**Part 1. The Hijacking Attack**

1. The boss launches an app **the\_boss\_console.exe**
2. The boss opens a new file **budget.txt** without sharing access

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_open budget.txt |

1. The boss writes the content **TeamA = $3,000,000 ; TeamB = $7,000,000**

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_write TeamA = $3,000,000 ; TeamB = $7,000,000 |

1. Let’s check that the budget has been written successfully:

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_read |

We can see that the content has been written and read successfully and the file is still opened.

1. An attacker launches an app **attacker.exe** to steal the file:
2. An attacker tries to steal the **budget.txt** by opening a file in a legal way:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_open budget.txt |

It receives an error message with STATUS\_SHARING\_VIOLATION (0xC000043). MSDN

1. An attacker copies a FILE\_OBJECT of **budget.txt** to launch a hijack attack:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_open\_by\_hijacking <FILE\_OBJECT> |

We can see that the attacker has received a handle of a new opened file.

1. Now attacker can read the file using f\_read:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_read |

The budget has been stolen.

1. The attacker modifies the opened file **budget.txt** using f\_write:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_write TeamA = $6,000,000 ; TeamB = $4,000,000 |

1. Let’s go back to the boss and read the file again:

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_read |

The **budget.txt** has been really modified.

Wait 10 hours and conclude that memory has been corrupted without BSOD.

Windows build-in security features do not protect FILE\_OBJECT structures.

**Part 2. The Prevention of Hijacking Attack**

1. Start **memranger\_console.exe** to protect FILE\_OBJECT structures.
2. The boss launches an app **the\_boss.exe**
3. The boss opens a new file **budget.txt** without sharing access and writes the content **TeamA = $3,000,000 ; TeamB = $7,000,000**

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_open budget.txt |
|  | f\_write TeamA = $3,000,000 ; TeamB = $7,000,000 |

1. Let’s check that the budget has been written successfully:

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_read |

We can see that the content has been written and read successfully and the file is still opened.

1. An attacker launches an app **attacker.exe** to steal the file:
2. An attacker tries to steal the **budget.txt** by opening a file in a legal way:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_open budget.txt |

It receives an error message with STATUS\_SHARING\_VIOLATION (0xC000043):

1. An attacker copies a FILE\_OBJECT of **budget.txt** to launch a hijack attack:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_open\_by\_hijacking <FILE\_OBJECT> |

We can see that the attacker has failed during hijacking attack.

1. Now attacker tries to read the file using f\_read:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_read |

The attacker has read nothing. The budget has not been stolen.

1. The attacker tries to modify the budget using f\_write:

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_write TeamA = TeamB = 0 |

The attacker has not changed the budget.

1. Let’s back to the boss and read the file again:

|  |  |
| --- | --- |
| **the\_boss\_console.exe** | f\_read |

The budget has been changed. MemoryRanger protects the OS.

1. The attacker closes its file and exists.

|  |  |
| --- | --- |
| **attacker\_console.exe** | f\_close |

1. The boss closes its file and exists

|  |  |
| --- | --- |
| **the\_boss.exe** | f\_close |

Open both files and see, that the attacker accesses only its file and cannot access the budget file.

**Subtitles for the Part 1. The Hijacking Attack.**

* Part 1 The Hijacking Attack  
  Illegal access to the file opened by kernel-mode driver in an exclusive mode  
  CDFSL 2019
* Launching the boss console to load its driver
* The boss opens a new budget.txt
* The budget file is opened.   
  File handle and FILE\_OBJECT have been created.
* The boss plans $3 and $7 mln for teams A and B
* Done
* The boss is checking the budget
* The budget is prepared.
* The attacker is launching its console to access the budget
* The attacker tries to open the budget file
* The attacker fails to open the file with STATUS\_SHARING\_VIOLATION (0xC0000043)
* The budget file is protected by sharing access flags
* The attacker copies the address of target FILE\_OBJECT
* Go back to the attacker
* The attacker uses the copied FILE\_OBJECT to hijack the budget
* The attacker opens the file hijacker and   
  copies fields from target FILE\_OBJECT
* The attacker reads the file hijacker
* The budget has been stolen! Wow!
* The attacker wants to modify the budget and copies its content
* The attacker overwrites the budget file by setting $6 and $4 mln
* Done
* Let's go back to the boss and check the budget
* The budget has been modified! Oh no!
* Let's wait for the PatchGuard reaction, e.g. BSOD
* 8 hours have passed - a BSOD hasn't appeared
* The OS is infected!

**Subtitles for the Part 2. The Attack Prevention.**

* Part 2 The Attack Prevention   
  Preventing hijacking of the FILE\_OBJECT  
  CDFSL 2019
* Launching MemoryRanger to protect the memory
* Launching the boss console to load its driver
* The boss opens a new budget.txt
* The budget file is opened.   
  File handle and FILE\_OBJECT have been created.
* The boss plans $3M and $7M for A and B teams
* Done
* The boss is checking the budget
* The budget is prepared
* The attacker is launching its console to access the budget
* The attacker tries to open the budget file
* The attacker fails to open the budget file.   
  The budget file is protected by sharing access flags.
* The attacker copies the address of target FILE\_OBJECT
* Go back to the attacker
* The attacker tries to hijack the copied target FILE\_OBJECT
* The attacker has failed to hijack the budget.   
  The attacker cannot access the target FILE\_OBJECT.
* The attacker reads the created file hijacker to access the budget
* The attacker fails to read the budget
* The attacker tries to overwrite the budget
* Done
* Let's go back to the boss and check the budget
* The budget has not been changed
* Let's close the file hijacker and budget file to complete file operations
* Closing the boss and the attacker consoles
* Opening the budget file and the file hijacker to check the content
* The budget file has boss data. The file hijacker has attacker's data.
* The OS is protected