Hiding Tasks via Hardware Task Switching

[Kyeong Joo Jung – Ajae.dll]
Hiding Tasks?

- Rootkits
- Malwares

- Task Switching Method
1. Intro

H/W Task Switching
- Used until Windows 3.1 (~1993)
- Uses structures defined by CPU Manufacturer
- Access to CPU directly

S/W Task Switching
- Used from Windows NT (1994~)
- Uses structures defined by OS
- Access CPU through OS Scheduler
H/W Task Switching
- Used until Windows 3.1 (~1993)
- Uses structures defined by CPU Manufacturer
- Access to CPU directly

S/W Task Switching
- Used from Windows NT (1994~)
- Uses structures defined by OS
- Access CPU through OS Scheduler
1. Intro

S/W
Task Switching

- Used from Windows NT (1994~)
- Uses structures defined by OS
- Access CPU through OS Scheduler

H/W
Task Switching
1. Intro

**JMP/CALL**

MY_TASK

H/W Task Switching

Intel CPU

<OS Scheduler>

Load

S/W Task Switching

TASK 1

TASK 2

TASK 3

April 9 ~ April 13 / 2018
1. Intro

JMP/CALL

MY_TASK

Intel CPU

H/W Task Switching

<OS Scheduler>

Load

TASK 1
TASK 2
TASK 3

S/W Task Switching

April 9 ~ April 13 / 2018
2. Explanation
H/W Task Switching?

- **JMP / CALL**
- **<Trigger>**
- **<Global Descriptor Table>**
- **<Call TSS>**
- **<Operate CPU>**

- **Task State Segment**
- **Defined by CPU manufacturer**
- **Saves states of Task**
2. Explanation

TSS

JMP / CALL

<Trigger>

<Global Descriptor Table>

<TSS DESCRIPTOR_1>

<TSS DESCRIPTOR_2>

<Call TSS>

<MY_TSS>

<Operate CPU>

<INTEL CPU>
2. Explanation

H/W Task Switching

JMP / CALL

(Trigger)

<TSS DESCRIPTOR_1>

<TSS DESCRIPTOR_2>

<Global Descriptor Table>

<Call TSS>

<MY_TSS>

<Operate CPU>

<INTEL CPU>

April 9 ~ April 13 / 2018
# 2. Explanation

## TSS Descriptor

<table>
<thead>
<tr>
<th>GDT INDEX</th>
<th>Descriptor Name &amp; State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>32-bit TSS (Busy)</td>
</tr>
<tr>
<td>0x50</td>
<td>32-bit TSS (Available)</td>
</tr>
<tr>
<td>0x58</td>
<td>32-bit TSS (Available)</td>
</tr>
<tr>
<td>0xA0</td>
<td>32-bit TSS (Available)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GDT INDEX</th>
<th>Descriptor Name &amp; State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x16</td>
<td>32-bit TSS (Busy)</td>
</tr>
<tr>
<td>0x31</td>
<td>32-bit Double Fault TSS (Available)</td>
</tr>
</tbody>
</table>
# Modifying TSS Descriptor

<table>
<thead>
<tr>
<th>GDT INDEX</th>
<th>Descriptor Name &amp; State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>32-bit TSS (Available)</td>
</tr>
<tr>
<td>0x50</td>
<td>32-bit TSS (Busy)</td>
</tr>
<tr>
<td>0x58</td>
<td>32-bit TSS (Available)</td>
</tr>
<tr>
<td>0xA0</td>
<td>32-bit TSS (Available)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GDT INDEX</th>
<th>Descriptor Name &amp; State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x16</td>
<td>32-bit TSS (Available)</td>
</tr>
<tr>
<td>0x31</td>
<td>32-bit Double Fault TSS (Busy)</td>
</tr>
</tbody>
</table>
3. Result
### 3. Result

<table>
<thead>
<tr>
<th>INDEX</th>
<th>DESCRIPTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>TSS (Busy)</td>
</tr>
<tr>
<td>0x50</td>
<td>MY_TSS (Available)</td>
</tr>
<tr>
<td>0x58</td>
<td>TSS (Available)</td>
</tr>
<tr>
<td>0xA0</td>
<td>TSS (Available)</td>
</tr>
</tbody>
</table>

**Before Task Switching**

----------

**After Task**

----------

SHOW ME THE AJAEDLL

----------

[After] [Success calling for '2' Times]

[Before] [Success calling for '2' Times]

----------

[Before Task Switching]

----------

[After Task]

----------

SHOW ME THE AJAEDLL

----------

[After] [Success calling for '3' Times]

[Before] [Success calling for '3' Times]
3. Result

Printing

<table>
<thead>
<tr>
<th>INDEX</th>
<th>DESCRIPTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>TSS (Available)</td>
</tr>
<tr>
<td>0x50</td>
<td>MY_TSS (Busy)</td>
</tr>
<tr>
<td>0x58</td>
<td>TSS (Available)</td>
</tr>
<tr>
<td>0xA0</td>
<td>TSS (Available)</td>
</tr>
</tbody>
</table>

[Before Task Switching]
---------[After Task]
--------- SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
--------- SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]
[Before][Success calling for '3' Times]
3. Result

Printing

<table>
<thead>
<tr>
<th>INDEX</th>
<th>DESCRIPTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>TSS (Busy)</td>
</tr>
<tr>
<td>0x50</td>
<td>MY_TSS (Available)</td>
</tr>
<tr>
<td>0x58</td>
<td>TSS (Available)</td>
</tr>
<tr>
<td>0xA0</td>
<td>TSS (Available)</td>
</tr>
</tbody>
</table>

[Before Task Switching]
---------[After Task]
---------SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]

[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
---------SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]

[Before][Success calling for '3' Times]
3. Result

Printing Results

Windows

Linux

[Before Task Switching]
---------[After Task]
---------SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '2' Times]
[Before][Success calling for '2' Times]

[Before Task Switching]
---------[After Task]
---------SHOW_ME_THE_AJAEDLL
---------[After][Success calling for '3' Times]
[Before][Success calling for '3' Times]
### 3. Result

#### CPU Usage Video

<table>
<thead>
<tr>
<th>Detection method</th>
<th>Tool</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature based detection</td>
<td>Icesword</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Malwarebytes anti-rootkit</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Sophos Virus Removal Tool</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>TDSSKiller</td>
<td>X</td>
</tr>
<tr>
<td>Behavior based detection</td>
<td>Alyac (Korean program)</td>
<td>X</td>
</tr>
<tr>
<td>Integrity based detection</td>
<td>Afick</td>
<td>X</td>
</tr>
<tr>
<td>Hooking detection</td>
<td>Gmer</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Radix</td>
<td>X</td>
</tr>
</tbody>
</table>
3. Result

JMP/CALL

MY_TASK

Intel CPU

Load

<OS Scheduler>

TASK 1

TASK 2

TASK 3

H/W Task Switching

S/W Task Switching
- 10 cases: increasing CPU utilization
- User will be looking at 55% only
- Sum of H/W and OS = actual required CPU usage
- User waits → suspicious → check

**Table 1: Result of task execution**

<table>
<thead>
<tr>
<th>Number of printed lines in task in hardware</th>
<th>0</th>
<th>1</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU utilization of task in hardware (%)</td>
<td>0</td>
<td>9</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>CPU utilization of task in OS (%)</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>Required CPU utilization of both tasks (%)</td>
<td>55</td>
<td>64</td>
<td>69</td>
<td>73</td>
</tr>
<tr>
<td>Execution time of task using OS time (sec)</td>
<td>13.99</td>
<td>14.03</td>
<td>14.44</td>
<td>14.1</td>
</tr>
<tr>
<td>Execution time of task using online (sec)</td>
<td>13.96</td>
<td>13.99</td>
<td>15.56</td>
<td>15.5</td>
</tr>
<tr>
<td>Time difference of both time (sec)</td>
<td>-0.04</td>
<td>-0.04</td>
<td>1.12</td>
<td>1.45</td>
</tr>
</tbody>
</table>

**Executed Time Comparison between OS Time and Real Time**

![Graph showing time comparison between OS and real time](image_url)
4. Conclusion
4. Conclusion

Problem

1. Gives chance to attackers
   - Servers, zombie PCs
   - Mining

2. Belief in OS
   - OS event log, time
   - Investigation

→ Power of ring 0
4. Conclusion

Limitation

1. 32 bit only yet
   - Still many out there
   - Working on 64 bit!!

2. Memory scan
   - Normal users won’t scan it
   - Don’t make it suspicious
1. New approach of hiding tasks

2. Potential Danger
   - Accessing files, network, CPU..

3. Ring 0 powerful than we think
Thank you!

Q&A

- AJAE.DLL -

kyeongjoo.jung@stonybrook.edu