PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
Mohamed Khider University – BISKRA
Faculty of Exact Sciences, Natural sciences and Life
Department of Computer Science

Order N°: RTIC10/M2/2022

THESIS

Presented for the Academic Master's degree in

Computer Science

Option: Information and communication networks and technologies

---

Web application vulnerabilities detection and reduction

By:

BENZEKRI MOHAMED EL AKHDAR

Presented in 26/06/2022 Board of Examiners:

:

Hamida Ammar	MAA	President
Boukhlouf Djemaa	MCB	Supervisor
Naidji Ilyes	MCB	Examiner

Academic year 2021-2022

الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة محمد خيضر بسكرة

تصريح شرفي

(خاص بالالتزام بقواعد النزاهة العلمية لإنجاز بحث)

أنا الممضي أسفله،

السيد (ة): بن زكري محمد الأخضر الصفة: ذكر **طالب :** ثانية ماستر شبكات وتكنولوجيات المعلومات والاتصالات.

الحامل لبطاقة التعريف الوطنية رقم: 664797 والصادرة بتاريخ: 2013/07/24

المسجل بكلية: العلوم الدقيقة و علوم الطبيعة و الحياة

قسم: الإعـــــلام الآلـي

والمكلف بإنجاز مذكرة تخرج في الماستر

عنوانها : Web application vulnerabilities detection and reduction

أصرح بشرفي أني ألتزم بمراعاة المعايير العلمية والمنهجية ومعايير الأخلاقيات المهنية والنزاهة الأكاديمية المطلوبة في إنجاز البحث المذكور أعلاه.

التاريخ: 20/06/2022

توقيع المعني:

.

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
Mohamed Khider University – BISKRA
Faculty of Exact Sciences, Natural sciences and Life
Department of Computer Science

Order N°: RTIC10/M2/2022

THESIS

Presented for the Academic Master's degree in

Computer Science

Option: Information and communication networks and technologies

---

Web application vulnerabilities detection and reduction

By:

BENZEKRI MOHAMED EL AKHDAR

Presented in 26/06/2022 Board of Examiners:

:

Hamida Ammar	MAA	President
Boukhlouf Djemaa	MCB	Supervisor
Naidji Ilyes	MCB	Examiner

Academic year 2021-2022

Dedication

First,I give thanks to Allah who helped me and gave me the strength and patience to endure all the difficulties to complete the work and who taught us the purpose of life .

I thank my good father Benzekri Kamel and my great mother Boucetta Razika , who gave me courage and financial and psychological support. I would like to sincerely thank my deep gratitude to Mr. Boukhlouf Djamaa, as supervisor of the dissertation. She has always been careful and advised, and for the effort .

I also dedicate this dissertation to all my family my brother Youcef , my sisters Manel , Nour and Selma , my grandmother Turkia , my Aunt Djahida and all my friends especially Mohamed Messaoud Kisrane , Abdelkader,Ikbal ,miloud and ahmed who have supported me throughout the process. I will always appreciate all they have done.

Abstract

With the emergence of web applications and the wide spread of their services and their use in business transactions and data exchange, which allowed attackers to exploit weaknesses in web applications and carry out various attacks such as Cross Site Scripting (XSS injection) and SQL code injection, defacement and session hijacking and doing damage to users.In this study we will focus on tow types of injection attacks (XSS and SQL) so that there are ways to detect and prevent vulnerabilities and protect them from attackers such as IDS, IPS, Firewall, and VPN They only reduce and do not do the job.

In this work we propose protection methods such as machine learning, deep learning, neural network to create one model and logistic regression to analyze SQL and XSS commands and data that is logged to enter into a web application, and we specifically suggest a convolutional neural network and logistic regression to analyze these inputs, where we propose a set of data imported from the Internet with adding data manually with a suggestion a model for including normal and unusual transactions.

Keywords: SQL injection, XSS injection, machine learning, convolutional neural network, data processing, training, testing, model ...

Résumé

Avec l'émergence des applications web et la grand diffusion de leurs services et leur utilisation dans les transactions commerciales et l'échange de données, ce qui a permis aux attaquants d'exploiter les faiblesses des applications web et de mener diverses attaques telles que cross site script (injection XSS) et injection de code SQL, la dégradation et le détournement de session et causant des dommages aux utilisateurs.Dans cette étude, nous nous concentrerons sur les deux types d'attaques par injection (XSS et SQL) afin qu'il existe des moyens de détecter et de prévenir les vulnérabilités et de les protéger contre les attaquants tels que IDS, IPS, Pare-feu et VPN Ils ne font que réduire et ne font pas le travail.

Dans ce travail, nous proposons des méthodes de protection telles que l'apprentissage automatique, l'apprentissage profond, le réseau neuronal pour créer un modèle et la régression logistique pour analyser les commandes SQL et XSS et les données qui sont enregistrées pour entrer dans une application Web, et nous suggérons spécifiquement un réseau neuronal convolutif et une régression logistique pour analyser ces entrées, où nous proposons un ensemble de données importées d'Internet avec l'ajout de données manuellement avec une suggestion Un modèle pour y compris les transactions normales et inhabituelles.

Mots-clés : Injection SQL, injection XSS, apprentissage automatique, réseau neuronal convolutif, traitement de données, formation, tests, modèle ...

ملخص

مع ظهور تطبيقات الويب والانتشار الواسع لخدماتها واستخدامها في المعاملات التجارية وتبادل البيانات ، مما سمح للمهاجمين باستغلال نقاط الضعف في تطبيقات الويب وتنفيذ هجمات مختلفة مثل البرمجة النصية عبر المواقع (XSS injection) وحقن رمز SQL والتشويه واختطاف الجلسة وإلحاق الضرر بالمستخدمين .

في هذه الدراسة سوف نركز على أنواع هجمات الحقن (XSS و SQL) بحيث تكون هناك طرق للكشف عن نقاط الضعف ومنعها وحمايتها من المهاجمين مثل IDS و IPS و Firewall و VPN فهي تقلل فقط ولا تقوم بهذه المهمة.

في هذا العمل نقترح طرق الحماية مثل التعلم الآلي والتعلم العميق والشبكة العصبية لإنشاء نموذج واحد والانحدار اللوجستي لتحليل أوامر SQL و XSS والبيانات التي يتم تسجيلها للدخول إلى تطبيق ويب، ونقترح على وجه التحديد شبكة عصبية الالتفافية وانحدار لوجستي لتحليل هذه المدخلات، حيث نقترح مجموعة من البيانات المستوردة من الإنترنت مع إضافة البيانات يدويا مع اقتراح نموذج ل بما في ذلك المعاملات العادية وغير العادية.

الكلمات المفتاحية: حقن SQL ، حقن XSS ، التعلم الآلي ، الشبكة العصبية الالتفافية ، معالجة البيانات ، التدريب ، الاختبار ، النموذج ...

Summary

List of Figures

List of Tables

General introduction

Web application security is the process of protecting websites and online services against different security threats that exploit vulnerabilities in an application code. Common targets for web application attacks are content management systems (WordPress), database administration tools (PhpMyAdmin) and SaaS applications. For years, security experts have warned about vulnerabilities in Web applications. Unfortunately, these warnings are now coming true. The news these days is dominated by the news of a particular hacker successfully infiltrating this or that web application.

With the fast evolution of technology, including web application services, where most commercial users resort to buying and selling, and most large companies rely on web applications to facilitate communication between company employees and other departments and sharing data, which has allowed attackers to increase hacking, intrusions and targeting of workers. Credit card theft, taking advantage of weaknesses, can access an information system and steal data, defraud or damage the reputation of the company.Therefore, the protection of the web application must be strengthened and attackers prevented. Developers have used methods to detect and prevent intrusions such as IDS (Intrusion Detection System), IPS (Intrusion Prevention System), WAF (Web Application Firewall)..., but they irreplaceable all objectives because they do not provide high security level .

In this study we will try to introduce new methods such as machine learning and the different algorithms, deep learning and neural networks. First we collect the largest number of data sets that attackers use to hack the web site using injection (SQL or XSS) query then where we create a file that contains a data set that contains tow data types normal and abnormal and we train the CNN Convolutional Neural Network model via the data set where at the end we get a CNN model trainer . This model is able to predict the normal commands of the other commands and categorizes them into injected (SQL or XSS) query or not injected (SQL or XSS) query. The obtained result were effective and compared with other methods (navibyes ,knn,svm) it gave a best accuracy.

This work is organized into four main chapters:

1. **The first chapter :** We will try to introduce the basic concepts and definitions of the Web application and its most important elements and components, in addition to its structure general, its working method and field of use, and then move on to web application security to know about vulnerabilities then we go to talk about attacks and its types and in finally show the security mechanisms and approaches for securing web applications .
2. **The second chapter :** we present top 10 important vulnerabilities according to the World Security Organization (OWASP) and we try to explain each element and how it is exploited by attackers and how to do the prevention and we talk about National vulnerability database (NVD) then explain vulnerability detection methods based on machine learning and in the finally we show three related work of different approaches.
3. **The third chapter :** A full description of our system design is provided with a detailed design of the convolutional neural network model as well as some diagrams.
4. **The last chapter :** we will present the environment and programming languages and tools used in this work, describing the database used and some images of the application used after vulnerability prevention, mentioning some other models and comparing them with the convolutional neural network model

Chapter 1

Web application security

1.1 Introduction

Web applications are becoming more popular and widely being used in all aspects of work and social activities. However, the exponential development of web technologies comes at a price, because the number of Web application security issues increases rapidly as well and Web applications are becoming more prone to worrisome vulnerabilities. [28]

In this chapter, we describe at first what Web applications are, which structure they usually have and, then we give a Web Application Architecture Diagram, we continue by web application terminology, then we talk about vulnerabilities types and the last thing about security mechanisms and approaches for securing web application.

1.2 Web application

1.2.1 definition

(1) Web application or web app is a client–server software application which the client (or user interface) runs in a web browser [15].

(2) According to the definition of OWASP, a Web application is a client/server software application that interacts with users or other systems using the Hypertext Transfer Protocol (HTTP) [16].

1.3 Web application architecture

Web application architecture are generally structured as three-tiered, which consist of:

- The first tier is a Web browser (client) such as Google Chrome, Mozilla Firefox and Opera, etc..
- The middle tier is an engine, which generates pages dynamically using technologies such as PHP, ASP and JSP.
- The third tier is a database; it enables Web applications to store data and other content elements. By using SQL.

As illustrated in (figure 1.1) , a client (Web browser) sends requests to the middle tier, which handles these

requests, searches information required by making SQL queries against the database and generates response pages using this information, and shows them to the user in the browser.[7]

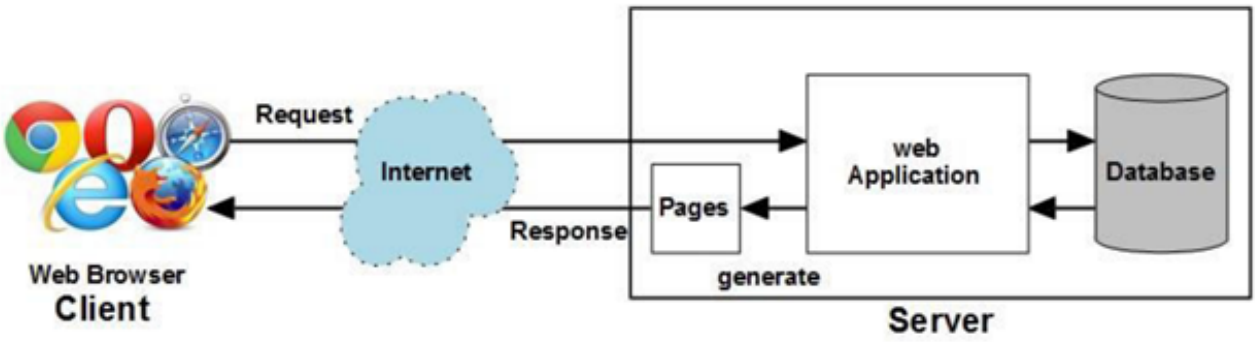

Figure 1.1: Dynamic Web Applications.[8]

1.3.1 Web application architecture components

Domain Name System (DNS)
is a fundamental system that helps search a domain name and IP address, and in this manner, a particular server receives a request sent by a user. We can say that DNS is like a phone book but for the Internet websites.[18]
Load Balancer:
Load Balancer primarily deals with horizontal scaling. With directing incoming requests to one of the multiple servers, the load balancer sends an answer to a user. Usually, web application servers exist in the form of multiple copies mirroring each other. Hence, any server processes requests in the same manner, and the load balancer distributes tasks among them, so they will not be overcharged.[18]
Web App Servers:
This component processes a user's request and sends documents (JSON, XML, etc.) back to a browser. To perform this task, it usually refers to back-end infrastructures such as database, cache server, job queue, and others. Besides, at least two servers, connected to the load balancer, manage to process the user's requests.[18]
Databases:
The name of this web application component speaks for itself. The database gives instruments for organizing, adding, searching, updating, deleting, and performing computations. In most cases, web application servers directly interact with the job servers.[18]
Caching Service:
Caching service provides storage for data, which allows storing and searching data. Whenever a user gets some information from the server, the results of this operation goes to cache. So, future requests return faster. In one word, caching allows you to refer to the previous result to make a computation much faster. Therefore, caching is effective when[18]:
* the computation is slow.
* computation is likely to occur many times.
* when the results are the same for a particular request.
Job Queue (optional):
Job queue consists of two components: the job queue itself and servers. These servers process jobs in the queue. It happens that most of the web-servers need to operate a vast amount of jobs that are not of primary importance. Therefore, when a job needs to be fulfilled, it goes to the job queue and is operated due to a schedule.[18]
Full-Text Search Service (optional):
Many web applications support the search by text function or so-called request, and then, an app sends the most relevant results to a user. This technology is named full-text search service. With the help of keywords,

it searches the needed data among a vast number of documents.[18]

Services:

When a web application reaches a specific level, services are created in the form of separate apps. They are not that visible among other web application components, but the web application and other services interact with them.[18]

Data Warehouse:

Almost every modern application implies the work with data, such as collecting, storing and analyzing.[18] These processes require three stages:

- The data is sent to the data "fire-hose", which provides a streaming interface for absorption and processing of data.
- Raw, processed, and additional data is sent to cloud storage.[18]
- And processed and additional data also go to a data warehouse.[18] It's a particular model of online storage and exchange of data through the Internet. The Data Warehouse can be used for storing a variety of files of different types such as videos, photos, or so on.

CDN:

CDN or Content Delivery System deals with sending HTML files, CSS files, JavaScript files, and images. It delivers the content of the end server throughout the world, so people can load various sources.[18]

The scheme of the user-server process can explain the essence of the web application architecture

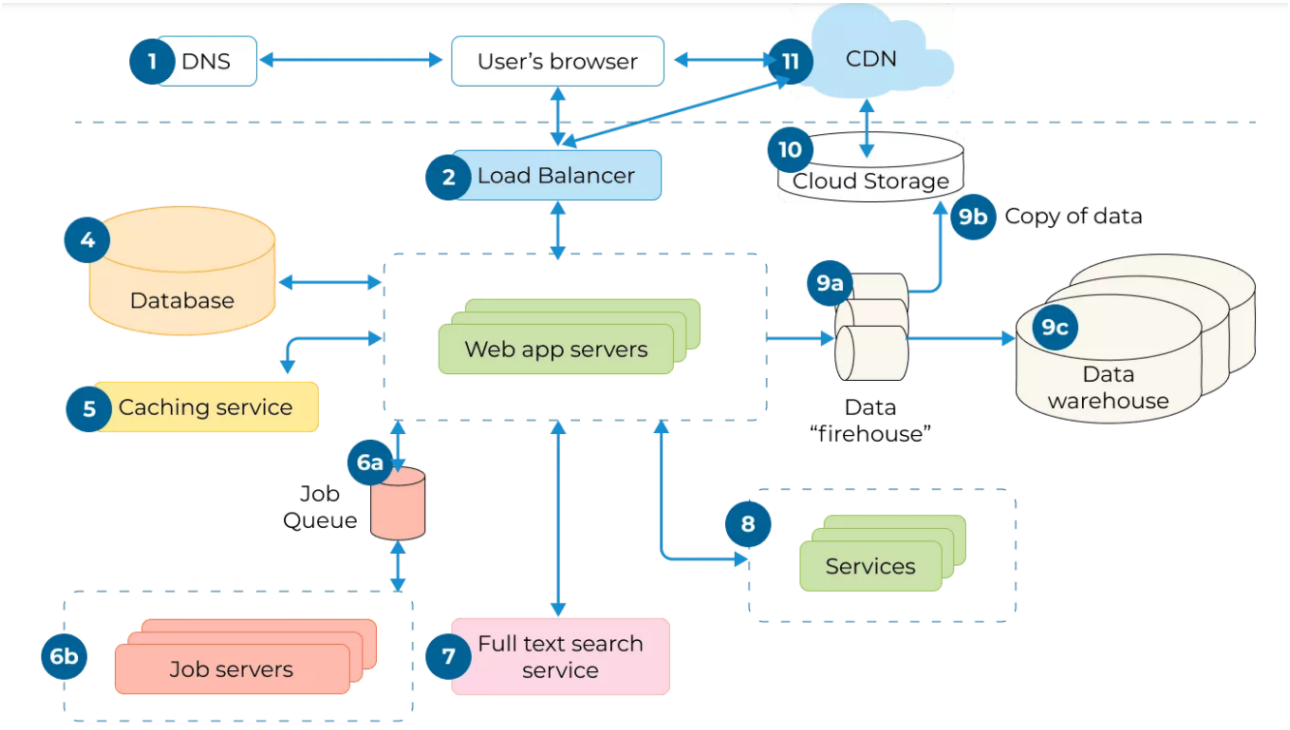

Figure 1.2: Web Application Architecture Diagram.[17]

1.4 Web application terminology

This section defines frequently used terms relating to web applications.

- **Computer security** Computer security is the use of technology, policies, and education to assure the confidentiality, integrity, and availability of data during its storage, processing, and transmission. To secure data, we pursue three activities: prevention, detection, and recovery.[29]

- **Vulnerabilities** A vulnerability is an inherent weakness in the design, configuration, or implementation of a network or system that renders it susceptible to a threat. Most vulnerabilities can usually be traced back to one of three sources: poor design, poor implementation, or poor management.[29]

- **Attacks** An attack is a specific technique used to exploit a vulnerability. For example, a threat could be a denial of service. a vulnerability is in the design of the operating system, and an attack could be a "ping of death". There are two general categories of attacks, passive and active.[29]

- **Threat** A threat is anything that can disrupt the operation, functioning, integrity, or availability of a network or system. There are different categories of threats. There are natural threats, occurrences such as floods, earthquakes, and storms. There are also unintentional threats that are the result of accidents and stupidity. Finally, there are intentional threats that are the result of malicious indent. Each type of threat can be deadly to a network.[29]

- **Security policy** Security policy is an action plan that a public or private organization establishes in order to reduce security risks. This plan usually includes specific plans such as a defence policy as well as other indirect policies, purchasing policies, and personnel selection, and also establishes control measures for the security of the organization. On brief, security policy is a statement of what is, and what is not allowed.[29]

- **Countermeasures** Countermeasures are the techniques or methods used to defend against attacks and to close or compensate for vulnerabilities in networks or systems.[29]

- **An application server :** is software that helps a web server process web pages containing server-side scripts or tags. When such a page is requested from the server, the web server hands the page off to the application server for processing before sending the page to the browser. For more information, see How a web application works.[29]

- **A database** is a collection of data stored in tables. Each row of a table constitutes one record and each column constitutes a field in the record, as shown in the following example.[29]

- **A dynamic page** is a web page customized by an application server before the page is sent to a browser. For more information, see How a web application works.[29]

- **A server technology** is the technology that an application server uses to modify dynamic pages at runtime.
  The Dreamweaver development environment supports the following server technologies:

  1. Macromedia ColdFusion

2. Microsoft ASP.NET

3. Microsoft Active Server Pages (ASP)

4. Sun Java Server Pages (JSP)

5. PHP: Hypertext Preprocessor (PHP)

You can also use the Dreamweaver coding environment to develop pages for any other server technology not listed.

- **A static page** is a web page that is not modified by an application server before the page is sent to a browser. For more information, see Processing static web pages.[29]

- **A web application** is a website that contains pages with partly or entirely undetermined content. The final content of these pages is determined only when a visitor requests a page from the web server. Because the final content of the page varies from request to request based on the visitor's actions, this kind of page is called a dynamic page.[29]

- **A web server** is software that sends out web pages in response to requests from web browsers. A page request is generated when a visitor clicks a link on a web page in the browser, selects a bookmark in the browser, or enters a URL in the browser's address text box.[29]

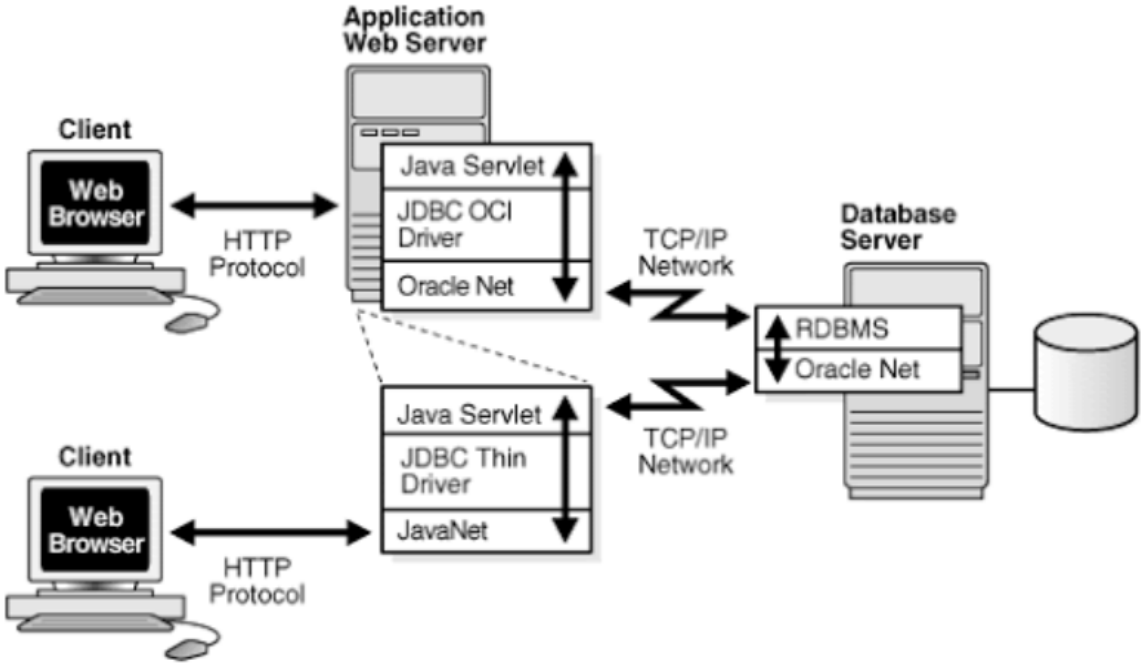

Figure 1.3: Web application terminology[45]

1.5 Why is web application not secure

1.5.1 What is a vulnerability ?

A vulnerability is a weakness or error in a system or device's code that, when exploited, can compromise the confidentiality, availability, and integrity of data stored in them through unauthorized access, elevation of privileges, or denial of service. A code or tool used to take advantage of a vulnerability is called an exploit. Most of the disclosed vulnerabilities are shared on the National Vulnerability Database (NVD) and enumerated in the Common Vulnerabilities and Exposures (CVE) List to make it easier to share data across separate vulnerability capabilities.[30]

1.5.2 Vulnerabilities classification

[42]

1. **Vulnerabilities related to physical domains:**
   - Lack of redundancy and resource at the equipment level.
   - Access to computer rooms not secured.
   - Absence or bad data backup strategy.
2. **Vulnerabilities related to organizational areas:**
   - Lack of: human resources and qualified personnel, communications.
   - Lack of: periodic inspections, procedural documents adapted to the company, means relating to the risks involved.
   - Too much functional complexity.
3. **Vulnerabilities related to technological fields :**
   - Numerous flaws in web services and applications and databases.
   - No operating system and patch updates.
   - Lack of sufficient control over malware.
   - Recurrence of flaws and lack of supervision of events.
   - Complex, unprotected, poorly organized, non-redundant networks.
   - Misuse of messaging.

1.5.3 Types of vulnerabilities

- **Mechanism authentication:**
  Failure to properly authenticate users.[21]

- **Management of the session:**
  Failure to adequately create, store, transmit and protect sensitive session information such as passwords.[21]

- **Permissions, privileges and access control:**
  Failure to enforce permissions and other resource access restrictions, or privilege management issues.[21]

- **Buffer:**
  Buffer overflow, caused by poor buffer management, allowing more information than possible and thus creating a potential code injection into memory.[21]

- **Cross-Site Request Forgery (CSRF):**
  Failure to verify that a web request made by a user is from the user.[21]

- **Cross-Site Scripting (XSS):**
  Failure of a site to properly validate, filter or encode information sent by a user before returning it.[21]

- **Cryptography:**
  use of an insecure encryption algorithm or misuse of an algorithm.[21]

- **Pathways access:**
  Failure to properly validate paths, allowing access to files outside the intended directory or directories.[21]

- **Injection:**
  Failure to validate user data or file uploads, allowing arbitrary code to be executed on the system 20.[21]

- **Configuration:**
  configuration error of an organization's system, allowing it to be used unsafely.[21]

- **Leakage of information:**
  Exposure of system, sensitive or private information.[21]

- **Competitive situation:**
  A system fault, characterized by a different result depending on the order in which components and clients of the system act.[21]

- **Architecture:**
  A design fault that is not caused by a layout or configuration problem.[21]
  thus, the types of vulnerabilities potentially present differ depending on the systems to be tested.[21]

In addition, a test may be restricted to certain types of vulnerabilities. For example, for a web vulnerability test, testing could be restricted to the top ten vulnerabilities identified by a recognized body such as the Open Web Application Security Project (OWASP). [21]

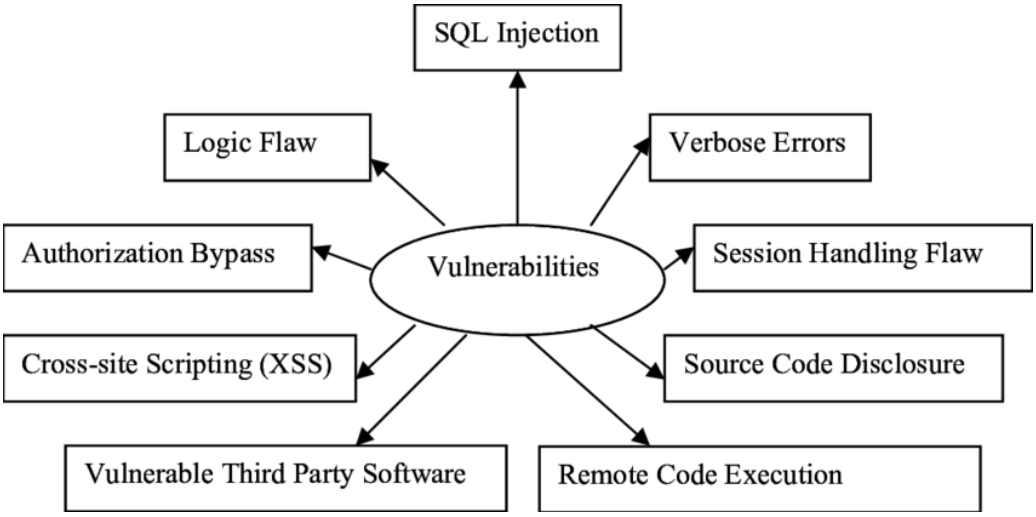

Figure 1.4: Types of vulnerabilities[31]

1.6 Web Attacks

1.6.1 Definition

Web attacks refer to threats that target vulnerabilities in web-based applications. Every time you enter information into a web application, you are initiating a command that generates a response. For example, if you are sending money to someone using an online banking application, the data you enter instructs the application to go into your account, take money out, and send it to someone else's account. Attackers work within the frameworks of these kinds of requests and use them to their advantage.

Some common web attacks include SQL injection and cross-site scripting (XSS), which will be discussed later in this article. Hackers also use cross-site request forgery (CSRF) attacks and parameter tampering. In a CSRF attack, the victim is fooled into performing an action that benefits the attacker. For example, they may click on something that launches a script designed to change the login credentials to access a web application. The hacker, armed with the new login credentials, can then log in as if they are the legitimate user.

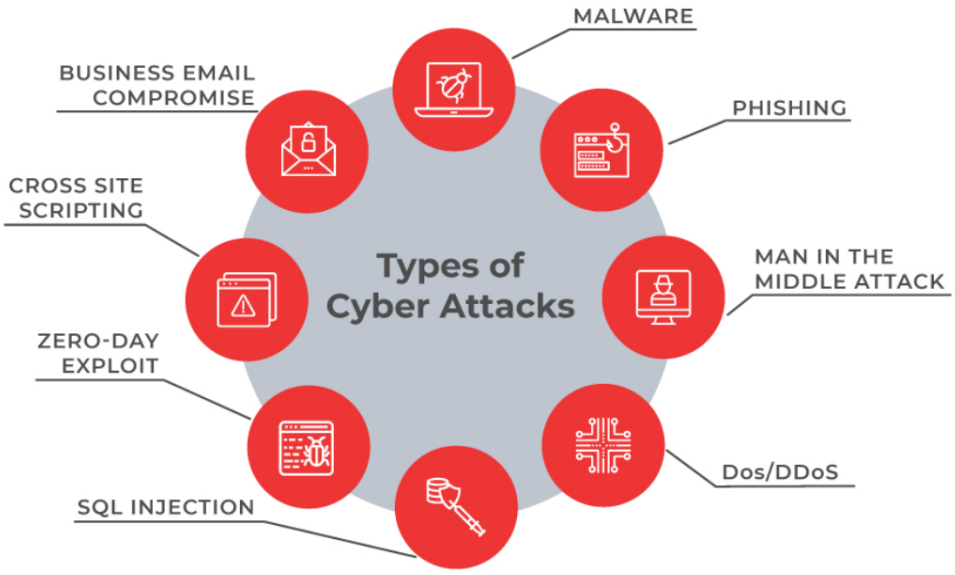

Figure 1.5: web Attacks[13]

1.6.2 Malware

A malware is a type of cyberattack where malicious software is installed on the victim's systems through executable files, usually without the user's knowledge. Malware includes malicious software, including spyware, ransomware, viruses, and worms. After installation, a malware can keep track of the user's activity or can trigger codes resulting into access to sensitive information, login details, credit cards or intellectual properties by the hacker.[13]

1.6.3 Phishing

Phishing refers to spoofing or deceptive communications activities performed by the attackers that appear to originate from a credible source such as emails, messages, legitimate websites that are disguised. Through phishing, attackers try to fetch sensitive information, user details, credit card numbers or make fraudulent attempts.[13]

1.6.4 Man-in-the-middle attack

These attacks happen with relaying or altering the communication channels. This can be communication between organisations and cloud server or over unsecured networks.[13]

1.6.5 DoS/DDoS

A DoS/DDoS attack aims at flooding the target website with overwhelming traffic to exhaust resources and bandwidth of the system. These are not to bring down a website, but to breach a security perimeter and smoke out the online systems. This can reduce a user base or may bring down the entire network.[13]

1.6.6 SQL Injection

This is injecting a nefarious code or statements into SQL queries or a database server to extract information from the database or to take a data dump of the complete database.[13]

1.6.7 Zero-day exploit

Zero-day is a software security flaw which is known to the software developers. Attackers try to exploit a vulnerability before a patch or solution is implemented to capture the system with known weaknesses.[13]

1.6.8 Cross Site Scripting

XSS attacks occur when a web app sends malicious code in the form of a side script to another user, thus bypassing access controls of the site as the same as the origin.[13]

1.6.9 Business Email compromise

This is an attack to spoof business emails and gain illegal access to company accounts and IDs to defraud the company or its employees.[13]

1.7 Security mechanisms and approaches for securing web applications

1.7.1 Security mechanisms

1. **Encryption**

   Cryptography is a mathematical science in which methods for transmitting data confidentially are studied. In order to protect a message, a transformation is applied to it that makes it incomprehensible; this is called encryption, which, from a clear text, gives a cipher or cryptogram. Conversely, decryption is the action that allows the reconstruction of the plain text from the cipher text. In modern cryptography, the transformations in question are mathematical functions, called cryptography algorithms, which depend on a parameter called a key . [14]

   In networks, to prevent information theft in the transmission path, cryptography techniques are used to encrypt and decrypt transmitted messages. There are currently two main encryption principles: symmetric encryption based on the use of a private key and asymmetric encryption based on two keys, one private and one public. encryption, which is based on two keys, one private and the other public.[14]

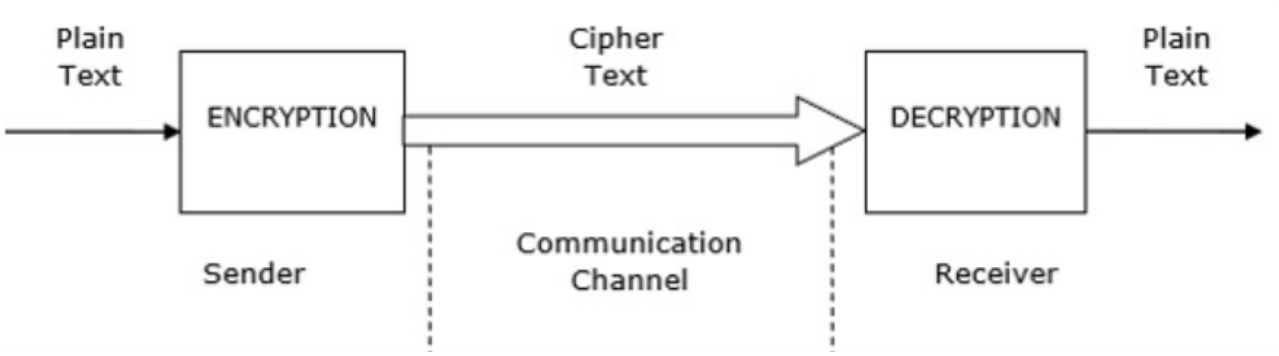

Figure 1.6: the operation of encryption[41]

2. **Firewall**

   A firewall is a hardware or software solution implemented within the network infrastructure to filter access to defined network resources. It allows only authorised users with a key or badge to enter and creates a protective layer between the network and the outside world. It has built-in filters that can prevent unauthorised or potentially dangerous material from entering the system. It also records attempted intrusions in a log that is sent to network administrators. It can also control access to applications and to prevent misuse [5].

   The firewall allows all or some of the packets they are allowed to pass through, and to block and log exchanges that are prohibited.

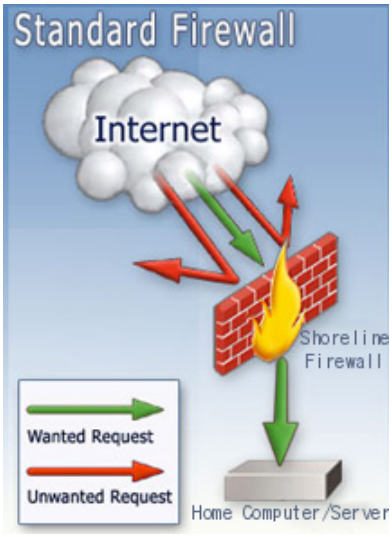

Figure 1.7: The firewall[33]

The firewall is an IDS, but it only detects attacks from outside. For Intranet, firewalls are necessary, but not sufficient, to start implementing a security policy.

Some firewalls only allow email to pass through. In this way, they prohibit any attack other than an attack based on the mail service. Other firewalls are less strict and only block services that are known to be dangerous. Typically, firewalls are configured to protect against unauthenticated access from the external network.

3. **antivirus**
   Antivirus is a kind of software used to prevent, scan, detect and delete viruses from a computer. Once installed, most antivirus software runs automatically in the background to provide real-time protection against virus attacks.Comprehensive virus protection programs help protect your files and hardware from malware such as worms, Trojan horses and spyware, and may also offer additional protection such as customizable firewalls and website blocking.[44]

4. **Intrusion detection and prevention**

   An intrusion detection system (IDS) is used to constantly monitor the network. It analyses the flow of data packets through the network, looking for unauthorised activity (such as hacker attacks) and allows users to address security breaches before systems are compromised.

   Today, IDS systems are evolving into so-called intrusion prevention systems (IPS) which, in addition to detection, provide active protection. An IPS system can decide, following alerts, to close ports and reject packets according to the parameters that have been set. [32]

1.7.2 Security approaches

1. **Signature approach**
   The signature approach (Misuse Detection) could be very just like the strategies utilized by antivirus, the not unusualplace precept of all strategies on this elegance is to apply a database, containing specs of assault signatures. The intrusion detector compares the determined behaviour of the machine to this database and increases an alert if this behaviour fits a predefined signature. "Thus, the entirety that isn't explicitly described is allowed" and the entirety this is explicitly described is prohibited.[12]

2. **Behavioural approach**
   Behavioural intrusion detection (Anomaly Detection) became the primary method proposed and developed. Anderson proposes to discover violations of the device's protection coverage through staring at the behaviour of customers and evaluating it to a version of behaviour taken into consideration everyday, known as a profile. In general, the behavioural method has phases: a studying segment wherein the profile is constructed through staring at the behaviour of the monitored entity and a detection segment wherein the IDS examine the behaviour of the entity, measures the similarity among the latter and the profile and problems an alert if the deviation is just too large. The most important concept of this method is to do not forget any deviation, any anomaly withinside the behaviour as an intrusion. This assumption is sincerely wrong: uncommon activities or behaviours can be valid from the factor of view of the device's protection coverage. The device is probably to emit fake positives. As lengthy because the quantity of fake positives stays low enough, the technique may be valid. This results in critical questions withinside the subject of behavioural intrusion detection, at the correctness and completeness of the of everyday behaviour .[12]

3. **Hybrid approach**
   The hybrid technique is an technique that mixes each approaches (the behavioural technique and the signature technique). First the behavioural technique seems for feasible intrusions after which those are surpassed to the signature technique for updating its database. surpassed to the signature technique for updating its database.[12]

1.8 Conclusion

Given the empowering nature of web applications, it is clear that securing web applications is important. Specifically, to focus the needs of the users: making sure that their data is safe, and that they are safe while browsing the web. To accomplish this, that we make strides to create automated tools that are able to automatically find security vulnerabilities. These tools can be used by developers with no security expertise, thus putting developers on a level playing field with the attackers.

Chapter 2

Vulnerabilities detection methods

2.1 Introduction

In this chapter, we talk about what vulnerabilities and top ten vulnerabilities first detection methods are, which structure they usually have and, then we give a Web Application Architecture Diagram, we continue by web application terminology, then we talk about types vulnerabilities and the last thing about security mechanisms and approaches for securing web application .

2.2 The Open Web Application Security Project (OWASP)

The Open Web Application Security Project (OWASP) is a non-profit foundation dedicated to improving software security. It operates under an "open community" model, which means that anyone can participate in and contribute to OWASP-related online chats, projects, and more. For everything from online tools and videos to forums and events, the OWASP ensures that its offerings remain free and easily accessible through its website. The OWASP Top 10 provides rankings of—and remediation guidance for—the top 10 most critical web application security risks. Leveraging the extensive knowledge and experience of the OWASP's open community contributors, the report is based on a consensus among security experts from around the world. Risks are ranked according to the frequency of discovered security defects, the severity of the uncovered vulnerabilities, and the magnitude of their potential impacts. The purpose of the report is to offer developers and web application security professionals insight into the most prevalent security risks so that they may fold the report's findings and recommendations into their own security practices, thereby minimizing the presence of known risks in their applications.[16]

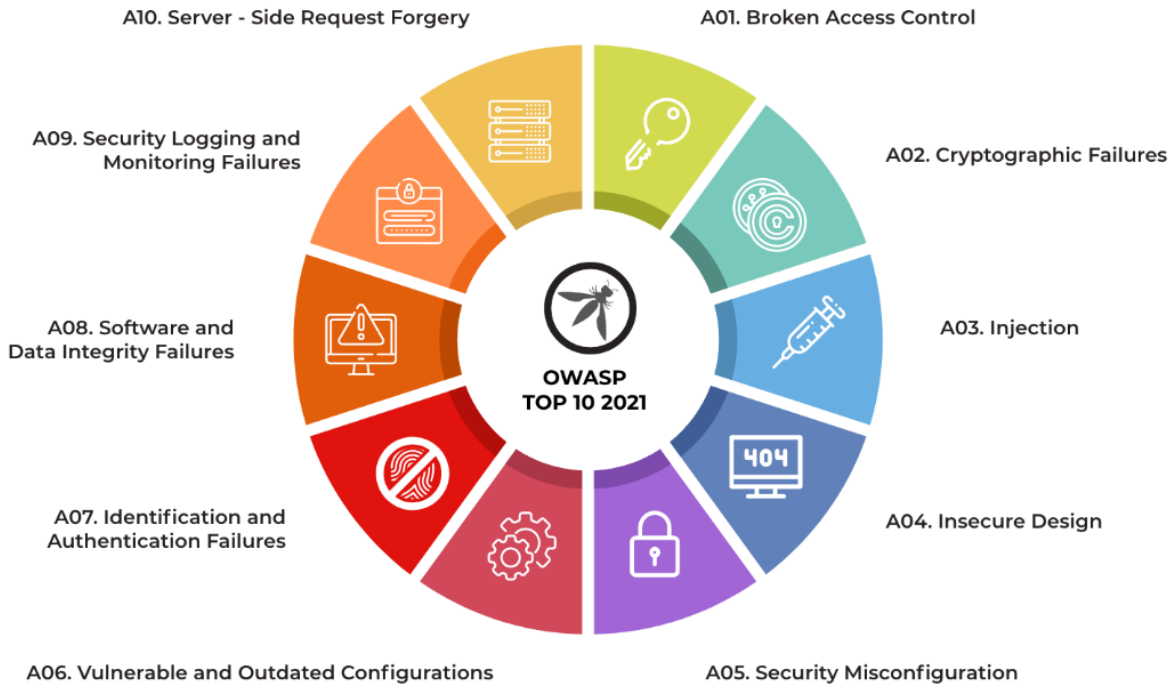

Figure 2.1: top 10 vulnerabilities [11]

2.3 Top 10 vulnerabilities

2.3.1 Broken access control

Access control imposes a policy such that users cannot act outside their intended permissions. Failures usually lead to unauthorized disclosure, modification or destruction of information from all data, or performing a business function outside the user's boundaries. [19]

Objectives:

* Bypass access controls by changing the URL, internal application state, or HTML page, or simply by using a custom API attack tool.

* Elevation of privilege.

Parades:

* Establish access control mechanisms.

* Disable the web server directory list and ensure that file metadata (e.g. .git) and backup files are not present in the web roots.[19]

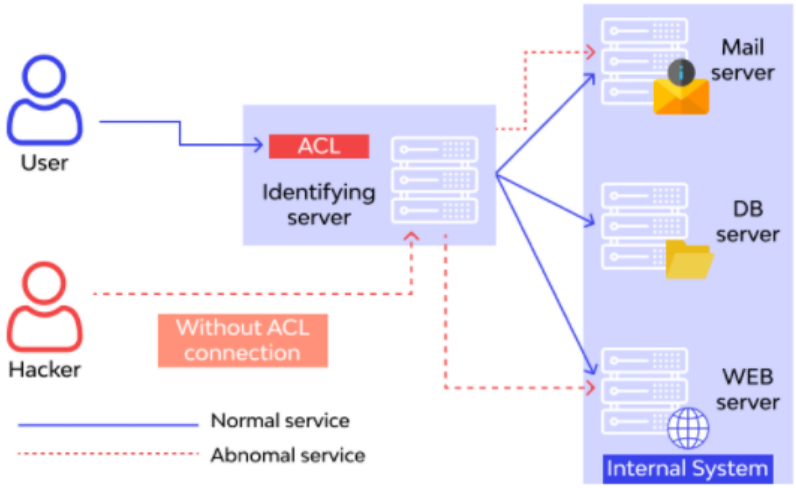

Figure 2.2: Broken access control [19]

2.3.2 Cryptographic failures

This flaw includes all vulnerabilities related to the protection of sensitive data.[19] **Objectives:**

* Access confidential data.
* Identity theft.

Parades:

* Use strong encryption algorithms.
* Do not store unnecessary information.
* Use TLS throughout the chain.
* Decryption keys must be stored separately from the data.

```php
<?php
$MotDePasse = $_POST['password'];
$GrainDeSel = rand();
// Association du mot de pass et d'un chiffre
// aléatoire qui sera stocké en clair en base
$nvPass     = $GrainDeSel.$MotDePasse
// encodage du mot de passe
$hash = sha1(md5($nvPass).md5($nvPass));
?>
```

Figure 2.3: Grain of salt technique [19]

2.3.3 Injection

An injection attack is a malicious code injected in the network which fetched all the information from the database to the attacker.[19]

Objectives:

* reading, deletion, alteration of data.

* access to the system without authentication.

Parades:

* Verification of the data entered (blacklists, whitelist, regular expressions...).

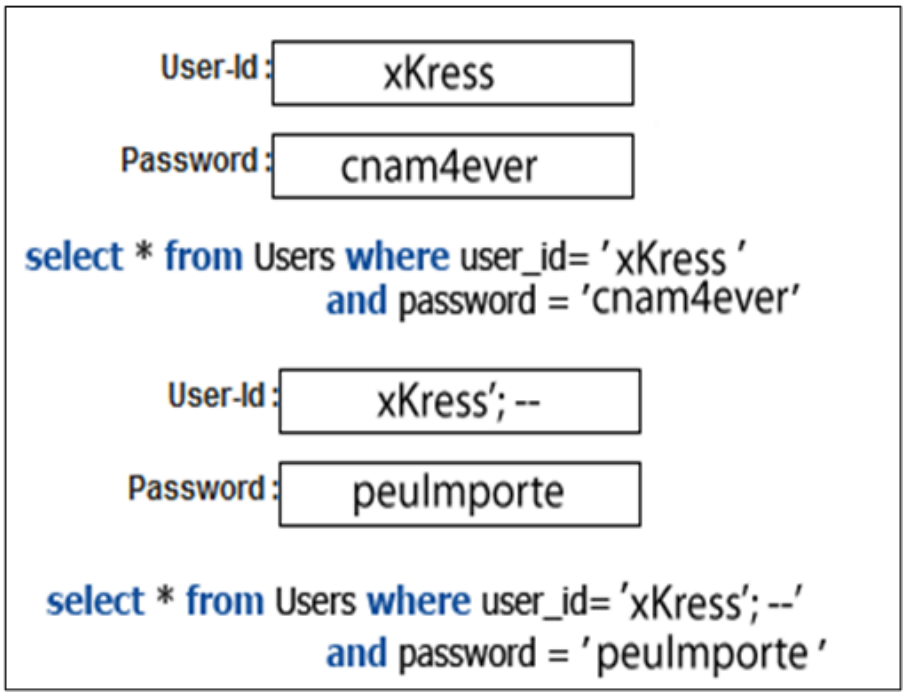

Figure 2.4: Injection [19]

2.3.4 Insecure design

A new category for 2021 focuses on risks related to design and architecture flaws, with a call for increased use of threat modeling, secure design patterns, and reference architectures. [19]

Parades:

* Establish and use a secure development life cycle with professionals (AppSec) to help assess and design security and privacy controls.

* Establish and use a library of secure design templates.

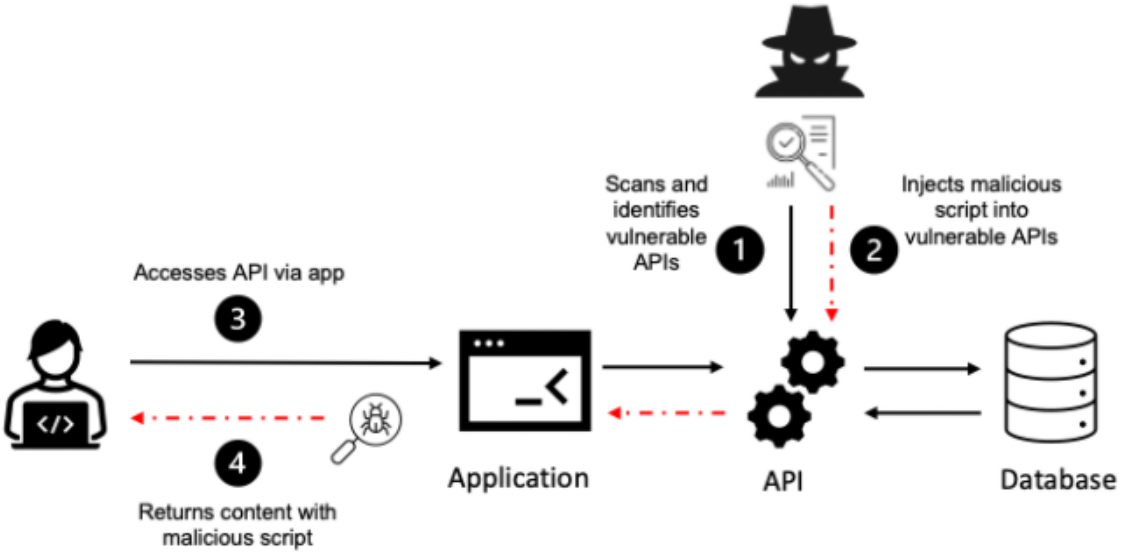

Figure 2.5: insecure design [19]

2.3.5 Security misconfiguration

This flaw includes all vulnerabilities related to configuration problems, on all elements of the application layer (servers, language, framework, components ...).[19]

Objectives:

* Access confidential information.
* Take control of a server.

Parades:

* Do not assign components more rights than necessary.
* Rigorously study the configuration.

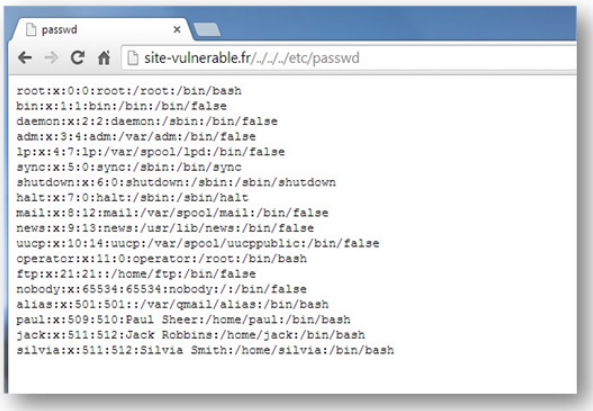

Figure 2.6: misconfiguration [19]

2.3.6 Vulnerable and Outdated components

This flaw concerns all components used for the operation of an application. They may have vulnerabilities (version, compatibility of libraries...) and must be the subject of special attention.[19] **Objectives:**
* Access confidential data.

* Take control of a server.

Parades:
* Maintain a list of the components used and their version.
* Update these components as soon as a vulnerability is detected and patched.

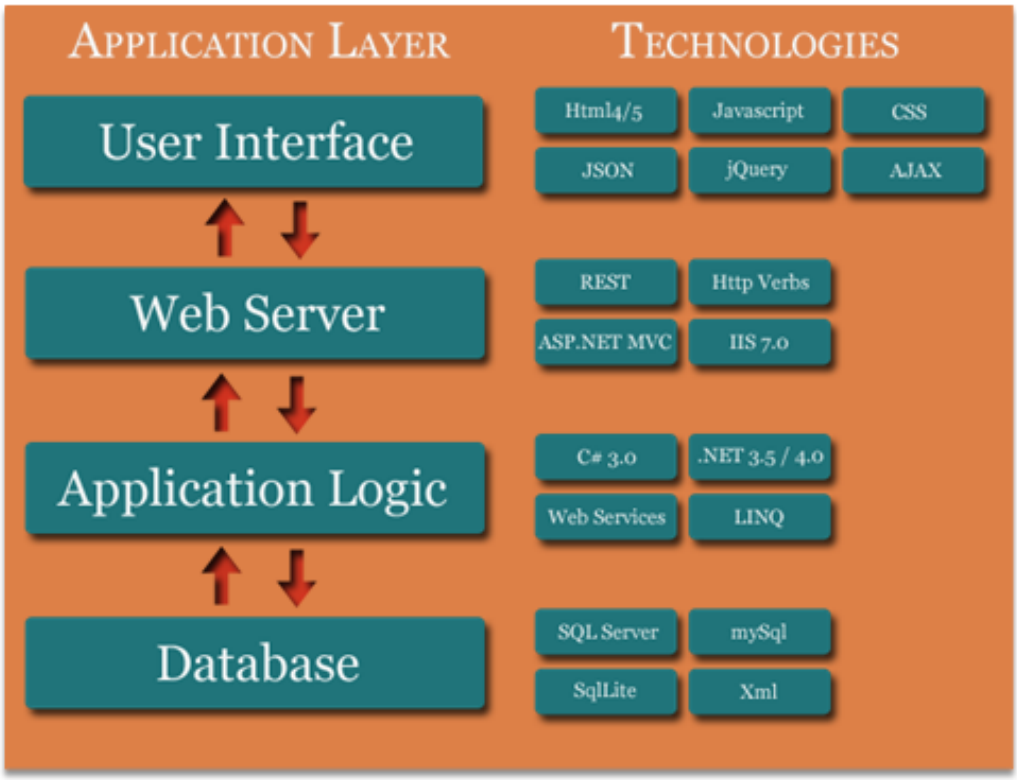

Figure 2.7: Vulnerable and Outdated components [19]

2.3.7 Identification and authentication failures

Objectives:
* Access features reserved for certain people.
* Access confidential data.

Parades:
* Require strong passwords.
* Use a captcha system.
* Use cookies to manage sessions.
* Set a maximum session duration.

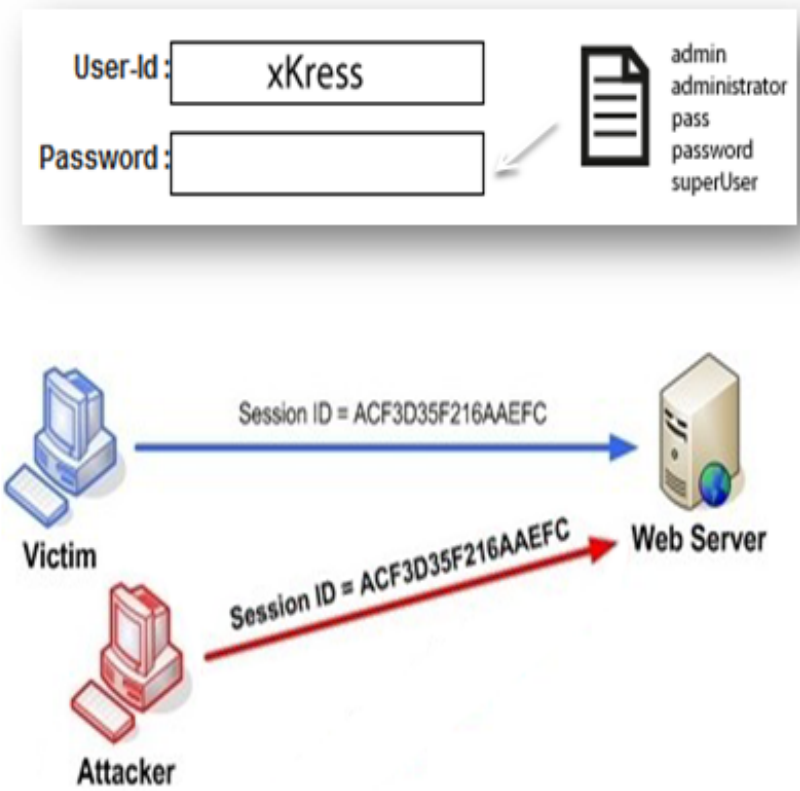

Figure 2.8: Authentication Management Violation and Session Theft [19]

2.3.8 Software and data integrity failures(XSS and insecure deserialization)

An example of this vulnerability is when an application relies on plugins, libraries, or modules from not trusted sources, repositories, and content delivery networks. Thus, many apps now include an automatic update feature, where updates are downloaded without sufficient integrity checks and applied to the previously approved app. Another example is where objects or data are encoded or serialized in a structure that an attacker can see and modify is vulnerable to insecure deserialization.[19]
Parades:

* Use digital signatures or similar mechanisms
* Ensure that unsigned or unencrypted serialized data is not sent to untrusted clients.
* Ensure that there is a process for reviewing code and configuration changes.

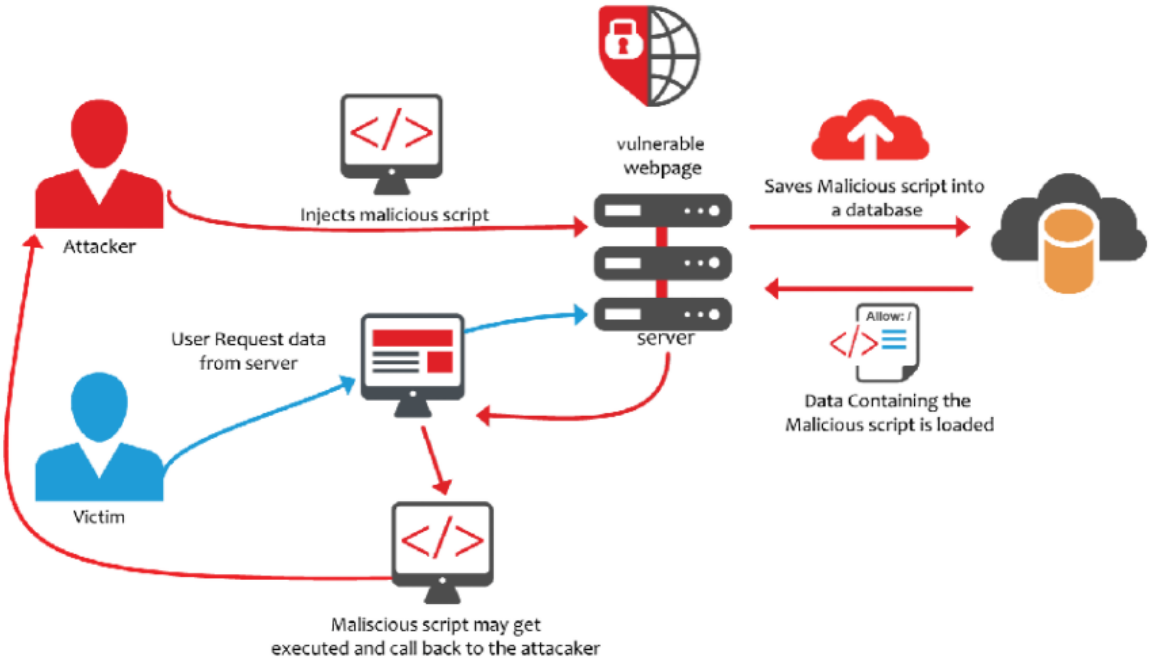

Figure 2.9: Cross Site Scripting [19]

2.3.9 Security Logging and Monitoring Failures

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show that the time to detect a breach exceeds 200 days, usually detected by external parties rather than by internal or monitoring processes.[19]

Parades:

* Performing penetration tests and security auditing.
* Ensure that log data is properly encoded to prevent injections or attacks on logging or monitoring systems.

2.3.10 Server-side request forgery (SSRF)

SSRF vulnerabilities occur whenever a Web application retrieves a remote resource without validating the URL provided by the user. It allows an attacker to force the application to send a specially crafted request to an unexpected destination, even when it is protected by a firewall, VPN, or other type of network access control list (ACL).[19]

Parades:

* Disable HTTP redirects
* Validate all input data provided by the customer.

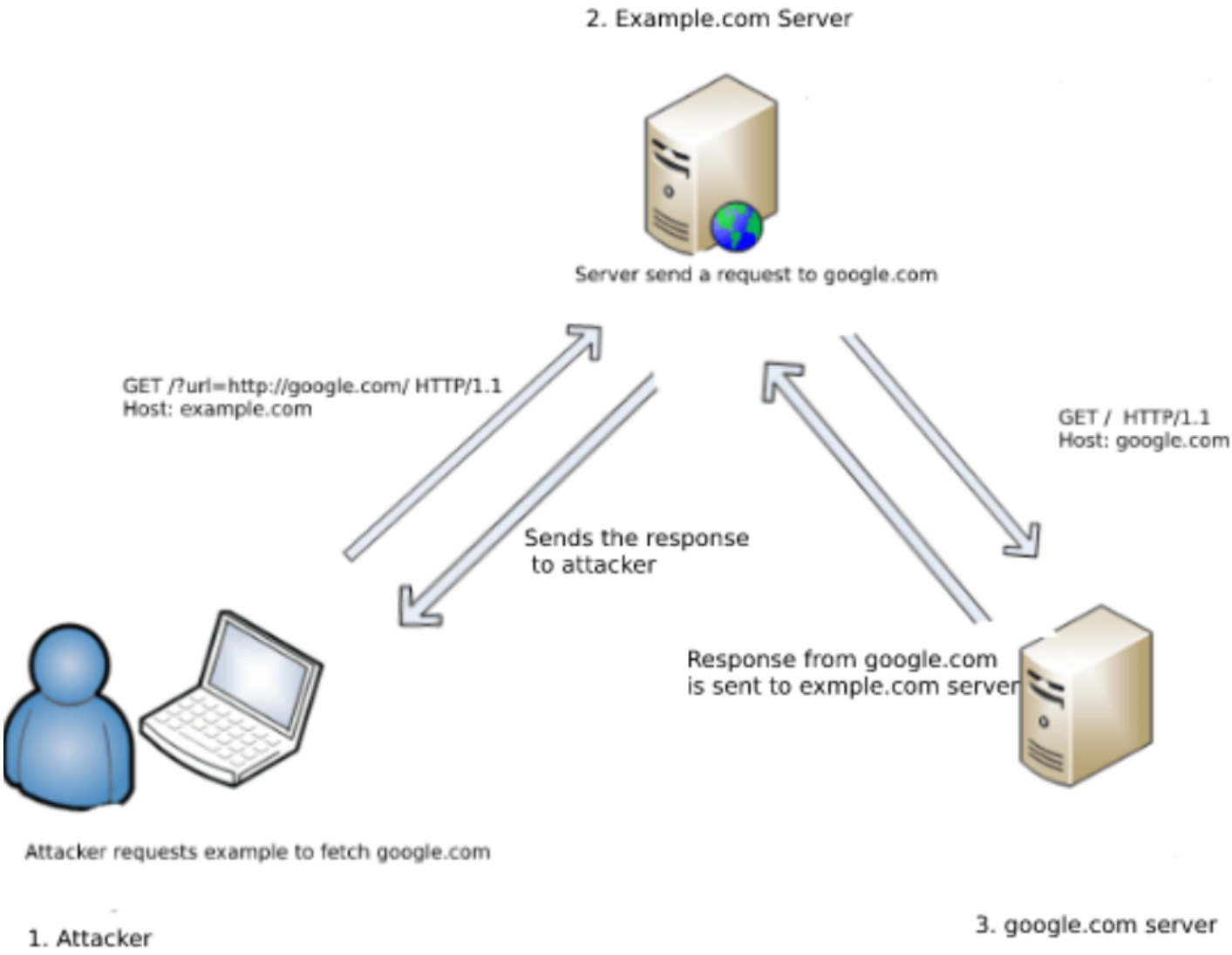

Figure 2.10: Server-side request forgery [19]

2.4 National vulnerability database (NVD)

The NVD is the U.S. government repository of standards based vulnerability management data, represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security related software flaws, misconfigurations, product names, and impact metrics.[34]
The NVD performs analysis on **Common Vulnerabilities and Exposures (CVEs)** that have been published to the CVE Dictionary. NVD staff are tasked with analysis of CVEs by aggregating data points from the description, references supplied and any supplemental data that can be found publicly at the time. This analysis results in association impact metrics **Common Vulnerability Scoring System (CVSS)**, vulnerability types (Common Weakness Enumeration - CWE), and applicability statements (Common Platform Enumeration - CPE), as well as other pertinent metadata. The NVD does not actively perform vulnerability testing, relying on vendors, third party security researchers and vulnerability coordinators to provide information that is then used to assign these attributes. As additional information becomes available, CVSS score , CWEs, and applicability statements are subject to change. The NVD endeavors to re-analyze CVEs that have been amended as time and resources allow, ensuring that the information offered is up-to-date.[34]

2.4.1 A Brief History of the NVD

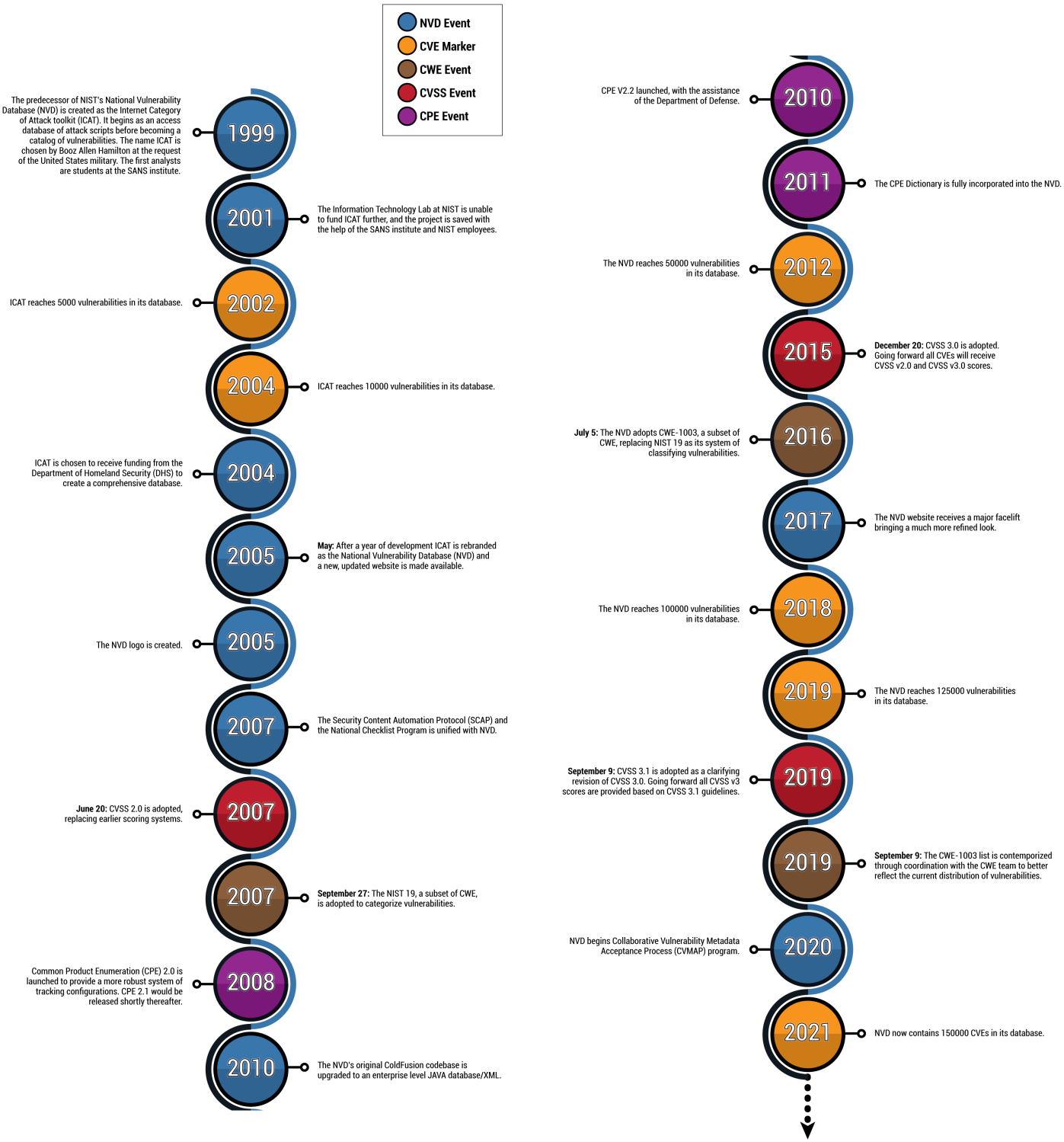

Figure 2.11: A Brief History of the NVD [34]

2.4.2 CVEs and the NVD Process

An Introduction

The Common Vulnerabilities and Exposures (CVE) program is a dictionary or glossary of vulnerabilities that have been identified for specific code bases, such as software applications or open libraries. This list allows

interested parties to acquire the details of vulnerabilities by referring to a unique identifier known as the CVE ID. It has garnered increasing awareness in recent years, making it important for participants and users to understand the fundamental elements of the program.[34]

The CVE Assignment and Vetting Process

CVE IDs are assigned by the CVE Assignment Team and CNAs. The diversity of CNAs provides varied yet specific areas of expertise for different types of vulnerabilities. Each CNA is given a realistic number of possible candidates based on their scope and ability to timely vet each one. Regular training and retraining of CNA staff and the establishment of a hierarchy of CNAs to govern various authorities help ensure that the guidelines for the process are strictly followed and that standards are being met.[34]

CNAs use a policy known as the Counting Process in addition to an inclusion decision tree to determine if an individual vulnerability should be included in the CVE list and if more than one CVE ID needs to be assigned. This process begins when a reporter (typically the original individual or organization(s) that discovered the bug) contacts the CVE Assignment Team or an appropriate CNA to request a CVE ID.[34]

Each CVE must include a description that is either provided by the reporter or created using the CVE Assignment Team's optional template. This description includes the type of vulnerability (e.g., a buffer overflow, NULL pointer dereference, or cross-site request forgery), the product's vendor, and the affected code base(s). Reporters can provide further information, such as the expected impact, attack vectors, or state of remediation. Once the vetting process is completed, a CVE ID is assigned.[34]

RESERVED tags are used when CVE IDs have been assigned or potentially assigned to vulnerabilities which need further details before they can be finalized. Should the vulnerability be unsuitable for publication, it will be denied a CVE ID and tagged REJECTED by the CNA. This may occur due to a lack of qualifying factors, irregularities in the reporting process, or a request to be withdrawn by the original reporter.[34]

A CVE ID also may be given a DISPUTED tag should the vendor or other authoritative entity challenge the validity of the vulnerability. This can occur before or after the National Vulnerability Database publishes their analysis (see below).[34]

NVD CVE Analysis

The National Vulnerability Database (NVD) is tasked with analyzing each CVE once it has been published to the CVE List, after which it is typically available in the NVD within an hour. Once a CVE is in the NVD, analysts can begin the analysis process. The processing time can vary depending on the CVE, the information available, and the quantity of CVEs published within a given timeframe. NVD analysts use the reference information provided with the CVE and any publicly available information at the time of analysis to associate Reference Tags, Common Vulnerability Scoring System (CVSS) v2.0, CVSS v3.1, CWE, and CPE Applicability statements.[34]

The following is a general overview of the analysis process for a given CVE:

1. An analyst reviews any reference material provided with the CVE record and assigns appropriate reference tags. This helps organize the various data sources to help researchers find the relevant information for their needs. The analyst also performs manual searches of the internet to ensure that any other available and relevant information is used for the analysis process. NVD analysts only use publicly available materials in the analysis process.

2. A common weakness enumeration (CWE) identifier is assigned that categorizes the vulnerability. NVD analysts use a subset of the full list of CWEs that best represents the distribution of specific types of vulnerabilities. This subset is known as the CWE-1003 view and was created through coordination with the MITRE CWE team.

3. CVSS V2.0 exploitability and impact metrics are assigned based on publicly available information and the guidelines of the specification.

4. CVSS V3.1 exploitability and impact metrics are assigned based on publicly available information and

the guidelines of the specification.

5. A Common Product Enumerator (CPE) Applicability Statement is associated with the vulnerability. The CPE match criteria identifies all potentially vulnerable software and/or hardware for the vulnerability. For example, an application may have several versions affected or must be running on a specific operating system to be vulnerable. Automated processes can reference match criteria within the applicability statements against the CPE dictionary to assist in identifying vulnerable products within an organization's information system.

6. Analysis results are given a quality assurance check by another, more senior analyst prior to being published to the website and data feeds.

CVE Maintenance Once a CVE is published and NVD analysis is provided, there may be additional maintenance or modifications made. References may be added, descriptions may be updated, or a request may be made to have a set of CVE IDs reorganized (such as one CVE ID being split into several). Furthermore, the validity of an individual CVE ID can be disputed by the vendor. The NVD does make efforts to reanalyze CVEs that have been changed after previous analysis. The NVD always appreciates and encourages feedback from the community to keep the database and CPE dictionary accurate and current.[34]

2.5 Vulnerability detection using machine learning

The first step in the study of computer algorithms and the use of mathematical methods in the field of computer science, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on a sample data, known as "training data", Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine lea mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytic. For example: Kernel machines are used to compute non-linearly separable functions into a higher dimension linearly separable function. [26]

2.5.1 Machine learning tasks

At a high-level, machine learning is simply the study of teaching a computer program or algorithm how to progressively improve upon a set task that it is given. On the research-side of things, machine learning can be viewed through the lens of theoretical and mathematical modeling of how this process works. However, more practically, it is the study of how to build applications that exhibit this iterative improvement. There are many ways to frame this idea, but largely there are three major recognized categories: supervised learning, unsupervised learning, and reinforcement learning.[43]

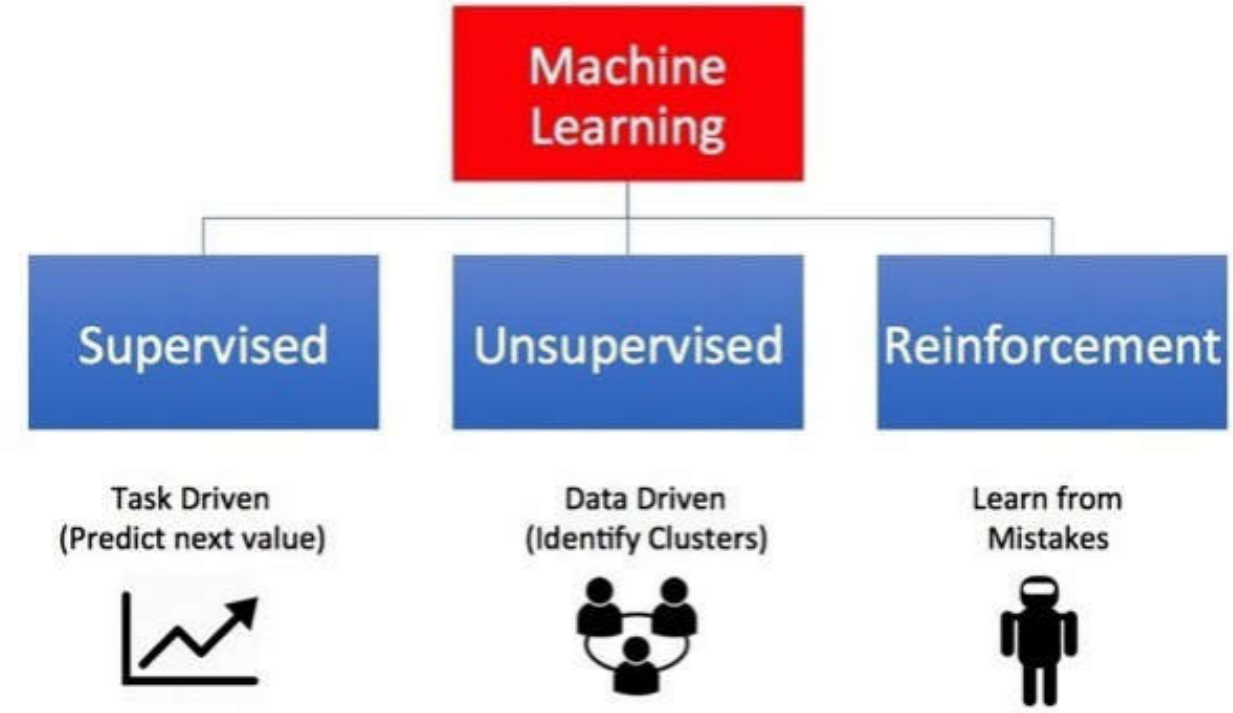

Figure 2.12: Machine learning tasks[24]

I. **Supervised Learning**

Supervised learning is the most popular paradigm for machine learning. It is the easiest to understand and the simplest to implement. Given data in the form of examples with labels, we can feed a learning algorithm these example-label pairs one by one, allowing the algorithm to predict the label for each example, and giving it feedback as to whether it predicted the right answer or not. Over time, the algorithm will learn to approximate the exact nature of the relationship between examples and their labels. When fully-trained, the supervised learning algorithm will be able to observe a new, never before seen example and predict a good label for it.[43]

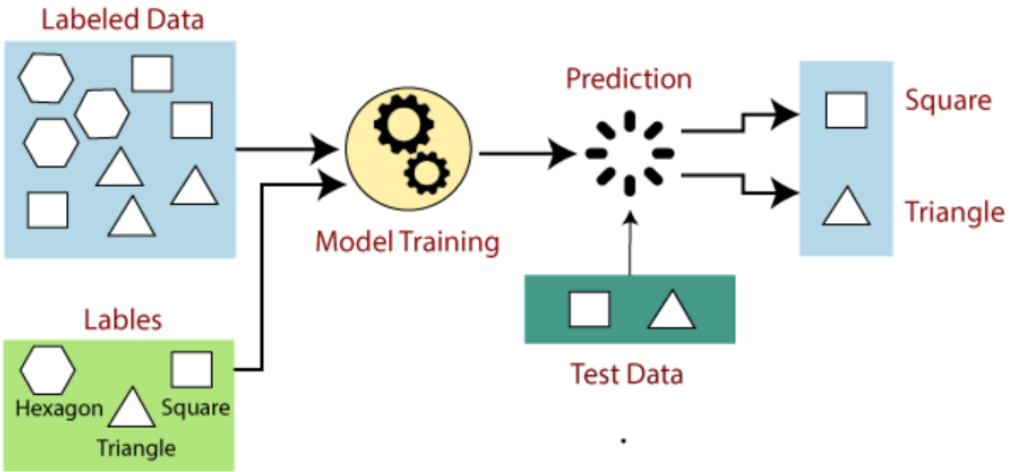

Figure 2.13: Supervised Learning[3]

Supervised learning is often described as task-oriented because of this. It is highly focused on a singular

task, feeding more and more examples to the algorithm until it can accurately perform on that task. This is the learning type that you will most likely encounter, as it is exhibited in many of the following common applications:

• **Advertisement Popularity** Selecting advertisements that will perform well is often a supervised learning task. Many of the ads you see as you browse the internet are placed there because a learning algorithm said that they were of reasonable popularity (and click ability). Furthermore, its placement associated on a certain site or with a certain query (if you find yourself using a search engine) is largely due to a learned algorithm saying that the matching between ad and placement will be effective.[43]

• **Spam Classification** If you use a modern email system, chances are you've encountered a spam filter. That spam filter is a supervised learning system. Fed email examples and labels (spam/not spam), these systems learn how to preemptively filter out malicious emails so that their user is not harassed by them. Many of these also behave in such a way that a user can provide new labels to the system, and it can learn user preference.[43]

• **Face Recognition** Do you use Facebook? Most likely, your face has been used in a supervised learning algorithm that is trained to recognize your face. Having a system that takes a photo, finds faces, and guesses who that is in the photo (suggesting a tag) is a supervised process. It has multiple layers to it, finding faces and then identifying them, but is still supervised nonetheless. [23]

II. **Unsupervised Learning**

Unsupervised learning is very much the opposite of supervised learning. It features no labels. Instead, our algorithm would be fed a lot of data and given the tools to understand the properties of the data. From there, it can learn to group, cluster, and/or organize the data in a way such that a human (or other intelligent algorithm) can come in and make sense of the newly organized data.[43]

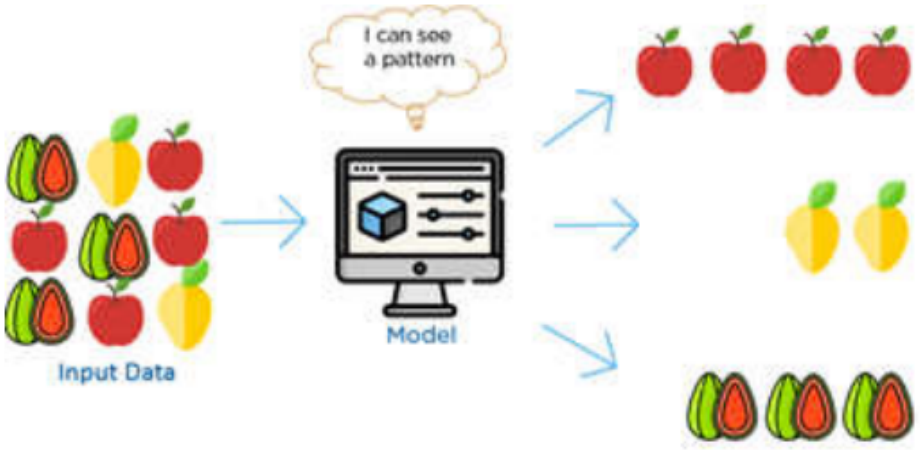

Figure 2.14: unsupervised Learning[10]

What makes unsupervised learning such an interesting area is that an overwhelming majority of data in this world is unlabeled. Having intelligent algorithms that can take our terabytes and terabytes of unlabeled data and make sense of it is a huge source of potential profit for many industries. That alone could help boost productivity in a number of fields.[43]

III. **Reinforcement Learning**

Reinforcement learning is fairly different when compared to supervised and unsupervised learning. Where we can easily see the relationship between supervised and unsupervised (the presence or absence of labels), the relationship to reinforcement learning is a bit murkier. Some people try to tie reinforcement learning closer to the two by describing it as a type of learning that relies on a time-dependent sequence of labels, however, my opinion is that simply makes things more confusing.I prefer to look at reinforcement learning

as learning from mistakes. Place a reinforcement learning algorithm into any environment, and it will make a lot of mistakes in the beginning. So long as we provide some sort of signal to the algorithm that associates good behaviors with a positive signal and bad behaviors with a negative one, we can reinforce our algorithm to prefer good behaviors over bad ones. Over time, our learning algorithm learns to make less mistakes than it used to.[43]

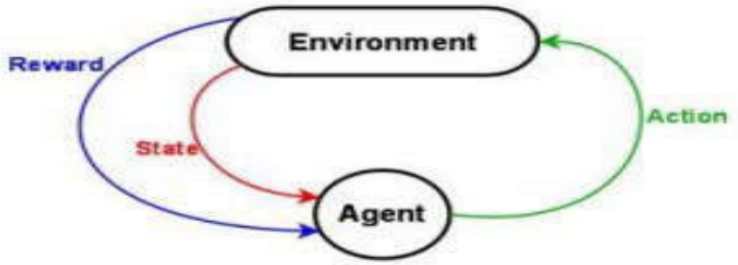

Figure 2.15: unsupervised Learning[10]

Reinforcement learning is very behavior driven. It has influences from the fields of neuroscience and psychology. If you've heard of Pavlov's dog, then you may already be familiar with the idea of reinforcing an agent, albeit a biological one. However, to truly understand reinforcement learning. For any reinforcement learning problem, we need an agent and an environment, as well as a way to connect the two through a feedback loop. To connect the agent to the environment, we give it a set of actions that it can take that affect the environment. To connect the environment to the agent, we have it continually issue two signals to the agent: an updated state and a reward (our reinforcement signal for behavior).[43]

2.5.2 Machine learning algorithms

A Machine Learning algorithm is an evolution of the regular algorithm. It makes your programs "smarter", by allowing them to automatically learn from the data you provide. The algorithm is mainly divided into:
Training Phase : You take a randomly selected specimen of apples from the market (training data), make a table of all the physical characteristics of each apple, like color, size, shape, grown in which part of the country, sold by which vendor, etc (features), along with the sweetness, juiciness, ripeness of that apple (output variables). You feed this data to the machine learning algorithm (classification/regression), and it learns a model of the correlation between an average apple's physical characteristics, and its quality.[43]
Testing Phase : Next time when you go shopping, you will measure the characteristics of the apples which you are purchasing (test data)and feed it to the Machine Learning algorithm. It will use the model which was computed earlier to predict if the apples are sweet, ripe and/or juicy. The algorithm may internally use the rules, similar to the one you manually wrote earlier (for eg, a decision tree). Finally, you can now shop for apples with great confidence, without worrying about the details of how to choose the best apples.[43]

2.5.3 types of machine learning algorithms

So, it can be categorized by the following three types. As the following figure shows.

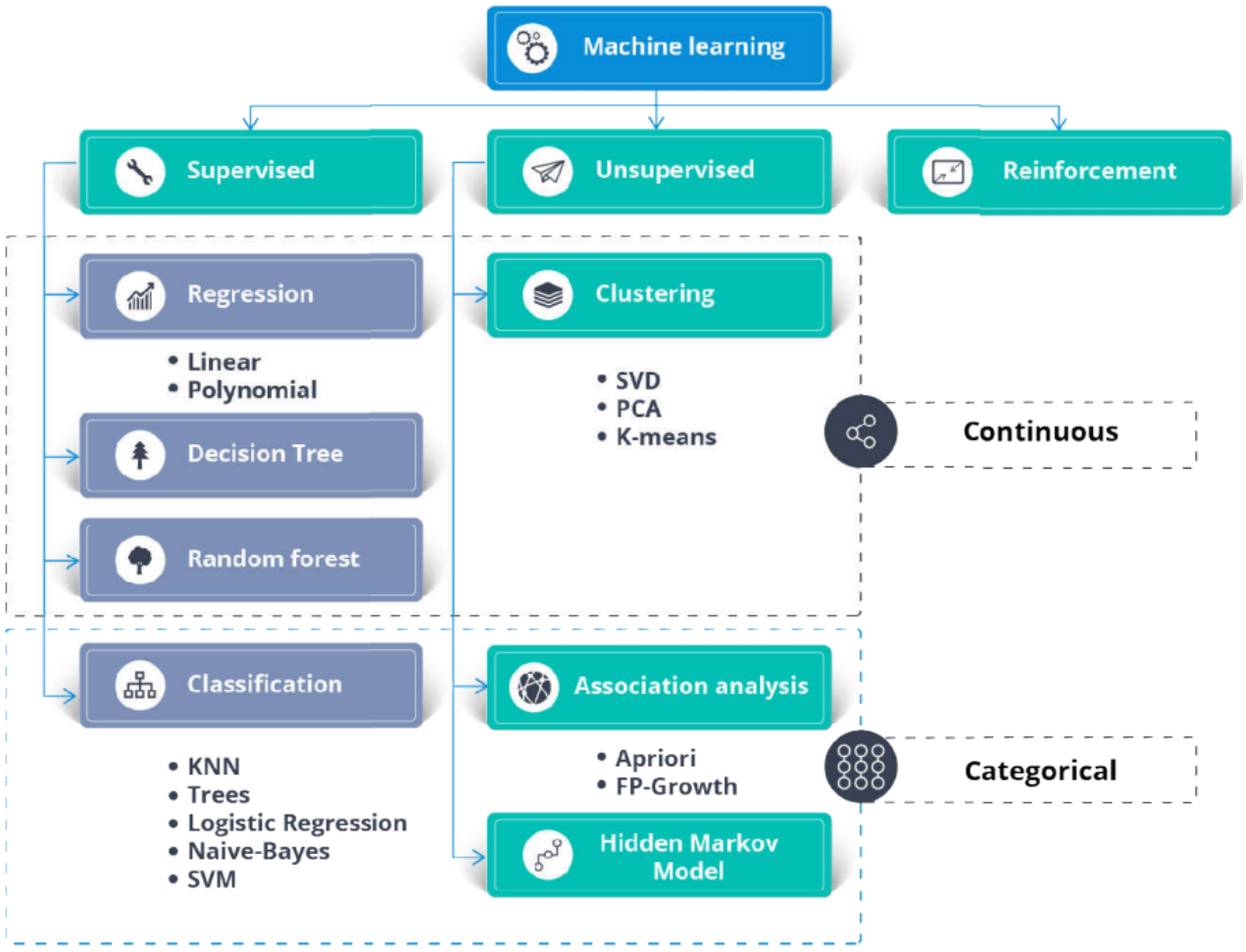

Figure 2.16: ML algorithms types[4]

2.6 Vulnerability detection using Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately, and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.[5]

2.6.1 How Deep Learning Works

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.Deep learning models are trained by using large sets of labelled data and neural network architectures that learn features directly from the data without the need for manual feature extraction .[5]

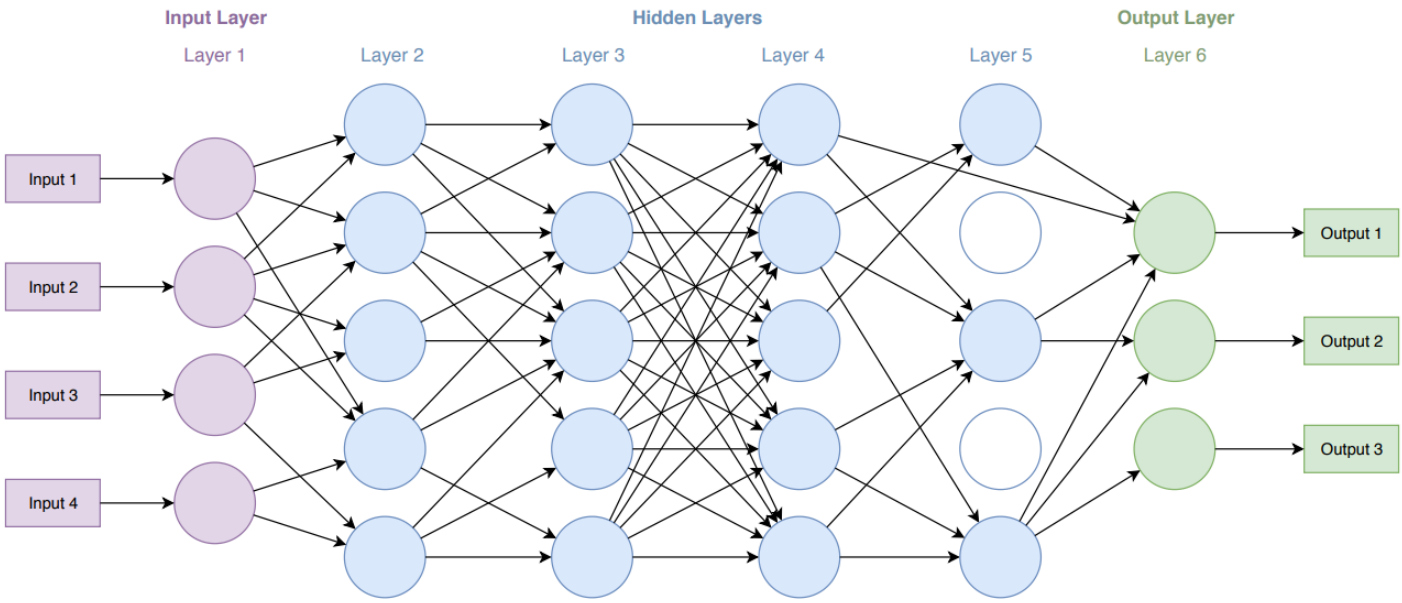

Figure 2.17: Neural networks[9]

2.6.2 Difference Between Machine Learning and Deep Learning

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs "end-to-end learning" – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically. Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network. A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.[5]

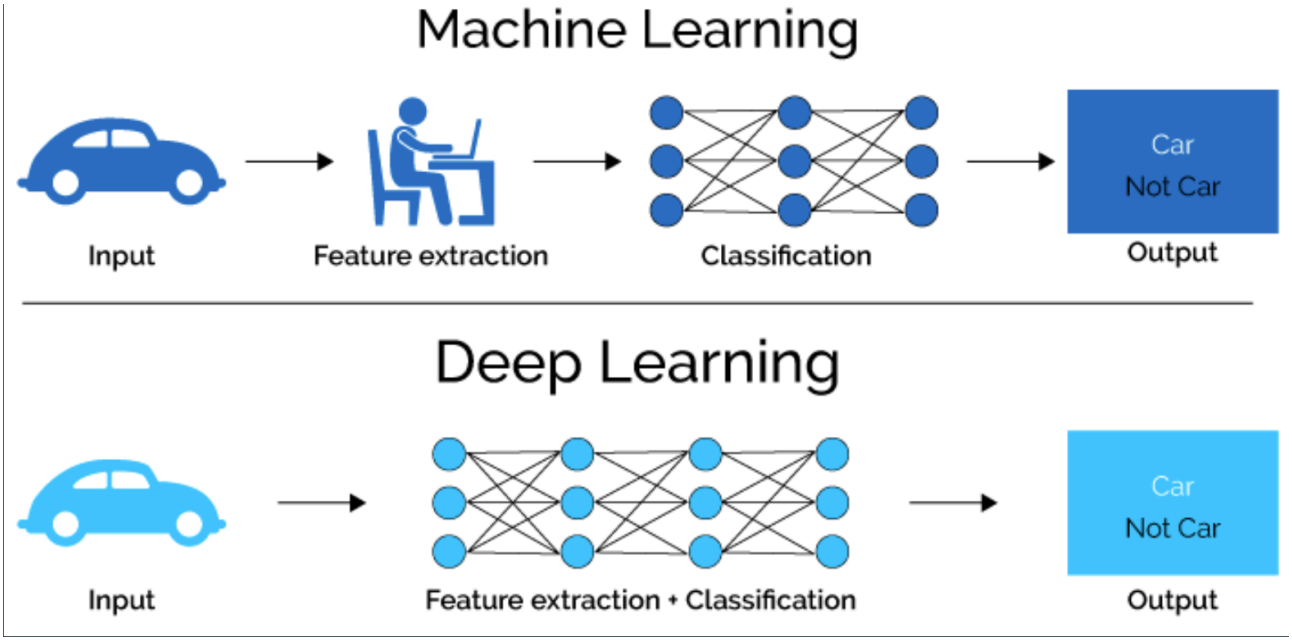

Figure 2.18: difference between Machine Learning and Deep Learning

2.6.3 How to create and train deep learning models

The three most common ways people use deep learning to perform object .[5] classification are:

• **Training from Scratch**

To train a deep network from scratch, you gather a very large labelled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have numerous output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train .

• **Transfer Learning**

Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pre-trained model. You start with an existing network, such as AlexNet or Google Net, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much fewer data (processing thousands of images, rather than millions), so computation time drops to minutes or hours. Transfer learning requires an interface to the internals of the pre-existing network, so it can be surgically modified and enhanced for the new task.

• **Feature Extraction**

A slightly less common, more specialized approach to deep learning is to use the network as a feature extractor. Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during the training process. These features can then be used as input to a machine learning model such as support vector machines (SVM).

2.6.4 Deep Neural Network

Deep Neural Networks (DNNs) are typically Feed Forward Networks (FFNNs) in which data flows from the input layer to the output layer without going backward[3] and the links between the layers are one way which is in the forward direction, and they never touch a node again.

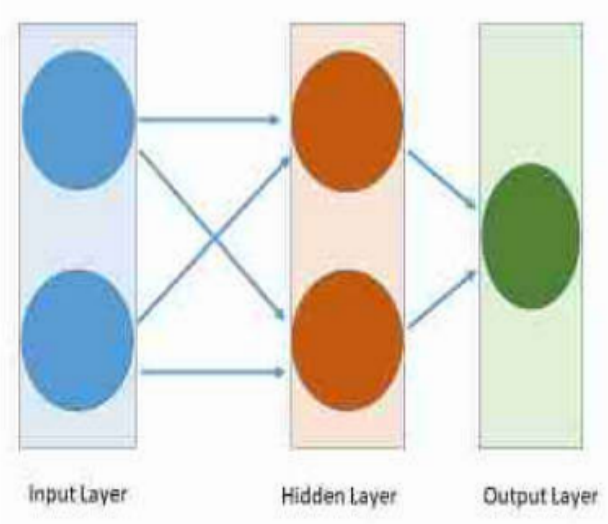

Figure 14: DNN

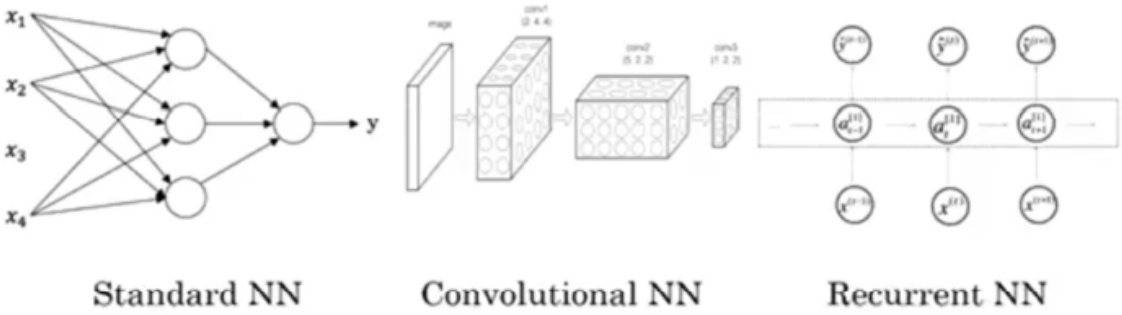

Figure 2.19: DNN types

2.7 Vulnerability detection using Natural Language Processing (NLP) technology

2.7.1 Natural Language Processing

Natural language processing (NLP) is the use of computers to model human natural language in order to solve the application of natural language in some related problems. In NLP, the problems that need to be solved can be divided into twocategories [1]:

• One is the natural language understanding (NLU) problem, including text classification, named entity recognition, relation extraction, reading comprehension ...

• The second is natural language generation (NLG) problems, including machine translation, text summary generation, automatic question and answer system, Image caption generation...

When NLP researchers studied and solved these two types of problems, they found that the underlying problems that constitute these problems are basically the same, such as embedding expressions of vocabulary. Now researchers are more inclined to use a unified model for modeling (pretraining stage), and then adjust the model according to specific problems (fine-tuning stage). Research at this stage has made great progress. It is believed that in the near future, machines can truly understand human language and even understand human thinking. Since 1980s, traditional NLP has increasingly relied on statistics, probability and shallow learning (traditional machine learning) [24], such as naive Bayes, hidden Markov model, conditional random field, support vector

machines and proximity algorithms, etc., these algorithms are still widely used in NLP today. But with the development of deep learning (DL), people are paying more and more attention to how to use DL models to solve the problems in NLP.[22]

2.7.2 How does Natural Language Processing Works

NLP entails applying algorithms to identify and extract the natural language rules such that the unstructured language data is converted into a form that computers can understand. When the text has been provided, the computer will utilize algorithms to extract meaning associated with every sentence and collect the essential data from them. Sometimes, the computer may fail to understand the meaning of a sentence well, leading to obscure results. For example, a humorous incident occurred in the 1950s during the translation of some words between the English and the Russian languages. Here is the biblical sentence that required translation[1]:

"The spirit is willing, but the flesh is weak."

Here is the result when the sentence was translated to Russian and back to English:

"The DRINK *is good, but the meat is rotten."*

Figure 2.20: biblical sentence that required translation

2.7.3 Deep Learning in Natural Language Processing

The main goal of DL is to learn the deep neural network model. The neural network model is composed of neurons and the edges connected to them. Each neuron can input and output. The data inside the neuron can be non-linearly transformed. According to the development of the timeline, we use the time point at which Transformer is proposed as the segmentation point. The model method before its appearance is called the basic model method, and the later one is called the modern model method (or attention model method). We will introduce them separately below Basic model method introduction [27]:

1. **Convolutional Neural Network (CNN) :** Due to the excellent abstract feature extraction ability of the convolution kernel, it has achieved great success in the field of computer vision (CV). In the field of NLP, CNN based algorithms have also appeared one after another, such as, etc. In the research related to vulnerability detection, some scholars have used CNN to mine vulnerabilities, as shown in figure.

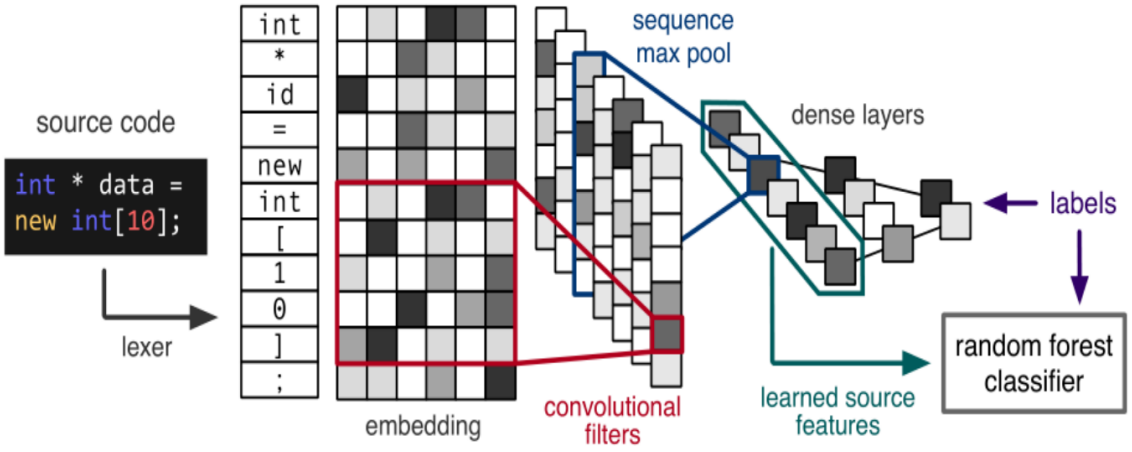

Figure 2.21: Using CNN to classify source code [27]

Although these models use CNN as a feature extractor to extract features from text data, because the feature dimensions of text data are not many, in text data, more attention is paid to the close connection between contexts, and the model is required to have a "memory" function, So CNN does not perform very impressively when processing tasks in NLP. But the latest research shows that with the development of multimodel technology, is some code generation tasks, such as image generation instructions, the use of CNN-based models has achieved good results.

2. **Recurrent Neural Network(RNN) [27]:**

   One of the characteristics of RNN is its "memory". RNN can take serialized data as input or output serialized data. For serialized data such as text, using RNN for processing has a natural advantage. In the output of the RNN, the above sequence information of the current token can be included, which makes the RNN have a "memory" function. When processing the data in this article, people often use a two-way RNN, that is, to process the above and below information of the current token separately. Let the token contain the current context information at the same time, which is very important for the model to understand the meaning of the sentence. However, the model with RNN structure cannot be processed in parallel. Today, with massive data, it greatly reduces the development of RNN in engineering applications. In NLP, CNN and RNN are used to extract the character-level representation of words, as shown in Figure.[27]

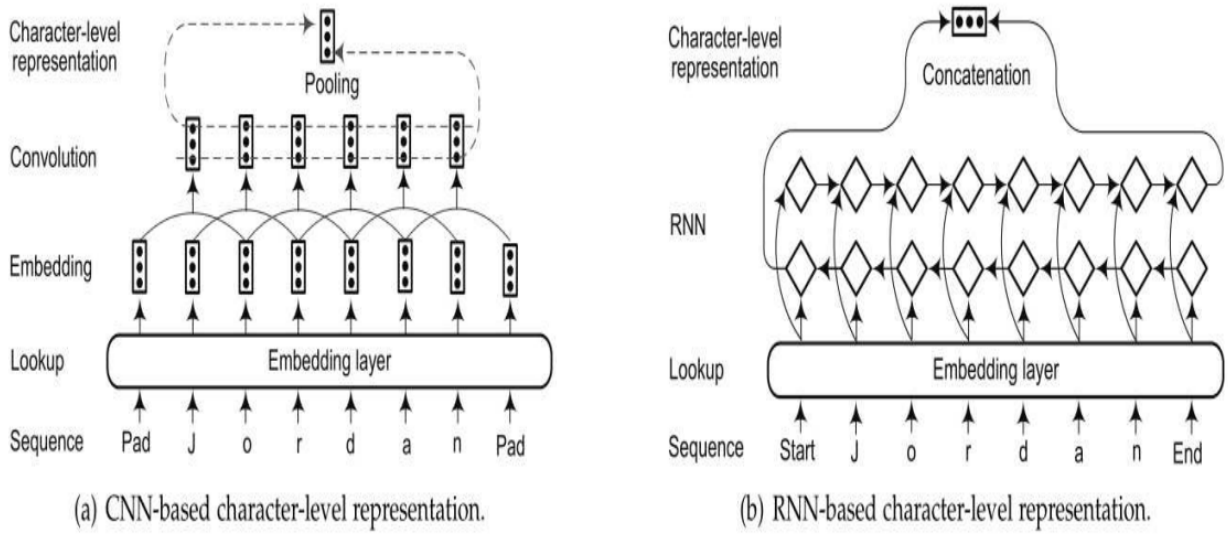

Figure 2.22: CNN RNN for extracting character-level representation for a word [27]

3. **Long Short-Term Memory Networks(LSTM) :** In addition to the structural limitations, RNN cannot capture long sequence text information due to the problem of vanishing gradient [42], so scholars modified RNN The LSTM model is proposed to solve the defect that RNN cannot process data in parallel. LSTM is one of the models with the strongest "memory" ability in NLP so far, and it is also one of the most widely used models. However, because LSTM has complex gating logic, it consumes a lot of space and time during training. Gated Recurrent Unit (GRU) is a model that is similar in structure to LSTM but more lightweight, and its performance in training is not worse than LSTM. For the comparison between the three basic models of CNN, GRU, and LSTM in NLP applications, please refer to . Since LSTM is a one-way model, in order to obtain the context information of the token, people often superimpose the LSTM/GRU model in two directions to obtain a two-way LSTM model (Bi-LSTM). In practical applications, the BiLSTM model is often used to extract the features of the sentence, and then the CRF algorithm is used to process the downstream tasks.[27]

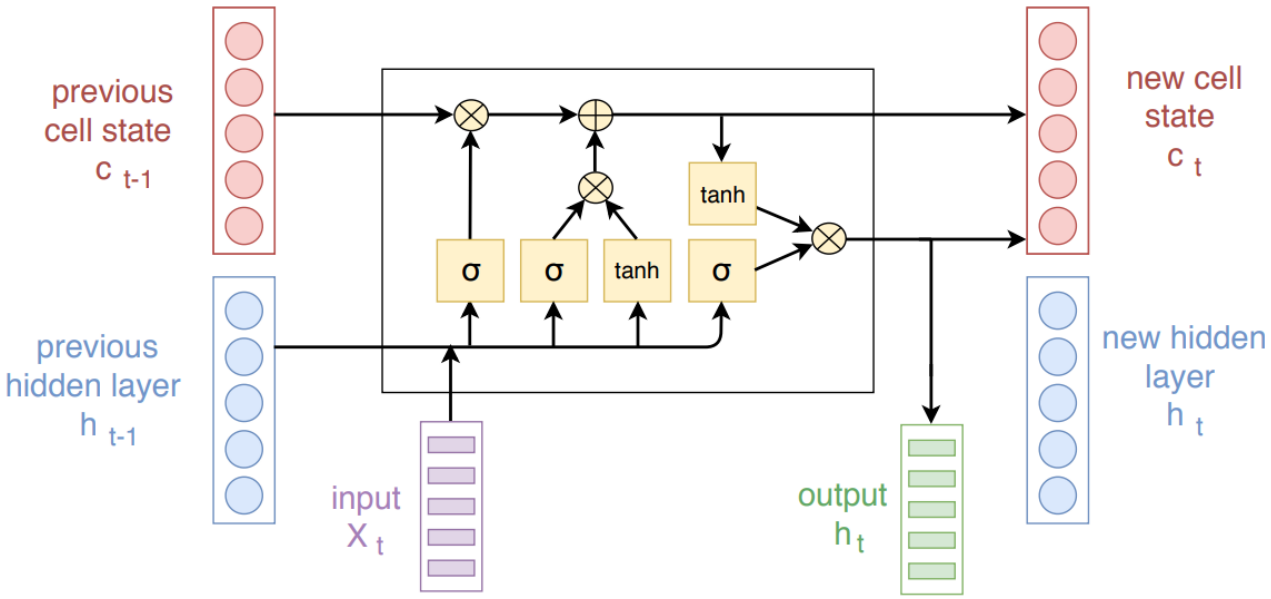

Figure 2.23: LSTM network [27]

So far, the basic concepts necessary for this work have been described. The next section describes previous work in finding vulnerabilities and also attempts a classification, although there are many criteria under which approaches can be compared. The advantages and disadvantages of the previous approaches are described.[27]

2.8 Related work

2.8.1 Vulnerability prediction based on metrics

What are the functions to apply while predicting whether code is susceptible or now no longer? For a lengthy time, the maximum typically used functions had been observed out of doors the supply code itself, with inside the shape of software program and developer metric. Those consist of length of the code (LOC), cycloramic complexity, code churn, developer activity, coupling, quantity of dependencies or legacy metrics. Such metrics had been universally used as functions for constructing fault prediction models, and are particularly applicable with inside the subject of software program first-class and reliability assurance. To provide simply one example, Nagappan et al. use organizational metrics to expect faults in a software program. It appears workable that the ones metrics may also be utilized in vulnerability prediction - however, there are a few issues with that. First, it's far viable that portions of code have the equal metrics (for instance, complexity) however a very extraordinary behavior, main to an extraordinary probability to be susceptible. In addition to that, additionally, they have a tendency to now no longer generalize nicely from one software program assignment to the next. The most powerful grievance is that such metrics do now no longer seize the semantics of the code, and this technique does now no longer take the real supply code, the program behavior, or the records circulate in account. The technique is correctly making use of a foregone end that sure meta functions might be associated with safety flaws, that's now no longer always true. For example, many vulnerabilities also can get up in very easy programs. In fact, regularly the trivial or direct method to an algorithmic problem does now no longer include the safeguards and precautions which might be required for stopping exploits, that's exactly the cause why software program builders who're below time constraints or lack revel in with safety concerns run into issues. Code complexity is now no longer a super predictor for safety flaws, and comparable arguments and counterexamples may be observed for the alternative metrics as nicely. However, it ought to be stated that as a minimum, a few insights may be won from software program metrics. This is illustrated with the subsequent works that use gadget getting to know procedures utilising code metrics to expect the prevalence of safety-associated flaws in software program. Shin et al. use 9 complexity metrics to expect vulnerabilities in JavaScript projects, attaining a low fake fine rate, however an incredibly excessive fake terrible rate. In a later work, the authors leveraged code complexity, code churn and developer metrics to expect vulnerabilities, attaining 80% to take into account and 25% positives with linear discriminant evaluation and Bayesian networks. Using complexity, coupling and brotherly love metrics (typically abbreviated as CCC), Chowdhury et al. try and expect software program vulnerabilities in the use of procedures that had been carried out to fault detection before. The behavior a take a look at on releases of Mozilla Firefox and use selection trees, random forest, logistic regression, and naive Bayes models to expect vulnerabilities, attaining round 70% precision and take into account, respectively. Zimmerman et al. delivered even extra metrics to the list: they investigated code churn, code complexity, code coverage, organizational measures and real dependencies. They observed a weak, however statistically good-sized correlation among the investigated metrics and used logistic regression to expect vulnerabilities primarily based totally on them, focusing on the proprietary code of Windows Vista. The metrics had been capable of expect vulnerabilities with a mean precision of 60%, however with an incredibly disappointing take into account of 40%. Neuhaus et al. checked out import statements with inside the Mozilla assignment, reporting a median precision of 70% and take into account of 40% while predicting vulnerabilities via way of means of import statements with assist vector machines. Yu et al. take many extraordinary viable functions into account, which includes software program metrics consisting of quantity of subclasses, or quantity of techniques in a file, in addition to crash functions and code tokens with their tf-idf scores. Their technique is consequently a

combination of many extraordinary angles. They expect vulnerabilities on the extent of entire documents and achieve very enjoyable consequences in narrowing down the quantity of code that needs to be inspected via way of means of human professionals to discover a vulnerability. Other researchers had been capable to make predictions simply with dedicate messages. Zhou et al. leverage a K-fold stacking set of rules to investigate dedicate messages to expect whether a dedicate includes vulnerabilities, reportedly with first-rate success. In contrast, Russel et al. observed that each human beings and Machine Learning algorithms accomplished poorly at predicting construct screw-ups or insects simply from dedicate messages. [37]

2.8.2 Anomaly detection approaches for finding vulnerabilities

Anomaly Detection refers to the problem of describing normal and expected behavior and detecting deviations from it. The assumption is that code not conforming to the implied rules can often be the cause of a defect. Data mining techniques have been used to analyze source code and extract normal coding patterns. To name one example, Li et al. developed a tool called PR-Miner that can find code patterns in any programming language and that has been proven to be quite useful. Their approach, which relies mostly on associating programming patterns that are used together with each other, is independent of any chosen language and violations reported by their tool have been confirmed as bugs in Linux, PostgreSQL and Apache. A fundamental problem is, however, that bugs that are themselves typical patterns (and therefore occur frequently in the code) are systematically overlooked, resulting in common flaws not being detected. At the same time, rare programming patterns or API usages can be flagged as false positive simply because they do not occur often. Several of the anomaly detection approaches have quite high false-positive rates.

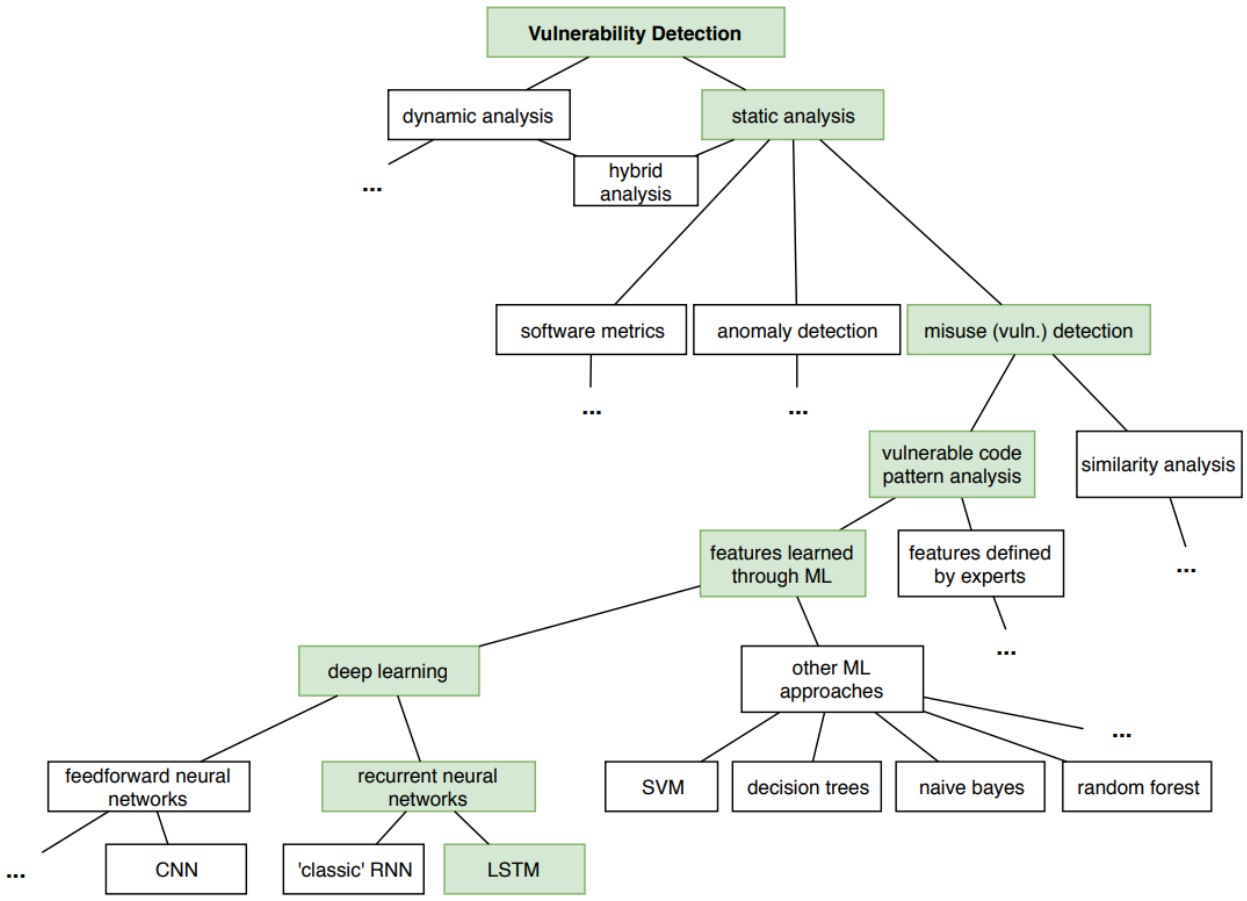

Figure 2.24: One way to structure different approaches for vulnerability detection

Specifically for finding security vulnerabilities (and not mere bugs that do not have any implication for security),

anomaly detection in code is not a straightforward approach, since it is hard to tell when a violation of common code patterns has an implication for security and when it does not. The approach in this work differs from typical anomaly detection insofar as explicit labels are used to train a model on vulnerable and (mostly) secure code, thereby avoiding the questionable assumption that 'typical' equals 'correct'. It belongs in the next category: vulnerable code pattern analysis.[20]

2.8.3 Vulnerable code pattern analysis and similarity analysis

Since the purpose is to discover vulnerabilities, in assessment to getting to know approximately summary metrics or the definition of accurate code, it appears nearly just like the maximum herbal desire to simply try to reply the question: What does prone code usually appearance like? There are two barely unique techniques to reply that question: prone code sample evaluation and similarity evaluation. Similarity evaluation does precisely what they call suggests. Given a prone code snippet, the purpose is to discover the maximum comparable code fragments, assuming that they're at threat to share the vulnerability. This method works nice for equal or almost equal code clones wherein the inherent shape of the in comparison code fragments could be very comparable, a state of affairs that happens pretty often, particularly through code sharing with inside the open supply community. In prone code sample evaluation, prone code segments are analyzed with data mining and device-getting to know strategies to extract their usual functions. Those functions constitute patterns, that may then be implemented on new code segments to discover vulnerabilities. Most of the works on this location acquire a huge dataset, procedures it to extract function vectors, after which makes use of device-getting to know algorithms on it, as defined through Ghaffarian et al. Both techniques are usually implemented to supply code without executing it, as a static evaluation, even though a few researchers additionally integrate their method with a dynamic evaluation. The crux of the problem is that during comparison to 'traditional' static evaluation, the functions are created mechanically or semi-mechanically, putting off the want for subjective human experts. By getting to know at once from a dataset of code what prone code entails, an independent version may be built. In many cases, the ones techniques additionally rely upon a completely hard granularity, classifying complete programs, documents, components or functions, which makes it not possible to pin down the precise vicinity of a vulnerability. Some, like Li et al. and Russell et al. use a greater fine-grained illustration of the code. Furthermore, the techniques vary in lots of aspects: the language used, the supply of the data (real-existence tasks or artificial databases) and the dimensions of the dataset, the introduction method for labels, the granularity stage of the evaluation (complete documents right all the way down to code tokens), the device getting to know version that become used, the tested varieties of vulnerabilities, and whether the version is usable in cross-venture predictions or simply at the venture it becomes educated on. First, a few fundamental techniques in the use of numerous device getting to know strategies might be defined. Afterwards, techniques that leverage deep getting to know are tested in greater detail. Morrison et al. study protection vulnerabilities in Windows 7 and Windows eight with numerous device getting to know strategies which includes logistic regression, naive Bayes, guide vector machines and random woodland classifiers, with particularly disappointing consequences, attaining very low precision and recollect values. In a completely truthful method, Pang et al. take labels from an online database and use a mixture of function choice and n-gram evaluation to categorise complete Java lessons as prone or now no longer prone. Working on a particularly small dataset of four Java android applications, they follow an easy n-gram version in aggregate with function choice (or better: ranking) techniques to mix associated functions and decrease the range of beside the point functions taken into account. Afterwards, they select out guide vector machines as getting to know algorithm, attaining round 92% curacy, 96precision and 87% recollect with inside the equal venture, and values round 65 % in cross-venture prediction (schooling on one venture and looking to classify prone documents in some other one). Shar et al. follow device getting to know to lessen fake positives in recognizing XSS and SQLI vulnerabilities in PHP code. They first select out a few code attributes manually and then teach a multi-layer perceptron to supplement static evaluation tools. Compared to a static evaluation tool, they detected much fewer vulnerabilities, however

additionally finished to decrease fake fantastic quotes in a universal fulfilling result. In their later paintings, they use a hybrid method which includes dynamic evaluation, enhancing their preceding consequences notably, as examined on six large PHP tasks. They additionally test with unsupervised predictors, which can be much less accurate, however nonetheless a promising location of research. Hovsepyan et al. examine uncooked supply code as textual content. As their example, they picked an Android e-mail purchaser written in Java and mainly centred on studying the supply code like a herbal language, processing documents as a complete. After filtering out comments, they remodel documents in function vectors made up from Java tokens with their respective counts with inside the document (in a bag-of-words-fashion method). Those function vectors are categorised in a binary scheme as prone or clean. Finally, the classifier (a guide vector device) is educated to expect if a document is prone. The accuracy finished through this classifier is 87%, with 85% precision and 88% recollect. Their achievement suggests that a great deal of perception may be gained without tricky fashions of code illustration, through simply taking the supply code as herbal textual content and studying it as-is. Unfortunately, their paintings are restrained through the utility on an unmarried software program repository. In a later painting, they used choice trees, k-nearest-neighbor, naive Bayes, random woodland and guide vector machines for the same task.[38]

2.9 conclusion

In this chapter, we have tried to cover ways to prevent gaps in web applications, including machine learning and deep learning, these are two interesting areas where it is widely used by many companies because it gives good results and gives solutions to problems through the data provided and advances data .In the next chapter, we will try to apply one of the learning algorithms, and they have been trained in the field of injection and XSS attacks prevention and some special designs.

Chapter 3

Conception

3.1 Introduction

In the previous chapter, we have seen some techniques and methods used to prevent gaps in Web applications and some work done, and we have chosen logistic regression and convolutional neural network and how it helps prevent SQL/XSS gaps and protect users from this gap . We will explain in this chapter the steps and units that make up our system and where to present the design of the system through its general design, then move on to its detailed design by explaining the components and elements of the system and identifying it and the principle of its operation.

3.2 System presentation

we will describe our system globally and give the shape of its structure, such as its components and purpose.

3.2.1 System objectives

The proposed system is a system that allows the analysis of (SQL) commands and Cross Site Scripting (XSS) and other words and symbols that the user enters into the web application for the purpose of accessing the platform by SQL and XSS injection OR input data in the request contains such special characters and Web applications generate response pages using this data, the model we have developed where everything that the user enters, We merged into one model each of (XSS and SQL) such as password, name or email, verifies and authenticates it by analyzing all the entries and comparing them to the database of known entries if there is consensus who passes and if otherwise he suspends and cancels them.

Any prevention and protection of our application is carried out against any vulnerability or attacker intended to cause damage or steal information... etc

3.2.2 Flow chart of the global system Architecture

This flow chart contains of information are exchanged between the user and our system (web application) where the user enters the information and then passes it on the CNN model so that model can expect if this information request vulnerable or not and if the request is vulnerable code the system will display sql or xss detected if not and if the user had account in database the system will display the interface user .

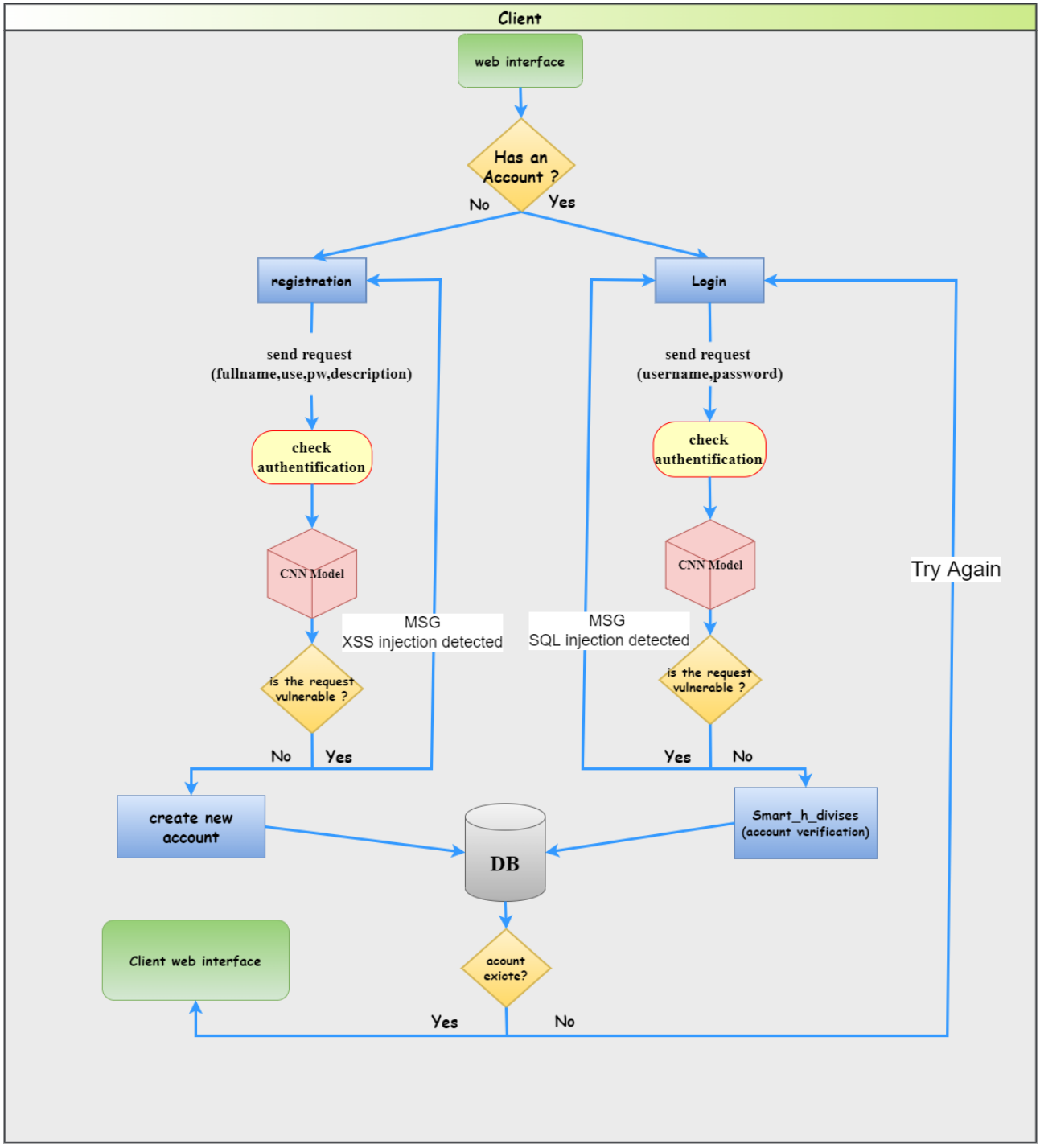

Figure 3.1: flow chart of the global system

3.3 Detailed System Design

3.3.1 Flow chart of creating CNN model

In general, we can represent the structure of the (SQL/XSS) CNN Model and evaluation system as follows :

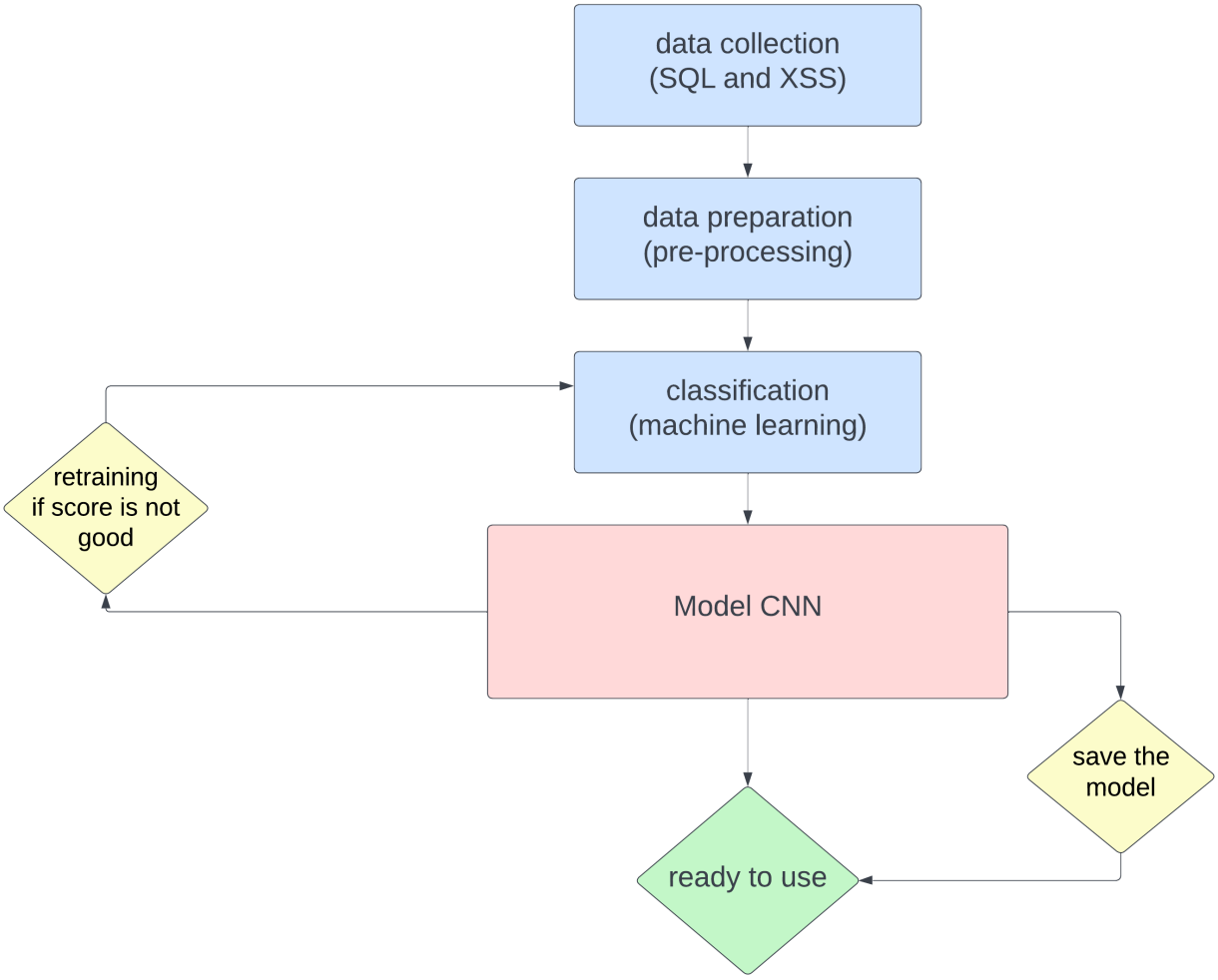

Figure 3.2: The General flow chart of CNN Model

The system can be divided into 3 components:

- Data Collection
- Data Preparation
- Classification and Training

3.3.2 Data Collection

In this unit we will collect as much data as possible, which is a collection of SQL and XSS query, letters, numbers and constraint codes, JavaScript .... Which in turn can be the cause of a vulnerability, leading to exploitation by attackers.
The process of collecting this dataset was difficult because most of the datasets were not free and did not have public access, but we succeeded to find one on the KAGGLE platform (kaggle.com), which did not contain much data, just a satisfactory amount.

this link from KAGGLE for dataset :

SQL : https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset
XSS : https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning

Dataset name/label	Dataset size	total entries	Malicious entries	Secure Entries	Source File Name
Database SQL Injection	723.15 KB	4200	1128	3072	Sqli.csv
Database XSS	16700 KB	13686	7373	6313	XSS-dataset.csv
all dataset	2393.15 KB	17886	8501	9385	SQL-XSS.csv

Table 3.1: Metadata of the Collected Data Set

The dataset contains two columns, a sentence, and a Label column. The value in column Label 1 means that the data is a SQL injection, the value 2 means the data is XSS vulnerability and 0 means that it is normal data.

3.3.3 Data Preparation

In this component we will try to train our system, which in turn learns in the field of classification and division of applications through specific data, where the algorithm vectorizes the data of the text or (SQL order and XSS order) into a matrix to be easily identified and manipulated, then preserve a model and if it is high resolution, let us use it or not use it using other more precise and effective parameters.

1. **Using TF-IDF in machine learning natural language processing (CNN)**
   In our system, we have chosen to use TF-IDF as a data vectoring technique, in which we apply ourselves to extract characteristics from our textual dataset.Machine learning algorithms often use numerical data, so when dealing with textual data or any natural language processing (NLP) task, a subfield of ML/AI dealing with text, that data first needs to be converted to a vector of numerical data by a process known as vectorization

2. **TF-IDF** vectorization involves calculating the TF-IDF score for every word in your corpus relative to that document and then putting that information into a vector (see image below using example documents “A” and “B”). Thus, each document in your corpus would have its own vector, and the vector would have a TF-IDF score for every single word in the entire collection of documents. Once you have these vectors, you can apply them to various use cases, such as seeing if two documents are similar by comparing their TF-IDF vector using.[40]

Word	TF		IDF	TF*IDF	
	A	B		A	B
select	1/7	1/7	log(2/2) = 0	0	0
from	1/7	0	log(2/1) = 0.3	0.043	0
user	0	1/7	log(2/1) = 0.3	0	0.043
href	1/7	1/7	log(2/2) = 0	0	0
class	1/7	1/7	log(2/2) = 0	0	0
web	1/7	1/7	log(2/2) = 0	0	0
if	1/7	1/7	log(2/2) = 0	0	0
var	1/7	0	log(2/1) = 0.3	0.043	0
and	0	1/7	log(2/1) = 0.3	0	0.043

Figure 3.3: example documents"A" and "B").

3.3.4 Classification and Training

In this part we will try to train our system, which in turn learns in the field of classification and division of commands from specific data, then saves a model and if it has high accuracy, we exploit it or do not reform it with other parameters.

We chose to use **convolutional neural network (CNN)** to train our model, which provided us with better accuracy compared to other machine learning algorithms such as SVM, Decision Tree and others, the following figure shows the training steps.

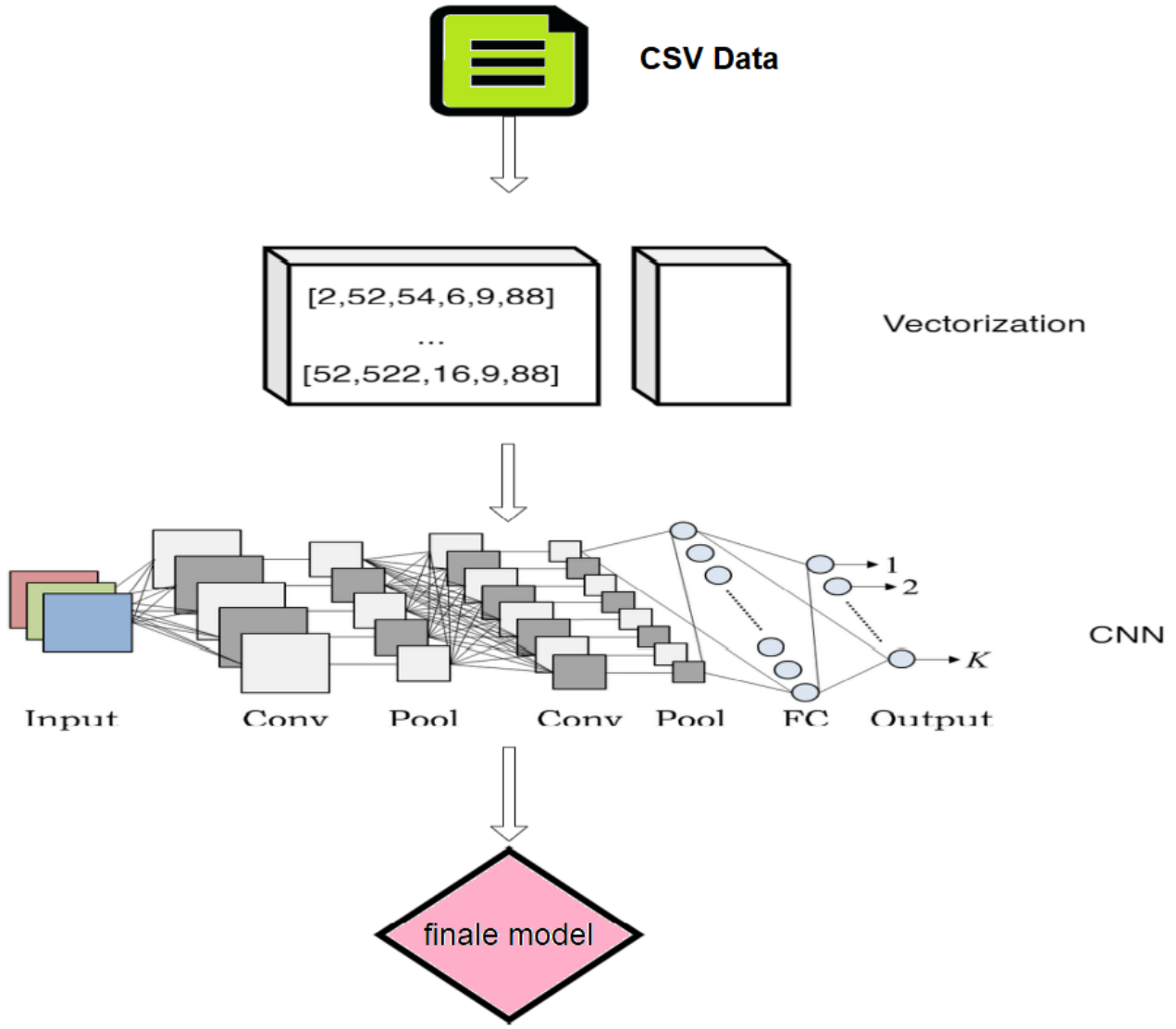

Figure 3.4: Training the Model

3.4 Model Testing

We will test our simple system which is to try the model of us by testing it, and we calculate the accuracy on the whole of which test provided that the model of us has not seen it and if the result is more than 75% means that it has passed, and therefore we run it, or we will retreat in another way which is the use of other good parameters and training again

3.4.1 Using the model

We can use our saved model as follows :

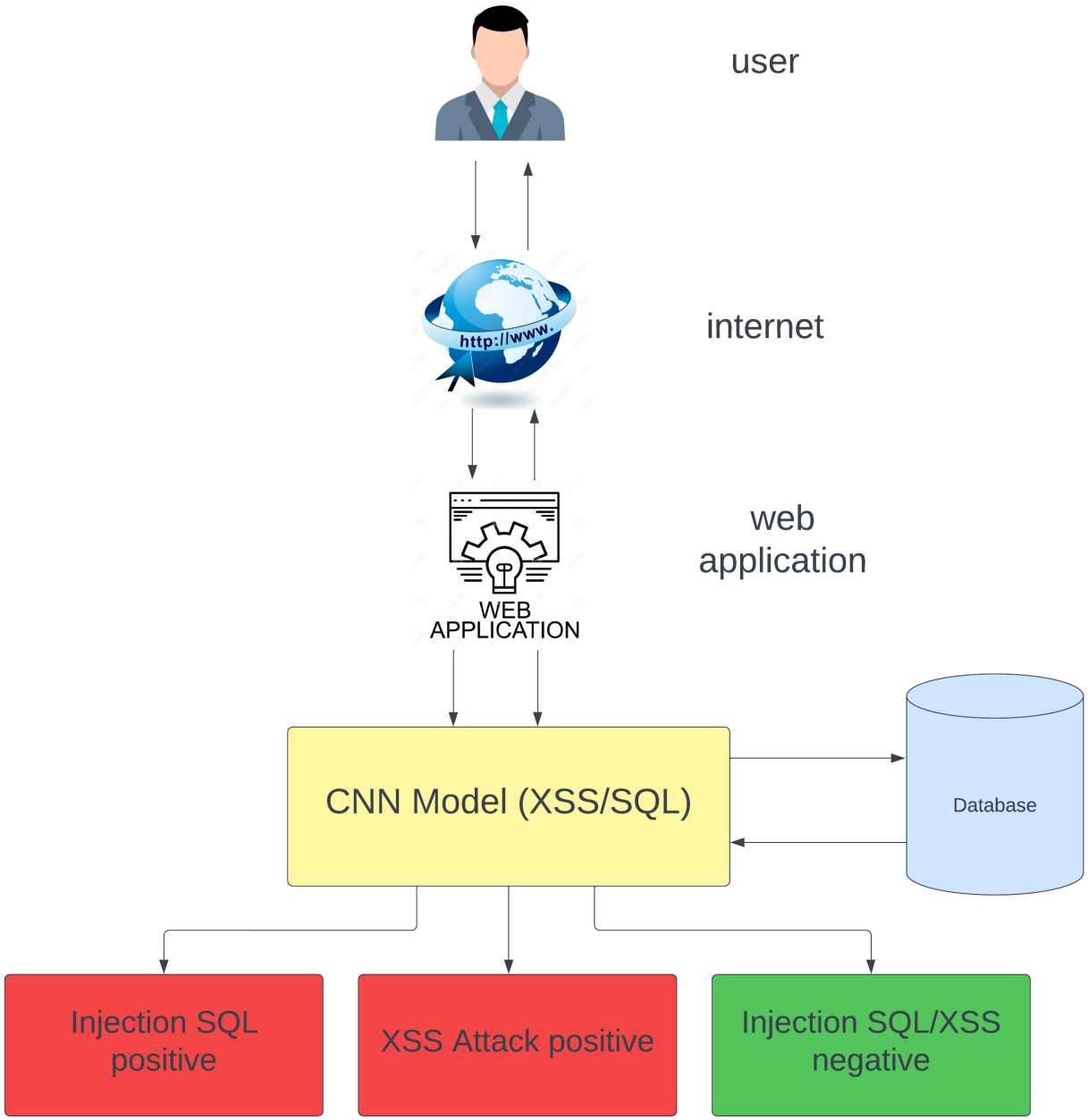

Figure 3.5: Using the Model

3.5 Designed by UML

3.5.1 Sequence diagram for " Registration "

This diagram explains how to create a new account in a web application where the system user accesses the record, and when the required interface is displayed, the fields are correct from the process where the system ensures that the data is entered and corrected after verification by (XSS-SQL-Model), and then stored and saved in the database as shown in the figure below.

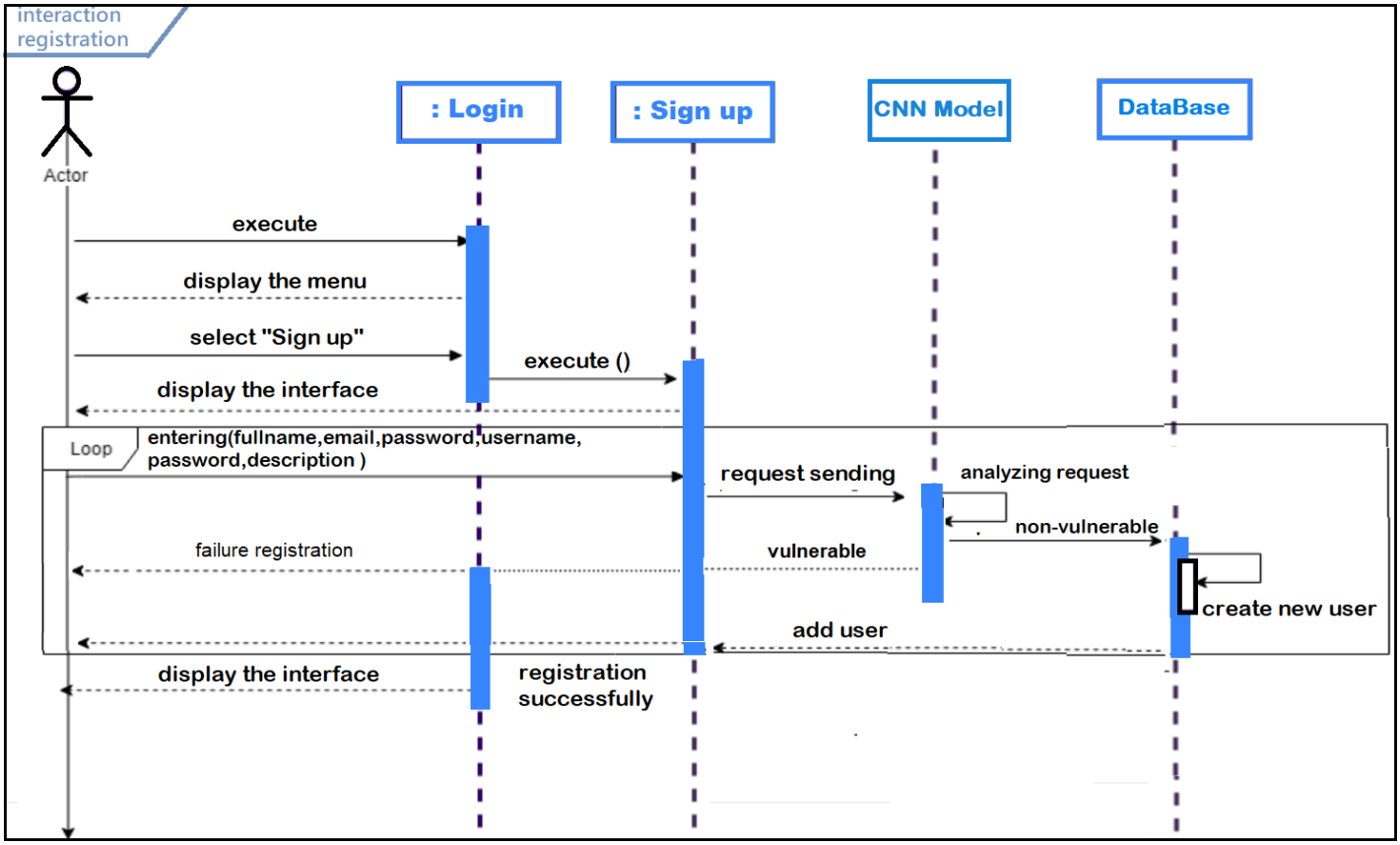

Figure 3.6: "Inscription" Sequence Diagram

3.5.2 Sequence diagram for " Authentication"

In this diagram, we have described the authentication scenario by displaying our system as an authentication form after the user has requested after writing down their complete information and submitting the form, and then the system reacts to perform the procedures to meet the requirements.

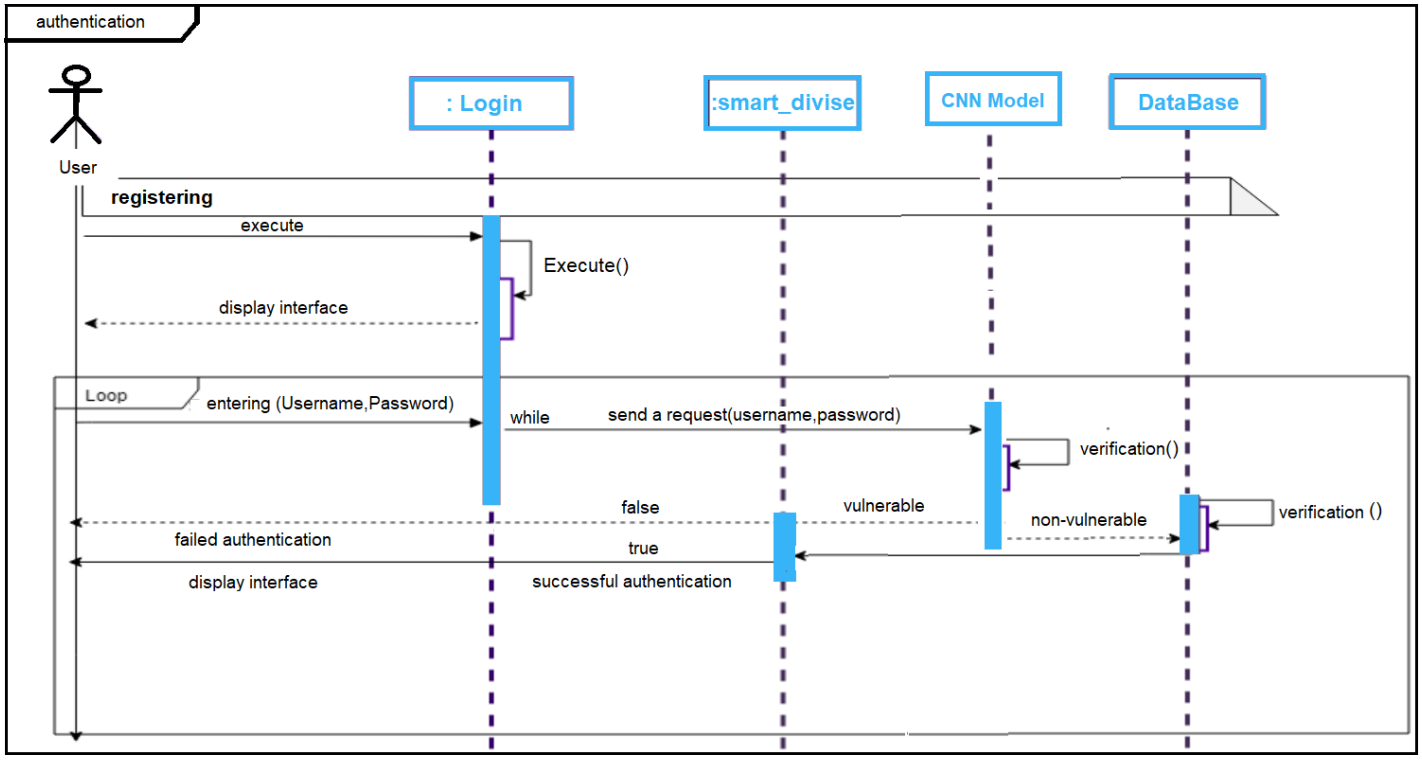

Figure 3.7: "Authentication" Sequence Diagram

3.6 Conclusion

In this chapter, we have presented how we designed our system, where we presented the overall design, and also detailed the steps we took to access our system, and we detailed the components used and the basic steps (data collection, data preparation, training and testing). We present, in the next chapter, the implementation of the system and the obtained results .

Chapter 4

Implementation

4.1 Introduction

In this chapter, we will introduce the work environment, the programming language, and the tools we used to build the system.Also we will Present the database used and some images of the application interface. Then, explain all the experiments we have applied to the proposed method and the results obtained . in the end we show a comparison with different models.

4.2 Development Environment

4.2.1 Python

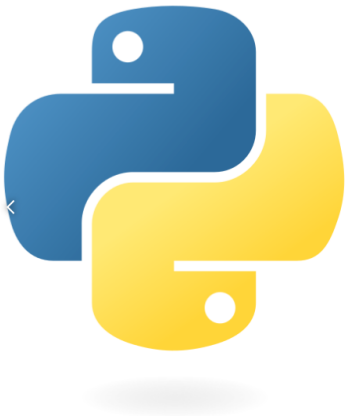

Figure 4.1: Python Logo

Python is the open source programming language most used by computer scientists. This language has propelled itself to the top of infrastructure management, data analysis or in the field of software development. Indeed, among its qualities, Python allows developers to focus on what they do rather than how they do it. It freed developers from the constraints of forms that occupied their time with older languages. Thus, developing code with Python is faster than with other languages. [2]

4.2.2 Environment using google colab for creating the model

Figure 4.2: google Colab Logo

Google Colab
was developed by Google to provide free access to GPU's and TPU's to anyone who needs them to build a machine learning or deep learning model. Google Colab can be defined as an improved version of Jupyter Notebook. Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs, which are sold per-use basis.[36]

What Colab Offers You?

- Write and execute code in Python
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

4.2.3 XAMPP

XAMPP is a set of software that is used to easily set up a Web server, ftp server, and e-mail server. It is a free software distribution (X Apache MySQL Perl PHP) offering good flexibility of use, recognized for its simple and fast installation. [**?**]

Figure 4.3: XAMPP Icon

Thus, it is within the reach of most people insofar as it does not require specific knowledge and works, moreover, on the most common operating devices

4.2.4 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.[6]
why choose Django ?

Figure 4.4: Ridiculously fast

Django was designed to help developers take applications from concept to completion as quickly as possible.

Figure 4.5: Reassuringly secure

Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.[6]

Figure 4.6: Exceedingly scalable

Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

What does Django code look like ?

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received, the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required, it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting

the retrieved data into placeholders in an HTML template.[6]
Django web applications typically group the code that handles each of these steps into separate files:

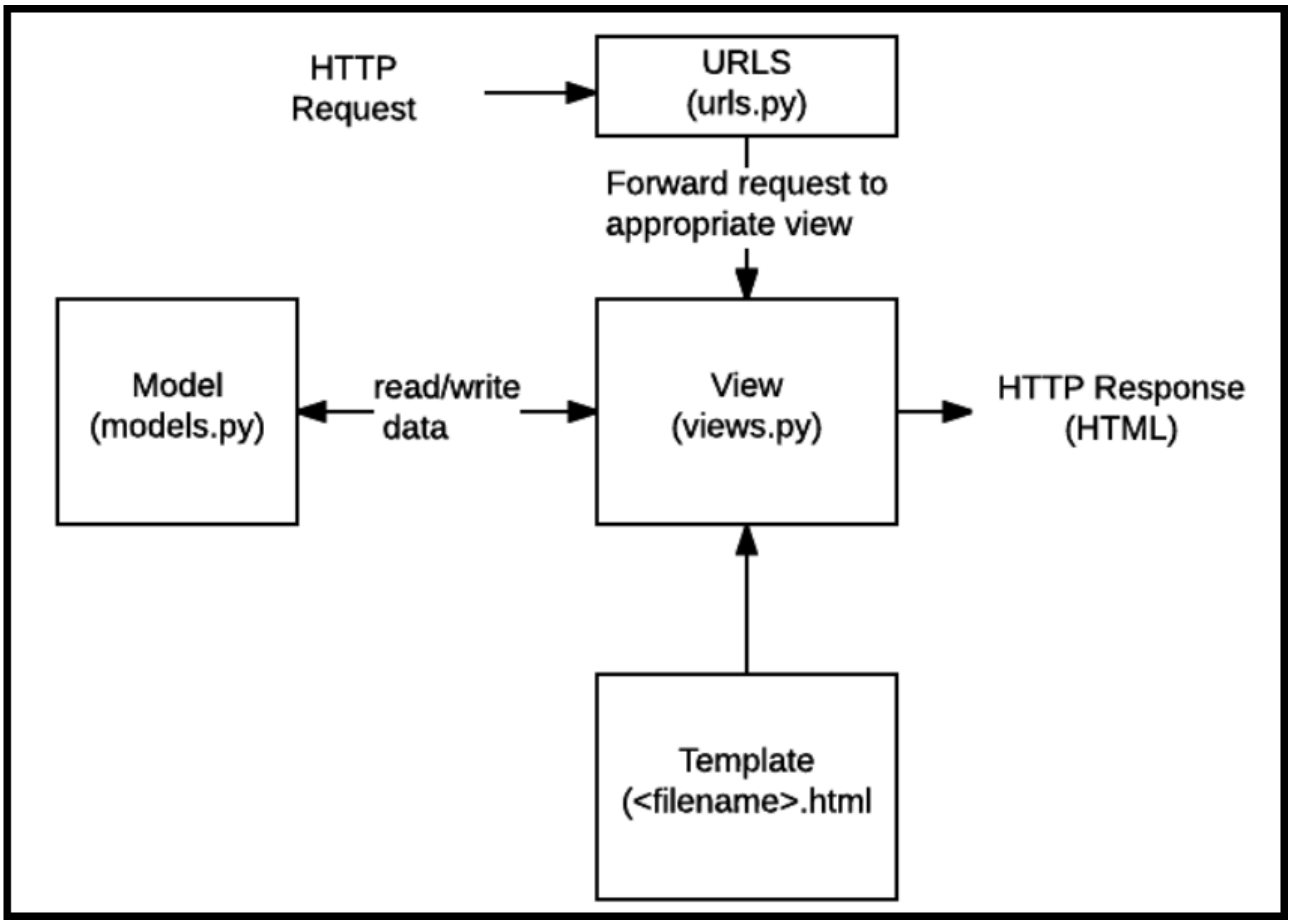

Figure 4.7: Django architecture

4.3 The used tools

4.3.1 Tensorflow

Figure 4.8: tensorflow logo

TensorFlow is an open-source library developed by the Google Brain team that initially used it internally. It implements machine learning methods based on the principle of deep neural networks (Deep Learning). A Python API is available, we can exploit it directly in a program written in Python. It enables easy deployment of compute across a variety of platforms (CPU, GPU, TPU), and from desktops to server clusters to mobile devices and peripherals. [25]

Why TensorFlow ?

- **Easy model building**
  Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.
- **Robust ML production anywhere**
  Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.
- **Powerful experimentation for research**
  A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

4.3.2 Keras

Figure 4.9: keras logo

Keras is an open source software library that provides multiple backends supported by Keras, including TensorFlow, Microsoft Cognitive Toolkit, Theano, only Tensor Flowis supported. Designed to enable rapid experimentation with deep neural networks, it focuses on usability, modularity, and extensibleness. It has been developed as part of the research effort of the ONEIROS (Open ended Neuro-Electronic Intelligent Robot Operating System) project[39]

4.4 Structures of Data

4.4.1 Part of the used dataset

For SQL

Sentence	Label
" or pg_sleep (__TIME__) --	1
create user name identified by pass123 temporary tablespace temp default tablespace users	
AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (table_name) FROM (SELECT DISTINCT (table_name) , ROWNUM AS LIMIT FROM sys.all_tables) WHERE LIMIT = 5)) AND 'i' = 'i	1
select * from users where id = '1' or @ @1 = 1 union select 1,version ( ) -- 1'	1
select * from users where id = 1 or 1#" (union select 1,version ( ) -- 1	1
select name from syscolumns where id = (select id from sysobjects where name = tablename') --	1
select * from users where id = 1 +$+ or 1 = 1 -- 1	1
1; (load_file (char (47,101,116,99,47,112,97,115,115,119,100))) ,1,1,1;	1
select * from users where id = '1' or ||/1 = 1 union select 1,version ( ) -- 1'	1
select * from users where id = '1' or \.<\ union select 1,@@VERSION -- 1'	1
? or 1 = 1 --	1
) or ('a' = 'a	1
admin' or 1 = 1#	1
select * from users where id = 1 or " ( ]" or 1 = 1 -- 1	1
or 1 = 1 --	1
AND 1 = utl_inaddr.get_host_address ((SELECT DISTINCT (column_name) FROM (SELECT DISTINCT (column_name) , ROWNUM AS LIMIT FROM all_tab_columns) WHERE LIMIT = 5)) AND 'i' = 'i	1

Figure 4.10: positive items

SELECT * FROM fat WHERE lower = 'hang' OR drew = 'purple'	0
SELECT * FROM mountain WHERE shape = 'street' OR package = 'tobacco'	0
SELECT * FROM running WHERE NOT frog = 'forest'	0
SELECT * FROM boat WHERE service = 'chart' AND (yourself = 'rapidly' OR wide = 'shout')	0
SELECT * FROM palace WHERE NOT settlers = 'pass' AND NOT bowl = 'leave'	0
SELECT * FROM prepare ORDER BY hearing	0
SELECT * FROM put ORDER BY cowboy DESC	0
SELECT * FROM threw ORDER BY stood, manner	0
SELECT * FROM package ORDER BY headed ASC, word DESC	0
INSERT INTO earlier (military, highest, anyone, house, almost, quite) VALUES ('jet', 'division'. one', 'film', 'shot', 'ago', 'clearly')	0
INSERT INTO product (instant, spent, pig) VALUES ('sick', 'plates', 'anybody')	0
SELECT molecular, queen, fought FROM blanket WHERE felt IS NULL	0
SELECT opportunity, ear, potFROM fat WHERE four IS NOT NULL	0

Figure 4.11: negative items

For XSS

<style>@keyframes x{from {left:0;}to {left: 1000px;}}:target {animation:10s ease-in-out 0s 1 x;}</style><frameset id=x style="position:absolu	1
<param id=x tabindex=1 ondeactivate=alert(1)></param><input id=y autofocus>	1
<th id=x tabindex=1 onbeforeactivate=alert(1)></th>	1
<data onfocusout=alert(1) tabindex=1 id=x></data><input autofocus>	1
<mark draggable="true" ondragleave="alert(1)">test</mark>	1
<em onbeforepaste="alert(1)" contenteditable>test	1
<menu onkeyup="alert(1)" contenteditable>test</menu>	1

Figure 4.12: positive items

Steering for the 1995 "<a href="/wiki/History_of_autonomous_cars#1990s" class="mw-redirect" title=	0
<cite class="citation web"><a rel="nofollow" class="external text" href="https://www.mileseducatio	0
. doi :<a rel="nofollow" class="external text" href="htt	0
<li id="cite_note-118">^ 	0
Contextualism 	0
<li id="cite_note-Representing_causation-95">^ <a href="#cite_ref-Representing_causation_95-0"><sup><i	0
<tr><td class="plainlist" style="padding:0 0.1em 0.4em">	0
	0

Figure 4.13: negative items

4.4.2 pre-processing Data

we will vectorize the dataset (SQL/XSS) words to numbers

Data pre_process

```
[ ]  #drop first column in xss_dataset
     df3 = df1.drop(['Unnamed: 0'], axis=1)
     df3["Label"].replace({1: 2}, inplace=True)
     #Merge two tables {df2 , df3} into one {mergerd_df}
     mergerd_df = pd.concat([df2, df3])
     #drop rows containing null values
     mergerd_df = mergerd_df.dropna()
```

```
[ ]  #display lenght of each table
     print(len(df2),len(df3),len(mergerd_df))

     4200 13686 17873
```

```
     #Convert first column values of merged_df to string
     mergerd_df.Sentence = [str (item) for item in mergerd_df.Sentence]

     #call TfIdf Vectorizer
     vec = TfidfVectorizer(analyzer='word',stop_words='english')
     xzx = vec.fit(mergerd_df.Sentence.values)
     features = xzx.transform(mergerd_df.Sentence.values)
     featuress = features.toarray()
```

Figure 4.14: code source TF-IDF

we drop the first row of XSS dataset because we don't need it and merge tow table or data tables (XSS and SQL) then convert first colomun values of merged to string and finally call TF-IDF to vectorized the data to be ready and use it CNN model .

4.4.3 Training

To train the CNN model, we used the Keras library, a library written in Python, and we used some super parameters as a comparison, where vectorize converts these parameters into a numerical matrix to make it easier to read and calculate the results to classify them as shown in the figure below.

Train Test Split

```
[ ]  X = featuress
     y = to_categorical(np.array(mergerd_df['Label']))
     X = X.reshape(-1, 1, 17152)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ]  model = models.Sequential()
     model.add(layers.Conv1D(32, 1, activation = 'relu', input_shape = (1,17152)))
     model.add(layers.Conv1D(32, 1, activation = 'relu'))
     model.add(layers.Flatten())
     model.add(layers.Dense(3, activation = 'softmax'))

     model.summary()
```

Figure 4.15: training CNN Model

4.4.4 Evaluation

For the purpose of prediction and classification, four key concepts are usually the basis for evaluation: true positives, true negatives, false positive and false negatives. They have been mentioned before but shall be defined properly here. Positive and negative refer to the prediction, meaning that (without loss of generality) in this work a prediction of 'vulnerable' would be a positive and a prediction of 'not vulnerable' would be a negative. The terms true and false refer to whether the prediction corresponds to the actual value or external judgment. Hence, a false positive is a clean code incorrectly labeled as vulnerable by the classifier, a true positive is a vulnerability that was correctly spotted, a false negative is an actual vulnerability that was not classified as such, and a true negative is a piece of code that was classified as 'not vulnerable' and is indeed harmless.

The precision
is the rate of true positives within all positives. It measures how precise the model is in terms of how many of the predicted positives are actual positives, or phrased differently,how much trust can be placed in the classification of a positive and how many false alarms are produced. [35]

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Figure 4.16: Equation of The precision

The recall
also called sensitivity, is a measurement for the rate of positives that were correctly identified in comparison to the total number of actual positives. One could take it as a measurement for how vigilantly the classifier spots all positives - or how much gets overlooked .[35]

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Figure 4.17: Equation of The recall

The accuracy
is the fraction of correct predictions compared to all predictions. For binary classification [35] , it is defined as following:

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives}}$$

Figure 4.18: Equation of The accuracy

4.4.5 Experiments and Obtained Results

To get a better model, we had several experiments, applied the dataset on several methods, including CNN, KNN, SVM, and Naivebayse, and we got some results summarized in the figures below.

- **Convolutional Neural Network (CNN)**
  To arrive at a good model, we tested our database on CNN and got the results below. we got effective results.

```
y_pred=model.predict(X_test)
pred=np.argmax(y_pred,axis=1)
ground = np.argmax(y_test,axis=1)
print(classification_report(ground,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      1919
           1       0.81      0.98      0.89       222
           2       1.00      1.00      1.00      1434

    accuracy                           0.98      3575
   macro avg       0.93      0.98      0.96      3575
weighted avg       0.99      0.98      0.98      3575
```

Figure 4.19: result CNN Model

- **NaiveBayes**
  The results of the Naive Bayes model did not give excellent results compared to the CNN.

```
[ ] accuracy,precision,recall=confusion_matrix(y_test,pred_gnb)
    print(" For Naive Bayes Accuracy : {0} \n Precision : {1} \n Recall : {2}".format(accuracy, precision, recall))
```

```
For Naive Bayes Accuracy : 0.8668012108980827
Precision : 0.6158357771260997
Recall : 0.995260663507109
```

Figure 4.20: result NaiveBayes

- **K-nearest Neighbors (KNN)**
  To arrive at a good model, we tested our database on KNN and got the results below, which are not enough.

```
accuracy,precision,recall=confusion_matrix(y_test,pred_knn)
print(" For KNN Accuracy : {0} \n Precision : {1} \n Recall : {2}".format(accuracy, precision, recall))
```

```
For KNN Accuracy : 0.798183652875883
Precision : 0.5190311418685121
Recall : 0.7109004739336493
```

Figure 4.21: result K-nearest Neighbors (KNN)

- **Support Vector Machine (SVM)**
  We also did the same on SVM algorithm, and it gave good results.

```
accuracy,precision,recall=confusion_matrix(y_test,pred_svm)
print(" For SVM Accuracy : {0} \n Precision : {1} \n Recall : {2}".format(accuracy, precision, recall))
```

```
For SVM Accuracy : 0.8395560040363269
Precision : 1.0
Recall : 0.24644549763033174
```

Figure 4.22: result Support Vector Machine (SVM)

- **Comparison accuracy result**

CNN	Naive Bayes	KNN	SVM
98%	86%	79%	83%

Table 4.1: Comparison accuracy result

So the best model is CNN because it is larger than them in terms of ratio, it is suitable and effective to detect and prevent SQL injections and XSS attack

4.4.6 Testing

- we will display CNN Model Accuracy and Loss History

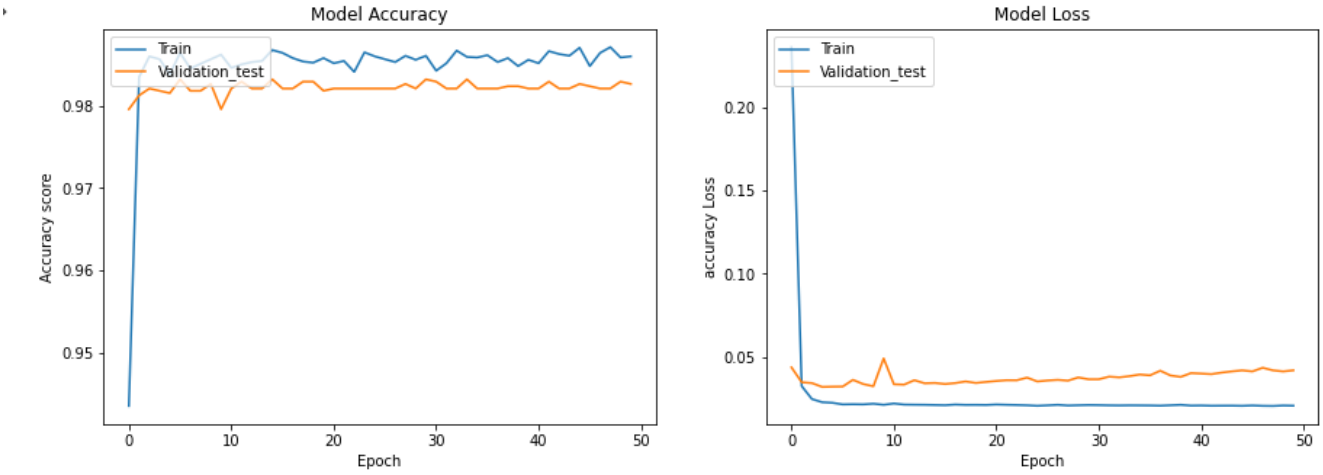

Figure 4.23: Model Accuracy and Model Loss

- we will display Confusion Matrix Of CNN Model

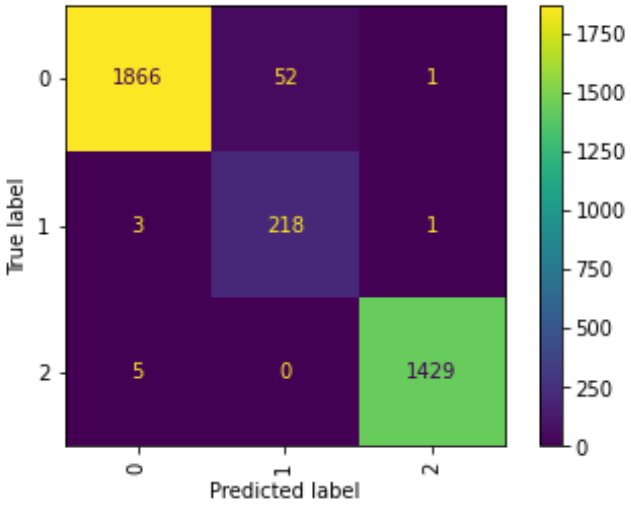

Figure 4.24: Matrix Of CNN Model

in the test :

- case 0 : 1866 case true , 52 case false(sql) , 1 case false (xss)
- case 1 : 3 case false , 218 case true , 1 case false (xss)
- case 2 : 2 case false , 0 case true(sql) , 1429 case true

- We trying to put a XSS injection example and see the result .

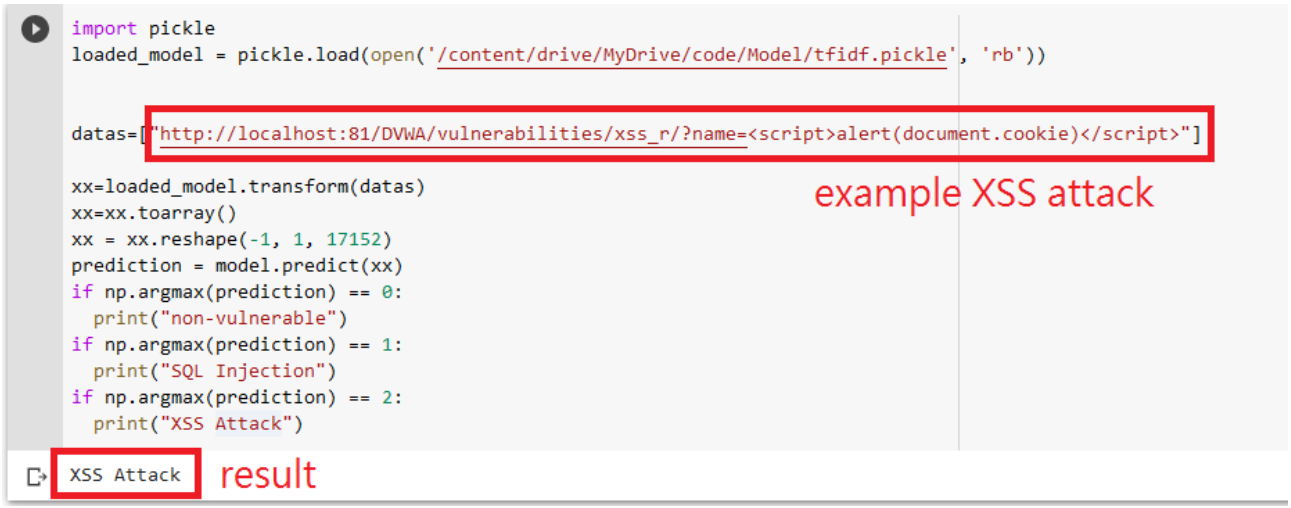

Figure 4.25: model testing

4.5 Presentation system

4.5.1 Database

The following database set was used: User database .

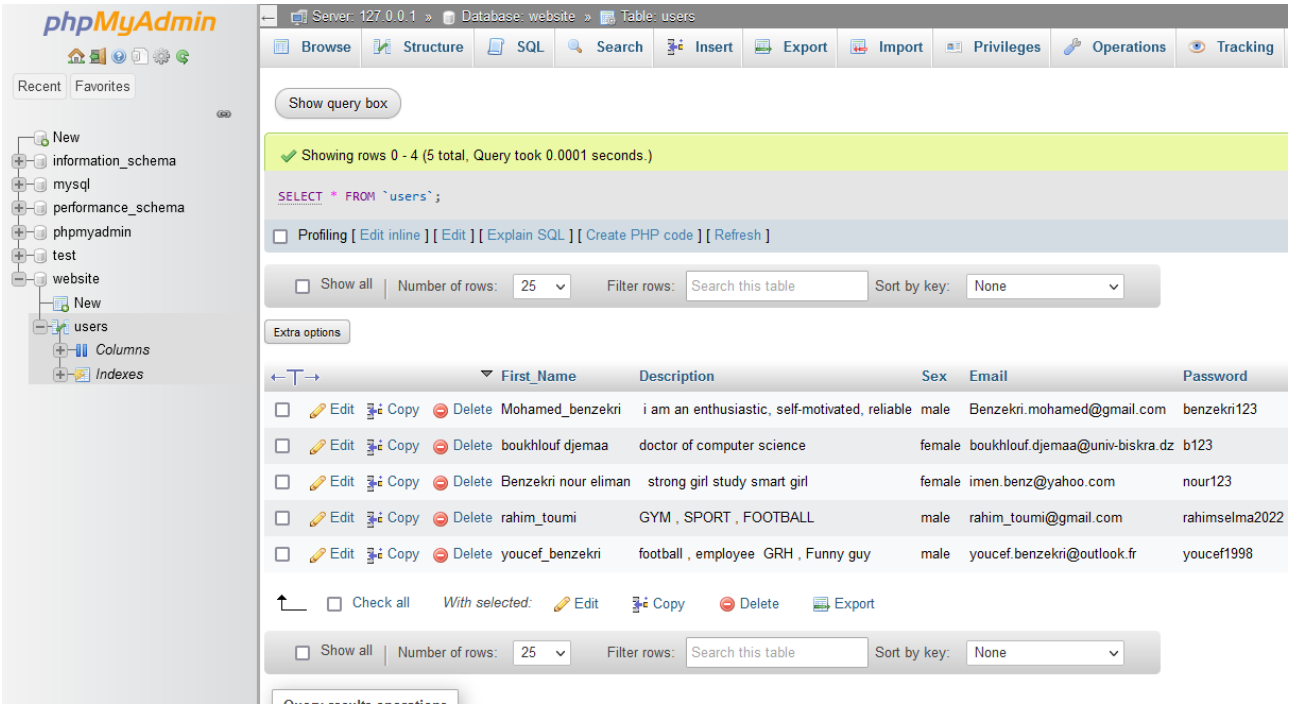

Figure 4.26: database myphp with xamp

4.5.2 Interface Already Registered "Login"

This interface allows the registered user in the database to enter as soon as they type their username and password.

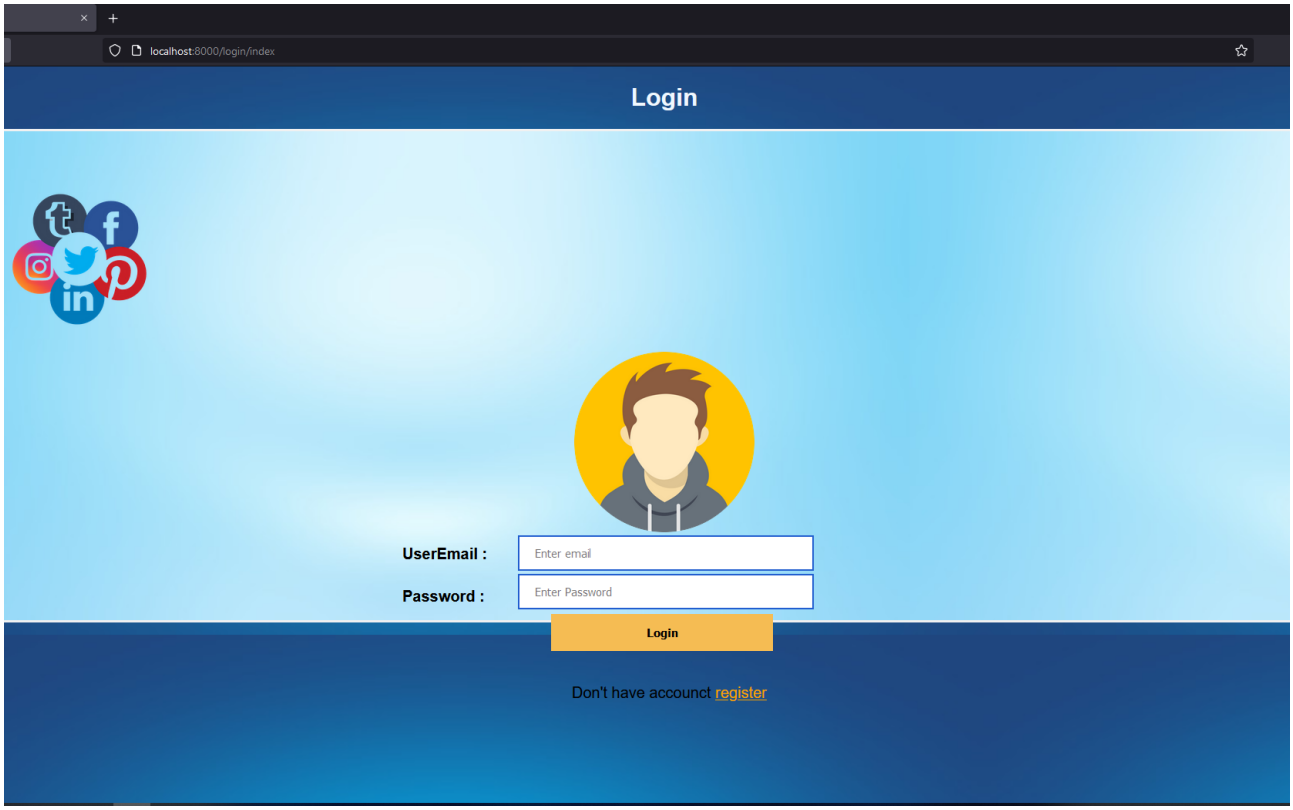

Figure 4.27: Interface "Login"

4.5.3 First Time Registration Interface "New User"

This interface allows the server to register for the first time because the interface contains certain features and allows you to type the username, sex, email, password, and Description , as shown in the figure below

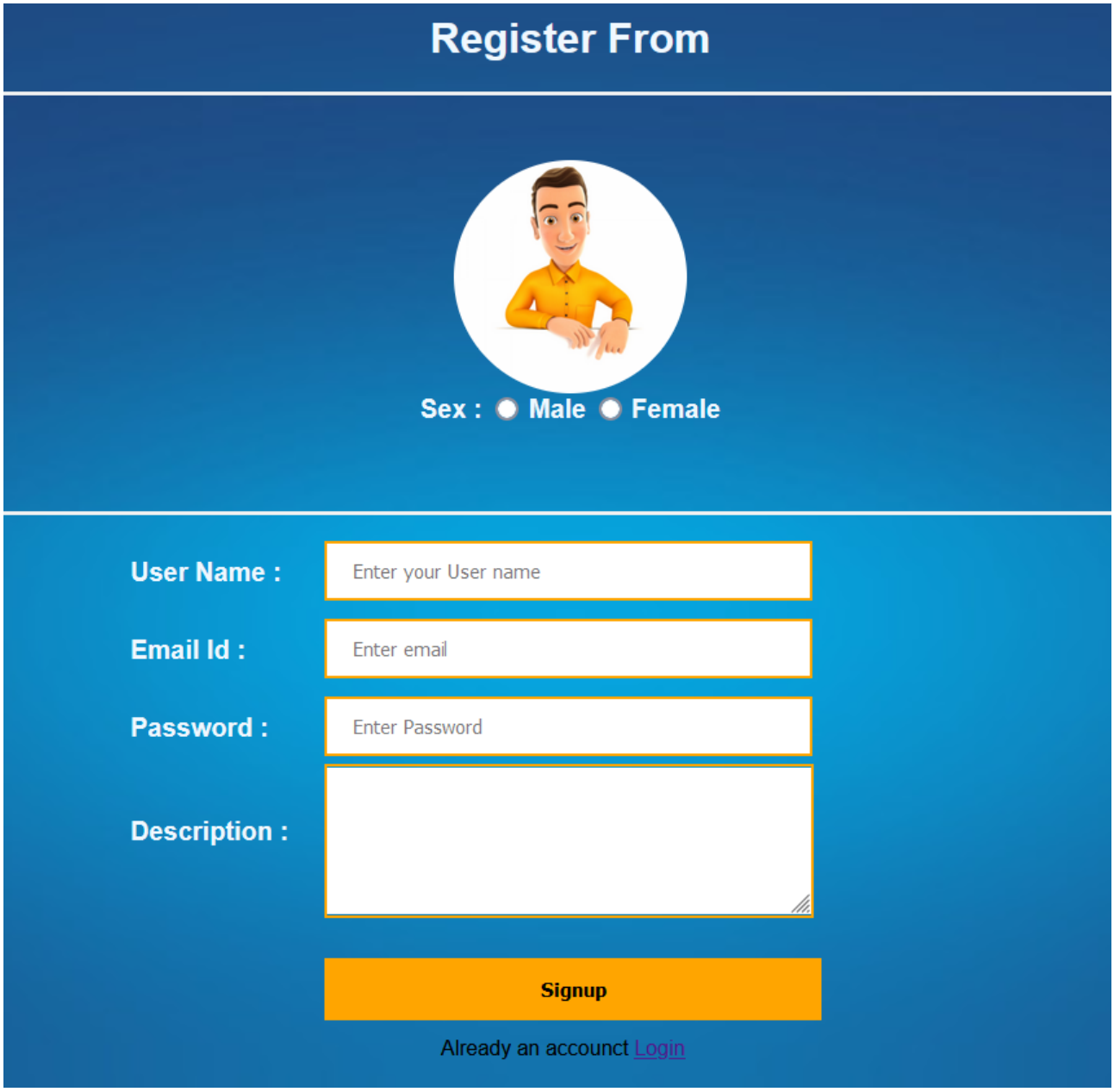

Figure 4.28: interface "New User"

4.5.4 First Time Registration Interface "New User"

This interface allows The user's interface is displayed with all the location data and this website is a social media, as shown in the figure below :

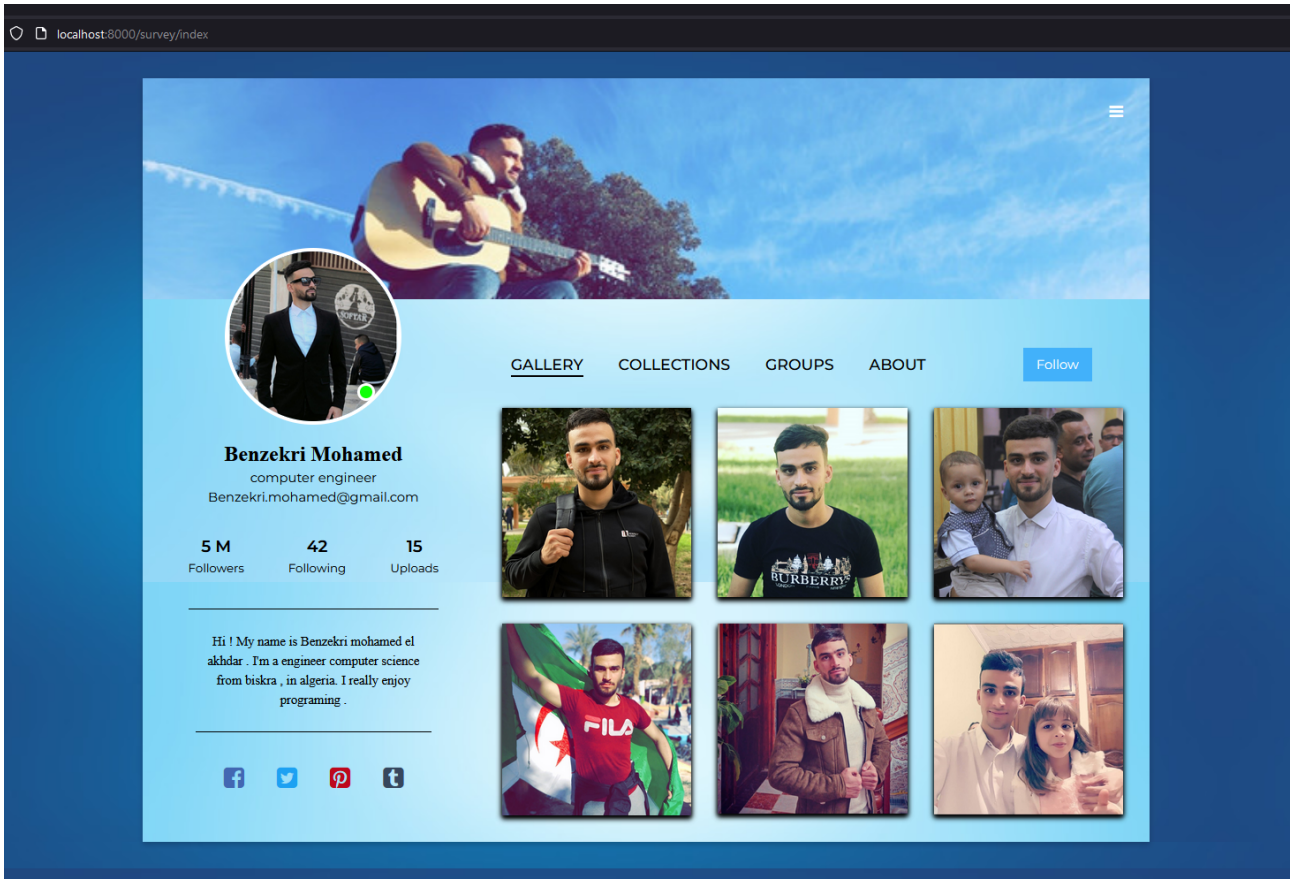

Figure 4.29: interface of User profile

4.5.5 Application After Prevention

Example 1

SQL injection :

In this example, we will apply SQL injection by attacker to enter the data of users. by using SQL injection attack : like this form : **"select * from users where id = 1 +$+ or 1 = 1 – 1 "**

other SQL examples didn't excite in dataset :

- hi') or ('a' = 'a
- ") or true–
- ") or sleep (TIME) = "
- select * from users where id = 1 or "?;" or 1 = 1 – 1

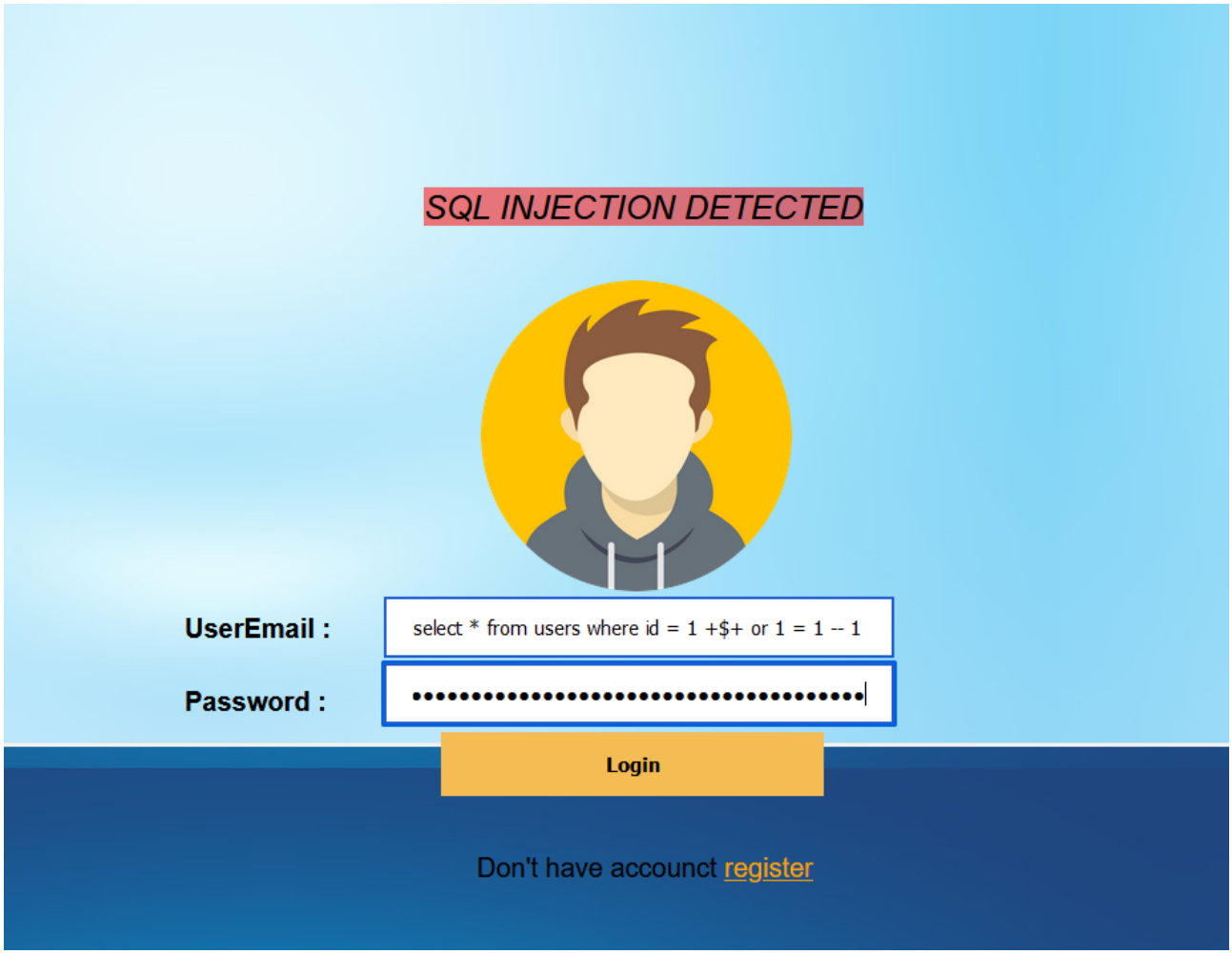

Figure 4.30: SQL injection try "New User"

Example 2

XSS injection :

In this example, we will apply XSS injection by attacker to enter the data of users. by using SQL injection attack : like this form **" ¡a onblur=alert(1) tabindex=1 id=x¿¡/a¿¡input autofocus¿ "**

other XSS examples didn't excite in dataset :

- $< aonblur = alert(1) tabindex = 1 id = x >< /a >< input autofocus >$
- $" < style > @keyframes x from left : 0; to left : 1000px; : target animation : 10 sease - in - out 0s 1x; < /style >< acronym id = x style = "position : absolute;" on animationcancel = "alert(1)" >< /acronym >$
- $< style > @keyframes slidein < /style >< dialog style = "animation - duration : 1s; animation - name : slidein; animation - iteration - count : 2" on animationiteration = "alert(1)" >< /dialog >$
- $< body on pointermove = alert(1) > XSS < /body >$

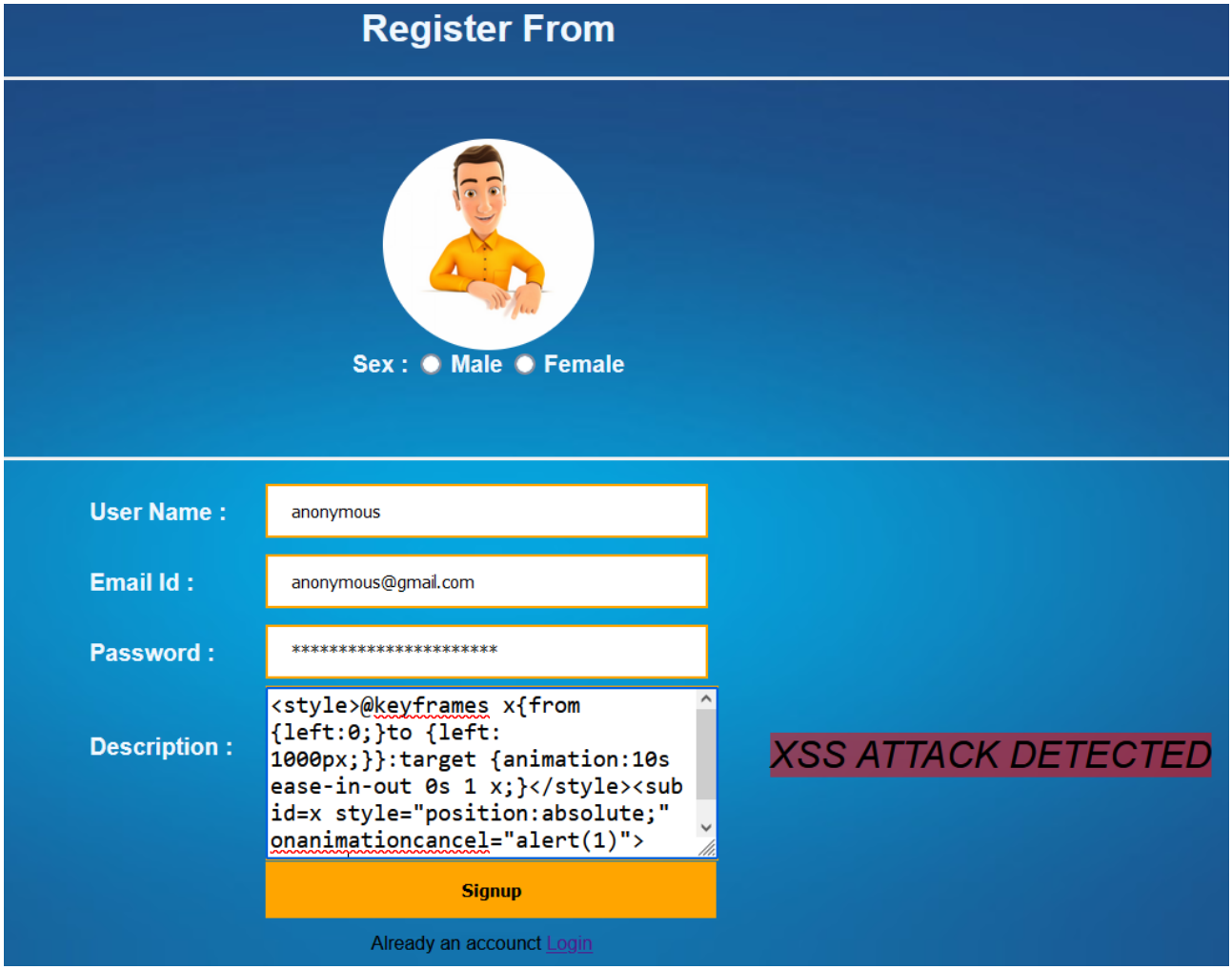

Figure 4.31: XSS injection in web site

4.6 Conclusion

In this chapter, we have described the implementation of our system, where we present the environment and the development tools we used. We used a database with different commands and a form. The database training and classification model was applied with the used parameters and tested, and we also showed the use of our model, such as we also provided graphical interfaces for our system. Finally, we explained the experiments and the obtained results .

General conclusion

Preventing vulnerabilities in web applications is not an easy task as hackers and attackers find new ways to break protections, making these techniques obsolete, researchers plan to apply machine learning . In the cybersecurity field to provide more dynamic and robust protection for new types of attacks that we have never seen before.
Machine learning is still a developing field in the field of cyber security, and there is a lack of open source libraries, frameworks, and tools to use for threat and attack issues.
In this work, we proposed a method to prevent two vulnerabilities of injection (SQL and XSS) common and dangerous that occurs in Web applications using relational database systems. The results were effective.
To continue work on this project, we could increase the size of the dataset by collecting more or even using the ones already built to increase efficiency. We could also manipulate the hyper parameters and recycle the model to get better results, we can also add protection against vulnerabilities other than SQL injection and XSS injection, such as "XSRF", "remote code execution", "path disclosure", "command injection", "open redirect" and other common and dangerous vulnerabilities.

Bibliography

[1] Deep learning, 06. 19 ,2022. URL: https://mc.ai/understanding-deep-learning-dnn-rnn-lstm-cnn-and-r-cnn/.

[2] python, urldate:06.19, 2022. URL: https://www.python.org/.

[3] Supervised machine learning, urldate:06.19, 2022. URL: https://tutorialforbeginner.com/supervised-machine-learning.

[4] types of machine learning methods, urldate:06.19, 2022. URL: https://www.sneakernews66.top/products.aspx?cname=types+of+machine+learning+methods&cid=66.

[5] What is deep learning?, urldate:06.19, 2022. URL: https://www.mathworks.com/discovery/deep-learning.html.

[6] Meet django, urldate:06.19, jun,20,2022. URL: https://www.djangoproject.com/.

[7] altexsoft. Web application architecture. 2019.

[8] A beginner's course for Biologists and Bioinformatics students. Adding a dynamic layer – introducing the php programming language, urldate:06.19, 2022. URL: http://www.cellbiol.com/bioinformatics_web_development/chapter-4-adding-a-dynamic-layer-introducing-the-php-programming-language/.

[9] Yoshua Bengio and Aaron Courville. *Ian Goodfellow,Deep learning*. MIT .Press, 2016.

[10] Diego Calvo. Unsupervised learning, urldate:06.19, Mar 24, 2019. URL: https://www.diegocalvo.es/en/learning-non-supervised/.

[11] Isaiah Chua. Real life examples of web vulnerabilities (owasp top 10), urldate:06.19, JAN 4 2022. URL: https://www.dnsstuff.com/intrusion-detection-system.

[12] Staff Contributor. What is an intrusion detection system? latest types and tools, urldate:06.19, 2019. URL: https://www.dnsstuff.com/intrusion-detection-system.

[13] Staff Contributor. Things you need to know about cyber attacks, threats risks, urldate:06.19, 2022. URL: https://blog.ecosystm360.com/cyber-attacks-threats-risks/.

[14] TechTarget Contributor. cryptology. 2022.

[15] Daniel. Web applications, urldate:06.19, 01.20.2014. URL: https://www.WebapplicationAbout.com.

[16] Daniel. *A Guide to Building Secure Web Applications*. The Open Web Application Security Project, 2015.

[17] Iryna Deremuk. Modern web application architecture explained: Components, best practices and more, urldate:06.19, Apr 23, 2021. URL: https://litslink.com/blog/web-application-architecture.

[18] Iryna Deremuk. Modern web application architecture explained: Components, best practices and more. Apr 23, 2021.

[19] Dr Mme. Boukhlouf Djemaa. *Cours Sécurité des systèmes d'Information et Web , Cours of university Mohamed Khider Biskra.* 2021.

[20] Seyed Mohammad Ghaffarian, Hamid Reza Shahriari. Software vulnerability analysis, discovery using machine learning, and 50(4):56 data-mining techniques: A survey. ACM Computing Surveys (CSUR). . Springer, 2017.

[21] Sébastien Gioria. Introduction à la sécurité des web services, confoo, montréal, canada. 10 Mars 2011.

[22] zhejiang hangzhou. Literature review on vulnerability detection using nlp technology. 23 Apr 2021.

[23] Hunter Heidenreich. What are the types of machine learning, urldate:06.19, 2018. URL: https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f.

[24] Hunter Heidenreich. What are the types of machine learning?, urldate:06.19, Dec 4, 2018. URL: https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f.

[25] Daniel Johnson. What is tensorflow?, urldate:06.19, May 14, 2022. URL: https://www.guru99.com/what-is-tensorflow.html/.

[26] karabaghli mouatez bellah. A deep learning approach for the analysis of feelings on social networks. 2019.

[27] Rebecca L. Automated vulnerability detection in source code using deep representation learninge. April, 2018.

[28] Anton Logvinenko. How to secure web applications from vulnerabilities in 2022. 2022.

[29] Marcus A Maloof. Machine learning and data mining for computer security :methods and applications. Springer, 2006.

[30] mark donson. Web applications security, urldate:06.19, 2020. URL: https://www.trendmicro.com/vinfo/us/security/definition/Vulnerability.

[31] Daniel Milodin. Non security – premise of cybercrime. April, 2002.

[32] Joëlle MUSSET. *Sécurité informatique Ethical Hacking Apprendre l'attaque pour mieux se défendre»Editions ENI.* 2009.

[33] Marius Nestor. Shoreline firewall beginners guide, urldate:06.19, Oct 20, 2006. URL: https://news.softpedia.com/news/Shoreline-Firewall-Beginners-Guide-38434.shtml.

[34] NVD. National vulnerability database, urldate:06.19, 2022. URL: https://nvd.nist.gov/general.

[35] Brendan Murphy Patrick Morrison, Kim Herzig and Laurie Williams. Challenges with applying vulnerability prediction models. In Proceedings of the 2015 Symposium and Bootcamp on the Science of Security, page 4, 2015.

[36] Prabanjan Raja. What is google colab? 28 Aug 2021.

[37] Lei Hamilton Tomo Lazovich Jacob Harer Onur Ozdemir Paul Ellingwood Rebecca Russell, Louis Kim, Marc McConley. Automated vulnerability detection in source code using deep representation learning. 17th IEEE International Conference on Machine Learning, and 2018. Applications (ICMLA), pages 757–762. . Springer, 2018.

[38] Aram Hovsepyan Riccardo Scandariato, James Walden and 40(10):993–1006 Wouter Joosen. Predicting vulnerable software components via text mining. IEEE Transactions on Software Engineering. . Springer, 2014.

[39] Mau Ruanova. Keras, urldate:06.19, Oct 14, 2020. URL: https://keras.io/.

[40] William Scott. Tf-idf from scratch in python on a real-world dataset. 15 Feb 2019.

[41] Dinesh Thakur. What is data encryption in dbms?, urldate:06.19, 2022. URL: https://ecomputernotes.com/database-system/adv-database/data-encryption.

[42] Rafael D Tordecilla. Vulnerabilities related to technological fields. urldate:06.19, 2021.

[43] Upasana. Machine learning algorithms. May 2022.

[44] verizon. Antivirus definition, urldate:06.19, 2022. URL: https://www.verizon.com/info/definitions/antivirus/.

[45] web site. Introduction to web application and web terminology, urldate:06.19, 20122. URL: https://dotnettutorials.net/lesson/introduction-to-web-application/.