**Project**

**MAC Spoofing Tool**

**CS4061**

**Ethical Hacking Concepts and Practices**

**Submitted by:** Faizan Pervaz

**Roll number:** 20I-0565

**Date:** 09-03-2024

**Table of Contents**

[ **Introduction** 3](#_Toc143364655)

[ **CodeStructure** 3](#_Toc143364656)

[ **Usage Instructions** 3](#_Toc143364656)

[ **Summary** 3](#_Toc143364657)

[ **References** 3](#_Toc143364658)

# **Introduction**

# The MAC Address Spoofing Tool is a Python script designed to provide a graphical user interface (GUI) for manipulating MAC addresses on Linux systems. It allows users to perform various tasks such as displaying developer information, viewing and changing MAC addresses, scanning the network for devices, assigning random MAC addresses, and resetting to the original MAC address.

# **Code Structure**

The code is organized into several functions, each serving a specific purpose:

1. **load\_mac\_vendor\_mappings(xml\_file):**
   * Loads MAC address vendor mappings from an XML file into a dictionary.
   * I have taken the database of mac address from cisco and used them for mac address spoofing.
2. def load\_mac\_vendor\_mappings(xml\_file):
3. try:
4. tree = ET.parse(xml\_file)
5. root = tree.getroot()
6. namespace = root.tag.split('}')[0] + '}'
7. for mapping in root.findall(namespace + 'VendorMapping'):
8. mac\_prefix = mapping.attrib['mac\_prefix']
9. vendor\_name = mapping.attrib['vendor\_name']
10. mac\_addresses\_full[mac\_prefix] = vendor\_name
11. except Exception as e:
12. print(f"Error loading MAC address vendor mappings: {e}")
13. **my\_info\_show():**
    * Displays developer information including name, roll number, degree, campus, course, and current date and time.
14. def my\_info\_show():
15. info = """
16. Faizan Pervaz
17. 20i-0565
18. CS B
19. BS Computer Science
20. ISB Campus
21. Ethical Hacking
22. Date and Time: {}
23. """.format(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
24. tk.messagebox.showinfo("Developer Information", info)
25. **network\_info\_cmd():**
    * Determines the appropriate command (**ifconfig** or **ip addr**) for retrieving network interface information based on availability.
26. def network\_info\_cmd():
27. if 'ifconfig' in subprocess.getoutput('which ifconfig'):
28. return 'ifconfig'
29. elif 'ip' in subprocess.getoutput('which ip'):
30. return 'ip addr'
31. else:
32. raise OSError("Neither 'ifconfig' nor 'ip' found on the system.")
33. **display\_current\_mac():**
    * Retrieves and displays the current MAC address of the system.
34. def display\_current\_mac():
35. try:
36. command = network\_info\_cmd()
37. output = subprocess.check\_output(command.split()).decode('utf-8')
38. if command == 'ifconfig':
39. mac\_address = re.search(r"ether (\S+)", output).group(1)
40. else:  # 'ip addr' command output parsing
41. mac\_address = re.search(r"link/ether (\S+)", output).group(1)
42. tk.messagebox.showinfo("MAC Address", f"MAC Address: {mac\_address}")
43. except subprocess.CalledProcessError as e:
44. tk.messagebox.showerror("Error", f"An error occurred while trying to get MAC address: {e}")
45. except Exception as e:
46. tk.messagebox.showerror("Error", f"An unexpected error occurred: {e}")
47. **get\_original\_mac(interface):**
    * Retrieves the original MAC address of a specified network interface.
48. def get\_original\_mac(interface):
49. try:
50. command = network\_info\_cmd()
51. output = subprocess.check\_output(command.split()).decode('utf-8')
52. if command == 'ifconfig':
53. match = re.search(r"ether (\S+)", output)
54. else:  # 'ip addr' command output parsing
55. match = re.search(r"link/ether (\S+)", output)
57. if match:
58. return match.group(1)
59. else:
60. tk.messagebox.showerror("Error", "MAC address not found.")
61. except subprocess.CalledProcessError as e:
62. tk.messagebox.showerror("Error", f"An error occurred while trying to get original MAC address: {e}")
63. except Exception as e:
64. tk.messagebox.showerror("Error", f"An unexpected error occurred: {e}")
65. **change\_mac\_linux(interface, new\_mac):**
    * Changes the MAC address of a specified network interface on Linux systems.
66. def change\_mac\_linux(interface, new\_mac):
67. try:
68. # Store the original MAC address before making any changes
69. if interface not in original\_mac\_addresses:
70. original\_mac\_addresses[interface] = get\_original\_mac(interface)
71. # Change the MAC address
72. command\_down = ["sudo", "ifconfig", interface, "down"]
73. subprocess.run(command\_down, check=True)
75. command\_change = ["sudo", "ifconfig", interface, "hw", "ether", new\_mac]
76. subprocess.run(command\_change, check=True)
78. command\_up = ["sudo", "ifconfig", interface, "up"]
79. subprocess.run(command\_up, check=True)
81. tk.messagebox.showinfo("Success", "MAC address has been changed successfully.")
82. except subprocess.CalledProcessError as e:
83. tk.messagebox.showerror("Error", f"An error occurred while trying to change MAC address: {e}")
84. except Exception as e:
85. tk.messagebox.showerror("Error", f"An unexpected error occurred: {e}")
86. **apply\_random\_mac\_address():**
    * Assigns a randomly generated MAC address to a specified network interface.
87. def apply\_random\_mac\_address():
88. interface = simpledialog.askstring("Interface", "Enter the network interface name (e.g., wlan0):")
89. if interface:
90. random\_mac = ':'.join(['{:02x}'.format(random.randint(0x00, 0xff)) for \_ in range(6)])
91. change\_mac\_linux(interface, random\_mac)
92. **reset\_to\_original\_mac():**
    * Resets the MAC address of a specified network interface to its original value.
93. def reset\_to\_original\_mac():
94. interface = simpledialog.askstring("Interface", "Enter the network interface name (e.g., wlan0):")
95. if interface:
96. if interface in original\_mac\_addresses:
97. original\_mac = original\_mac\_addresses[interface]
98. change\_mac\_linux(interface, original\_mac)
99. else:
100. tk.messagebox.showinfo("Info", "Original MAC address not found. No changes made.")
101. **select\_and\_change\_mac():**
     * Allows the user to select a MAC address from a list and change it on the specified network interface.
102. def select\_and\_change\_mac():
103. def on\_select(evt):
104. # Check if there is a selection first to prevent IndexError
105. if not lb.curselection():
106. tk.messagebox.showerror("Error", "No MAC address selected.")
107. return
108. index = int(lb.curselection()[0])
109. selected\_mac = list(filtered\_mac\_addresses.keys())[index]
110. vendor\_name = filtered\_mac\_addresses[selected\_mac]
111. interface = simpledialog.askstring("Interface", "Enter the network interface name (e.g., wlan0):")
112. if interface:
113. change\_mac\_linux(interface, selected\_mac)
114. tk.messagebox.showinfo("MAC Address Changed", f"MAC address has been changed to {selected\_mac} ({vendor\_name})")
115. top.destroy()
116. def on\_search():
117. search\_text = search\_entry.get().lower()
118. filtered\_mac\_addresses.clear()
119. for mac, vendor in mac\_addresses\_full.items():
120. if search\_text in vendor.lower():
121. filtered\_mac\_addresses[mac] = vendor
122. update\_listbox()
123. def update\_listbox():
124. lb.delete(0, tk.END)
125. for mac, vendor in filtered\_mac\_addresses.items():
126. lb.insert(tk.END, f"{mac} - {vendor}")
127. top = Toplevel()
128. top.title("Select MAC Address")
129. search\_frame = tk.Frame(top)
130. search\_frame.pack(fill=tk.X)
131. search\_label = tk.Label(search\_frame, text="Search Vendor Name:")
132. search\_label.pack(side=tk.LEFT, padx=(5, 0))
133. search\_entry = tk.Entry(search\_frame)
134. search\_entry.pack(side=tk.LEFT, padx=5, fill=tk.X, expand=True)
135. search\_button = tk.Button(search\_frame, text="Search", command=on\_search)
136. search\_button.pack(side=tk.LEFT, padx=(0, 5))
137. lb = Listbox(top, width=50, height=20)
138. lb.pack(padx=5, pady=5, fill=tk.BOTH, expand=True)
139. scrollbar = Scrollbar(top, orient=tk.VERTICAL)
140. scrollbar.config(command=lb.yview)
141. scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
142. filtered\_mac\_addresses = mac\_addresses\_full.copy()
143. update\_listbox()
144. lb.bind('<<ListboxSelect>>', on\_select)
145. **scan\_network(network\_cidr):**
     * Scans the network for devices using ARP requests and returns their MAC addresses.
146. def scan\_network(network\_cidr):
147. """
148. Scans the network for devices using ARP requests and attempts to resolve their hostnames.
149. :param network\_cidr: The network CIDR notation (e.g., '192.168.1.0/24') to scan.
150. """
151. packet = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=network\_cidr)
152. result, \_ = srp(packet, timeout=3, verbose=0)
154. mac\_addresses = []
155. for \_, received in result:
156. mac\_addresses.append(received.hwsrc)
157. return mac\_addresses
158. **scan\_network\_and\_display():**
     * Scans the network for devices and displays their MAC addresses in a listbox.
159. def scan\_network\_and\_display():
160. try:
161. local\_ip = socket.gethostbyname(socket.gethostname())
162. network\_cidr = '.'.join(local\_ip.split('.')[:3]) + '.0/24'
163. except Exception:
164. network\_cidr = '192.168.1.0/24'  # Fallback CIDR, change as needed
165. mac\_addresses = scan\_network(network\_cidr)
166. # Display the results in a listbox
167. top = Toplevel()
168. top.title("Scanned MAC Addresses")
169. lb = Listbox(top, width=50, height=20)
170. lb.pack(padx=5, pady=5, fill=tk.BOTH, expand=True)
171. scrollbar = Scrollbar(top, orient=tk.VERTICAL)
172. scrollbar.config(command=lb.yview)
173. scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
174. for mac in mac\_addresses:
175. lb.insert(tk.END, mac)
176. lb.config(yscrollcommand=scrollbar.set)
178. **display\_welcome\_screen() & second\_screen():**
     * Displays a welcome screen with a background image.
     * Displays the main interface of the MAC Address Spoofing Tool.
179. def display\_welcome\_screen():
180. welcome\_root = tk.Tk()
181. welcome\_root.title("Welcome to MAC Address Spoofing Tool")
182. welcome\_root.geometry("800x600")
183. welcome\_root.resizable(False, False)
184. background\_image = Image.open("hacker.jpeg")
185. background\_image = background\_image.resize((800, 600))
186. background\_photo = ImageTk.PhotoImage(background\_image)
188. background\_label = tk.Label(welcome\_root, image=background\_photo)
189. background\_label.place(x=0, y=0, relwidth=1, relheight=1)
191. welcome\_frame = tk.Frame(welcome\_root, bg="#ffffff", bd=5)
192. welcome\_frame.place(relx=0.5, rely=0.5, anchor='center')
193. welcome\_label = tk.Label(welcome\_frame, text="Welcome to MAC Address Spoofing Tool", font=("Arial", 18), bg="#ffffff")
194. welcome\_label.pack(pady=(20, 30))
195. continue\_button = tk.Button(welcome\_frame, text="Continue", command=welcome\_root.destroy, font=('Arial', 14), padx=15, pady=8)
196. continue\_button.pack()
197. welcome\_root.mainloop()
198. def second\_screen():
199. display\_welcome\_screen()
200. root = tk.Tk()
201. root.title("MAC Address Spoofing Tool")
202. root.geometry("800x600")
203. root.resizable(False, False)
204. load\_mac\_vendor\_mappings("vendorMacs.xml")
206. background\_image = Image.open("hacker.jpeg")
207. background\_image = background\_image.resize((800, 600))
208. background\_photo = ImageTk.PhotoImage(background\_image)
209. background\_label = tk.Label(root, image=background\_photo)
210. background\_label.place(x=0, y=0, relwidth=1, relheight=1)
211. frame = tk.Frame(root, bg="#ffffff", bd=5)
212. frame.place(relx=0.5, rely=0.5, anchor='center')
213. title\_label = tk.Label(frame, text="MAC Address Spoofing Tool", font=("Arial", 18), bg="#ffffff")
214. title\_label.pack(pady=(20, 30))
215. button\_font = ('Arial', 14)
216. left\_buttons\_frame = tk.Frame(frame, bg="#ffffff")
217. left\_buttons\_frame.pack(side=tk.LEFT, padx=20)
218. right\_buttons\_frame = tk.Frame(frame, bg="#ffffff")
219. right\_buttons\_frame.pack(side=tk.RIGHT, padx=20)
220. tk.Button(left\_buttons\_frame, text="Display Developer Info", command=my\_info\_show, font=button\_font, padx=15, pady=8).pack(fill=tk.X, pady=10)
221. tk.Button(left\_buttons\_frame, text="Display Current MAC", command=display\_current\_mac, font=button\_font, padx=15, pady=8).pack(fill=tk.X, pady=10)
222. tk.Button(left\_buttons\_frame, text="Change MAC From Manufacturer", command=select\_and\_change\_mac, font=button\_font, padx=15, pady=8).pack(fill=tk.X, pady=10)
223. tk.Button(right\_buttons\_frame, text="Network Scanning", command=scan\_network\_and\_display, font=button\_font, padx=15, pady=8).pack(fill=tk.X, pady=10)
224. tk.Button(right\_buttons\_frame, text="Assign Random MAC Address", command=apply\_random\_mac\_address, font=button\_font, padx=15, pady=8).pack(fill=tk.X, pady=10)
225. tk.Button(right\_buttons\_frame, text="Reset to Original MAC Address", command=reset\_to\_original\_mac, font=button\_font, padx=15, pady=8).pack(fill=tk.X, pady=10)
226. root.mainloop()

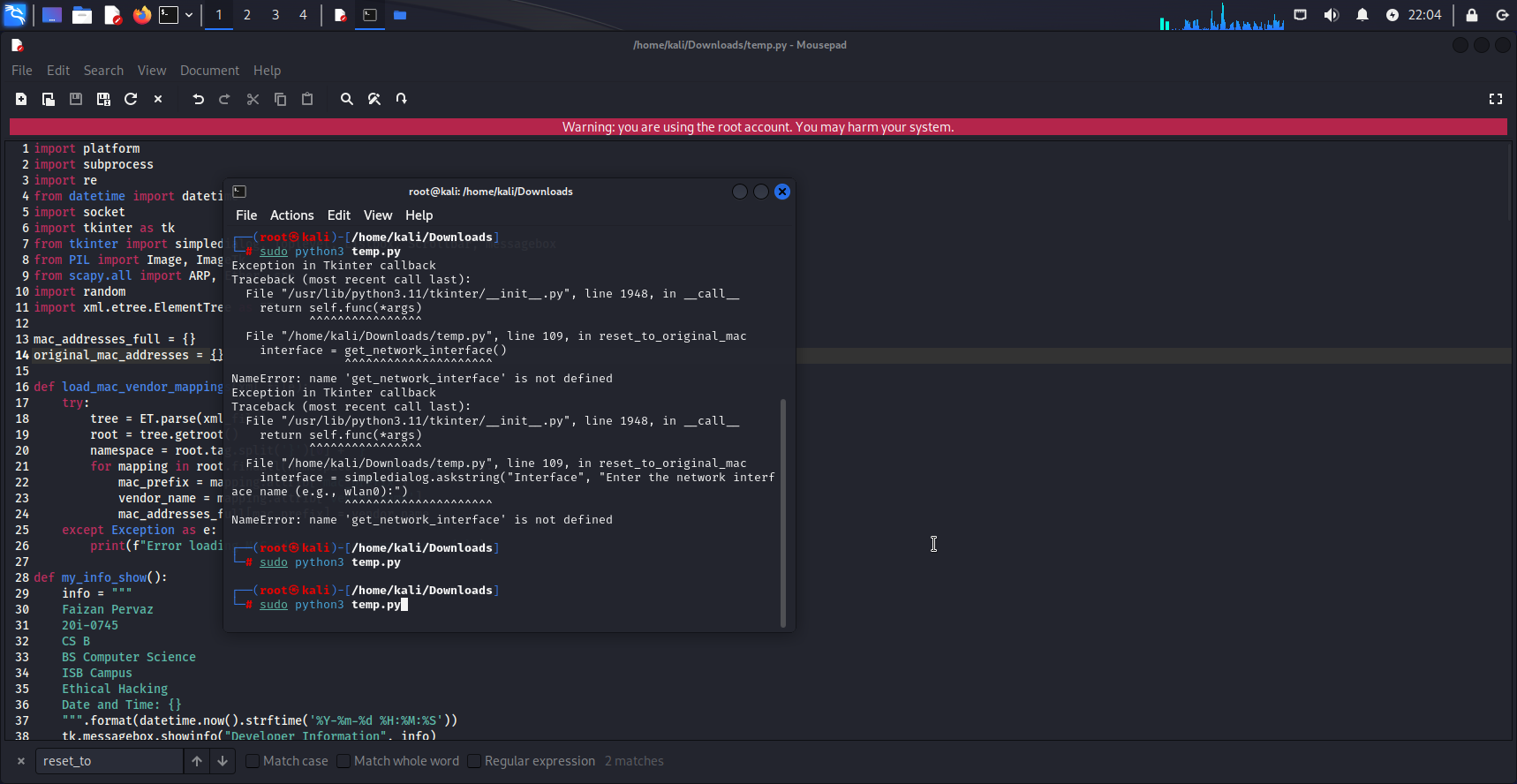
# **Usage Instructions**

1. Ensure Python and required dependencies (Tkinter, scapy, etc.) are installed.
2. Run the script in a Python environment.
3. Follow the on-screen instructions:
   * Use the buttons to perform various tasks such as displaying developer information, viewing current MAC address, changing MAC address, scanning the network, assigning random MAC address, and resetting to original MAC address.
   * Enter interface names when prompted.
   * Use the search functionality to filter MAC addresses.
   * Click the "Continue" button on the welcome screen to access the main interface.

# **Summary**

The MAC Address Spoofing Tool offers a convenient solution for managing MAC addresses on Linux systems, empowering users to enhance network security and privacy through effective MAC address manipulation.

Overall, the project provides a valuable tool for cybersecurity enthusiasts, network administrators, and individuals seeking to protect their privacy in network environments.



A computer screen shot of a person holding a keyhole

Description automatically generated

A screenshot of a computer

Description automatically generated

# **References**

* <https://devtools360.com/en/macaddress/vendorMacs.xml>
* <https://dnschecker.org/mac-lookup.php>
* <https://maclookup.app/>