Tejas Tripathi

CVSS Calculator

In recent years, cyber-attacks have emerged as one of the most significant threats facing organisations of all sizes. The Internet and other network operations have created risks that were non-existent less than a decade ago. When cyber-attacks (such as data breaches and hacks) occur, they can result in devastating damage, such as business disruptions, revenue loss, legal fees, a permanently tainted reputation, and more.

It is important to remember that no organisation is immune to the impact of cybercrime. As a result, cyber liability insurance has become an essential component to any risk management programme.

As a blue team design Risk calculator **(Exploitability Part only**) on basis of CVSS Matrix Equations mentioned below:

ISS =1 - [(1 - Confidentiality) × (1 - Integrity) × (1 - Availability)]

Impact:
If Scope is Unchanged --> 6.42 × ISS
If Scope is Changed --> 7.52 × (ISS - 0.029) - 3.25 × (ISS - 0.02)^15
Exploitability = 8.22 × AttackVector × AttackComplexity × PrivilegesRequired × UserInteraction

BaseScore:
If Impact \<= 0 --> 0, else
If Scope is Unchanged --> Roundup (Minimum [(Impact + Exploitability), 10])
If Scope is Changed --> Roundup (Minimum [1.08 × (Impact + Exploitability), 10])

Metric Values

Each metric value has an associated constant which is used in the formulas, as defined in Table

Metric	Metric Value	Numerical Value
Attack Vector / Modified Attack Vector	Network	0.85
	Adjacent	0.62
	Local	0.55
	Physical	0.2
Attack Complexity / Modified Attack Complexity	Low	0.77
	High	0.44
Privileges Required / Modified Privileges Required	None	0.85
	Low	0.62 (or 0.68 if Scope / Modified Scope is Changed)
	High	0.27 (or 0.5 if Scope / Modified Scope is Changed)
User Interaction / Modified User Interaction	None	0.85
	Required	0.62
Confidentiality / Integrity / Availability / Modified Confidentiality / Modified Integrity / Modified Availability	High	0.56
	Low	0.22
	None	0

Implementing the GUI

CODE:

```
from tkinter import *
import win32gui, win32con

hide = win32gui.GetForegroundWindow()
win32gui.ShowWindow(hide , win32con.SW_HIDE)

root = Tk()
root.geometry("800x600")
root.title("CYD Practical-3")

def Network():
   global av
   b1.configure(bg='red')
   b2.configure(bg='black', fg='green')
   b3.configure(bg='black', fg='green')
   b4.configure(bg='black', fg='green')
   av=0.85

def Adjacent():
   global av
   b2.configure(bg='red')
   b1.configure(bg='black', fg='green')
   b3.configure(bg='black', fg='green')
   b4.configure(bg='black', fg='green')
```

```
    av=0.62

def Local():
    global av
    b3.configure(bg='red')
    b2.configure(bg='black', fg='green')
    b1.configure(bg='black', fg='green')
    b4.configure(bg='black', fg='green')
    av=0.55

def Physical():
    global av
    b4.configure(bg='red')
    b2.configure(bg='black', fg='green')
    b3.configure(bg='black', fg='green')
    b1.configure(bg='black', fg='green')
    av=0.2

def Low():
    global ac
    b5.configure(bg='red')
    b6.configure(bg='black', fg='green')
    ac=0.77

def High():
    global ac
```

```
    b6.configure(bg='red')
    b5.configure(bg='black', fg='green')
    ac=0.44

def Lo():
    global pr
    b9.configure(bg='red')
    b8.configure(bg='black', fg='green')
    b7.configure(bg='black', fg='green')
    pr=0.62

def Hig():
    global pr
    b8.configure(bg='red')
    b7.configure(bg='black', fg='green')
    b9.configure(bg='black', fg='green')
    pr=0.27

def Non():
    global pr
    b7.configure(bg='red')
    b8.configure(bg='black', fg='green')
    b9.configure(bg='black', fg='green')
    pr=0.85

def No():
```

```
    global ui
    b10.configure(bg='red')
    b11.configure(bg='black', fg='green')
    ui=0.85

def Required():
    global ui
    b11.configure(bg='red')
    b10.configure(bg='black', fg='green')
    ui=0.62

def Unchanged():
    global s
    b12.configure(bg='red')
    b13.configure(bg='black', fg='green')
    s='Unchanged'

def Changed():
    global s
    b13.configure(bg='red')
    b12.configure(bg='black', fg='green')
    s='Changed'

def Locon():
    global confi
    b17.configure(bg='red')
```

```
    b15.configure(bg='black', fg='green')
    b16.configure(bg='black', fg='green')
    confi=0.22

def Higcon():
    global confi
    b16.configure(bg='red')
    b15.configure(bg='black', fg='green')
    b17.configure(bg='black', fg='green')
    confi=0.56

def Noncon():
    global confi
    b15.configure(bg='red')
    b16.configure(bg='black', fg='green')
    b17.configure(bg='black', fg='green')
    confi=0

def Loint():
    global inti
    b20.configure(bg='red')
    b19.configure(bg='black', fg='green')
    b18.configure(bg='black', fg='green')
    inti=0.22

def Higint():
```

```
    global inti
    b19.configure(bg='red')
    b18.configure(bg='black', fg='green')
    b20.configure(bg='black', fg='green')
    inti=0.56

def Nonint():
    global inti
    b18.configure(bg='red')
    b19.configure(bg='black', fg='green')
    b20.configure(bg='black', fg='green')
    inti=0



def Loava():
    global avai
    b23.configure(bg='red')
    b22.configure(bg='black', fg='green')
    b21.configure(bg='black', fg='green')
    avai=0.22

def Higava():
    global avai
    b22.configure(bg='red')
    b21.configure(bg='black', fg='green')
    b23.configure(bg='black', fg='green')
```

```
    avai=0.56


def Nonava():
    global avai
    b21.configure(bg='red')
    b22.configure(bg='black', fg='green')
    b23.configure(bg='black', fg='green')
    avai=0









def result():
    global pr
    global expre
    global impact
    global base
    if s=='Changed' and pr==0.62:
        pr=0.68
    elif s=='Changed' and pr==0.27:
        pr=0.5
    else:
        A=1
```

```
    expre=8.22*av*ac*pr*ui
    expre=round(expre, 2)
    iss=1-((1-confi)*(1-avai)*(1-inti))

    if s=='Changed':
        impact=7.52*(iss-0.029) - 3.25*(iss-0.02)**15
    else:
        impact=6.42*iss


    if impact<=0:
        base=0
    elif s=='Changed':
        base=round(min((1.08*(impact+expre)), 10))
    else:
        base=round(min((impact+expre), 10))

    impact=round(impact, 1)

    s_label.configure(text=f'Exploitability: {expre}\n\nImpact:
{impact}\n\nBase Score: {base}', font='comicsansms 10 bold')




#Title
title_label = Label(text ='''CVSS CALCULATOR''', bg ="black", fg="green",
```

```
  padx=1, pady=9, font="comicsansms 20 bold", borderwidth=10,
  relief=RIDGE)
title_label.grid(row=0, column=4)



#Attack Vector
av_label=Label(text='Attack Vector', font="comicsansms 10 bold")
av_label.grid(row=1, column=2, pady='15')

#Frame
f_av = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_av.grid(row=2, column=2)



# Buttons
b1 = Button(f_av, bg='black', fg="green", text="Network(N)",
          command=Network)
b1.grid(row=2, column=3, padx='4')

b2 = Button(f_av, bg='black', fg="green", text="Adjacent(A)",
          command=Adjacent)
b2.grid(row=2, column=4, padx='4')

b3 = Button(f_av, bg='black', fg="green", text="Local(L)",
```

```
        command=Local)
b3.grid(row=2, column=5, padx='4')

b4 = Button(f_av, bg='black', fg="green", text="Physical(P)",
        command=Physical)
b4.grid(row=2, column=6, padx='4')




# #Attack Complexity
ac_label=Label(text='Attack Complexity', font="comicsansms 10 bold")
ac_label.grid(row=3, column=2, pady='15')

#Frame
f_ac = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_ac.grid(row=4, column=2)




# Buttons
b5 = Button(f_ac, bg='black', fg="green", text="Low(L)",
        command=Low)
b5.grid(row=4, column=2, padx='4')
```

```
b6 = Button(f_ac, bg='black', fg="green", text="High(H)",
        command=High)
b6.grid(row=4, column=3, padx='4')




# Privileges Required(PR)
pr_label=Label(text='Privileges Required (PR)', font="comicsansms 10
bold")
pr_label.grid(row=5, column=2, pady='15')

#Frame
f_p = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_p.grid(row=6, column=2)




# Buttons
b7 = Button(f_p, bg='black', fg="green", text="None(N)",
        command=Non)
b7.grid(row=6, column=2, padx='4')

b8 = Button(f_p, bg='black', fg="green", text="High(H)",
        command=Hig)
```

```
b8.grid(row=6, column=3, padx='4')

b9 = Button(f_p, bg='black', fg="green", text="Low(L)",
        command=Lo)
b9.grid(row=6, column=4, padx='4')




#user Interface
ui_label=Label(text='User Interface (UI)', font="comicsansms 10 bold")
ui_label.grid(row=7, column=2, pady='15')

#Frame
f_ui = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_ui.grid(row=8, column=2)

# Buttons
b10 = Button(f_ui, bg='black', fg="green", text="None(N)",
        command=No)
b10.grid(row=8, column=2, padx='4')

b11 = Button(f_ui, bg='black', fg="green", text="Required(R)",
        command=Required)
```

```
b11.grid(row=8, column=3, padx='4')



#scope
s_label=Label(text='Scope (S)', font="comicsansms 10 bold")
s_label.grid(row=9, column=2, pady='15')

#Frame
f_s = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_s.grid(row=10, column=2)

# Buttons
b12 = Button(f_s, bg='black', fg="green", text="Unchanged(U)",
        command=Unchanged)
b12.grid(row=10, column=2, padx='4')

b13 = Button(f_s, bg='black', fg="green", text="Changed(C)",
        command=Changed)
b13.grid(row=10, column=3, padx='4')



# Confidentiality(C)
conf_label=Label(text='Confidentiality (C)', font="comicsansms 10 bold")
conf_label.grid(row=1, column=4, pady='15')
```

```
#Frame
f_con = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_con.grid(row=2, column=4)

# Buttons
b15 = Button(f_con, bg='black', fg="green", text="None(N)",
        command=Noncon)
b15.grid(row=2, column=4, padx='4')

b16 = Button(f_con, bg='black', fg="green", text="High(H)",
        command=Higcon)
b16.grid(row=2, column=5, padx='4')

b17 = Button(f_con, bg='black', fg="green", text="Low(L)",
        command=Locon)
b17.grid(row=2, column=6, padx='4')



# Intigrity(I)
inti_label=Label(text='Intigrity (I)', font="comicsansms 10 bold")
inti_label.grid(row=3, column=4, pady='15')

#Frame
f_inti = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
```

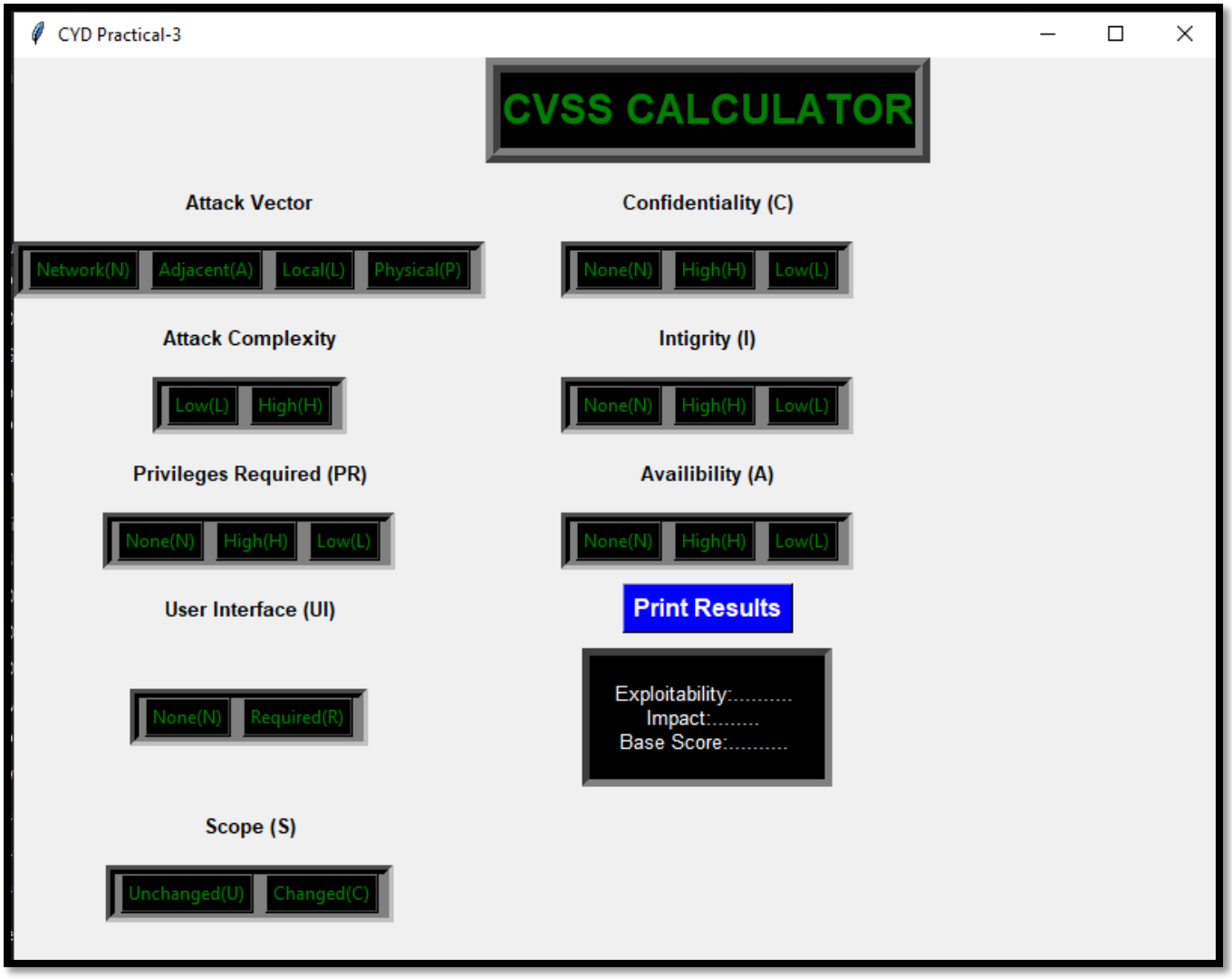
```
f_inti.grid(row=4, column=4)

# Buttons
b18 = Button(f_inti, bg='black', fg="green", text="None(N)",
        command=Nonint)
b18.grid(row=4, column=4, padx='4')

b19 = Button(f_inti, bg='black', fg="green", text="High(H)",
        command=Higint)
b19.grid(row=4, column=5, padx='4')

b20 = Button(f_inti, bg='black', fg="green", text="Low(L)",
        command=Loint)
b20.grid(row=4, column=6, padx='4')



# Availibility(A)
ava_label=Label(text='Availibility (A)', font="comicsansms 10 bold")
ava_label.grid(row=5, column=4, pady='15')

#Frame
f_ava = Frame(root, borderwidth=6, bg="grey", relief=SUNKEN)
f_ava.grid(row=6, column=4)
```

```
# Buttons
b21 = Button(f_ava, bg='black', fg="green", text="None(N)",
        command=Nonava)
b21.grid(row=6, column=4, padx='4')

b22 = Button(f_ava, bg='black', fg="green", text="High(H)",
        command=Higava)
b22.grid(row=6, column=5, padx='4')

b23 = Button(f_ava, bg='black', fg="green", text="Low(L)",
        command=Loava)
b23.grid(row=6, column=6, padx='4')



#getting the result
b14 = Button(bg='blue', fg="white", text="Print Results",
         font="comicsansms 12 bold", command=result)
b14.grid(row=7, column=4, padx='4')

f_res = Frame(root, borderwidth=9, bg="black", relief=SUNKEN)
f_res.grid(row=8, column=4)

s_label=Label(f_res, text='Exploitability:..... \nImpact:..... \nBase
Score:..... ',
         font='comicsansms 10', bg='black', fg='white')
s_label.grid(row=8, column=4, pady='10', padx='10')
```

root.mainloop()

OUTPUT:

Double click the python file



Now select the all the options and then click on the print results

Tejas Tripathi