD pipelines, introducing security early in the software development lifecycle is critical to preventing XSS vulnerabilities from reaching production. The frequency of attacks is likely to rise, making proactive prevention essential.

The following best practices can help safeguard your users against XSS attacks:

Sanitize user input

- Validate all user-provided input to detect potentially malicious content.
- Encode output to prevent untrusted data from triggering automatic execution in the browser.

Limit use of user-provided data

- Only accept and use user input where absolutely necessary.

Utilize a Content Security Policy (CSP):

- CSPs provide an extra layer of protection against XSS by restricting the sources from which scripts can be executed.

Run vulnerability scans regularly:

- Use a web application vulnerability scanning tool to detect XSS and other injection flaws in your application.

Conduct manual penetration testing

- In addition to automated tools, penetration testing helps uncover complex or hidden XSS vulnerabilities by simulating real-world attack scenarios.

Provide security awareness training

- Human error and social engineering often play a role in the success of XSS attacks.
- Implementing regular security awareness training can help employees recognize suspicious links, phishing attempts, and unsafe browsing behaviors that may expose them to malicious scripts.

Cross-site request forgery (CSRF)

A Cross-Site Request Forgery (CSRF) attack is when a victim is forced to perform an unintended action on a web application they are logged into. The web application will have already deemed the victim and their browser trustworthy, and so executes an action intended by the hacker when the victim is tricked into submitting a malicious request to the application. This has been used for everything from harmless pranks on users to illicit money transfers.

One way website owners can help cut down on their chance of attack is to have advanced validation techniques in place for anyone who may visit pages on their site or app, especially when it comes to social media or community sites. This will enable them to identify the user's browser and session to verify their authenticity.

While there are a variety of ways a hacker may infiltrate an application due to web application vulnerabilities, there are also a variety of ways to defend against them—including deploying a web application firewall (WAF) to block malicious requests and monitor for known attack patterns. There are web application security testing tools specially designed to monitor even the most public of applications. Using these scanners reduce your chances of being the victim of a hack by showing you exactly where to make the changes needed for more secure applications.

How does CSRF work?

When users attempt to access a site, their browser often automatically includes any credentials associated with the site along with their request so that the login process is more convenient. These credentials can include the user's session cookie, basic authentication credentials, IP address, Windows domain credentials, and so on. Once the user is authenticated to the site, however, the site has no way to distinguish a forged request from a legitimate user request.

By co-opting the victim's identity and access via a CSRF attack, an attacker can make a user perform unintended actions. Typically, the attacker persuades a victim to click on a link by using a social-engineering technique via an email, chat message, or a similar form of communication. The user may then unknowingly encounter malicious HTML or JavaScript code in the email message or after loading a site page that requests a specific task URL. The task then executes, either directly or by using a cross-site scripting flaw. The user is often unaware that anything has happened until after a malicious action has occurred.

CSRF attacks usually target functions that cause a state change on the server but can also be used to access sensitive data. Upon performing a successful CSRF attack on a victim's account, a malicious actor can initiate a transfer of funds, purchase an item, place a product in a shopping cart, alter account information such as a shipping address, change a password, or use any other function that is available on the vulnerable website.

Stored CSRF attacks and their impact

In some cases, it is possible to store a CSRF attack directly on the vulnerable site itself. Such vulnerabilities are called stored CSRF flaws. An attacker can create a stored CSRF flaw simply by storing an IMG or IFRAME tag in a field that accepts HTML, or by conducting a more complex cross-site scripting (XSS) attack. The Samy MySpace worm is a notable case in which XSS techniques compromised a site on a mass scale.

If an attacker is able to store a CSRF attack on the target site, the impact can be far more severe. In this case, since the page containing the malicious payload is now contained within the site and therefore appears entirely legitimate, the victim is more likely to view and trust the page containing the attack than a random page on the internet. And since the victim has already been authenticated to the site in this scenario, the attacker will have an even better opportunity to target them with a CSRF attack.

How to prevent CSRF attacks

There are several methods for strengthening your web application security program so that you will be less vulnerable to a potential CSRF attack. As with other web application security measures, the best defense involves regularly scanning and testing the security of your web applications:

- **Enable CSRF protection in web applications**

If your web application does not currently have CSRF protection, it could be vulnerable to this form of attack. Web application security tools can help you quickly determine whether such a vulnerability exists within your web application and provide you with steps to remediate the issue.

- **Use CSRF tokens and advanced validation**

You can help reduce the likelihood of a CSRF attack by having advanced validation techniques in place for anyone who may visit pages on your site, especially if you are operating a social media or community site. CSRF tokens, which are sometimes also referred to as anti-CSRF tokens since they are intended to deflect CSRF attacks, are one such example. Typically comprised of a large, random string of numbers that is unique to both the individual session and the user, they make it much harder for attackers to guess the proper token required to create a valid request.

By implementing CSRF tokens in your form submissions and side-effect URLs, you can better ensure that every form submission or request is tied to an authenticated user and shielded from a potential CSRF attack. In cases involving highly sensitive operations, OWASP notes that you may also want to consider implementing a user interaction based protection (either re-authentication/one-time token along) along with token based mitigation techniques.

- **Conduct regular web application security testing**

Even after you have successfully resolved a vulnerability in a web application that would have enabled a CSRF attack, it is still possible for vulnerabilities to arise in the future as the application is updated and changes are made to its code. For this reason, it's wise to continually scan and test your web applications for any security vulnerabilities they may harbor—including those related to CSRF—using tools that perform dynamic application security testing (DAST) to detect issues during runtime. In addition to automated tools, regular penetration testing can identify CSRF attack paths that require a deeper understanding of business logic or user flows.

Although CSRF attacks only work on users that are currently authenticated to a site, these exploits can be devastating when successful. An attacker who has impersonated a user can then proceed to perform a range of actions without their knowledge or consent, stealing money or committing fraud.

A company can find its reputation severely damaged as a result, experiencing a loss of customer trust and even facing regulatory fines in some cases. By proactively implementing a comprehensive application security program, your business can reduce the possibility of such an attack.

What Are Web Application Attacks?

Web application attacks are malicious activities that target web applications by exploiting vulnerabilities in their design or implementation. These attacks can result in unauthorized access, data theft, or other harmful consequences.

Common types of web application attacks include SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and file inclusion attacks. Attackers may use automated tools or manually craft their attacks to bypass security measures and gain access to sensitive information or systems.

Organizations can prevent or mitigate web application attacks by implementing strong security measures, such as input validation, user authentication, and regular vulnerability testing.

What Are the Consequences of Web Application Attacks?

Web application attacks can have a wide range of consequences for organizations, users, and other stakeholders. Some of the potential consequences of web application attacks include:

- **Data breaches:** Attackers may gain unauthorized access to sensitive data, such as personal information, financial data, or intellectual property, leading to data breaches. This can result in severe financial, reputational, and legal consequences for the affected organization.
- **Identity theft:** Attackers may steal personal information during web application attacks, leading to identity theft. Victims of identity theft may face financial losses, credit issues, and time-consuming recovery processes.

- **Financial loss:** Web application attacks may lead to direct financial losses for businesses, either through theft of funds, fraud, or the costs associated with remediation and recovery.
- **Damage to reputation:** A successful web application attack can damage an organization's reputation, leading to loss of customer trust, negative publicity, and reduced business opportunities.
- **Legal consequences:** Organizations that fail to protect their web applications may face legal consequences, such as fines, lawsuits, or regulatory penalties, particularly if the attack results in a data breach involving personal information.
- **Business disruption:** Web application attacks can disrupt business operations by causing system downtime, impacting the availability of online services, or compromising critical infrastructure.

Common Types of Web Application Attacks

1. Cross-Site Scripting (XSS)

Cross-site scripting (XSS) is a type of web application attack that involves injecting malicious scripts into web pages that are viewed by other users. This is typically accomplished by injecting the script into a form input field or URL parameter that is then stored in the web application's database.

When another user views the page that contains the malicious script, the script is executed in their browser, allowing the attacker to steal data or perform other malicious actions on the user's behalf. XSS attacks can be prevented by properly sanitizing user input, using content security policy (CSP) headers, and escaping untrusted data.

2. Cross-Site Request Forgery (CSRF)

Cross-site request forgery (CSRF) is a type of web application attack that tricks a user into executing an unwanted action on a web application that they are already authenticated with. This is typically accomplished by sending a specially crafted link or script to the user, which then performs the unwanted action when clicked.

For example, a CSRF attack could be used to make unauthorized purchases or change account settings. CSRF attacks can be prevented by using anti-CSRF tokens, which are unique tokens that are generated by the web application for each user session and must be included in every request to the application.

3. XML External Entity (XXE)

XML External Entity (XXE) is a type of web application attack that involves exploiting vulnerabilities in XML parsers used by a web application. This can allow an attacker to read sensitive data or execute unauthorized actions on the web application's server.

XXE attacks typically involve injecting specially crafted XML payloads that exploit the XML parser's ability to read external entities. XXE attacks can be prevented by disabling external entity parsing or using secure XML parsers that properly sanitize input data.

4. Injection Attacks

Injection attacks involve inserting malicious code into a web application, typically in the form of input data such as SQL queries, commands, or scripts. Injection attacks are successful when an application fails to properly validate and sanitize input data. These attacks can be prevented by properly validating and sanitizing input data and using parameterized queries to access databases.

5. Fuzz Testing (Fuzzing)

Fuzz testing, also known as fuzzing, is a technique used to discover vulnerabilities in a web application by sending it random or invalid input data. The goal of fuzz testing is to identify how the web application responds to different inputs and to find errors and crashes.

Fuzz testing can be performed manually or with the help of automated tools. Fuzz testing can uncover vulnerabilities that may not be detected by other security testing methods such as penetration testing. To perform effective fuzz testing, a tester needs to understand the web application's input and output mechanisms and the types of data that the application processes.

6. DDoS (Distributed Denial-of-Service)

A Distributed Denial-of-Service (DDoS) attack is a type of web application attack that involves overwhelming a web application with a large volume of traffic from multiple sources, such as botnets or compromised devices. This can cause the web application to become unavailable to legitimate users.

DDoS attacks can be prevented by using network security devices, such as firewalls and intrusion prevention systems, that can detect and block malicious traffic. Additionally, web application developers can use content delivery networks (CDNs) and load balancers to distribute traffic across multiple servers to help mitigate the effects of DDoS attacks.

7. Brute Force Attack

A brute force attack is an automated method of guessing a username and password combination to gain unauthorized access to a web application. Attackers use software tools to try different combinations of usernames and passwords until they successfully guess the correct one.

To prevent brute force attacks, web applications can implement rate-limiting and account lockout policies. Rate-limiting limits the number of login attempts from a single IP address, while account lockout temporarily blocks access to an account after a certain number of failed login attempts.

8. Path Traversal

Path traversal is a type of web application attack that involves manipulating file paths in a web application in order to access unauthorized files or directories on the server. Path traversal attacks typically occur when a web application does not properly validate user input, allowing an attacker to traverse up and down directory structures to access sensitive files.

Path traversal attacks can be prevented by properly validating user input and sanitizing file paths, as well as using secure file access methods that restrict access to sensitive files and directories.

Web Application Security Strategies

Here are some web application security strategies that organizations can implement to protect their web applications:

- **Secure coding practices:** Adopt secure coding practices, such as the OWASP Top 10 guidelines, to ensure that web applications are built with security in mind. This includes measures like input validation, output encoding, and secure authentication mechanisms.
- **Regular security testing:** Perform regular security testing, such as penetration testing and vulnerability scanning, to identify and address security vulnerabilities in web applications.
- **Access control:** Implement access controls to ensure that only authorized users can access sensitive data or functionality within web applications. This includes measures like role-based access control and multi-factor authentication.
- **Secure communication:** Use secure communication protocols, such as HTTPS, to ensure that data transmitted between web applications and users is encrypted and protected from interception.
- **Server and network security:** Implement server and network security measures, such as firewalls and intrusion detection systems, to protect web applications from attacks like DDoS and SQL injection.
- **Regular updates and patches:** Keep web applications and supporting software up-to-date with the latest security patches and updates to address known vulnerabilities.
- **User education:** Educate users on best practices for safe web browsing, such as avoiding clicking on suspicious links or downloading attachments from unknown sources.
- **Incident response planning:** Develop and test incident response plans to ensure that web application security incidents are identified and addressed in a timely and effective manner.

What is Sniffing?

Sniffing is the technique of continuously monitoring and recording all data packets that transit via a network. Network or system administrators employ sniffers to monitor and troubleshoot network traffic. Hackers use sniffers to capture data packets containing sensitive data such as passwords and account information. Attackers install sniffers as hardware or software in the system.

Types of Sniffing Attacks

Active Sniffing

Active Sniffing is sniffing in the switch. It is a network device that connects two points. This switch monitors the MAC addresses on each port, which ensures that data is passed only to the appropriate destination. To sniff the traffic between targets, sniffers must actively inject traffic into the LAN. There are several ways to accomplish this.

Passive Sniffing

The process of Sniffing through the hub is called passive Sniffing. All machines on an unbridged or non-switched network segment will be able to see any traffic passing through it. They operate on the data link layer of the network. A hacker transmits a network packet across the LAN, where it reaches every machine connected to it. Attackers can passively capture data by sending sniffers.

Active Sniffing involves infesting the switch content address memory (CAM) table with address resolution protocols (ARPs). Consequently, the attacker sniffs data from the switch by redirecting legitimate traffic to other ports. There are several active Sniffing techniques, including Spoofing, DHCP, and DNS poisoning

What is Spoofing?

Using a spoof to represent a communication coming from a known and trusted source is Spoofing. It can be as simple as email Spoofing, phone Spoofing, website Spoofing, or more technical such as a computer Spoofing an IP address, ARP, or DNS server.

The purpose of a Spoofing attack is to gain access to sensitive data or information by posing as a trustworthy source. Spamming can be done through websites, emails, phone calls, texts, IP addresses, and servers.

Different Types of Spoofing Attacks

The different types of Spoofing attacks are listed here –

- Caller ID Spoofing

Spoofing takes place when the caller ID is changed by using false information. To hide their identity, phone scammers use Caller ID Spoofing to make it impossible to block a number. In some cases, scammers will use your area code to disguise the call as being local.

Scammers often use Voice over Internet Protocol (VoIP) to spoof caller ID by creating fake phone numbers and names. Scammers will attempt to get vital information from the call recipient, once they answer the phone.

- Email Spoofing

Scammers use fake sender addresses to harm your computer, steal your information, or infect your computer with malware through email Spoofing. Such emails look like they came from a friend or co-worker. This is so that you can be fooled into thinking that the emails are legit.

Using alternative numbers or letters to look slightly different from the original will get you this result, or disguising the “from” field to become an address that belongs to someone in your contact list.

- Website Spoofing

Scammers use legitimate fonts, colors, and logos to make a dangerous website appear to be a secure one. Scammers replicate a trusted website so that users visit a phishing or malicious site. Most of these copied sites look authentic at first glance due to the similar website addresses. Nevertheless, their primary purpose is to gather visitor information.

- DNS Server Spoofing

DNS Spoofing, also known as cache poisoning, is the process of rerouting traffic to a different IP address. Malicious websites will be redirected to this page. Scammers do this by replacing the DNS server’s IP addresses with their own.

- GPS Spoofing

A GPS Spoofing attack occurs when fake signals resemble real signals and are broadcast to fool GPS receivers. Essentially, scammers pretend to be in one place, while, in reality, being in another place.

Scammers use this type of attack to interfere with GPS signals of ships, buildings, or aircrafts such as to drive them to wrong addresses. Apps that rely on the location data from a smartphone are potentially vulnerable to this type of attack.

- ARP Spoofing

ARP Spoofing is to manipulate and steal data as well as hijack sessions. As a result, spammers will connect their media access control to the IP address to access the data belonging to the owner of that address.

- Man-in-the-Middle (MitM) attack

MitM attacks occur when scammers hack a Wi-Fi network or create an identical counterfeit network to intercept web traffic between two parties. This allows scammers to reroute credit card numbers or login information to themselves.

- Text Message Spoofing

The practice of Spoofing texts occurs when scammers use another person’s phone number to send a text message. Scammers use alpha-numeric sender IDs to mask their identity, and they normally link to phishing or malware downloads. Make sure that you are familiar with mobile security tips, if you fear your data is being compromised.

- Extension Spoofing

Extension Spoofing is used by scammers to mask malware extension folders. These hacker files are often renamed as filename.txt.exe, and have malware hidden inside. The malicious program that runs when a file appears to be a text document is a text document.

Protection Against Sniffing and Spoofing

The development of technology brings more and more new cyber threats, so staying informed about the protection measures is imperative to be able to combat and defend against illegal hackers. We have listed a few points that you can follow to keep your devices safe from hackers.

Protection against sniffing

- Set up a strong antivirus on your device
- Secure your data with a VPN
- Avoid visiting unencrypted websites
- Avoid using public Wi-Fi
- Do not use unencrypted messaging apps

Protection against spoofing

- Implement packet filtering with deep packet inspection
- Verify the authenticity of users and systems
- Use Spoofing detection software
- Implement encrypted and authenticated protocols

Intrusion Detection System

An intrusion detection system (IDS) is an application that monitors network traffic and searches for known threats and suspicious or malicious activity. The IDS sends alerts to IT and security teams when it detects any security risks and threats.

Most IDS solutions simply monitor and report suspicious activity and traffic when they detect an anomaly. However, some can go a step further by taking action when it detects anomalous activity, such as blocking malicious or suspicious traffic.

IDS tools typically are software applications that run on organizations' hardware or as a network security solution. There are also cloud-based IDS solutions that protect organizations' data, resources, and systems in their cloud deployments and environments.

What Is An Intrusion In Cybersecurity?

The answer to "what is intrusion" is typically an attacker gaining unauthorized access to a device, network, or system. Cyber criminals use increasingly sophisticated techniques and tactics to infiltrate organizations without being discovered. This includes common techniques like:

1. **Address spoofing:** The source of an attack is hidden using spoofed, misconfigured, and poorly secured proxy servers, which makes it difficult for organizations to discover attackers.

2. **Fragmentation:** Fragmented packets enable attackers to bypass organizations' detection systems.
3. **Pattern evasion:** Hackers adjust their attack architectures to avoid the patterns that IDS solutions use to spot a threat.
4. **Coordinated attack:** A network scan threat allocates numerous hosts or ports to different attackers, making it difficult for the IDS to work out what is happening.

Types Of Intrusion Detection Systems (IDS)

IDS solutions come in a range of different types and varying capabilities. Common types of intrusion detection systems (IDS) include:

1. **Network intrusion detection system (NIDS):** A NIDS solution is deployed at strategic points within an organization's network to monitor incoming and outgoing traffic. This IDS approach monitors and detects malicious and suspicious traffic coming to and going from all devices connected to the network.
2. **Host intrusion detection system (HIDS):** A HIDS system is installed on individual devices that are connected to the internet and an organization's internal network. This solution can detect packets that come from inside the business and additional malicious traffic that a NIDS solution cannot. It can also discover malicious threats coming from the host, such as a host being infected with malware attempting to spread it across the organization's system.
3. **Signature-based intrusion detection system (SIDS):** A SIDS solution monitors all packets on an organization's network and compares them with attack signatures on a database of known threats.
4. **Anomaly-based intrusion detection system (AIDS):** This solution monitors traffic on a network and compares it with a predefined baseline that is considered "normal." It detects anomalous activity and behavior across the network, including bandwidth, devices, ports, and protocols. An AIDS solution uses machine-learning techniques to build a baseline of normal behavior and establish a corresponding security policy. This ensures businesses can discover new, evolving threats that solutions like SIDS cannot.
5. **Perimeter intrusion detection system (PIDS):** A PIDS solution is placed on a network to detect intrusion attempts taking place on the perimeter of organizations' critical infrastructures.
6. **Virtual machine-based intrusion detection system (VMIDS):** A VMIDS solution detects intrusions by monitoring virtual machines. It enables organizations to monitor traffic across all the devices and systems that their devices are connected to.
7. **Stack-based intrusion detection system (SBIDS):** SBIDS is integrated into an organization's **Transmission Control Protocol/Internet Protocol (TCP/IP)**, which is used as a communications protocol on private networks. This approach enables the IDS to watch packets as they move through the organization's network

and pulls malicious packets before applications or the operating system can process them.

How Does An Intrusion Detection System Work? What Are Its Uses?

IDS solutions excel in monitoring network traffic and detecting anomalous activity. They are placed at strategic locations across a network or on devices themselves to analyze network traffic and recognize signs of a potential attack.

An IDS works by looking for the signature of known attack types or detecting activity that deviates from a prescribed normal. It then alerts or reports these anomalies and potentially malicious actions to administrators so they can be examined at the application and protocol layers.

This enables organizations to detect the potential signs of an attack beginning or being carried out by an attacker. IDS solutions do this through several capabilities, including:

1. **Monitoring the performance** of key [firewalls](#), files, routers, and servers to detect, prevent, and recover from cyberattacks
2. **Enabling system administrators** to organize and understand their relevant operating system audit trails and logs that are often difficult to manage and track
3. **Providing an easy-to-use interface** that allows staff who are not security experts to help with the management of an organization's systems
4. **Providing an extensive database** of attack signatures that can be used to match and detect known threats
5. **Providing a quick and effective reporting system** when anomalous or malicious activity occurs, which enables the threat to be passed up the stack
6. **Generating alarms** that notify the necessary individuals, such as system administrators and security teams, when a breach occurs
7. **In some cases, reacting to potentially malicious actors** by blocking them and their access to the server or network to prevent them from carrying out any further action

The increasingly connected nature of business environments and infrastructures means they demand highly secure systems and techniques to establish trusted lines of communication. IDS has an important role within [modern cybersecurity strategies](#) to safeguard organizations from hackers attempting to gain unauthorized access to networks and stealing corporate data.

Why Are Intrusion Detection Systems (IDS) Important?

An intrusion detection system provides an extra layer of protection, making it a critical element of an effective cybersecurity strategy. You can use it alongside your other cybersecurity tools to catch threats that are able to penetrate your primary defenses. So even if your main system fails, you are still alerted to the presence of a threat.

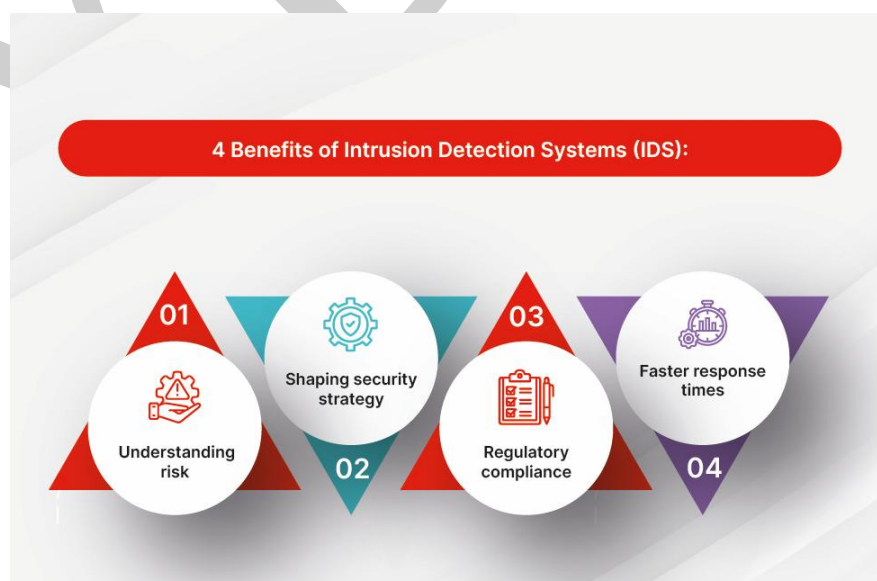
A healthcare organization, for example, can deploy an IDS to signal to the IT team that a range of threats has infiltrated its network, including those that have managed to bypass

its firewalls. In this way, the IDS helps the organization to stay in compliance with data security regulations.

Benefits of intrusion detection systems

IDS solutions offer major benefits to organizations, primarily around identifying potential security threats being posed to their networks and users. A few common benefits of deploying an IDS include:

1. **Understanding risk:** An IDS tool helps businesses understand the number of attacks being targeted at them and the type and level of sophistication of risks they face.
2. **Shaping security strategy:** Understanding risk is crucial to establishing and evolving a comprehensive cybersecurity strategy that can stand up to the modern threat landscape. An IDS can also be used to identify bugs and potential flaws in organizations' devices and networks, then assess and adapt their defenses to address the risks they may face in the future.
3. **Regulatory compliance:** Organizations now face an ever-evolving list of increasingly stringent regulations that they must comply with. An IDS tool provides them with visibility on what is happening across their networks, which eases the process of meeting these regulations. The information it gathers and saves in its logs is also vital for businesses to document that they are meeting their compliance requirements.
4. **Faster response times:** The immediate alerts that IDS solutions initiate allow organizations to discover and prevent attackers more quickly than they would through manual monitoring of their networks. The sensors that an IDS uses can also inspect data in network packets and operating systems, which is also faster than manually collecting this information.



Intrusion detection system (IDS) challenges

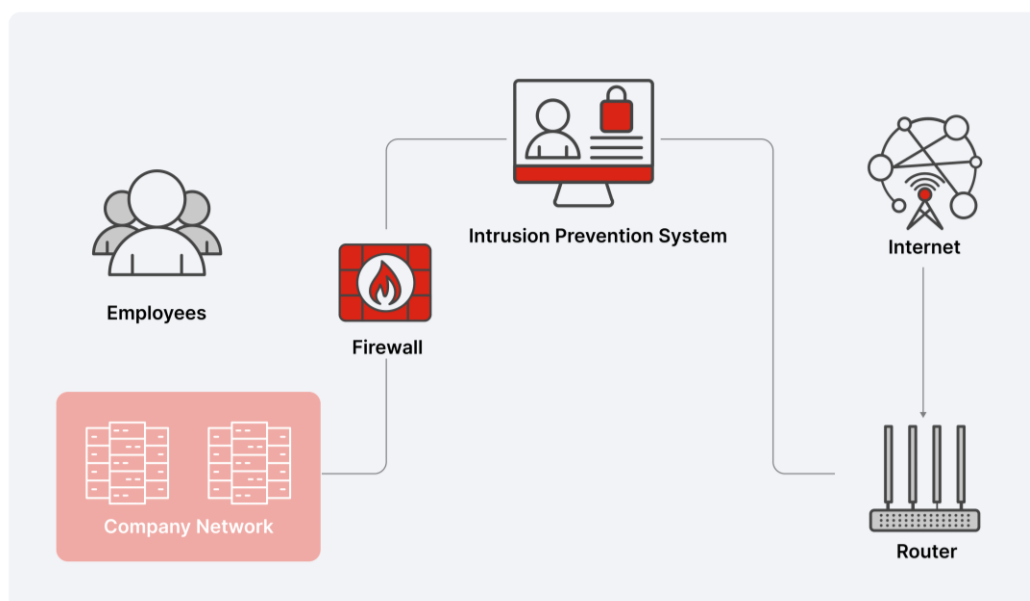
While IDS solutions are important tools in monitoring and detecting potential threats, they are not without their challenges. These include:

1. **False alarms:** Also known as false positives, these leave IDS solutions vulnerable to identifying potential threats that are not a true risk to the organization. To avoid this, organizations must configure their IDS to understand what normal looks like, and as a result, what should be considered as malicious activity.
2. **False negatives:** This is a bigger concern, as the IDS solution mistakes an actual security threat for legitimate traffic. An attacker is allowed to pass into the organization's network, with IT and security teams oblivious to the fact that their systems have been infiltrated.

As the threat landscape evolves and attackers become more sophisticated, it is preferable for IDS solutions to provide false positives than false negatives. In other words, it is better to discover a potential threat and prove it to be wrong than for the IDS to mistake attackers for legitimate users. Furthermore, IDS solutions increasingly need to be capable of quickly detecting new threats and signs of malicious behavior.

What is an Intrusion Prevention System? An essential part of Intrusion Prevention System is the **network security technology** that constantly monitors network traffic to identify threats. Under the general meaning of IPS, IPS technology is also an intrusion detection prevention system (IDPS).

What Is an **Intrusion Prevention System (IPS)**?



Why is IPS important for system security?

Organizations choose IPS technologies over traditional reactive network security efforts because IPS proactively detects and prevents harm from malicious traffic. IPS protection identifies potential threats by monitoring network traffic in real time by using network behavior analysis.

If an unauthorized attacker gains network access, the IPS identifies the suspicious activity, records the IP address, and launches an automated response to the threat based on rules set up in advance by the network administrator.

IPS is an adaptable safeguard technology for system security

IPS includes anti-virus/anti-malware software, firewall, anti-spoofing software, and network traffic monitoring. Enterprises use IPS to document threats, uncover problems with security policies, and block external or insider security violations.

How Intrusion Prevention Systems (IPS) work?

An [IPS security service](#) is typically deployed “in-line” where they sit in the direct communication path between the source and the destination, where it can analyze in real-time all the network traffic flow along that path and take automated preventive action. The IPS can be deployed anywhere in the network but their most common deployments locations are:

- Enterprise Edge, Perimeter
- [Hyperscale data center](#), Enterprise Data Center

An IPS can be deployed as a standalone IPS or the same capability can be turned on in the consolidated IPS function inside a next-generation firewall (NGFW). An IPS uses signatures which can be both vulnerability or exploit specific to identify malicious traffic. Typically, these employ signature-based detection or statistical anomaly-based detection to identify malicious activity.

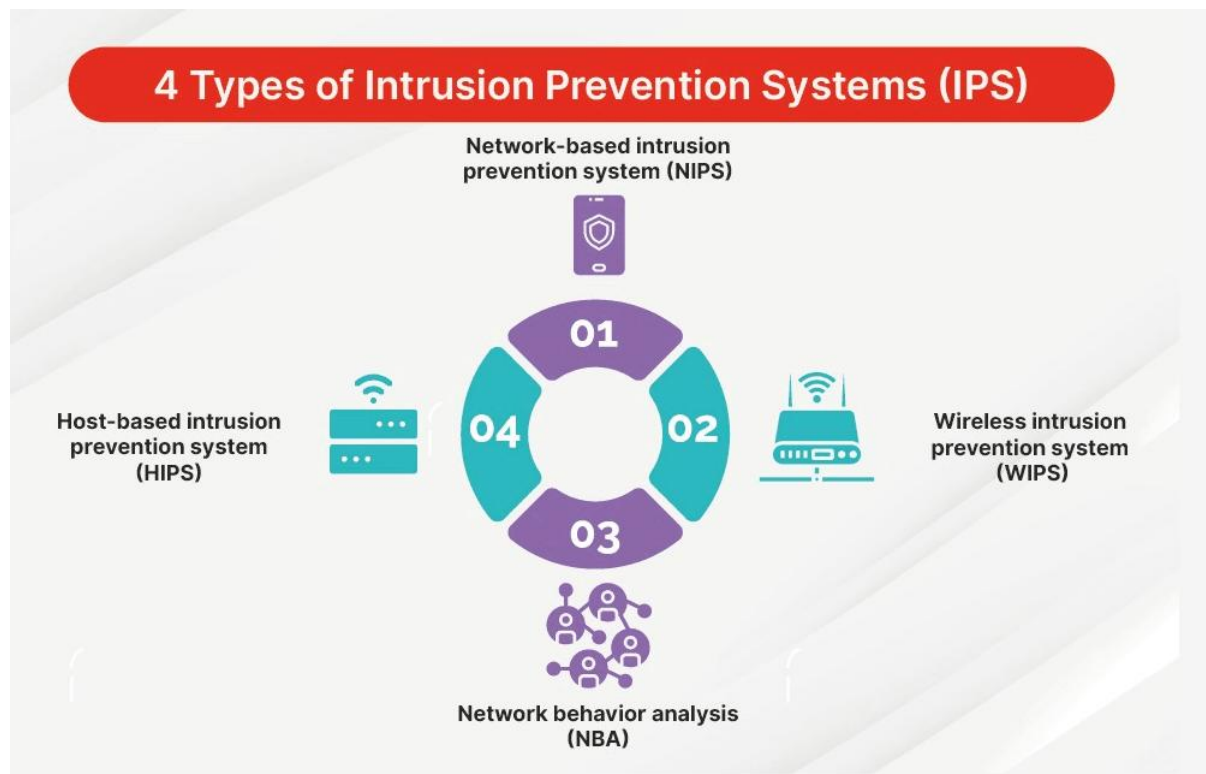
1. **Signature-based Detection:** It uses uniquely identifiable signatures that are located in exploit code. When exploits are discovered, their signatures go into an increasingly expanding database. Signature-based detection for IPS involves either exploit-facing signatures, which identify the individual exploits themselves, or vulnerability-facing signatures, which identify the vulnerability in the system being targeted for attack. Vulnerability-facing signatures are important for identifying potential exploit variants that haven't been previously observed, but they also increase the risk of false positive results (benign packets mislabeled as threats).
2. **Statistical Anomaly-based Detection:** This randomly samples network traffic and compares samples to performance level baselines. When samples are identified as being outside the baseline, the IPS triggers an action to prevent a potential attack.

Once the IPS identifies the malicious traffic that can be network exploitable it deploys what is known as a virtual patch for protection. Virtual patch, acts as a safety measure

against threats that exploit known and unknown vulnerabilities. It works by implementing layers of security policies and rules that prevent and intercept an exploit from taking network paths to and from a vulnerability, thereby offering coverage against that vulnerability at the network level rather than the host level.

Types Of Intrusion Prevention Systems (IPS)

There are four noteworthy types of intrusion prevention systems. Each type has its own unique defense specialty.



1. Network-based Intrusion Prevention System (NIPS)

Typically, a network-based intrusion prevention system is placed at key network locations, where it monitors traffic and scans for cyberthreats.

2. Wireless Intrusion Prevention System (WIPS)

As you would expect, wireless intrusion prevention systems monitor Wi-Fi networks, acting as a gatekeeper and removing unauthorized devices.

3. Host-based Intrusion Prevention System (HIPS)

Installed on endpoints like PCs, host-based intrusion prevention systems monitor inbound and outbound traffic from that device only. HIPS works best in tandem with a NIPS and serves to block threats that have made it past the NIPS.

4. Network Behavior Analysis (NBA)

Not be confused with professional basketball, NBA is focused on network traffic to detect odd movement and flows that might be associated with distributed denial of service (DDoS) attacks.

Attacks Detected And Prevented By IPS

An IPS security solution needs to handle various types of attacks, such as:

- **Address Resolution Protocol (ARP) Spoofing:** This attack re-directs traffic from a legitimate system to the attacker. Fake ARP messages sent by an attacker create a link between the attacker's MAC address and the IP address of an attacked system.
- **Buffer Overflow:** This attack uses vulnerabilities in the buffer overflow to overwrite memory and corrupt the execution of an application.
- **Distributed Denial of Service (DDoS):** A DDoS attack is a massive flood of traffic from distributed computers meant to overwhelm a system making it unavailable for legitimate requests.
- **IP Fragmentation:** This attack exploits datagram fragmentation mechanisms confusing the targeted system about how to reassemble TCP/UDP datagrams.
- **Operating System (OS) Fingerprinting:** These attacks exploit vulnerabilities in the OS.
- **Ping of Death:** Using a ping command, an attacker sends oversized or malformed packets that crash a system.
- **Port Scanning:** This is a port attack, scanning for an open, unprotected port to exploit.
- **Server Message Block (SMB) Probes:** This is a capture of SMB protocol authentication requests to relay them to the attacker's host.
- **Smurf:** This is a DDoS attack using **Internet Control Message Protocol (ICMP)** packets to overwhelm a system.
- **Secure Sockets Layer (SSL) Evasion:** This exploits SSL and Transport Layer Security (TLS) encryption that hides malicious content to avoid detection and get past network security.
- **SYN Flood:** Under this attack, a considerable volume of SYN (synchronize) packets sent as connection requests overwhelm a server or firewall.