```
import os
import mimetypes
import tkinter as tk
from tkinter import filedialog
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

DOWNLOADS_FOLDER = os.path.expanduser("~/Downloads")

class DownloadHandler(FileSystemEventHandler):
    def on_created(self, event):
        if not event.is_directory:
            display_output(f"[📩 New Download] {event.src_path}")
            scan_file(event.src_path)

def display_output(text):
    output_box.config(state="normal")
    output_box.insert(tk.END, text + "\n")
    output_box.see(tk.END)
    output_box.config(state="disabled")

def scan_file(filepath):
    # Extract the filename from the given file path
    filename = os.path.basename(filepath)
    try:
        # Retreive file size in bytes
        size_bytes = os.path.getsize(filepath)
        # Convert file size to a human-readable form
        if size_bytes >= 1024 * 1024:
            size_str = f"{size_bytes / (1024 * 1024):.2f} MB"
        elif size_bytes >= 1024:
            size_str = f"{size_bytes / 1024:.2f} KB"
        else:
            size_str = f"{size_bytes} Bytes"
    except Exception as e:
        # Handle any errors that occur while retrieving the file size
        size_str = "Unknown"
        display_output(f"⚠️ Could not determine file size: {e}")

    filetype, _ = mimetypes.guess_type(filepath)
    ext = os.path.splitext(filename)[1].lower()
    risky_exts = ['.exe', '.js', '.bat', '.vbs', '.scr', '.ps1']

    result = [
        "----- FILE SCAN -----",
        f"📄 Name: {filename}",
        f"📁 Type: {filetype or 'Unknown'}",
        f"📦 Size: {size_str}",
        "⚠️  Risk Level: HIGH - Executable or potentially harmful file." if ext in
risky_exts else "✅ Risk Level: LOW - Common file type.",
        "----------------------"
    ]
    display_output("\n".join(result))

observer = None

def start_file_watcher():
    global observer
    event_handler = DownloadHandler()
    observer = Observer()
    observer.schedule(event_handler, DOWNLOADS_FOLDER, recursive=False)
```

```
    observer.start()
    display_output("✅ File watcher enabled.")

def stop_file_watcher():
    global observer
    if observer:
        observer.stop()
        observer.join()
        observer = None
        display_output("🛑 File watcher disabled.")

def manual_file_scan():
    filepath = filedialog.askopenfilename()
    if filepath:
        display_output(f"📂 Manually scanning: {filepath}")
        scan_file(filepath)

print("Created by Samuel Zizzo")

window = tk.Tk()
window.title("ScanPro File Scanner")
window.geometry("500x400")

icon_path = os.path.join("assets", "fire_file_icon.png")
try:
    icon = tk.PhotoImage(file=icon_path)
    window.iconphoto(True, icon)
except Exception:
    pass  # Simplified: Removed unnecessary error message for icon loading

title_frame = tk.Frame(window, bg="#f0f0f0")
title_frame.pack(anchor="n", pady=10)

try:
    small_icon = tk.PhotoImage(file=icon_path).subsample(10, 10)
    title_label = tk.Label(
        title_frame,
        text="ScanPro File Scanner",
        image=small_icon,
        compound="left",
        font=("Helvetica", 20, "bold"),
        bg="#f0f0f0",
        fg="black"
    )
    title_label.image = small_icon
    title_label.pack()
except Exception:
    pass  # Simplified: Removed unnecessary error message for small icon loading

top_frame = tk.Frame(window)
top_frame.pack(anchor="nw", padx=10, pady=10, fill="x")

toggle_button = tk.Button(top_frame, text="🟢 Enable File Watcher", command=lambda:
enable_watcher())
toggle_button.pack(side="left")

clear_button = tk.Button(top_frame, text="🧹 Clear Output", command=lambda: clear_output())
clear_button.pack(side="right", padx=10)

label = tk.Label(window, text="Click to scan a file manually:", pady=5)
label.pack()

scan_button = tk.Button(window, text="📂 Choose File to Scan", command=manual_file_scan)
```

```
scan_button.pack(pady=5)

output_box = tk.Text(window, height=10, wrap="word", state="disabled")
output_box.pack(padx=10, pady=(10, 0), fill="both", expand=True)

output_box.config(state="normal")
output_box.insert(tk.END, "Created by Samuel Zizzo\n")
output_box.config(state="disabled")

def enable_watcher():
    start_file_watcher()
    toggle_button.config(text="🔴 Disable File Watcher", command=disable_watcher)

def disable_watcher():
    stop_file_watcher()
    toggle_button.config(text="🟢 Enable File Watcher", command=enable_watcher)

def clear_output():
    output_box.config(state="normal")
    output_box.delete(1.0, tk.END)
    output_box.config(state="disabled")

if __name__ == "__main__":
    window.mainloop()
```