

## Lab Report - Lab 3 – 4

Name: Tyler Burleson

Date: 2/12/2024

### Summary

In today's lab we focused on two main parts, spinning up a Window server with Process Explorer and creating a script to easily monitor processes in Ubuntu. We started by creating two Window servers, one was used for today's lab and the other will be for a future lab. We configured these servers similarly to our Ubuntu instances, but we used a Windows image instead. Additionally, we added a new rule in our security group so we could access our server remotely. We did this by adding the Remote Desktop Protocol rule to our group through port 3389 and allowing it to be accessed through all ipv4's.

Once our server was initialized, we RDP'd into the machine and used Ninite to install some essential applications. After these applications finished installing, we navigated to Microsoft Docs and installed Sysinternal Process Explorer. We used this application to have a more accurate and detailed list of the running processes on our machine. While using this application we saw how not properly killing a process tree can create zombies and how we can use some of the other Sysinternal tools.

After we finished working on our Windows server, we moved back into our Ubuntu server we made last week. During this section we followed a similar set of tasks to the one we did on the Windows server.

This section focused on Linux process management commands and writing a script to write a memory report for us. As we followed the lab guide, we used 3 commands "top," "ps aux," and "htop."

Additionally, we accessed a memory log through "/proc." When we used "top" we were given a list of processes that are actively updating and their resource usage. By using "ps aux" and "grep" we're given a snapshot of these processes and can search for certain fields with grep. We then used "htop" which is similar to "top," but had a friendlier interface. After seeing the demonstration of these commands, we decided to make a shell script to easily store a log of processes and their memory consumption. By using

bash, we were able to write a script to use the proc directory that would store a snapshot every 5 seconds into a memory log file.

## Screen Shots

```

Hostname : EC2AMAZ-5IUHVEV
Instance ID : i-0593c47aa44d6b271
Private IP Address : 172.30.1.10
Public IP Address : 54.85.18.107
Instance Size : t2.micro
Availability Zone : us-east-1e
Architecture : AMD64
Total Memory : 1024
Network : Low to Moderate

```

Figure 1 (Screenshot of the windows DC server)

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		8,016 K	40,824 K	76		
System Idle Process	22.53	60 K	8 K	0		
System	1.41	36 K	12 K	4		
Interrupts	< 0.01	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		1,076 K	16 K	436		
csrss.exe	< 0.01	1,920 K	1,596 K	596		
csrss.exe	< 0.01	1,700 K	136 K	668		
wininit.exe		1,316 K	4 K	712		
services.exe	< 0.01	3,516 K	3,992 K	796		
svchost.exe	< 0.01	6,808 K	9,180 K	908	Host Process for Windows S...	Microsoft Corporation
StartMenuExperienceHo...		19,964 K	8,876 K	4936		
RuntimeBroker.exe		4,280 K	1,928 K	4276	Runtime Broker	Microsoft Corporation
SearchApp.exe	Susp...	94,404 K	1,148 K	4668	Search application	Microsoft Corporation
RuntimeBroker.exe		21,980 K	6,304 K	4712	Runtime Broker	Microsoft Corporation
RuntimeBroker.exe		2,568 K	6,100 K	4020	Runtime Broker	Microsoft Corporation
TextInputHost.exe		10,268 K	10,652 K	3456		Microsoft Corporation
dllhost.exe	< 0.01	3,464 K	2,640 K	1528	COM Surrogate	Microsoft Corporation
ApplicationFrameHoste...		4,380 K	3,396 K	4100	Application Frame Host	Microsoft Corporation
smartscreen.exe		7,552 K	4,000 K	5476	Windows Defender SmartScr...	Microsoft Corporation
WmiPrvSE.exe		2,488 K	8,904 K	3260	WMI Provider Host	Microsoft Corporation
ScreenClippingHost.exe	< 0.01	8,744 K	40,252 K	3284		Microsoft Corporation
RuntimeBroker.exe	1.41	2,696 K	15,252 K	5000	Runtime Broker	Microsoft Corporation
svchost.exe	< 0.01	5,220 K	7,620 K	1012	Host Process for Windows S...	Microsoft Corporation
svchost.exe	1.41	80,972 K	48,720 K	568	Host Process for Windows S...	Microsoft Corporation
rdpclip.exe	< 0.01	3,184 K	3,084 K	2520	RDP Clipboard Monitor	Microsoft Corporation
rdpinput.exe		1,516 K	560 K	3572	RDP Session Input Handler	Microsoft Corporation
svchost.exe	< 0.01	16,432 K	7,252 K	792	Host Process for Windows S...	Microsoft Corporation
ctfmon.exe		3,800 K	6,612 K	3520	CTF Loader	Microsoft Corporation
TabTip.exe	< 0.01	4,320 K	5,260 K	3704	Touch Keyboard and Handwr...	Microsoft Corporation
TabTip32.exe		1,272 K	112 K	1916	Touch Keyboard and Handwr...	Microsoft Corporation

CPU Usage: 77.46% Commit Charge: 63.45% Processes: 76 Physical Usage: 90.94%

Figure 2 (Sysinternal Process Explorer)

```

CPU[| 0.7%] Tasks: 44, 65 thr; 1 running
Mem[|||||||||||||||||544M/950M] Load average: 0.00 0.00 0.00
Swp[| 0K/0K] Uptime: 00:44:41

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
526	mysql	20	0	1293M	352M	3968	S	0.7	37.1	0:10.57	/usr/sbin/mysql
652	mysql	20	0	1293M	352M	3968	R	0.7	37.1	0:06.82	/usr/sbin/mysql
2375	root	20	0	8380	3968	3328	R	0.7	0.4	0:00.04	htop
1	root	20	0	163M	10692	6084	S	0.0	1.1	0:04.52	/sbin/init
112	root	19	-1	48128	13096	12072	S	0.0	1.3	0:00.40	/lib/systemd/sy
150	root	RT	0	282M	27392	8960	S	0.0	2.8	0:00.15	/sbin/multipath
152	root	20	0	282M	27392	8960	S	0.0	2.8	0:00.00	/sbin/multipath
154	root	RT	0	282M	27392	8960	S	0.0	2.8	0:00.00	/sbin/multipath
155	root	RT	0	282M	27392	8960	S	0.0	2.8	0:00.00	/sbin/multipath
156	root	RT	0	282M	27392	8960	S	0.0	2.8	0:00.00	/sbin/multipath
157	root	RT	0	282M	27392	8960	S	0.0	2.8	0:00.11	/sbin/multipath
158	root	RT	0	282M	27392	8960	S	0.0	2.8	0:00.00	/sbin/multipath
160	root	20	0	11484	4912	3120	S	0.0	0.5	0:00.15	/lib/systemd/sy
303	systemd-n	20	0	16252	6144	5120	S	0.0	0.6	0:00.05	/lib/systemd/sy
305	systemd-r	20	0	25536	9580	5376	S	0.0	1.0	0:00.11	/lib/systemd/sy
338	root	20	0	2816	2048	1920	S	0.0	0.2	0:00.00	/usr/sbin/acpid
353	root	20	0	7288	2816	2560	S	0.0	0.3	0:00.00	/usr/sbin/cron

Figure 3 (Htop)

```

*****
Mon Feb  5 21:40:25 UTC 2024
*****
MemTotal:      972500 kB
MemFree:       65884 kB
MemAvailable:  258556 kB
Buffers:       10604 kB
Cached:        312084 kB
SwapCached:    0 kB
Active:        169296 kB
Inactive:      576584 kB
Active(anon):   836 kB
Inactive(anon): 441896 kB
Active(file):   168460 kB
Inactive(file): 134688 kB
Unevictable:   34736 kB
Mlocked:       27620 kB
SwapTotal:     0 kB
SwapFree:      0 kB
Zswap:         0 kB
Zswapped:      0 kB
Dirty:         0 kB
Writeback:     0 kB
memorylog.txt

```

Figure 4 (First iteration of our script)

```
*****  
Mon Feb  5 21:43:11 UTC 2024  
*****  
Active:           170072 kB  
Inactive:         576276 kB
```

Figure 5 (Second iteration of our script)

## Research Questions

1. The server configuration we're using (mostly) for this class is AWS's t2.micro. Would that be a good configuration to use on production servers?  
No, t2.micro isn't suitable for production environments and is intended for educational use.
2. We opened communications in the Security group for all traffic on the 172.30.0.0/16 VPC. In your opinion, is that a good or bad idea?  
172.30.0.0/16 is isolated from the internet and trusted. Its probably ok to leave it how it is, but in a production environment we would want to limit the access it has.
3. In Process Explorer, what is the structure for the grouping of the Processes?  
It is hierarchical/ parent-child.
4. When we ran Notepad, why is notepad.exe located under explorer.exe?  
Notepad is listed under the explorer.exe since it is a user application and not a system application.
5. Why do you think 'Autostart Location' (Under the Image Tab) for notepad.exe is 'n/a'?  
This is because notepad.exe doesn't autostart with windows when the system boots up.
6. What does this do? (See image below)  
The target tool can be used to identify a process by clicking its window.
7. See #48  
Process explorer highlights the notepad.exe process.

8. Define every portion of the output for the top command (hint: try man top and look in section, maybe, 3), i.e., "PID," "USER," "PR," "NI," etc. (See #72)
  - a. PID: the unique process id
  - b. USER: name of the owner of the task
  - c. PR: priority of the task.
  - d. NI: nice value. The lower the number, the higher the priority.
  - e. VIRT: the total amount of virtual memory that the task uses.
  - f. RES: Resident Memory is a subset of VIRT. This includes non-swapped physical memory.
  - g. SHR: is a subset of RES that other processes can use.
    - i. S: status of task
    - ii. D: uninterruptible sleep
    - iii. R: running
    - iv. S: sleeping
    - v. T: stopped by control signal
    - vi. t: stopped by debugger
    - vii. Z: zombie
    - viii. %CPU: percent of elapsed time and how much of the CPU is used
    - ix. %MEM: a task's share of physical memory
    - x. TIME+: total CPU time that the task has used while running
    - xi. COMMAND: displays the command that was used to start the task
9. (See #83) What could that possibly mean? Feel free to Google it  
PTS is a virtual/ remote equivalent to the original TTY
10. Why didn't we have to worry about licensing our Windows servers when we launched them  
AWS / Amazon handles the Microsoft licensing for us.