Undetectable Task Bypassing OS Scheduler via Hardware Task Switching

Kyeong Joo Jung[1,2], Bang Hun Lee[3], Yeon Nam Gung[4], Jun Seok Kim[5], Hyung Suk Kim[4], Ju Seong Han[6], Tomaspeter Kim[4], Bong Jun (David) Choi[1,2]
Stonybrook University[1], SUNY Korea[2], Woosuk University[3], BoB(Best of the Best) KITRI[4], Ahnlab Inc.[5], Bluehole Inc.[6]

Motivation

Recently, malicious mining using CPUs has become a trend – mining which the task is not detected by the users is even more of a threat. In this poster, we focused on discovering a new IA-32 vulnerability and found an undetectable task using hardware task switching method. By manually manipulating hardware task switching, which is directly managed by the CPU, we show that it is possible to create a hidden scheduler aside from the ones created by the operating system.

Results of Task Manager when executing through normal scheduler(Left) and proposed method scheduler(Right)

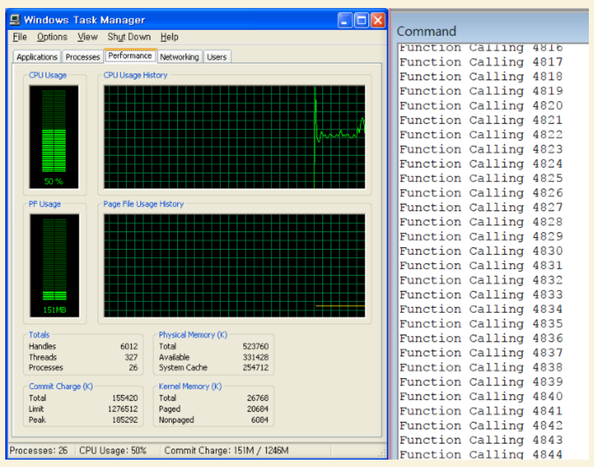

Figure 1 Normal task function call shows 55% of CPU usage.

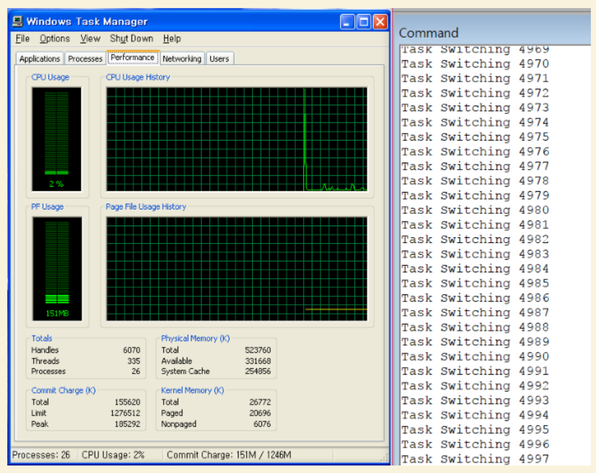

Figure 2 Hardware task switching call shows 2% of CPU usage.

Contribution

- Propose a new hiding attack using hardware task switching method
- Expose a new threat that uses ring 0 authority to modify GDT and TSS values that have not been done before
- Made a recommendation to reduce the permissions of ring 0 authority.

Limitation

- Currently, the proposed method has been implemented and tested under 32-bit Windows - extension to 64-bit Windows and other operating systems is in progress.

Reliability of OS

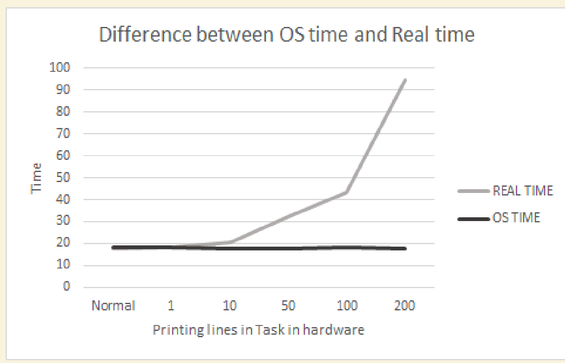

Figure 3 Executed time comparison between OS time and real time.

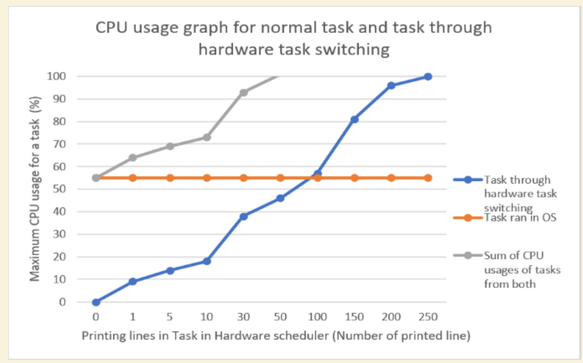

Figure 4 CPU utilization graph for normal task and task through hardware task switching.

Table 1 Result of task execution.

Number of printed lines in task in hardware	0	1	5	10	30	50	100	150	200	250
CPU utilization of task in hardware (%)	0	9	14	18	38	46	57	81	96	100
CPU utilization of task in OS (%)	55	55	55	55	55	55	55	55	55	55
Required CPU utilization of both tasks (%)	55	64	69	73	93	**101**	**112**	**136**	**151**	**155**
Execution time of task using OS time (sec)	13.99	14.03	14.44	14.13	14.55	14.07	14.21	14.42	13.93	13.36
Execution time of task using online (sec)	13.96	13.99	15.56	15.58	17.81	20.55	36.8	50.27	58.45	69.53
Time difference of both time (sec)	-0.04	-0.04	1.12	1.45	3.26	**6.48**	**22.58**	**35.85**	**44.52**	**56.17**

- OS stops when tasks from proposed method executes.
- Actual log data and the log data shown from OS becomes different.

⇨ Reliability of the OS becomes untrustful which is really dangerous (ex) forensic investigation)

Conclusion

- The ring 0 authority can be achieved easily through installing a newly created driver.
- Create a new scheduler outside of the OS management boundary can make the system vulnerable and can give rise to various new attacks.
- Attacks can dominate server's CPU usage or mine cryptocurrencies from the users' computers without being noticed.

⇨ Therefore, further research effort is need to detect such attacks by designing more sophisticated detection methods that also examines the GDT and TSS structure.

Acknowledgement

This research was funded by the MSIT, Korea, under the "ICT Consilience Creative Program" (IITP-2017-R0346-16-1007) supervisedby the IITP, by the KEIT, Korea, under the "Global Advanced Technology Center" (10053204), and by the NRF, Korea, under the "Basic Science Research Program" (NRF-2015R1C1A1A01053788).

Contact: kyeongjoo.jung@stonybrook.edu