**Introduction**

We are increasingly relying on web, and accessing important information as well as transmitting data through it. At the same time, quantity and impact of security vulnerabilities in such applications has grown as well. Billions of transactions are performed online with the help of various kinds of web applications.

Almost in all of them user is authenticated before providing access to backend database for storing all the information.

In this whole scenario a well-designed injection can provide access to malicious or unauthorized users and mostly achieved through SQL injection and Cross site scripting (XSS).

In this paper we are going to provide a detailed survey of various kinds of SQL injection, XSS attacks and approaches to detect and prevent them.

Furthermore we are also going to provide an Enhanced VAIT against these attacks.

**Definitions**

**SQL Injection Attack**

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.



**Cross-Site Scripting (XSS)**

Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts (also commonly referred to as a malicious payload) into a legitimate website or web application. XSS is amongst the most rampant of web application vulnerabilities and occurs when a web application makes use of invalidated or unencoded user input within the output it generates.

By leveraging XSS, an attacker does not target a victim directly. Instead, an attacker would exploit vulnerability within a website or web application that the victim would visit, essentially using the vulnerable website as a vehicle to deliver a malicious script to the victim’s browser.

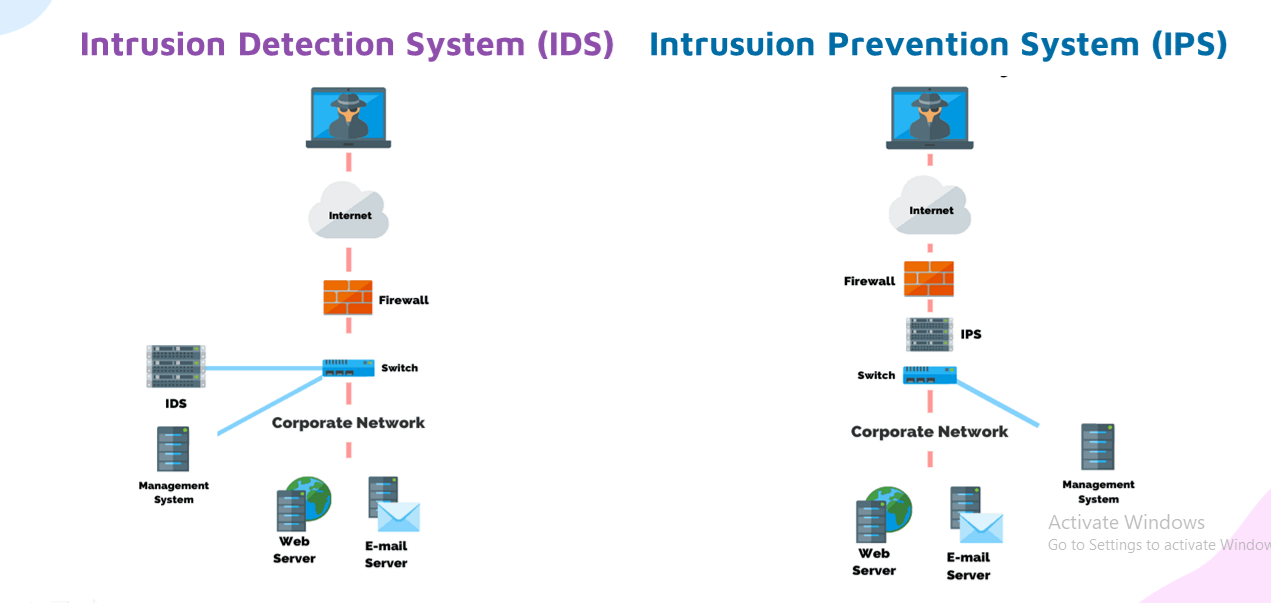


**Intrusion Detection System (IDS)**

analyze and monitor network traffic for signs that indicate attackers are using a known cyberthreat to infiltrate or steal data from your network. IDS systems compare the current network activity to a known threat database to detect several kinds of behaviors like security policy violations, malware, and port scanners.

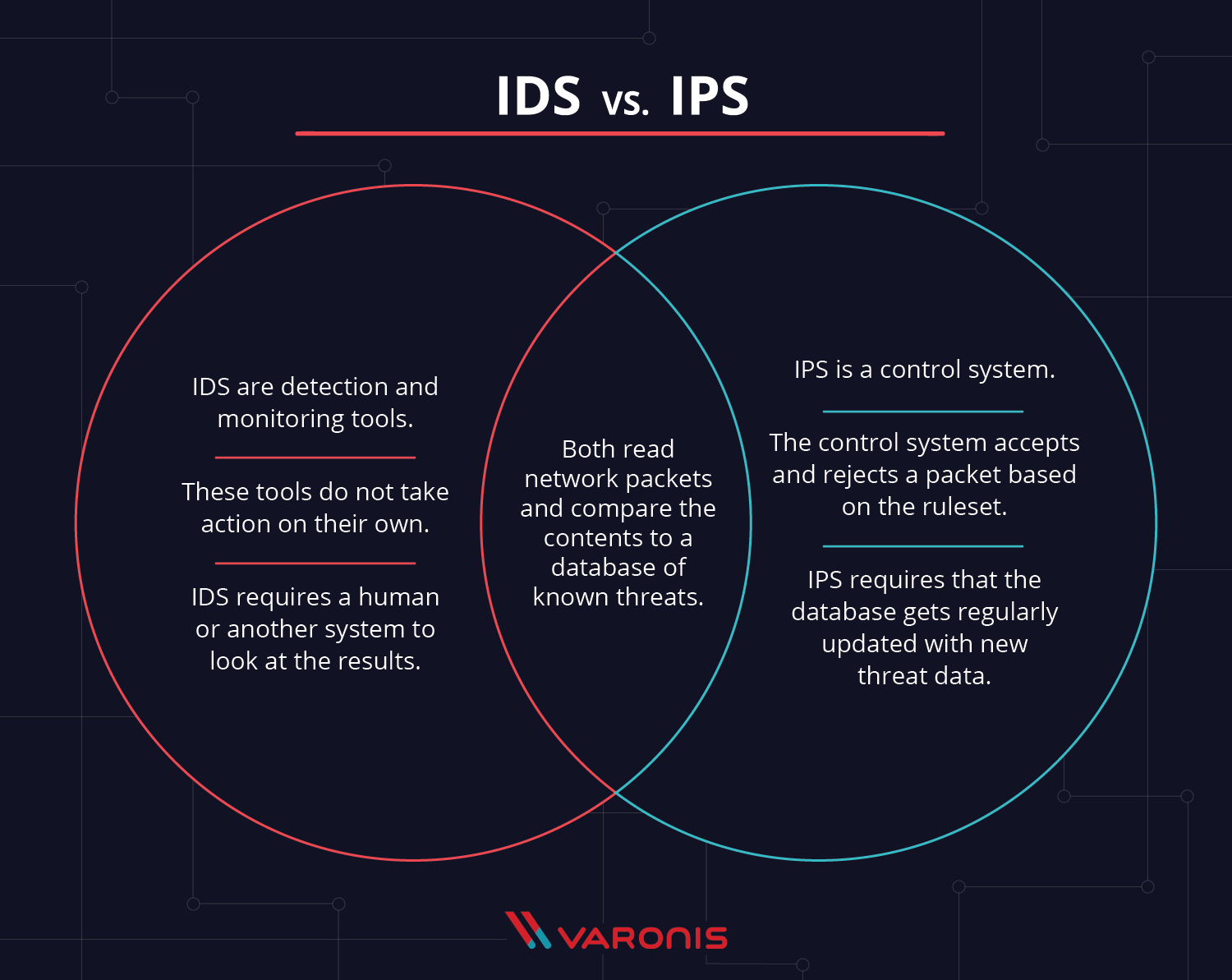
**Intrusion Prevention System (IPS)**

Live in the same area of the network as a firewall, between the outside world and the internal network. IPS proactively deny network traffic based on a security profile if that packet represents a known security threat.



**The Differences between IDS and IPS**

Both IDS/IPS read network packets and compare the contents to a database of known threats. The primary difference between them is what happens next. IDS are detection and monitoring tools that don’t take action on their own. IPS is a control system that accepts or rejects a packet based on the ruleset.



**Challenges**

**Firewall**

Firewalls and SSL provide no protection against web application hacking, simply because access to the website has to be made public – All modern database systems (e.g. Microsoft SQL Server, Oracle and MySQL) may be accessed through specific ports (e.g., port 80 and 443) and anyone can attempt direct connections to the databases effectively bypassing the security mechanisms used by the operating system.

These ports remain open to allow communication with legitimate traffic and therefore constitute a major vulnerability.

**VAIT**

Our Existing system uses VAIT tool to prevent attacks.

**XSS**

This VAIT is evaluating Sql Injection but not supports effectively to xss attacks.

**Problem Definition**

Web applications do raise a number of security concerns stemming from improper coding. Serious weaknesses or vulnerabilities, allow hackers to gain direct and public access to databases in order to churn sensitive data.

Many of these databases contain valuable information (e.g., personal and financial details) making them a frequent target of hackers.

Recent research shows that 75% of cyber-attacks are done at web application level.

1) Websites and related web applications must be available 24 hours a day, 7 days a week, to provide the required service to customers, employees, suppliers and other stakeholders.

2) Firewalls and SSL provide no protection against web application hacking, simply because access to the website has to be made public – All modern database systems (e.g. Microsoft SQL Server, Oracle and MySQL) may be accessed through specific ports (e.g., port 80 and 443) and anyone can attempt direct connections to the databases effectively bypassing the security mechanisms used by the operating system.

These ports remain open to allow communication with legitimate traffic and therefore constitute a major vulnerability.

3) Web applications often have direct access to backend data such as customer databases and, hence, control valuable data and are much more difficult to secure.

Those that do not have access will have some form of script that allows data capture and transmission. If a hacker becomes aware of weaknesses in such a script, he may easily reroute unwitting traffic to another location and illegitimately hive off personal details.

4) Most web applications are custom-made and, therefore, involve a lesser degree of testing than off-theshelf software. Consequently, custom applications are more susceptible to attack.

**Proposed Solution**

In this system we are going to provide a detailed survey of various kinds of SQL injection, XSS attacks and approaches to detect and prevent them. Furthermore we are also going to provide a Enhanced VAIT against these attacks. And then we are also going to present our findings and note down future expectations and expected development of counter measures against these attacks.

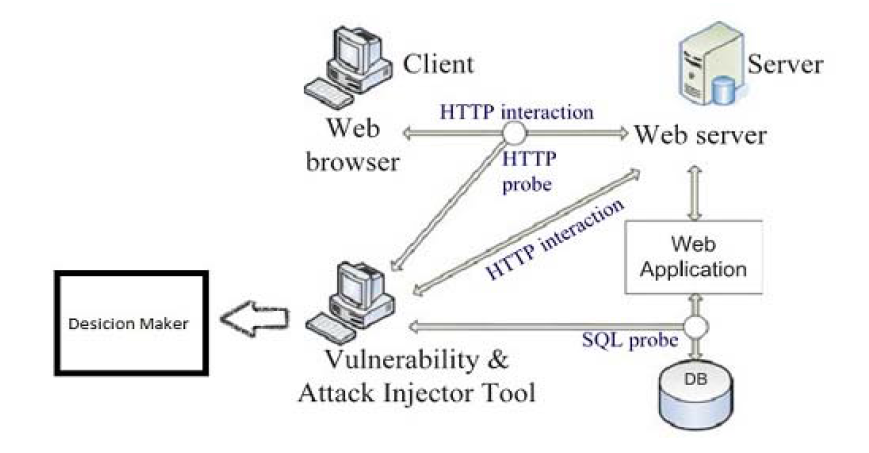
**Advantages of Proposed System:**

(i) Our proposed system effectively evaluates both t-xss attack and sql injection.

(ii) It uses EVAIT (Enhanced Vulnerability & Attack Injection) tool.

**Proposed Architecture**

To demonstrate the proposed VAIT they conducted three groups of experiments.

* Injecting vulnerabilities into three web applications to verify the quality of the vulnerabilities injected and the attack performance.
* Testing an IDS for databases by using it in line with the VAIT.
* Evaluating two top commercial web application vulnerability scanners regarding the detection of vulnerabilities that may be exploited by adhoc SQLi attacks.

The server runs **Linux** and the web server is **Apache**. This server hosts a **PHP** web application that uses a **Mysql** database.

**Module Description**

1. **Vulnerability Process**

The implementation of the Vulnerability & Attack Injector Tool (VAIT) that allows the automation of the entire process. We used this tool to run a set of experiments that demonstrate the feasibility and the effectiveness of the proposed methodology. The experiments include the evaluation of coverage and false positives of an intrusion detection system for SQL Injection attacks and the assessment of the effectiveness of two top commercial web application vulnerability scanners.

Results show that the injection of vulnerabilities and attacks is indeed an effective way to evaluate security mechanisms and to point out not only their weaknesses but also ways for their improvement.

1. **Attacker System**

It is in this vulnerability injection stage that vulnerabilities are injected into the web application. For this purpose, it needs information about which input variables carry relevant information that can be used to execute attacks to the web application. This stage starts by analyzing the source code of the web application files searching for locations where vulnerabilities can be injected the injection of vulnerabilities is done by removing the protection of the target variables, like the call to a sanitizing function. This process follows the realistic patterns resulting from the field study presented.

Once it finds a possible location, it performs a specific code mutation in order to inject one vulnerability in that particular location.

1. **Vulnerability Injection Stage:**

It is in this vulnerability injection stage that vulnerabilities are injected into the web application. For this purpose, it needs information about which input variables carry relevant information that can be used to execute attacks to the web application. This stage starts by analyzing the source code of the web application files searching for locations where vulnerabilities can be injected. The injection of vulnerabilities is done by removing the protection of the target variables, like the call to a sanitizing function. This process follows the realistic patterns resulting from the field study presented.

Once it finds a possible location, it performs a specific code mutation in order to inject one vulnerability in that particular location.

1. **Vait Components**

In this section we present the methodology for testing security mechanisms in the context of web applications.

The methodology is based on the injection of realistic vulnerabilities and the subsequent controlled exploit of those vulnerabilities in order to attack the system. It gets value from input validation section and process the input for vulnerability attack codes if found block the code for prevent execution and otherwise it process through it.

**1) Xss**

For cross site attack check user inputs such as username, passwords, search queries and etc.

**2) Sql Injection**

In sql injection validate the input queries before it executes by database server.

**3) Decision Maker**

In this section makes a decision for actions such as if found any unauthorized activities blocks the user and protect the data that he tried to hack.

**Questions after Discussion**

**How does XSS work?**

Cross-site scripting works by manipulating a vulnerable web site so that it returns malicious JavaScript to users. When the malicious code executes inside a victim's browser, the attacker can fully compromise their interaction with the application.

**What are the types of XSS attacks?**

There are three main types of XSS attacks. These are:

* [**Reflected XSS**](https://portswigger.net/web-security/cross-site-scripting#reflected-cross-site-scripting): where the malicious script comes from the current HTTP request.
* [**Stored XSS**](https://portswigger.net/web-security/cross-site-scripting#stored-cross-site-scripting): where the malicious script comes from the website's database.
* [**DOM-based XSS**](https://portswigger.net/web-security/cross-site-scripting#dom-based-cross-site-scripting): where the vulnerability exists in client-side code rather than server-side code.
* **Stored XSS (Persistent XSS)**

The most damaging type of XSS is [Stored XSS (Persistent XSS)](https://www.acunetix.com/blog/articles/persistent-cross-site-scripting/). An attacker uses Stored XSS to inject malicious content (referred to as the payload), most often JavaScript code, into the target application. If there is no input validation, this malicious code is permanently stored (persisted) by the target application, for example within a database. For example, an attacker may enter a malicious script into a user input field such as a blog comment field or in a forum post.

When a victim opens the affected web page in a browser, the XSS attack payload is served to the victim’s browser as part of the HTML code (just like a legitimate comment would). This means that victims will end up executing the malicious script once the page is viewed in their browser.

* **Reflected XSS (Non-persistent XSS)**

The second and the most common type of XSS is Reflected XSS (Non-persistent XSS). In this case, the attacker’s payload has to be a part of the request that is sent to the web server (URL). It is then reflected back in such a way that the HTTP response includes the payload from the HTTP request. Attackers use malicious links, phishing emails, and other social engineering techniques to lure the victim into making a request to the server. The reflected XSS payload is then executed in the user’s browser.

Reflected XSS is not a persistent attack, so the attacker needs to deliver the payload to each victim. These attacks are often made using social networks.

* **DOM-based XSS**

[DOM-based XSS](https://www.acunetix.com/blog/articles/dom-xss-explained/) is an advanced XSS attack. It is possible if the web application’s client-side scripts write data provided by the user to the Document Object Model (DOM). The data is subsequently read from the DOM by the web application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which will be stored as part of the DOM and executed when the data is read back from the DOM.

A DOM-based XSS attack is often a client-side attack and the malicious payload is never sent to the server. This makes it even more difficult to detect for Web Application Firewalls (WAFs) and security engineers who analyze server logs because they will never even see the attack. DOM objects that are most often manipulated include the URL (document.URL), the anchor part of the URL (location.hash), and the Referrer (document.referrer).

**XSS Discovery and Prevention**

Cross-site Scripting is a very old technique but XSS vulnerabilities remain one of the most common ones on the web. They are still mentioned by the Open Web Application Security Project (OWASP) as one of the top-10 security risks.

An easy way to test if your website or web application is vulnerable to XSS and other vulnerabilities is to run an automated web scan using the Acunetix [vulnerability scanner](https://www.acunetix.com/vulnerability-scanner/), which includes a specialized [XSS scanner](https://www.acunetix.com/vulnerability-scanner/xss-vulnerability-scanning/) module. [Take a demo](https://www.acunetix.com/web-vulnerability-scanner/demo/) and find out more about running XSS scans against your website or web application.

[Cross Site Scripting (XSS)](https://www.acunetix.com/websitesecurity/cross-site-scripting/) attacks are amongst the most common types of attacks against web applications. [XSS vulnerabilities](http://www.acunetix.com/websitesecurity/xss/) all fall under the same category, however, a more detailed look at the techniques employed during XSS attacks reveals a multitude of tactics that exploit a variety of attack vectors. This article describes the two most common and useful XSS prevention mechanisms – *filtering* and *escaping*.

**Filtering for XSS**

All XSS attacks affect your web site through some form of client-side user input. Malicious code could come from a simple <form> submitted by your users or could take a more complex route such as a JSON script, XML web service, or even an exploited cookie. In all cases, the web developer should be aware that the data is coming from an external source and therefore must not be trusted since it may introduce a security vulnerability.

The simplest and arguably the easiest form of cross-site scripting vulnerability elimination would be to pass all external data through a filter. Such a filter would remove dangerous keywords, for example, the infamous <script> tag, JavaScript commands, CSS styles, and other dangerous HTML markups (such as those that contain event handlers.)

Many web developers choose to implement their own XSS filter mechanisms. They usually write server-side code (in PHP, ASP, or some other web-enabled development language) to search for keywords and replace them with empty strings. A lot of code uses regular expressions for filtering and replacing.

This technique is not a bad one in itself, however, the hackers usually have more experience than web developers and often manage to circumvent simple filters by using techniques such as hex encoding, Unicode character variations, line breaks, and null characters in strings. These techniques must all be catered for and that is why it is recommended to use some sort of library that has been tried and tested by the community at large.

Many libraries exist to choose from and your choice will primarily depend on the back-end technology that your web server uses. What is important is that you choose a library that is regularly maintained by a reliable source. XSS techniques keep changing and new ones emerge all the time so your filters will need to be updated periodically to keep abreast with the changing attacks. If you are using Java, then a good place to go to is the [OWASP Java Encoder Project](https://www.owasp.org/index.php/OWASP_Java_Encoder_Project). For PHP, there is a comprehensive library called [HTML Purifier](http://htmlpurifier.org/), which boasts strict standards compliance and better features than other filters.

The side effect of filtering techniques is that legitimate text is often removed because it matches forbidden keywords. For example, this article would be incomplete if the Acunetix web server simply filtered out all HTML markup. It would be impossible to include text like <script> and alert('you have been hacked'). If you want to preserve the original data (and its formatting) as best as possible you need to relax your filters and employ HTML, Script, and CSS escaping techniques.

**Escaping from XSS**

Escaping is the primary means to avoid cross-site scripting attacks. When escaping, you are effectively telling the web browser that the data you are sending should be treated as data and should not be interpreted in any other way. If an attacker manages to put a malicious script on your page, the victim will not be affected because the browser will not execute the script if it is properly escaped. In HTML, you can escape dangerous characters by using HTML entities, for example, the &# sequence followed by its character code.

Escaping HTML is fairly easy. However, to guarantee web application security, you must also escape JavaScript code, Cascading Style Sheets, and sometimes XML data. There are also many pitfalls if you try to do all the escaping by yourself. This is where an escaping library comes useful.

The two most popular escaping libraries available are the [ESAPI](https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API) provided by OWASP and AntiXSS provided for Microsoft. ESAPI can plug into various technologies such as Java, .NET, PHP, Classic ASP, Cold Fusion, Python, and Haskell. AntiXSS exclusively protects Microsoft technologies and is, therefore, better suited in an all-Microsoft environment. Both libraries are constantly updated to keep up with the latest hacker techniques and are maintained by industry experts who understand changing tactics and emerging technologies.

**When to Escape**

You cannot just simply escape everything or else your own scripts and HTML markup will not work, rendering your page useless. There are several places on your web page which you need to ensure are properly escaped. You can use your own escaping functions (not recommended) or existing libraries.

* **HTML Escaping**

Use HTML escaping when untrusted data is inserted between HTML opening and closing tags. These are standards tags such as <body>, <div>, <table>, etc. For example:

<div>If this data is untrusted, it must be HTML-escaped.</div>

* **JavaScript Escaping**

Use JavaScript escaping when untrusted data is inserted inside one of your scripts, or in a place where malicious JavaScript may be included. This includes certain HTML attributes such as style and all event handlers such as onmouseover and onload. For example:

<script>alert('If this data is untrusted, it must be JavaScript-escaped.')</script>

<body onload="If this data is untrusted, it must be JavaScript-escaped.">

* **CSS Escaping**

Use CSS escaping when untrusted data is inserted inside your CSS styles. Many CSS styles can be used to smuggle a script into your page. For example:

<div style="background-image: If this data is untrusted, it must be CSS-escaped.">

**XSS Attacks Are a Moving Target**

Recommendations in this article are by no means exhaustive, however, they should be a good starting point to secure applications. Technology is changing and hacker attacks are getting more sophisticated but by understanding the basics, you can be prepared to prevent future attack techniques that will most definitely arise.

The first step in defending against all XSS attack vectors is to code your web applications carefully and use proper escaping mechanisms in the right places. After that, comprehensive code review and testing should be performed, ideally using an [automated XSS scanner such as Acunetix](https://www.acunetix.com/vulnerability-scanner/xss-vulnerability-scanning/). When updates are made to your web applications, you should scan the affected pages again to ensure that no new vulnerabilities have been exposed.

**How Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) Work**

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are both parts of the network infrastructure. IDS/IPS compare network packets to a cyberthreat database containing known signatures of [cyberattacks](https://www.varonis.com/blog/cyber-attack-prep/) — and flag any matching packets.

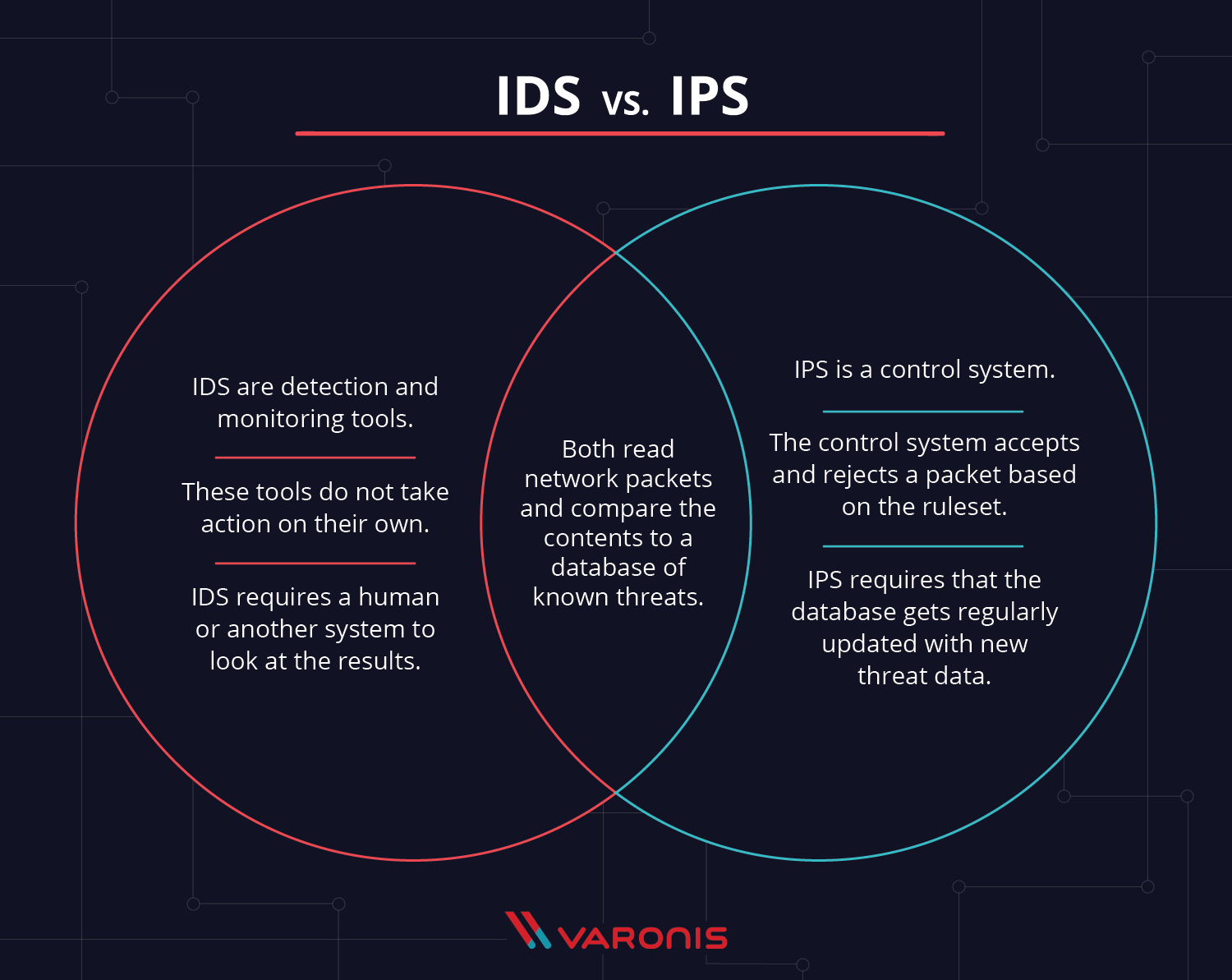
The main difference between them is that IDS is a monitoring system, while IPS is a control system.

IDS doesn’t alter the network packets in any way, whereas IPS prevents the packet from delivery based on the contents of the packet, much like how a firewall prevents traffic by IP address.

* **Intrusion Detection Systems (IDS):** analyze and monitor network traffic for signs that indicate attackers are using a known cyberthreat to infiltrate or steal data from your network. IDS systems compare the current network activity to a known threat database to detect several kinds of behaviors like security policy violations, malware, and port scanners.
* **Intrusion Prevention Systems (IPS):** live in the same area of the network as a firewall, between the outside world and the internal network. IPS proactively deny network traffic based on a security profile if that packet represents a known security threat.

Many IDS/IPS vendors have integrated newer IPS systems with firewalls to create a Unified Threat Management (UTM) technology that combines the functionality of those two similar systems into a single unit. Some systems provide both IDS and IPS functionality in one unit.

**The Differences between IDS and IPS**

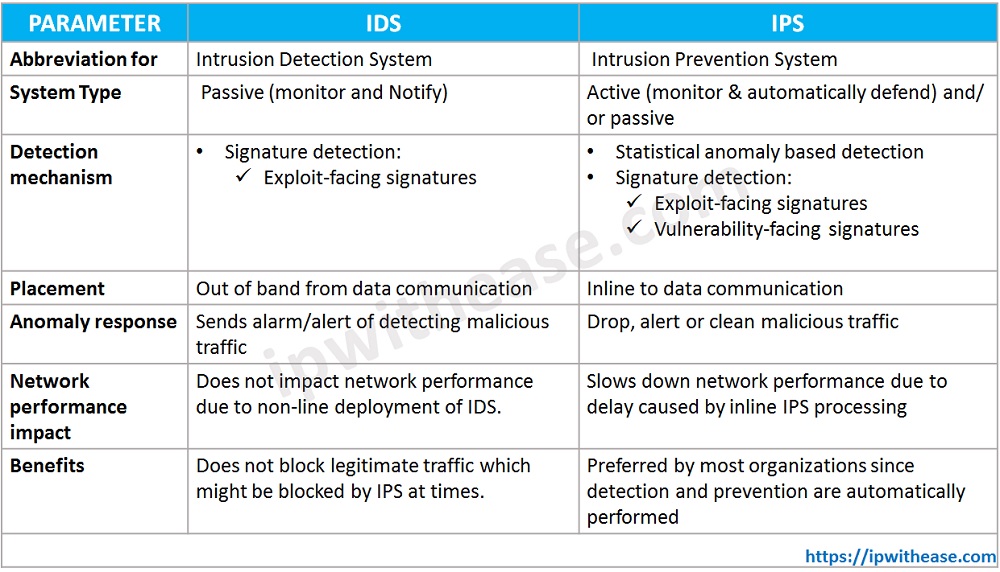


Both IDS/IPS read network packets and compare the contents to a database of known threats. The primary difference between them is what happens next. IDS are detection and monitoring tools that don’t take action on their own. IPS is a control system that accepts or rejects a packet based on the ruleset.

IDS requires a human or another system to look at the results and determine what actions to take next, which could be a full time job depending on the amount of network traffic generated each day. IDS makes a better post-mortem forensics tool for the CSIRT to use as part of their security incident investigations.

The purpose of the IPS, on the other hand, is to catch dangerous packets and drop them before they reach their target. It’s more passive than an IDS, simply requiring that the database gets regularly updated with new threat data.

\*Point of emphasis: IDS/IPS are only as effective as their cyberattack databases. Keep them updated and be prepared to make manual adjustments when a new attack breaks out in the wild and/or the attack signature isn’t in the database.



**IPS Preventing Methods and techniques**

IPS is an advance combination of IDS, personal firewalls and anti-viruses etc.

The purpose of an Intrusion Prevention System (IPS) is not only to detect an attack that is trying to interrupt, but also to stop it by responding automatically such as:

* Logging off the user.
* Shutting down the system.
* Stopping the process.
* Disabling the connection.

Similar to IDS, IPS can be divided into two types, i.e. Host-Based Intrusion Prevention Systems and Network-Based Intrusion Prevention Systems.

**IPS APPROACHES**

Some of the approaches being used are.

• Software based heuristic approach: This approach is similar to IDS anomaly detection using neural networks with the added ability to act against intrusions and block them.

• **Sandbox approach:** Mobile code like ActiveX, Java applets and various scripting languages are quarantined in a sandbox - an area with restricted access to the rest of the system resources. The system then runs the code in this sandbox and monitors it’s behavior. If the code violate a predefined policy it’s stopped and prevented from executing, thwarting the attack (Conry-Murray).

• Hybrid approach: On network-based IPS (NIPS), various detection methods, some proprietary including protocol anomaly, traffic anomaly, and signature detection work together to determine an imminent attack and block traffic coming from an inline router.

• Kernel based protection approach: Used on hostbased IPS (HIPS). Most operating systems restrict access to the kernel by a user application. The kernel controls access to system resources like memory, I/O devices, and CPU, preventing direct user access. In order to use resources user applications send requests or system calls to the kernel, which then carries out the operation. Any exploit code will execute at least one system call to gain access to privileged resources or services. Kernel based IPS prevents execution of malicious system calls.

**TYPES OF IPS**

**Host based Intrusion Prevention (HIP)**

1. Storm watch: OKENA’s Storm Watch uses a kernel-based approach and works on servers and workstations. Policies - collections of access control rules based on acceptable behavior, is available out-of-the-box for common applications such as Microsoft SQL Server, Instant Messenger, and IIS Server. Policies control what resource is being used, what operation is being invoked, and which application is invoking it. Storm Watch hooks into the kernel and intercepts system calls (Okena).

It has four interceptors:

• File System interceptor - intercepts all file read and write requests.

• Network interceptor - intercepts packet events at the driver (NDIS) or transport (TDI) level.

• Configuration interceptor - intercepts read/write requests to the registry on Windows or to files on UNIX.

• Execution space (Run-time environment) interceptor - requests to write to memory not owned by the requesting application will be blocked by this interceptor.

2. ENTERCEPT’s Standard: Entercept, a pioneer in kernel-based protection, proactively protects the host by intercepting system calls (Entercept). Unlike Okena’s Storm Watch it uses both, signatures and behavior rules to stop and detect attacks. In an article by Ed Skoudis on “infosec’s WORST NIGHTMARES”, some Night mares that he mentions are stealthier attacks and “super” worms – “Fast spreading, multiplatform, multiexploit, zero-day, metamorphic worms”. He goes on to say that one way of preparing for these coming “super” worms is to, “Utilize host-based intrusion detection and prevention tools such as Entercept Security Technologies and OKENA’s Storm Watch on critical systems to block or rapidly discover attacks.

**Network based Intrusion Prevention (NIP)**

NIPS are generally appliance-based systems that sit in line, and block suspicious traffic after detecting an attack. They utilize different detection methods, signature detection, anomaly detection, and some proprietary methods, to block specific attacks. Some of the methods adopted by vendors are

• Stateful Signature detection – It looks at relevant portions of traffic, where the attack can be perpetrated. It does this by tracking state and based on the context specified by the user detects an attack. It is not completely automatic, as the user needs to have some prior knowledge about the attack.

• Protocol anomaly detection - All vendors do detailed packet analysis with protocol decode engines to ensure packets meet protocol requirements. Traffic normalization is also done to remove protocol ambiguities and ensures that traffic interpreted by the NIPS.

**Methods used for Detection and Prevention:**

1. **Detection:**

There are three methods used for detection. They are Misuse detection or Signature detection (knowledge based), Anomaly detection (behavior based), Stateful protocol analysis method

**Misuse detection** discovers attacks based on patterns extracted from known intrusions. Anomaly detection identifies attacks based on significant deviations from normal activities. Misuse detection has low false positive rate, but cannot detect novel attacks.

**Anomaly detection** can detect unknown attacks, but usually has a high false positive rate. To combine the advantages of misuse and anomaly detection, many hybrid approaches have been proposed. Data mining is the analysis of large data sets to discover understandable patterns or models. Here there are some examples of Signatures given.

• A telnet attempt with a username of “root”, which is a violation of an organization’s security policy.

• An e-mail with a subject of “Free pictures!” and an attachment filename of “freepics.exe”, which are characteristics of a known form of malware.

• An operating system log entry with a status code value of 645, which indicates that the host’s auditing has been disabled.

• Signature-based detection is very effective at detecting known threats but largely ineffective at detecting previously unknown threats, threats disguised by the use of evasion techniques, and many variants of known threats.

**Stateful protocol** analysis is the process of comparing predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations. Stateful protocol analysis relies on vendor-developed universal profiles that specify how particular protocols should and should not be used.

1. **Prevention methods:**

An IPS is a preemptive network security approach that uses advanced techniques to detect and block (or prevent) possible intrusion attempts into a computer system. An IPS thoroughly scans the traffic flowing to and from a computer system or computer network for security breaches. If a threat is detected, the system is able to take defensive actions such as dropping a particular data packet or dropping the whole connection. The scan captures details, the action report is logged in a file, and an alert is sent to the system or network administrator. IPSs differ in how they scan the data streams to detect a threat or intrusion. Some of the most popular methods are described below.

**Signature method:** In the signature method, the IPS compares the real-time data stream patterns with a huge database of attack patterns that have already been detected. In this process, each data packet is scanned, byte by byte, for a particular pattern or string that represents complete or partial code associated with a known attack. The pattern or string could be anything, such as a command name or a specific set of characters. some examples of signature matching:

• Matching the subject description or attachment name of an email with details of a known or detected malicious email.

• Tracking the denial-of-service attack by counting the number of times a command is executed and matching it with known statistics of a similar kind of attack.

• Matching a user activity prior to authentication or login with a known attack pattern. **Profile method:** In the profile method, the IPS collects a pattern of data stream flowing to and from a computer system (or computer network) in controlled or trusted conditions. This pattern is treated as a baseline profile and compared against the real-time data stream patterns. A real-time data stream pattern that is found to be suspiciously different from the baseline profile is treated as an attack, and preventive action is taken against it. A standard baseline profile can represent normal behavior of things such as network connections, applications, users, and hosts. For example, if a real-time data stream is observed to be accessing a crucial system file that wasn't accessed when the baseline profile was generated in the controlled environment, this attempt is treated as malicious stateful protocol method.

**State protocol method:** Data packets are wrapped with various protocol headers. Each layer of the TCP/IP or Open Systems Interconnection (OSI) model adds the header of the protocol (the protocol being used for that layer, which is to the received packet. Protocols follow a standard document format known as Requests for Comments (RFCs). An RFC completely explains the protocol and describes how it should be used. The RFC forms the basis of the stateful protocol method. In this method, each protocol header is peeled apart and scanned for its consistency with what its RFC specifies. A deviation from the RFC is considered alarming, and an alert is raised. For example, a TCP packet with only SYN and FIN flags on is a deviation from what the TCP RFC specifies. If a data packet with the TCP header contains both these flags on, then this needs to be reported.

**References**

**Paper**

<https://www.acadpubl.eu/hub/2018-119-14/articles/2/124.pdf>

**XSS**

<https://portswigger.net/web-security/cross-site-scripting>

<https://www.acunetix.com/websitesecurity/xss/>

<https://www.acunetix.com/blog/articles/preventing-xss-attacks/>

**IPS/IDS**

<https://www.varonis.com/blog/ids-vs-ips/>

<https://www.bitlyft.com/resources/what-is-the-difference-between-ids-and-ips>

<https://ipwithease.com/difference-between-ips-and-ids-in-network-security/>

\* <https://openaccess.city.ac.uk/id/eprint/1737/1/>

\*<https://ijcsi.org/papers/IJCSI-13-2-38-43.pdf>