Universidad Nacional Autónoma de México

Facultad de ingeniería

Sistemas operativos

Proyecto 3:

Asignación de memoria en un sistema real

Profesor: Gunnar Eyal Wolf Iszaevich

Alumnos: Christian Alejandro Leyva Mercado y Velasco Pachuca Bryan

Grupo: 06

Introducción:

El tercer proyecto consta en realizar una reimplementación del pmap que le permita al usuario ingresar un identificador de proceso *PID* y obtener de manera visual e identificable las secciones en memoria, indicando el tamaño de cada una, así como las correspondientes a bibliotecas del sistema y otros archivos en memoria con una representación visual que sea agradable para el usuario.

La funcionalidad se basa sobre una reimplementación de pmap, comando usado para mostrar el mapa de memoria de un proceso, en otras palabras, muestra en donde se encuentra distribuida la memoria del proceso.

Descripción del entorno:

- **¿Qué lenguaje se emplea?**
  Para el diseño del programa se empleó Python 3.9.10
- **¿Qué bibliotecas del lenguajes se emplean?**
  Se emplearon las bibliotecas:
  1. sys: Acceso a variables y funciones del intérprete de Python.
  2. webbrowser: Muestra archivos web directamente en el navegador.
  3. tabulate: Impresión de información en forma de tablas sencillas

  ****IMPORTANTE****

  Podría llegar a ser necesario contar con el módulos de Python *tabulate*, en caso de no contar con él, instalarlo a través de la siguiente instrucción en terminal:

  ```
  pip install tabulate
  ```

  Los módulos *sys* y *webbrowser* ya vienen por default con la instalación normal de Python desde su versión 2.7.

- **¿En qué sistema operativo se desarrolló?**
  Terminal de Ubuntu en Windows 10 y máquina virtual Debian/Kali Linux

Descripción del programa:

- **Descripción de los procesos:**
  - main():

```
def main():
    try:
        PID = getArgs()
    except:
        print("\nERROR:\n\tPor favor ingresa el PID del proceso a leer. \n
        return
    try:
        textoMap = leerMAP(PID)
    except:
        print("\nERROR:\n\tNo se pudo acceder al map, verifica que el PID
        return
    Pmap = ObtenerPmap(textoMap)
    PulirPMap(Pmap)
    # Para mostrar de manera 'bonita' el Pmap, se utiliza la biblioteca ta
    print(tabulate(Pmap,headers="firstrow",tablefmt='fancy_grid'))

    # A partir de aqui se comienza la realizacion de la tabla en html con
    filename = "Pmap_PID_" + PID + ".html"
    crearHTML(PID,createPmapHTML(Pmap),filename)
    # Se abre automáticamente el archivo generado:
    # Si no se desea que se abra automáticamente puede comentar la siguien
    webbrowser.open_new_tab(filename)
```

    Primer proceso en ejecutarse será el proceso principal sobre el que se llamarán a los demás procesos, valida que se haya ingresado un PID correcto y se pueda acceder correctamente a su mapa.
  - getArgs() y leerMap(PID):

```
def getArgs():
    return sys.argv[1]

def leerMAP(PID):
    ruta = "/proc/"+PID + "/smaps"
    with open(ruta, 'r') as archivoMap:
        textoMap = archivoMap.read()
    return textoMap
```

    El proceso *getArgs()* obtiene el PID que se proporciona cuando se ejecuta el programa (python proyecto3_chrisco.py **{PID}**) y el proceso *leerMAP(PID)* recibe el PID obtenido para buscar su mapa de proceso a partir de la ruta que se crea al unirlo con el comando *smaps /proc/**{PID}**/smaps*, regresa el texto que contiene toda la información del proceso.

- ObtenerPmap(textoLargo:str):

```
def ObtenerPmap(textoLargo:str):
    lineasPMAP = textoLargo.splitlines()
    salida = []
    i = 0
    out ={}
    out['uso'] = 'Uso'
    out['desde'] = 'De pág.'
    out['hasta'] = 'A pág.'
    out['size'] = 'Tamaño'
    out['sizeUnit'] = ''
    out['paginas'] = 'Paginas'
    out['permisos'] ='Permisos'
    out['mapeo'] = 'Uso o mapeo'
    salida.append(out)

    while i < len(lineasPMAP):
        out = {}
        primera = lineasPMAP[i].split(' ')
        out['desde'] = primera[0].split('-')[0]
        out['hasta'] = primera[0].split('-')[1]
        out['permisos'] = primera[1]
        out['mapeo'] = primera[-1]
        segunda = lineasPMAP[i+1].split(' ')
        out['size'] = segunda[-2]
        out['sizeUnit'] = segunda[-1].lower()
        salida.append(out)
        i += 23
    return salida
```

Tras haber obtenido el texto con toda la información este es procesado para separarlo en la información que se mostrará, la cual se almacena en un diccionario que será añadido a la lista que se retornará, en el diccionario se guardará desde que página a que página se encuentra un determinado elemento, su tamaño y cual es el mapeo que le corresponde.

El tipo de uso se añadirá más adelante cuando se analicen los permisos otorgados, al igual que obtener la cantidad de páginas, pues se requiere cierta aritmética que se explicará más adelante.

- PulirPMap(Pmap:List):

```
def PulirPMap(Pmap:list):
    for line in Pmap:
        if 'uso' in line:
            continue
        line['paginas'] = getnPaginas(int(line['size']),line['sizeUnit'])
        line['uso'] = getUso(line['mapeo'], line['permisos'])
```

El método *PulirPMap()* recibe la lista con la información previamente procesada para realizar el cálculo de la cantidad de páginas y el tipo de uso y guardarlos en sus respectivos apartados, nótese que se ignora la primer línea que contiene los títulos de cada columna.

- getnPaginas(valor,unidad):

```python
def getnPaginas(valor,unidad):
    if unidad == 'kb':
        return str(int(valor/4))

    if unidad == 'mb':
        return str((valor*1000)/4)

    if unidad == 'gb':
        return str((valor*1000000)/4)

    if unidad == 'tb':
        return str((valor*1000000000)/4)
```

Con el método *getnPaginas()* se realiza el cálculo de cuantas páginas corresponden por cada elemento, para esto se necesita conocer el tamaño en kilobytes que tiene cada uno, para casos donde se tienen elementos de tamaños iguales o mayores a megabytes se realiza la conversión a kilobytes, de tal manera que el resultado se divida entre 4 (considerando una paginación con páginas de 4 kilobytes) y se obtengan la cantidad de páginas.

- getUso(mapeo,permisos):

```
def getUso(mapeo,permisos):
    if mapeo == '[stack]':
        return "Stack"
    elif mapeo == '[heap]':
        return "Heap"
    elif mapeo == '[anon]':
        return 'Anónimo'
    elif mapeo in ('[vdso]', '[vsyscall]', '[vectors]'):
        return "Llamada al Sistema"
    elif mapeo == '[vvar]':
        return 'Procesos Var (Kernel)'
    elif mapeo == "":
        return "vació"
    elif permisos[0].lower() == 'r' and permisos[2].lower() == "x"
                                    and "/lib" in mapeo:
        return "Bib→Texto"
    elif permisos[0].lower() == 'r' and "lib" in mapeo:
        return "Bib→Datos"
    elif permisos[0].lower() == 'r' and permisos[2].lower() == "x"
                                    and "/bin" in mapeo:
        return "Texto"
    elif permisos[0].lower() == 'r' and "/bin" in mapeo:
        return "Datos"
    else:
        return 'Desconocido'
```

  Este método recibe el mapeo correspondiente al elemento (dirección del archivo) y sus permisos otorgados, con esto se determina el uso del elemento, revisando si corresponde al Stack, Heap, si es anónimo (listo para su uso, pero vacío), si es una llamada al sistema, si es un proceso del kernel o si es espacio vacío.
  Para los elementos no filtrados previamente, se realiza un análisis en sus permisos y su dirección, de tal manera que si tiene permisos read (r) y execution (x) su uso es de tipo texto, y si únicamente tiene permisos read (r) su uso es de tipo datos, solo quedaría determinar si pertenece a una biblioteca (/lib) o no (/bin).
- tabulate():

```
print(tabulate(Pmap,headers="firstrow",tablefmt='fancy_grid'))
```

  El método *tabulate()* recibe la lista con los elementos previamente obtenidos y se le indica que en la primera fila se encuentran los títulos de las columnas y se desea se imprima como tabla.
- crearHTML():

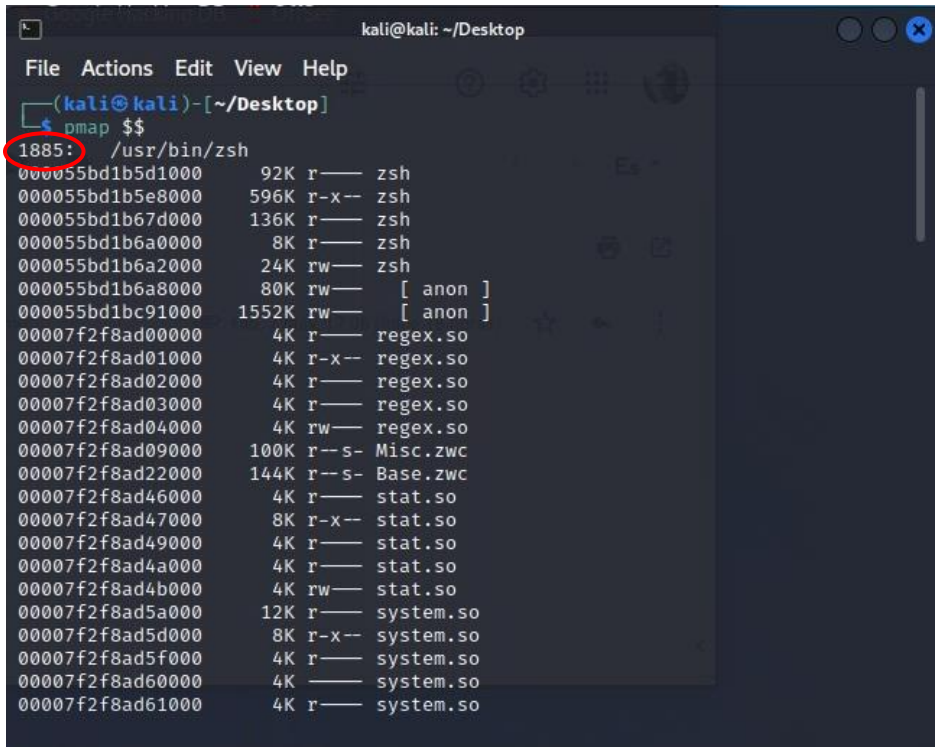
```
filename = "Pmap_PID_" + PID + ".html"
crearHTML(PID,createPmapHTML(Pmap),filename)
```

  El método *crearHTML() es el encargado* de crear un bonito diseño con la lista que contiene todos los elementos guardados y analizados previamente, para esto se crearon todos los métodos en el archivo ***PmapToHTML_chrisco.py*** cuyo funcionamiento por ser alejado al objetivo del proyecto no se explicará.

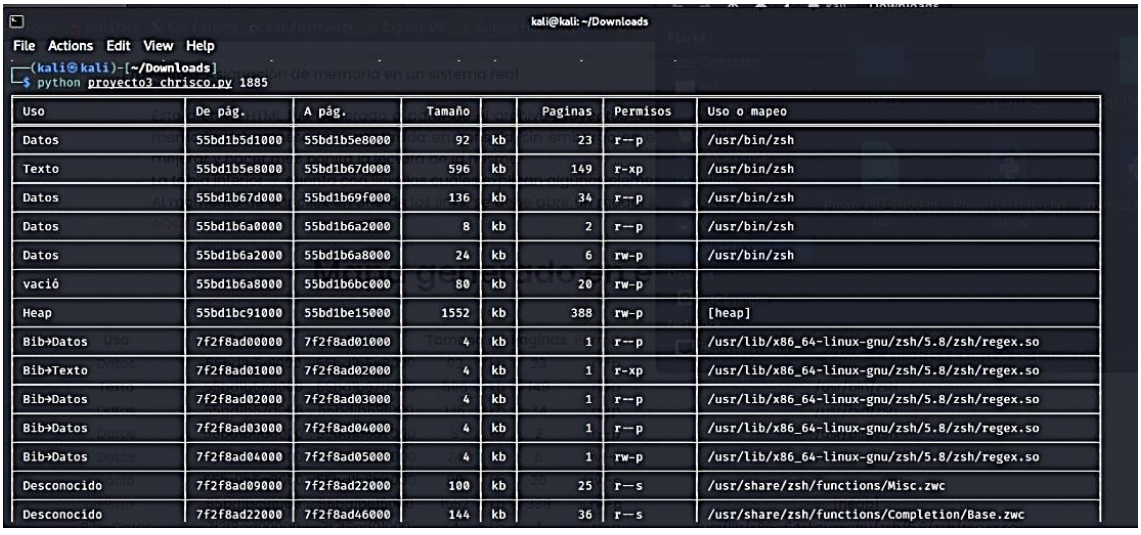
Capturas del programa en funcionamiento:

EJEMPLO 1:

Para el proceso 1855 del shell zsh:

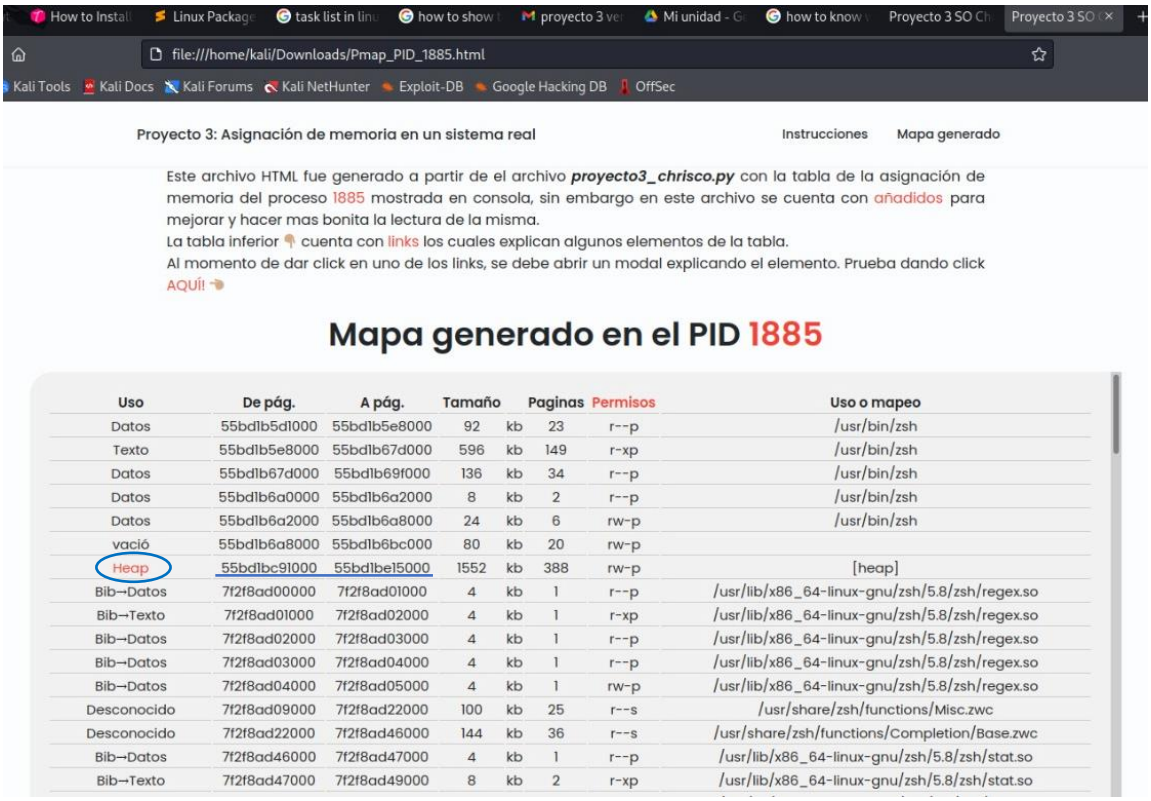


Se obtuvo el siguiente mapa de memoria:



Uso	De pág.	A pág.	Tamaño		Paginas	Permisos	Uso o mapeo
Datos	55bd1b5d1000	55bd1b5e8000	92	kb	23	r--p	/usr/bin/zsh
Texto	55bd1b5e8000	55bd1b67d000	596	kb	149	r-xp	/usr/bin/zsh
Datos	55bd1b67d000	55bd1b69f000	136	kb	34	r--p	/usr/bin/zsh
Datos	55bd1b6a0000	55bd1b6a2000	8	kb	2	r--p	/usr/bin/zsh
Datos	55bd1b6a2000	55bd1b6a8000	24	kb	6	rw-p	/usr/bin/zsh
vació	55bd1b6a8000	55bd1b6bc000	80	kb	20	rw-p	
Heap	55bd1bc91000	55bd1be15000	1552	kb	388	rw-p	[heap]
Bib→Datos	7f2f8ad00000	7f2f8ad01000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Texto	7f2f8ad01000	7f2f8ad02000	4	kb	1	r-xp	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Datos	7f2f8ad02000	7f2f8ad03000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Datos	7f2f8ad03000	7f2f8ad04000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Datos	7f2f8ad04000	7f2f8ad05000	4	kb	1	rw-p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Desconocido	7f2f8ad09000	7f2f8ad22000	100	kb	25	r--s	/usr/share/zsh/functions/Misc.zwc
Desconocido	7f2f8ad22000	7f2f8ad46000	144	kb	36	r--s	/usr/share/zsh/functions/Completion/Base.zwc

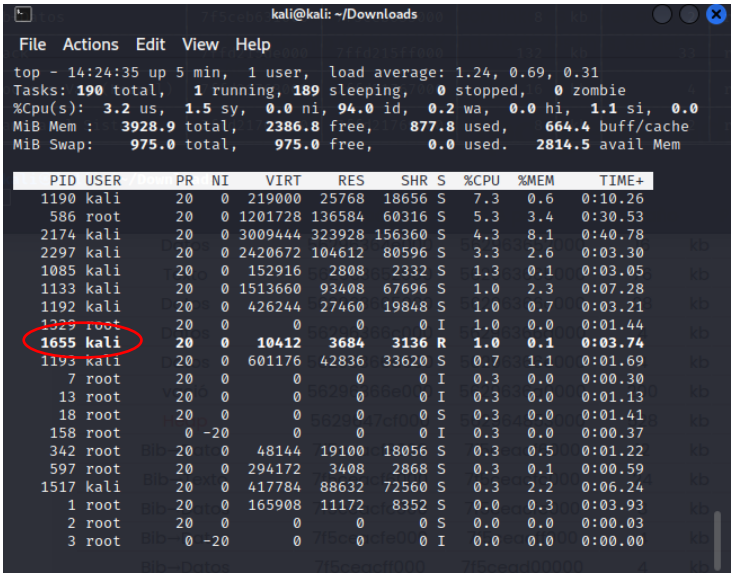
Y también se obtuvo el siguiente HTML, nótese la obtención de la dirección del heap:



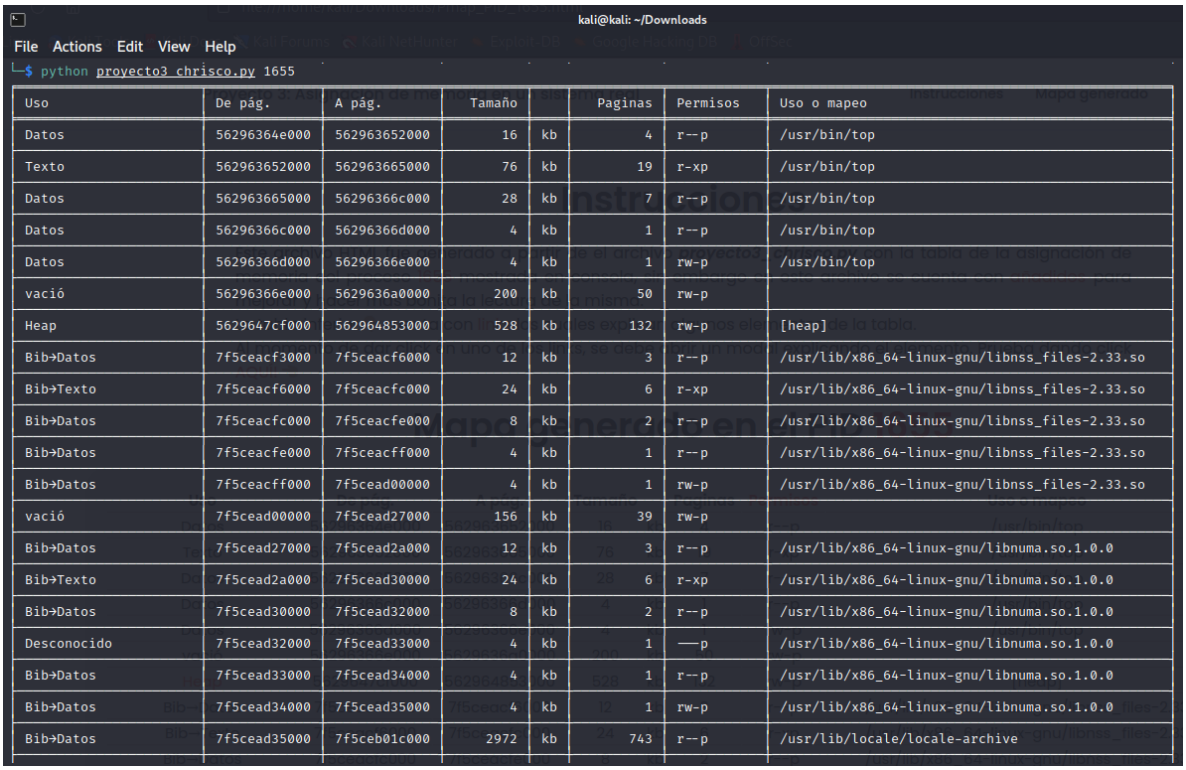
Uso	De pág.	A pág.	Tamaño		Paginas	Permisos	Uso o mapeo
Datos	55bd1b5d1000	55bd1b5e8000	92	kb	23	r--p	/usr/bin/zsh
Texto	55bd1b5e8000	55bd1b67d000	596	kb	149	r-xp	/usr/bin/zsh
Datos	55bd1b67d000	55bd1b69f000	136	kb	34	r--p	/usr/bin/zsh
Datos	55bd1b6a0000	55bd1b6a2000	8	kb	2	r--p	/usr/bin/zsh
Datos	55bd1b6a2000	55bd1b6a8000	24	kb	6	rw-p	/usr/bin/zsh
vació	55bd1b6a8000	55bd1b6bc000	80	kb	20	rw-p	
Heap	55bd1bc91000	55bd1be15000	1552	kb	388	rw-p	[heap]
Bib→Datos	7f2f8ad00000	7f2f8ad01000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Texto	7f2f8ad01000	7f2f8ad02000	4	kb	1	r-xp	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Datos	7f2f8ad02000	7f2f8ad03000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Datos	7f2f8ad03000	7f2f8ad04000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Bib→Datos	7f2f8ad04000	7f2f8ad05000	4	kb	1	rw-p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/regex.so
Desconocido	7f2f8ad09000	7f2f8ad22000	100	kb	25	r--s	/usr/share/zsh/functions/Misc.zwc
Desconocido	7f2f8ad22000	7f2f8ad46000	144	kb	36	r--s	/usr/share/zsh/functions/Completion/Base.zwc
Bib→Datos	7f2f8ad46000	7f2f8ad47000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/stat.so
Bib→Texto	7f2f8ad47000	7f2f8ad49000	8	kb	2	r-xp	/usr/lib/x86_64-linux-gnu/zsh/5.8/zsh/stat.so

EJEMPLO 2:

En los procesos en ejecución se seleccionó el 1655:



Con el que se obtuvo el siguiente mapa de memoria.



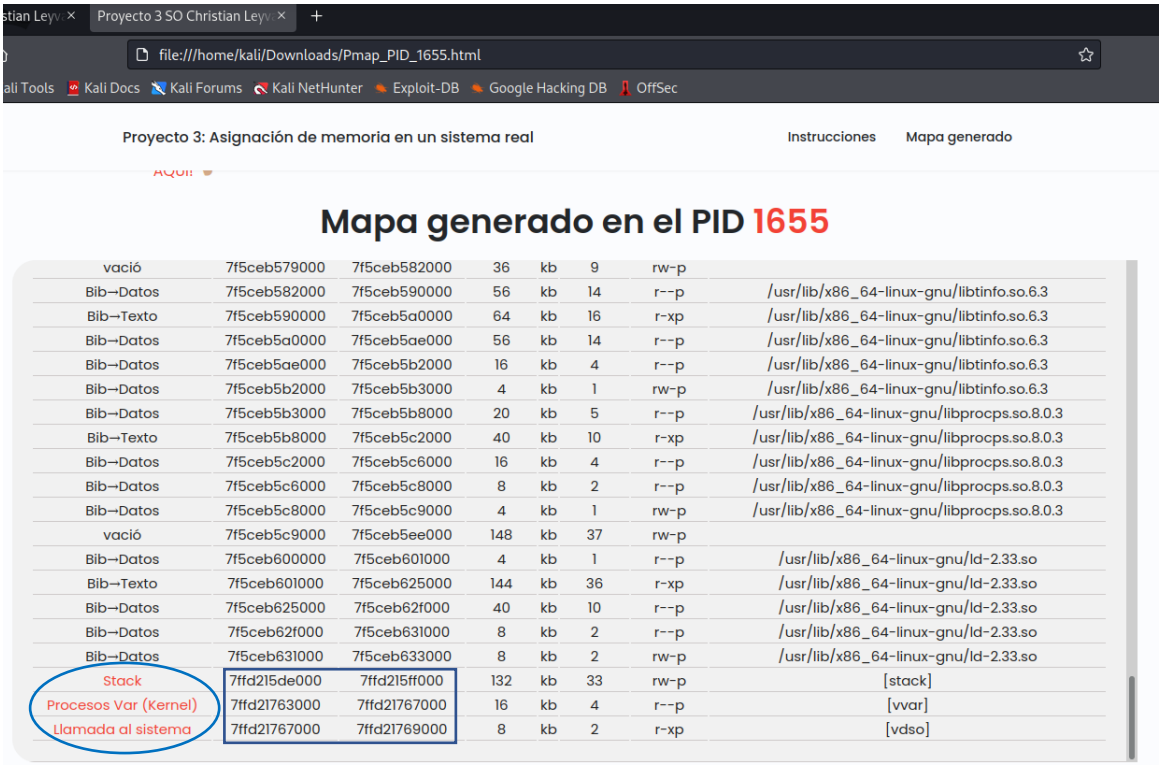
kali@kali: ~/Downloads

File Actions Edit View Help

```
└─$ python proyecto3_chrisco.py 1655
```

Uso	De pág.	A pág.	Tamaño		Paginas	Permisos	Uso o mapeo
Datos	56296364e000	562963652000	16	kb	4	r--p	/usr/bin/top
Texto	562963652000	562963665000	76	kb	19	r-xp	/usr/bin/top
Datos	562963665000	56296366c000	28	kb	7	r--p	/usr/bin/top
Datos	56296366c000	56296366d000	4	kb	1	r--p	/usr/bin/top
Datos	56296366d000	56296366e000	4	kb	1	rw-p	/usr/bin/top
vació	56296366e000	5629636a0000	200	kb	50	rw-p	
Heap	5629647cf000	562964853000	528	kb	132	rw-p	[heap]
Bib→Datos	7f5ceacf3000	7f5ceacf6000	12	kb	3	r--p	/usr/lib/x86_64-linux-gnu/libnss_files-2.33.so
Bib→Texto	7f5ceacf6000	7f5ceacfc000	24	kb	6	r-xp	/usr/lib/x86_64-linux-gnu/libnss_files-2.33.so
Bib→Datos	7f5ceacfc000	7f5ceacfe000	8	kb	2	r--p	/usr/lib/x86_64-linux-gnu/libnss_files-2.33.so
Bib→Datos	7f5ceacfe000	7f5ceacff000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/libnss_files-2.33.so
Bib→Datos	7f5ceacff000	7f5cead00000	4	kb	1	rw-p	/usr/lib/x86_64-linux-gnu/libnss_files-2.33.so
vació	7f5cead00000	7f5cead27000	156	kb	39	rw-p	
Bib→Datos	7f5cead27000	7f5cead2a000	12	kb	3	r--p	/usr/lib/x86_64-linux-gnu/libnuma.so.1.0.0
Bib→Texto	7f5cead2a000	7f5cead30000	24	kb	6	r-xp	/usr/lib/x86_64-linux-gnu/libnuma.so.1.0.0
Bib→Datos	7f5cead30000	7f5cead32000	8	kb	2	r--p	/usr/lib/x86_64-linux-gnu/libnuma.so.1.0.0
Desconocido	7f5cead32000	7f5cead33000	4	kb	1	---p	/usr/lib/x86_64-linux-gnu/libnuma.so.1.0.0
Bib→Datos	7f5cead33000	7f5cead34000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/libnuma.so.1.0.0
Bib→Datos	7f5cead34000	7f5cead35000	4	kb	1	rw-p	/usr/lib/x86_64-linux-gnu/libnuma.so.1.0.0
Bib→Datos	7f5cead35000	7f5ceb01c000	2972	kb	743	r--p	/usr/lib/locale/locale-archive

Y el siguiente HTML, nótese la obtención de la dirección del Stack, así como procesos del kernel y llamadas al sistema.



Proyecto 3 SO Christian Leyv

file:///home/kali/Downloads/Pmap_PID_1655.html

Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Proyecto 3: Asignación de memoria en un sistema real — Instrucciones — Mapa generado

Mapa generado en el PID 1655

vació	7f5ceb579000	7f5ceb582000	36	kb	9	rw-p	
Bib→Datos	7f5ceb582000	7f5ceb590000	56	kb	14	r--p	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
Bib→Texto	7f5ceb590000	7f5ceb5a0000	64	kb	16	r-xp	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
Bib→Datos	7f5ceb5a0000	7f5ceb5ae000	56	kb	14	r--p	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
Bib→Datos	7f5ceb5ae000	7f5ceb5b2000	16	kb	4	r--p	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
Bib→Datos	7f5ceb5b2000	7f5ceb5b3000	4	kb	1	rw-p	/usr/lib/x86_64-linux-gnu/libtinfo.so.6.3
Bib→Datos	7f5ceb5b3000	7f5ceb5b8000	20	kb	5	r--p	/usr/lib/x86_64-linux-gnu/libprocps.so.8.0.3
Bib→Texto	7f5ceb5b8000	7f5ceb5c2000	40	kb	10	r-xp	/usr/lib/x86_64-linux-gnu/libprocps.so.8.0.3
Bib→Datos	7f5ceb5c2000	7f5ceb5c6000	16	kb	4	r--p	/usr/lib/x86_64-linux-gnu/libprocps.so.8.0.3
Bib→Datos	7f5ceb5c6000	7f5ceb5c8000	8	kb	2	r--p	/usr/lib/x86_64-linux-gnu/libprocps.so.8.0.3
Bib→Datos	7f5ceb5c8000	7f5ceb5c9000	4	kb	1	rw-p	/usr/lib/x86_64-linux-gnu/libprocps.so.8.0.3
vació	7f5ceb5c9000	7f5ceb5ee000	148	kb	37	rw-p	
Bib→Datos	7f5ceb600000	7f5ceb601000	4	kb	1	r--p	/usr/lib/x86_64-linux-gnu/ld-2.33.so
Bib→Texto	7f5ceb601000	7f5ceb625000	144	kb	36	r-xp	/usr/lib/x86_64-linux-gnu/ld-2.33.so
Bib→Datos	7f5ceb625000	7f5ceb62f000	40	kb	10	r--p	/usr/lib/x86_64-linux-gnu/ld-2.33.so
Bib→Datos	7f5ceb62f000	7f5ceb631000	8	kb	2	r--p	/usr/lib/x86_64-linux-gnu/ld-2.33.so
Bib→Datos	7f5ceb631000	7f5ceb633000	8	kb	2	rw-p	/usr/lib/x86_64-linux-gnu/ld-2.33.so
Stack	7ffd215de000	7ffd215ff000	132	kb	33	rw-p	[stack]
Procesos Var (Kernel)	7ffd21763000	7ffd21767000	16	kb	4	r--p	[vvar]
Llamada al sistema	7ffd21767000	7ffd21769000	8	kb	2	r-xp	[vdso]

Referencias:

1. https://www.geeksforgeeks.org/pmap-command-in-linux-with-examples/
2. https://docs.python.org/es/3.10/library/sys.html
3. https://pypi.org/project/tabulate/
4. https://stackoverflow.com/questions/34042915/what-is-the-purpose-of-map-anonymous-flag-in-mmap-system-call
5. https://docs.oracle.com/cd/E88353_01/html/E37839/pmap-1.html