Informe Evaluación Sumativa 03

“Mejoras en la Seguridad del Algoritmo de Consultas a Pacientes: Aplicación de SAMM y SDL”

NOMBRE	Alex Camus, Antonio Morales
CARRERA	Ingeniería en Informática
ASIGNATURA	[TI3V11] Introducción a la Programación Segura
PROFESOR	Luis Yáñez
FECHA	24/07/24

Índice

1 Introducción

En el contexto del desarrollo de software seguro, la implementación de medidas adecuadas es crucial para proteger los datos sensibles y garantizar la eficiencia operativa de las aplicaciones.

Este informe se centra en la modificación de un algoritmo de consultas a pacientes, con el objetivo de fortalecer sus defensas y asegurar su comportamiento óptimo.

Utilizando las metodologías de Security Assurance Maturity Model (SAMM) y Security Development Lifecycle (SDL), se han aplicado prácticas y controles específicos para abordar aspectos críticos como la confidencialidad, la integridad de los datos y la trazabilidad de las operaciones.

A lo largo de este documento, se detallan las modificaciones realizadas, las tecnologías utilizadas y se justifican las decisiones tomadas para optimizar la seguridad y el rendimiento del sistema.

2 Descripción e Impacto del Problema

El algoritmo de consultas a pacientes presenta vulnerabilidades significativas en la manipulación y visualización de datos sensibles, lo cual requiere mejoras urgentes en términos de seguridad.

A continuación, se describen los problemas identificados y el impacto potencial de no abordar estas vulnerabilidades.

2.1 Confidencialidad

Descripción: El programa permite acceso no autorizado a cualquier usuario que tenga conocimientos sobre cómo operar el software y manipular sus datos.

Impacto: Los pacientes pueden generar preocupaciones sobre la seguridad y divulgación de sus datos, afectar negativamente su confianza en los servicios de salud y en los profesionales médicos. Como consecuencia, puede llevar a demandas legales por reparación de daños y multas regulatorias.

2.2 Vulnerabilidad

Descripción: El programa presenta deficiencias en la validación de entrada de datos, lo que permite la inyección de código malicioso y comprometer la integridad de los datos de los pacientes.

Impacto: Puede llevar a accesos y modificaciones de registros médicos no autorizados, además de poner en riesgo la confidencialidad de la información de los pacientes.

2.3 Trazabilidad y No Repudio

Descripción: El programa carece de un mecanismo para registrar y auditar las acciones realizadas por los usuarios, lo que dificulta la detección de actividades maliciosas o errores involuntarios.

Impacto: Limita la capacidad de investigar incidentes de seguridad, responder a auditorías regulatorias y garantizar el cumplimiento de normativas de protección de datos.

2.4 Confiabilidad

Descripción: El programa no distingue en el ingreso de datos incoherentes, como por ejemplo edad superior a 120 años, nombres de más de 100 caracteres, o un Rut que no siga la estructura `12345678-9`.

Impacto: Puede dañar la reputación de la organización y afectar negativamente la confianza de los pacientes y su percepción de la calidad del servicio médico ofrecido.

3 Metodología

Para implementar un desarrollo seguro en el programa de consultas de pacientes, se siguieron los siguientes pasos:

3.1 Pruebas Exhaustivas

Se llevaron a cabo pruebas exhaustivas para identificar y localizar errores, vulnerabilidades y posibles puntos débiles en el sistema.

3.2 Detección de Errores y Vulnerabilidades

- Rut distinto a estructura `12345678-9`: *no confiable*
- Nombre mayor a 100 caracteres: *no confiable*
- Edad superior a 120 años: *no confiable*
- Opción de menú distinto a 1,2,3,4: *valueError*
- Ingreso de datos incoherentes (scripts): *vulnerable*
- Acceso indiscriminado al programa: *vulnerable*

3.3 SAMM y SDL

Se aplicaron las metodologías SAMM y SDL vistas en clases para solucionar los errores y vulnerabilidades encontrados y así garantizar un desarrollo seguro y robusto del programa.

4 Implementación de 3 Medidas según SAMM

4.1 Evaluación Inicial (SAMM Level 1)

- Se realizan las distintas pruebas exhaustivas para localizar errores, mencionados anteriormente.
- Se propone el diseño de funciones que permitan validar los datos ingresados.
- Se propone el uso de `try-except` para los distintos tipos de errores.
- Se propone el uso de librerías para la sanitización de datos y evitar inyección de código malicioso.
- Se propone el diseño de una función para restringir el acceso al programa a usuarios designados.
- Se propone el uso de tablas hash para encriptar las contraseñas y así evitar la fuga de datos sensibles.

4.2 Definición de Estrategia (SAMM Level 2)

4.2.1 Creación de Políticas de Seguridad Claras: Login para Médicos

- Se crea un diccionario `DIC_usuarios` con 3 usuarios y sus contraseñas.
- Se utiliza el módulo `hashlib` para encriptar las contraseñas, mediante el método `sha256()` que acepta como parámetro la contraseña en formato `bytes`. Para la conversión a `bytes` se utiliza el método `encode()`.
- Se solicita ingresar un nombre de usuario y contraseña, para compararlos con el diccionario utilizando la función `iniciarSesion()`
- Si la función devuelve `False`, imprime `"Usuario o Contraseña NO VÁLIDOS"`
- En caso contrario continua la ejecución del programa.

```
DIC_usuarios = {
        'user1': hashlib.sha256('pass1'.encode()).hexdigest(),
        'user2': hashlib.sha256('pass2'.encode()).hexdigest(),
        'user3': hashlib.sha256('pass3'.encode()).hexdigest()
}

usuario = input('Ingrese nombre de usuario: ')
password = input('Ingrese contraseña: ')

if not iniciarSesion(DIC_usuarios,usuario,password):
    print('Usuario o Contraseña NO VÁLIDOS')

else:
    ...
    ...
    ...
```

4.3 Implementación (SAMM Level 3)

4.3.1 Integración de Controles de Seguridad: Manejo de Errores

- Se implementa la estructura `try-except` para el manejo de errores, en el caso que el usuario ingrese una opción distinta a un valor numérico entero, provocando un error de tipo `ValueError`.

```
    while op != 4:
    mostrarMenu()
    try:
        op = int(input('Ingrese Opción: '))

      except ValueError:
        print('\nError Opción NO VALIDA')
        logger.error(f'Error Opción NO VALIDA: {op}')

      else:
        ...
        ...
```

- También se implementa `try-except` para la función `validarRut()` en el caso que el usuario ingrese un Rut que no tenga la forma `12345678-9`, ya que esto puede generar un error de tipo `ValueError` al calcular la suma de los productos entre el dígito del rut y su factor.

```
def validarRut(rut):
    try:
        ultDigito = rut[-1].upper()
        rut = rut[:-2].zfill(8)
        sum = 0
        for i,j in zip(rut,'32765432'):
            sum = sum + (int(i) * int(j))
        decimal = (sum/11) - int(sum/11)
        digito = round(11-(11*decimal))
        digito = 'K' if (digito == 10) else 0 if (digito == 11) else digito
        return (str(digito) == ultDigito)
    except ValueError:
        return False
```

4.3.2 Integración de Controles de Seguridad: Implementación de Registros

- Se implementa una función `configurarLogger`, que registra todas las acciones realizadas por el usuario.
- Genera un archivo de texto `log_user#.txt`, donde `#` es 1, 2, o 3 según el usuario que inicia sesión.
- El registro imprime cada acción en el formato: `tiempo - tipo de acción - mensaje`

```
def configurarLogger(usuario):
    logger = logging.getLogger('miLogger')
    logger.setLevel(logging.DEBUG)

    file_handler = logging.FileHandler(f'log_{usuario}.txt')
    file_handler.setLevel(logging.DEBUG)

    formato = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
    file_handler.setFormatter(formato)
    logger.addHandler(file_handler)
    return logger
```

5 Cambios Realizados según SDL

5.1 Validación de Datos de Entrada

Se diseñan las distintas funciones que permiten solucionar los errores y debilidades encontrados en la evaluación inicial en *SAMM Level 1*

5.1.1 Rut

- Se reutiliza el código de la evaluación 2 del ramo para validar un Rut como dato de entrada.
- Se agrega `try-except` en caso que el Rut ingresado no sea de la forma `12345678-9`

```
def validarRut(rut):
    try:
        ultDigito = rut[-1].upper()
        rut = rut[:-2].zfill(8)
        sum = 0
        for i,j in zip(rut,'32765432'):
            sum = sum + (int(i) * int(j))
        decimal = (sum/11) - int(sum/11)
        digito = round(11-(11*decimal))
        digito = 'K' if (digito == 10) else 0 if (digito == 11) else digito
        return (str(digito) == ultDigito)
    except ValueError:
        return False
```

5.1.2 Nombre

- Se diseña una función para validar el nombre como dato de entrada.
- Si el nombre ingresado contiene solo espacios en blancos o supera los 100 caracteres, devuelve `False`.
- En caso contrario devuelve `True`.

```
def validarNombre(nombre):
    nombre_null = len(nombre.strip()) == 0
    nombre_overflow = len(nombre) > 100
    if nombre_null or nombre_overflow:
        valido = False
    else:
        valido = True
    return valido
```

5.1.3 Edad

- Se diseña una función para validar la edad como dato de entrada.
- Si el tipo de dato no es numérico, devuelve `False`.
- Si la edad ingresada no está en el rango $0 <$ edad $<= 120$, devuelve `False`.
- En caso contrario, devuelve `True`.

```
def validarEdad(edad):
    try:
        int(edad)
    except:
        valido = False
    else:
        edad = int(edad)
        if (edad > 0) and (edad <= 120):
            valido = True
        else:
            valido = False

        return valido
```

5.1.4 Diagnóstico

- Se diseña una función para validar el diagnóstico como dato de entrada.
- Si el dato es una secuencia de espacios en blanco o su número de caracteres es mayor a 100, devuelve `False`.
- En caso contrario devuelve `True`.

```
def validarDiagnostico(diag):
    diag_null = len(diag.strip()) == 0
    diag_overflow = len(diag) > 100
    if diag_null or diag_overflow:
        valido = False
    else:
        valido = True
    return valido
```

5.2 Sanitización de Datos

- Se diseña una función para prevenir la inyección de código malicioso.
- Se elimina cualquier exceso de espacios en blanco al inicio y final del dato utilizando el método `strip()`
- Se localizan y eliminan caracteres que permitan activar instrucciones de código. Se utiliza el módulo `re` para expresiones regulares.
- Se utiliza el método `escape()` del módulo `html` para *escapar* caracteres especiales y no se interpreten como código HTML.

```
def sanitizar(dato):
    dato = dato.strip()
    dato = re.sub(r'<.*?>', '', dato)
    dato = html.escape(dato)
    dato = re.sub(r'[^\w\s@.-]','',dato)
    return dato
```

5.3 Autenticación

- Se diseña una función que permita comparar los datos ingresados de usuario y contraseña con los de un diccionario.
- Si el usuario ingresado está en el diccionario, encripta la contraseña ingresada y la compara con la del diccionario.
- Si son iguales devuelve `True`, en caso contrario devuelve `False`.

```
def iniciarSesion(DIC_usuarios,usuario,password):
    if usuario in DIC_usuarios:
        password_hash = hashlib.sha256(password.encode()).hexdigest()
        valido = DIC_usuarios[usuario] == password_hash
    else:
        valido = False
    return valido
```

6 Uso y Funcionamiento de Librerías y Módulos

6.1 html

Posee una función `escape()`. Permite *escapar* caracteres especiales que se puedan interpretar como código HTML, por ejemplo `<script>` que permite introducir código en javascript.

Es un módulo de la librería estándar de Python. Se utiliza con la siguiente instrucción:

```
import html
```

6.2 re

Posee una función `sub()`. Permite reemplazar las ocurrencias de un patrón de expresión regular que pueden ser dañinos para el programa y datos.

Es un módulo que se encuentra en la librería estándar de Python.

```
import re
```

6.3 hashlib

Posee una función `sha256()`. Permite encriptar el parámetro ingresado de tipo `byte`, a un hash de 256 bits.

Es un módulo de la librería estándar de Python.

```
import hashlib
```

6.4 logging

Es un módulo de la librería estándar de Python. Posee distintos métodos y clases que facilitan el diseño de un registro.

- `Logger()`: Clase. Permite configurar el comportamiento del registro y emitir mensajes.
- `FileHandler()`: Clase. Escribe datos de los registros a un archivo, para su posterior lectura.
- `Formater()`: Clase. Define el formato de salida del registro.
- `getLogger()`: Método. Devuelve una instancia de la clase `Logger()`.
- `setLevel()`: Método. Establece el nivel de severidad. Para el diseño solo se utilizan 2: INFO, ERROR.

```
import logging
```

7 Pruebas de Aceptación

7.1 Confidencialidad

1. Ingresar usuario incorrecto.

```
Ingrese nombre de usuario: asldkfj
Ingrese contraseña: pass1
Usuario o Contraseña NO VALIDOS
```

2. Ingresar contraseña incorrecta.

```
Ingrese nombre de usuario: user1
Ingrese contraseña: asdfkajl
Usuario o Contraseña NO VALIDOS
```

7.2 Vulnerabilidad

1. Ingresar una etiqueta html `<script>console.log('startHack')</script>`

```
Ingrese nombre de usuario: user1
Ingrese contraseña: pass1

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 11111111-1
Ingrese Nombre: <script>console.log('startHack')</script>
Ingrese Edad: 25
Diagnóstico: cancer etapa 3

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 11111111-1
Rut        : 11111111-1
Nombre     : console.logx27startHackx27
Edad       : 25 años
Diagnóstico: cancer etapa 3
```

2. Ingresar nombre con espacios extras al inicio y final.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 22222222-2
Ingrese Nombre:                           Jose Diaz
Ingrese Edad: 30
Diagnóstico: lupus

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 22222222-2
Rut        : 22222222-2
Nombre     : Jose Diaz
Edad       : 30 años
Diagnóstico: lupus
```

3. Ingresar nombre mayor a 100 caracteres.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 33333333-3
Ingrese Nombre: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
El nombre NO ES VALIDO
```

4. Ingresar diagnóstico con espacios extras al inicio y final.

```
Ingrese Rut: 44444444-4
Ingrese Nombre: Matias
Ingrese Edad: 30
Diagnóstico:                           Trastorno de Estrés Postraumático

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 44444444-4
Rut        : 44444444-4
Nombre     : Matias
Edad       : 30 años
Diagnóstico: Trastorno de Estrés Postraumático
```

5. Ingresar un dato como cadena de texto de espacios.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 33333333-3
Ingrese Nombre:
El nombre NO ES VALIDO
```

6. Ingresar diagnóstico mayor a 100 caracteres.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 11111111-1
Ingrese Nombre: Jorge
Ingrese Edad: 21
Diagnóstico: ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
ddddddddddddddddddddddddddddddddddddddddd
El diagnóstico NO ES VALIDO
```

7. Ingresar edad fuera del rango 0-120 años.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 55555555-5
Ingrese Nombre: Ximena
Ingrese Edad: 2500
La edad NO ES VALIDA
```

7.3 Trazabilidad y No Repudio

1. Al finalizar sesión de usuario, revisar registro de acciones realizadas.

```
Ingrese nombre de usuario: user3
Ingrese contraseña: pass3

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 44444444-4
Ingrese Nombre: Pedro
Ingrese Edad: 45
Diagnóstico: Anemia

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 11111111-1
El paciente consultado no se encuentra registrado

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 44444444-4
Rut        : 44444444-4
Nombre     : Pedro
Edad       : 45 años
Diagnóstico: Anemia

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 3

Ingrese Rut a eliminar: 44444444-4
Se ha eliminado la atención del paciente

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 4

Fin de las atenciones, los registros serán validados y traspasados
```

```
2024-07-24 01:33:38,252 - INFO - Inicio de sesión como user3
2024-07-24 01:34:13,526 - INFO - Paciente Ingresado: 44444444-4, ['Pedro', '45',
'Anemia']
2024-07-24 01:34:21,958 - ERROR - El paciente consultado no se encuentra registr
ado: 11111111-1
```

```
2024-07-24 01:34:35,362 - INFO - Consulta realizada para paciente 44444444-4
2024-07-24 01:34:48,156 - INFO - Paciente eliminado: 44444444-4
2024-07-24 01:34:50,402 - INFO - Fin de las atenciones
```

7.4 Confiabilidad

1. Ingresar una opción del menú fuera del rango 1-4.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 50

Opción NO VALIDA
```

2. Ingresar una opción del menú incoherente.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: asdlñgkbj3qlñkj3242l35kj

Error Opción NO VALIDA
```

3. Ingresar rut incoherente.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 2lkj452lñknñlaf's03lknjm111111-3
El rut NO ES VALIDO
```

4. Ingresar edad incoherente.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 55555555-5
Ingrese Nombre: Iris
Ingrese Edad: flñk234lk23j45'ganm
La edad NO ES VALIDA
```

5. Ingresar el rut que no está en la tupla de pacientes.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 1

Ingrese Rut: 999999999-9
El rut ingresado no es un paciente agendado
```

6. Consultar el rut que no está en la tupla de pacientes.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 66666666-6
El rut ingresado no es un paciente agendado
```

7. Consultar el rut de un paciente no registrado (paciente pero aun sin datos).

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 2

Ingrese Rut a consultar: 66666666-6
El rut ingresado no es un paciente agendado
```

8. Eliminar un rut que no está en la tupla de pacientes.

```
1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 3

Ingrese Rut a eliminar: 77777777-7
El rut ingresado no es un paciente agendado
```

9. Eliminar un rut de un paciente no agendado.

```
Ingrese nombre de usuario: user2
Ingrese contraseña: pass2

1. Iniciar Atención  2. Consultar Atención  3. Eliminar Atención  4. Salir
Ingrese Opción: 3

Ingrese Rut a eliminar: 33333333-3
El paciente consultado no se encuentra registrado
```

8 Conclusiones

Siguiendo las metodologías SAMM y SDL, hemos mejorado notablemente la seguridad del algoritmo de consultas a pacientes. Ahora, el sistema protege mucho mejor la confidencialidad e integridad de los datos y permite rastrear todas las operaciones realizadas. Las nuevas validaciones aseguran que solo se ingresen datos coherentes y válidos, y las autenticaciones robustas previenen accesos no autorizados. Además, hemos implementado mecanismos para evitar la inyección de código malicioso y añadido un sistema de registros para auditar las acciones de los usuarios. Estos cambios no solo optimizan la seguridad y el rendimiento del sistema, sino que también generan una mayor confianza en los usuarios sobre el manejo de su información sensible. En resumen, estas mejoras ofrecen un entorno más seguro y eficiente para la gestión de datos de pacientes, siguiendo las mejores prácticas de desarrollo de software seguro.

9 Bibliografía

- Clase 22: Desarrollo Seguro, Introducción a la Programación Segura, Profesor Luis Yáñez.
- Clase 23: Encriptación, Introducción a la Programación Segura, Profesor Luis Yáñez.
- Visión General del Modelo SAMM, https://owaspsamm.org/es/model/
- El Ciclo de Vida del Desarrollo Seguro, https://www.blmovil.com/ciclo-de-vida-del-desarrollo-seguro-sdl/
- Módulo `re` - expresiones regulares, https://rico-schmidt.name/pymotw-3/re/index.html
- `html.escape()` documentation, https://docs.python.org/3/library/html.html
- `logging` - Logging facility for Python, https://docs.python.org/3/library/logging.html
- `hashlib` - Secure hashes and message digest, https://docs.python.org/3/library/hashlib.html