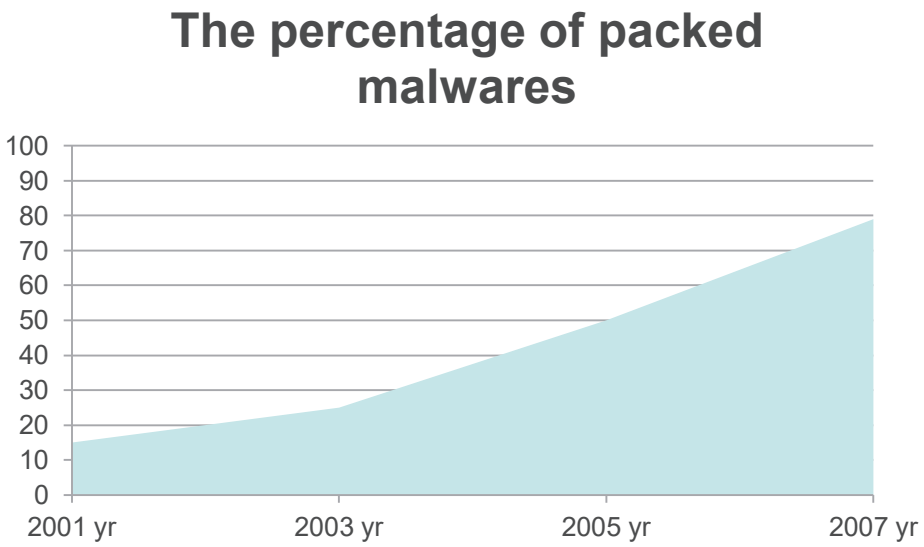
“egg” – A Stealth fine grained code analyzer

Fourteenforty Research Institute, Inc.
http://www.fourteenforty.jp

Agenda

- Background and problems
- Introduce “egg”
  - Demonstration its basic functions
- Implementation (Taint tracing approach in ring-0)
  - Demonstration of the taint tracing behavior
- Discuss a limitation of “egg”
- Conclusion

Too many malwares!



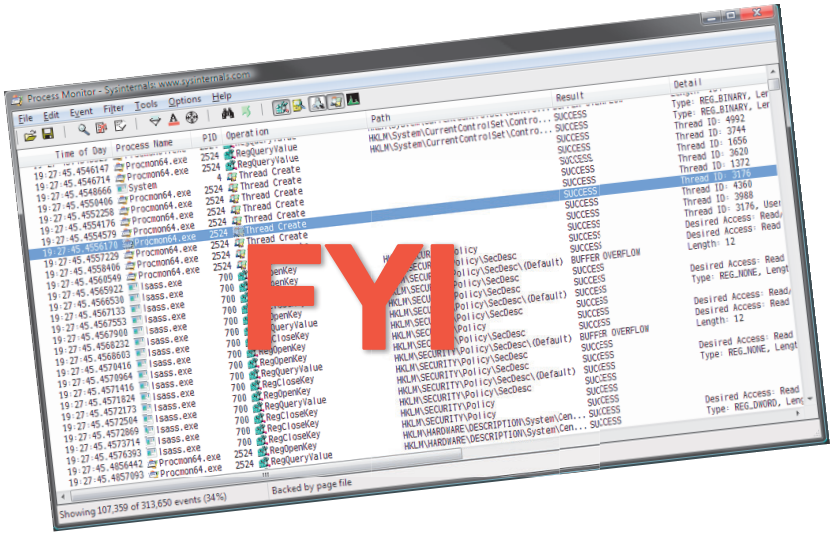
80% of malwares ware packed in 2007

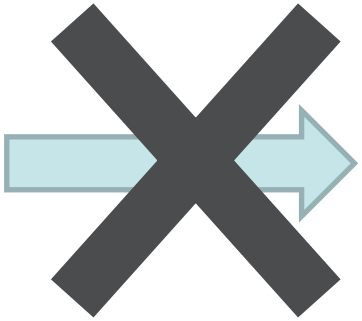
- We can't manually analyze each malware.
- Automatic approaches have become more important.

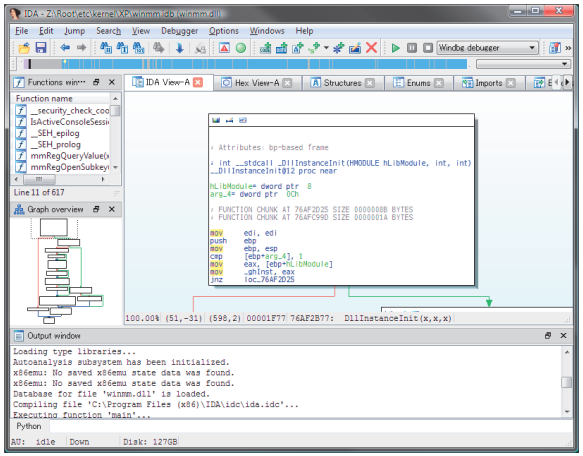
Source:
2001-2005 : McAfee Sage vol.1 issue 1
2007 : Panda Research (http://research.pandasecurity.com/malwareformation-statistics/)

Problems of traditional dynamic analyzers

- We can't get useful information for more intensive analysis.
- We can't analyze a kernel mode code.
- It's difficult to analyze a spreading malware over the process.







Innovative analyzers (based on VM environments)

- Innovative analyzers have already resolved the above problems☺
  - Anubis
  - Ether
    - It's able to analyze a kernel mode code and perform an instruction level analysis.
  - BitBlaze and Renovo
    - Also these analyze a spreading malware automatically with approach called "taint tracing".

- However these systems are detected by VM detection techniques☹

Summary table of problems

Type of system	Traditional	Innovative (Based on virtual environments)
Getting useful information	Insufficient	Good
Analyzing a kernel mode code	Insufficient	Good
Analyzing a spreading malware.	Insufficient	Good
Not affected by VM detection techniques	Good	Insufficient

- I developed “egg” to try and resolve these problems.

What is egg?

- “egg” is a dynamic analyzer based on a Windows device driver.

- egg has following capabilities:

1. It can obtain more detailed information.
2. It can analyze a kernel mode code.
3. It can automatically trace a spreading malware.

- Of course, It’s not affected by VM detection techniques.
- Also most common anti-debug tech can’t detect “egg”.

What kind of information does ”egg” collect?

1. API arguments for IN, OUT (,INOUT), and return value

```
BOOL WINAPI ReadFile(
  __in          HANDLE hFile,
  __out         LPVOID lpBuffer,
  __in          DWORD nNumberOfBytesToRead,
  __out_opt     LPDWORD lpNumberOfBytesRead,
  __inout_opt   LPOVERLAPPED lpOverlapped
);
```

What kind of information does ”egg” collect?

1. API arguments for IN, OUT (,INOUT), and return value

```
BOOL WINAPI ReadFile(
  __in          HANDLE hFile,
  __out         LPVOID lpBuffer,
  __in          DWORD nNumberOfBytesToRead,
  __out_opt     LPDWORD lpNumberOfBytesRead,
  __inout_opt   LPOVERLAPPED lpOverlapped
);
```

```
call to kernel32.dll!ReadFile(
    Arg 1 : 00000064 = File : ¥Device¥HarddiskVolume1¥WINDOWS¥(...) ,
    Arg 3 : 00000800(2048)
)
```

What kind of information does ”egg” collect?

1. API arguments for IN, OUT (,INOUT), and return value

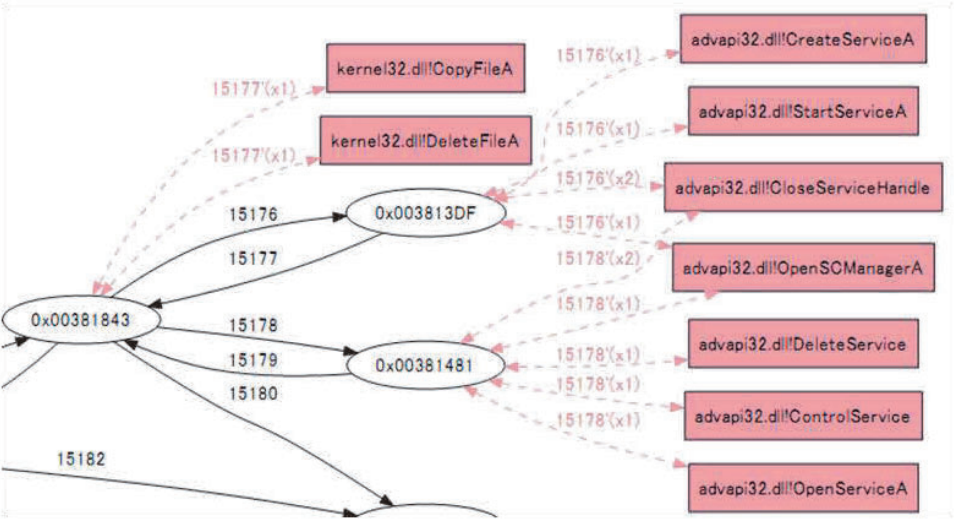
```
BOOL WINAPI ReadFile(
  __in          HANDLE hFile,
  __out         LPVOID lpBuffer,
  __in          DWORD nNumberOfBytesToRead,
  __out_opt     LPDWORD lpNumberOfBytesRead,
  __inout_opt   LPOVERLAPPED lpOverlapped
);
```

```
call to kernel32.dll!ReadFile(
    Arg 1 : 00000064 = File : ¥Device¥HarddiskVolume1¥WINDOWS¥(...) ,
    Arg 3 : 00000800(2048)
)
```

```
returned from kernel32.dll!ReadFile(
    Arg 2 : 0012F184 - 0012F983 is dumped as ¥(...)¥(...)ReadFile_Arg02.bin
) => 00000001(1)
```

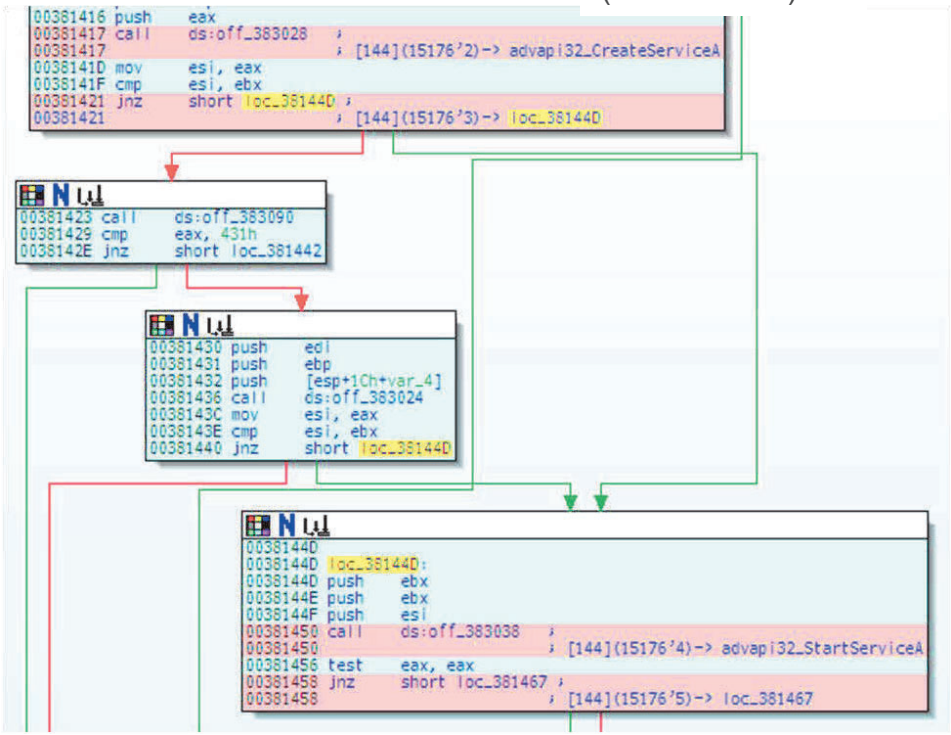
What kind of information does ”egg” collect?

2. Callgraph
3. Branch information



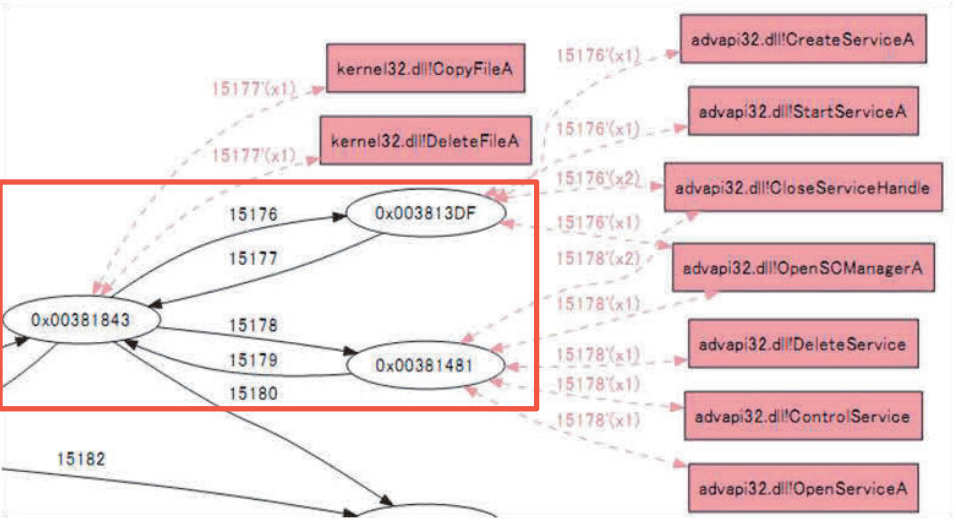
Callgraph
(made with Graphviz)

Branch Info
(with IDA Pro)



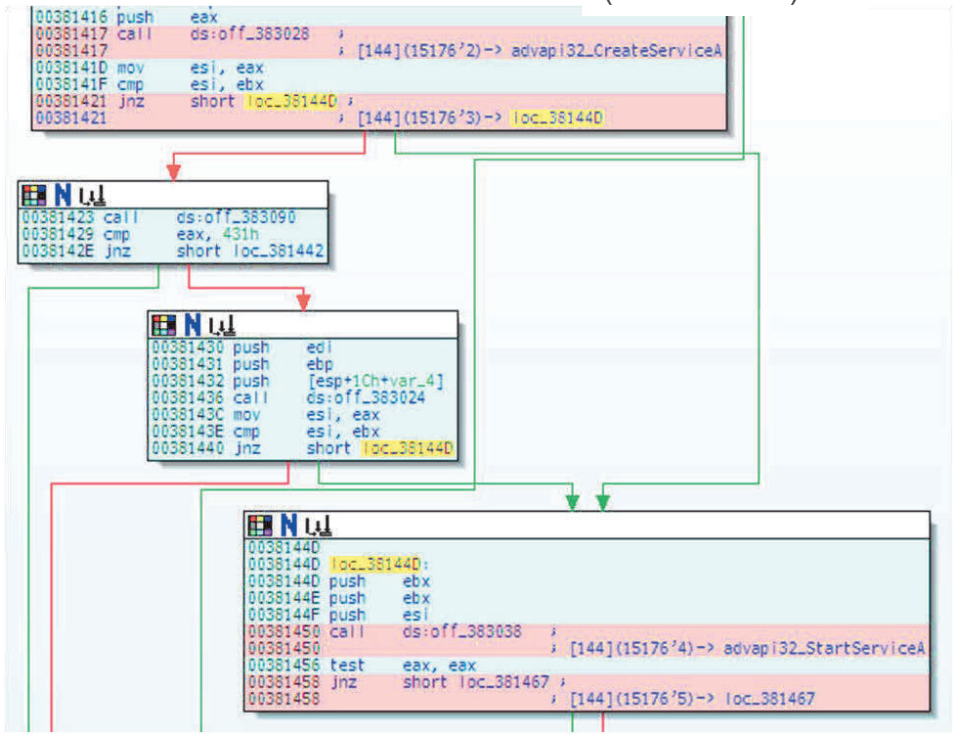
What kind of information does ”egg” collect?

2. Callgraph
3. Branch information



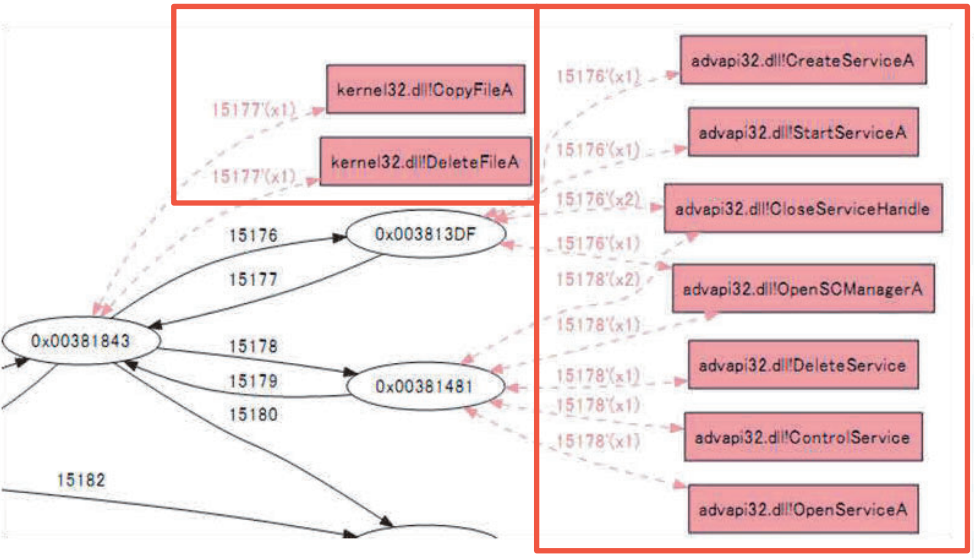
Callgraph
(made with Graphviz)

Branch Info
(with IDA Pro)



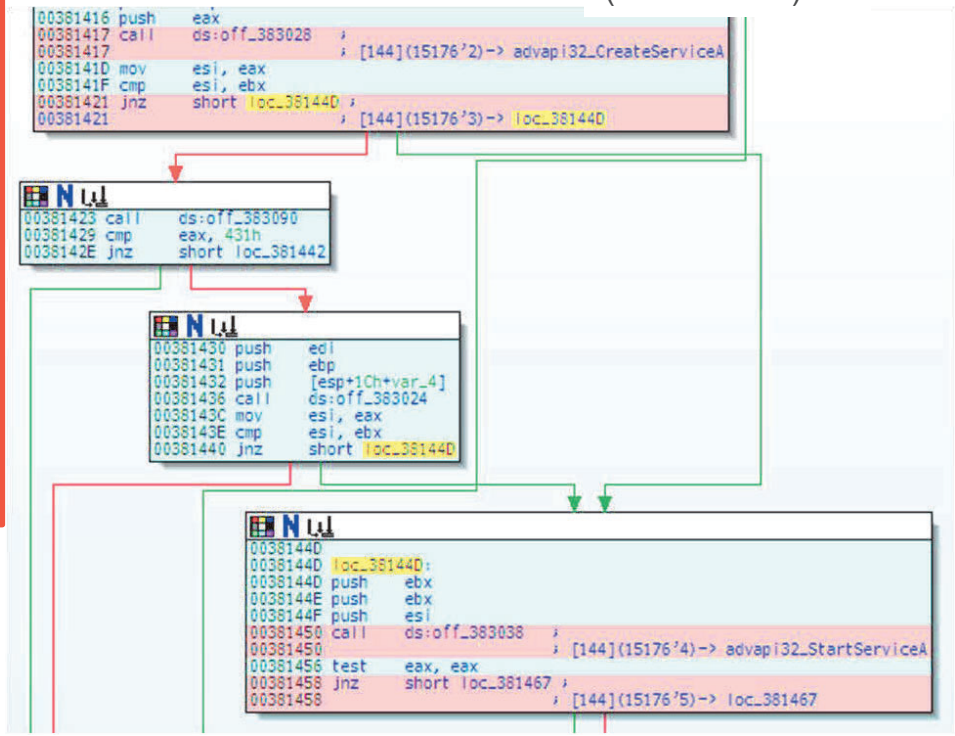
What kind of information does ”egg” collect?

2. Callgraph
3. Branch information



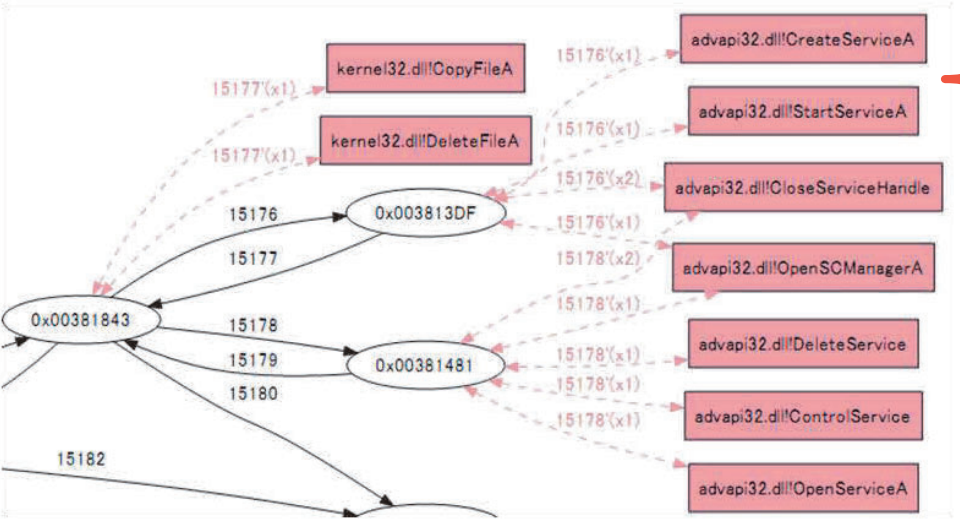
Callgraph
(made with Graphviz)

Branch Info
(with IDA Pro)



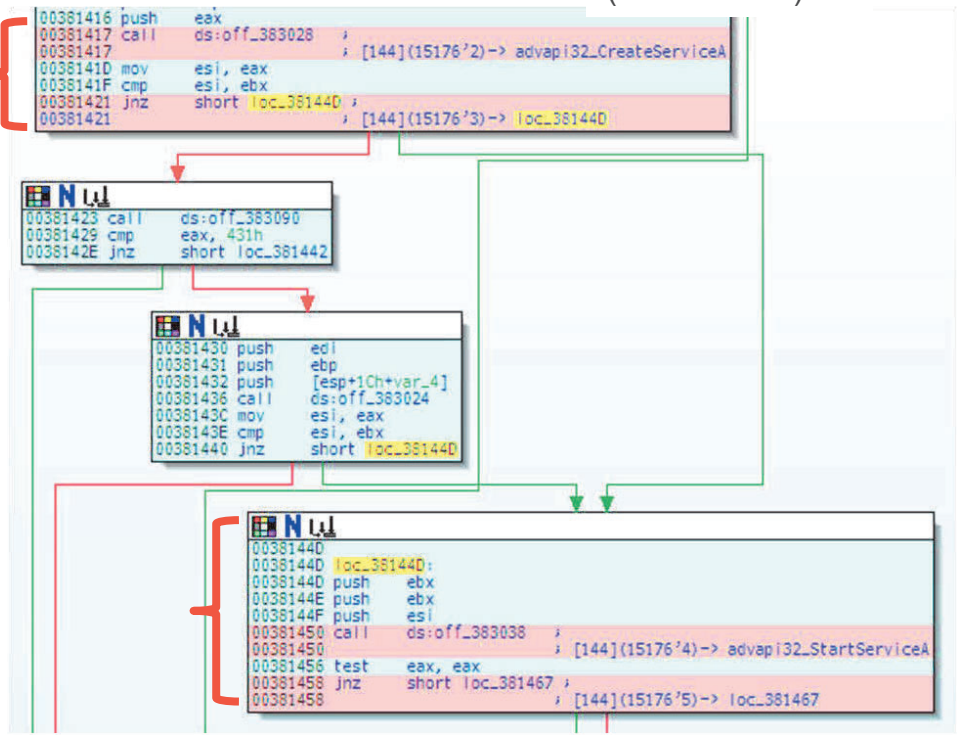
What kind of information does ”egg” collect?

2. Callgraph
3. Branch information



Callgraph
(made with Graphviz)

Branch Info
(with IDA Pro)



Demonstration of basic functions(movie)

- Analyzing sample.exe.

- Sample.exe overwrites original beep driver (beep.sys).
- Then restarts beep service to install this driver in the kernel.

- “egg” analyzes sample.exe and the modified beep driver.

Implementation of the fine-grained code analysis

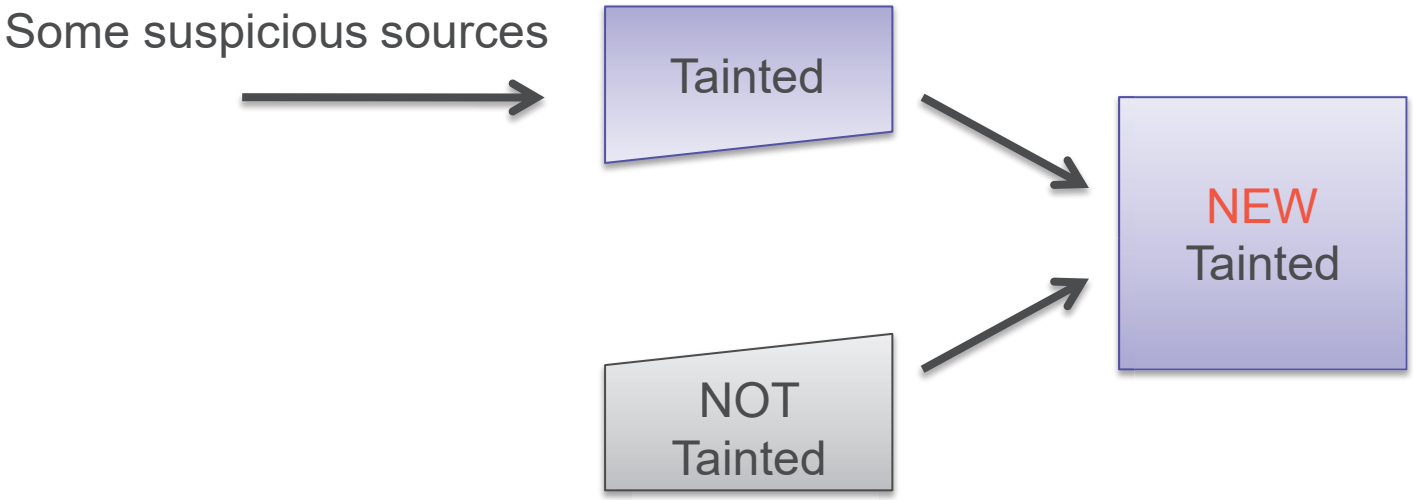
- Based on the page protection and the trap flag.
- Published by the paper “Stealth Breakpoints”.

- We can run analysis codes for each instruction execution.
- It can applies to both a kernel and user modes, and even works transparently in the user mode code.

Stealth Breakpoints
http://www.acsac.org/2005/abstracts/72.html

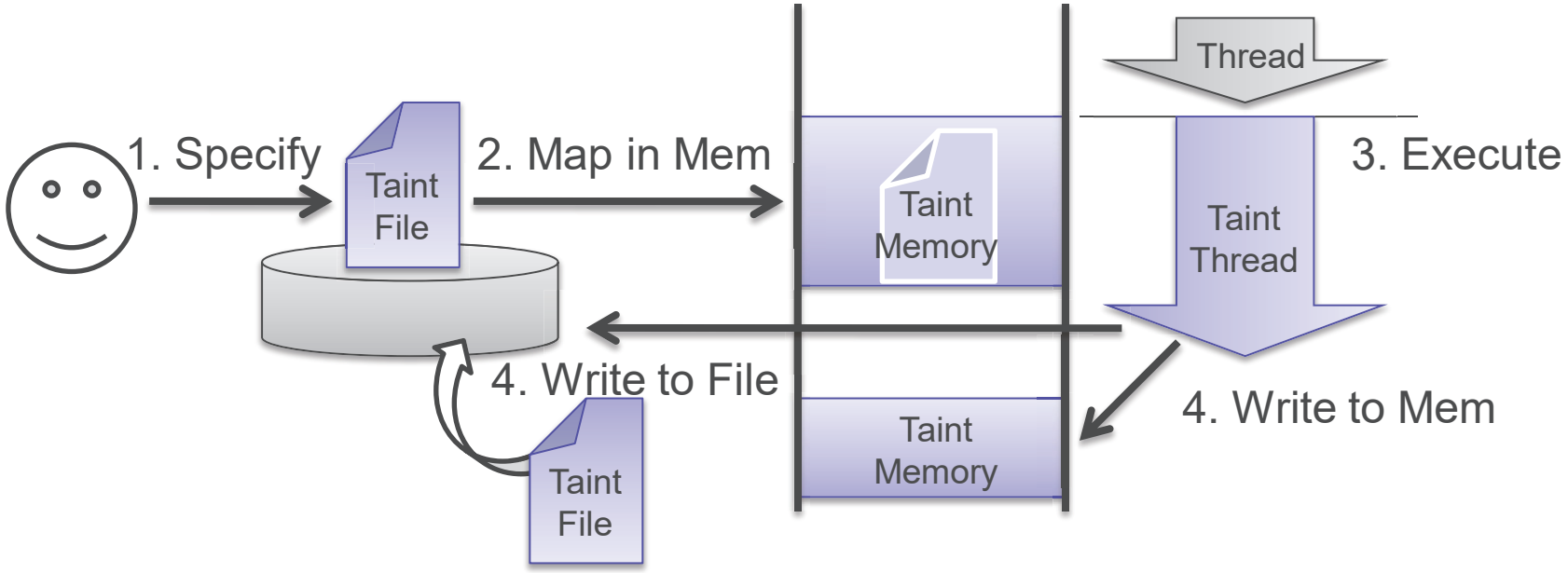
What is taint tracing?

- It can automatically trace suspicious elements.
- A suspicious element is marked as tainted.
- A taint automatically influences new elements that used tainted elements.



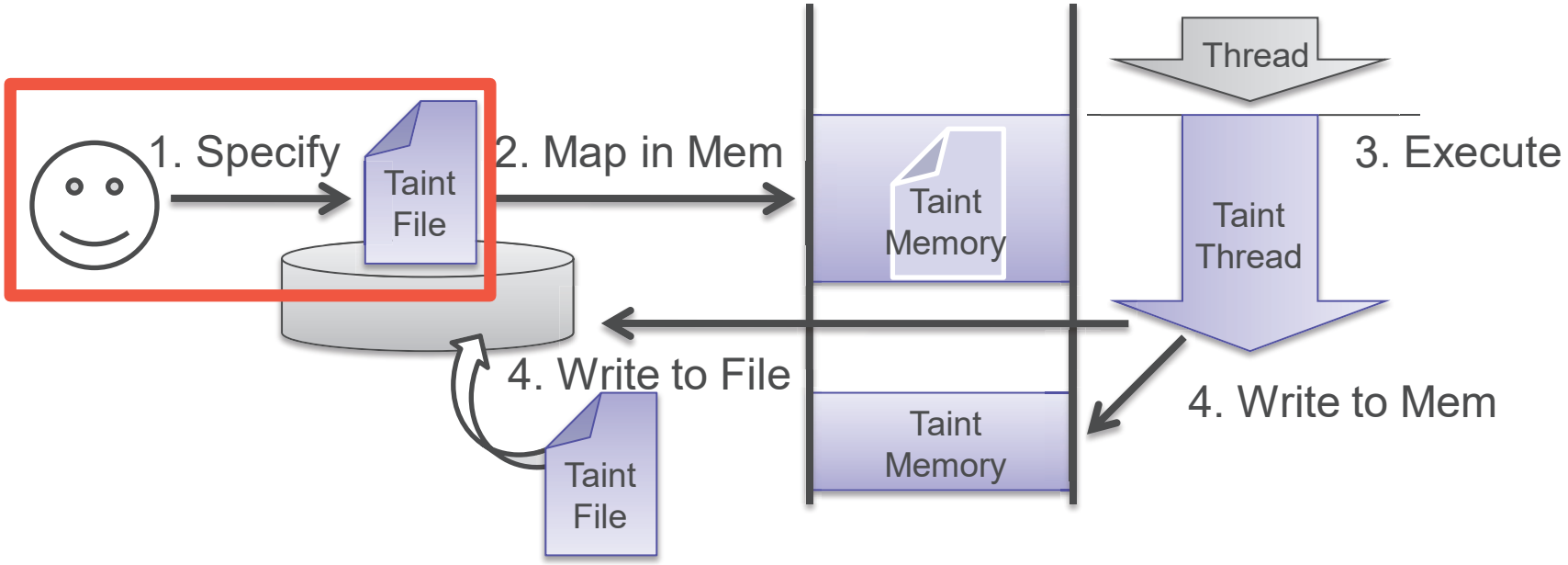
An overview of taint tracing approach of “egg”

- egg takes a novel approach to implement the taint tracing.
- In case of egg, “Elements” are Files, Virtual memory and Threads.

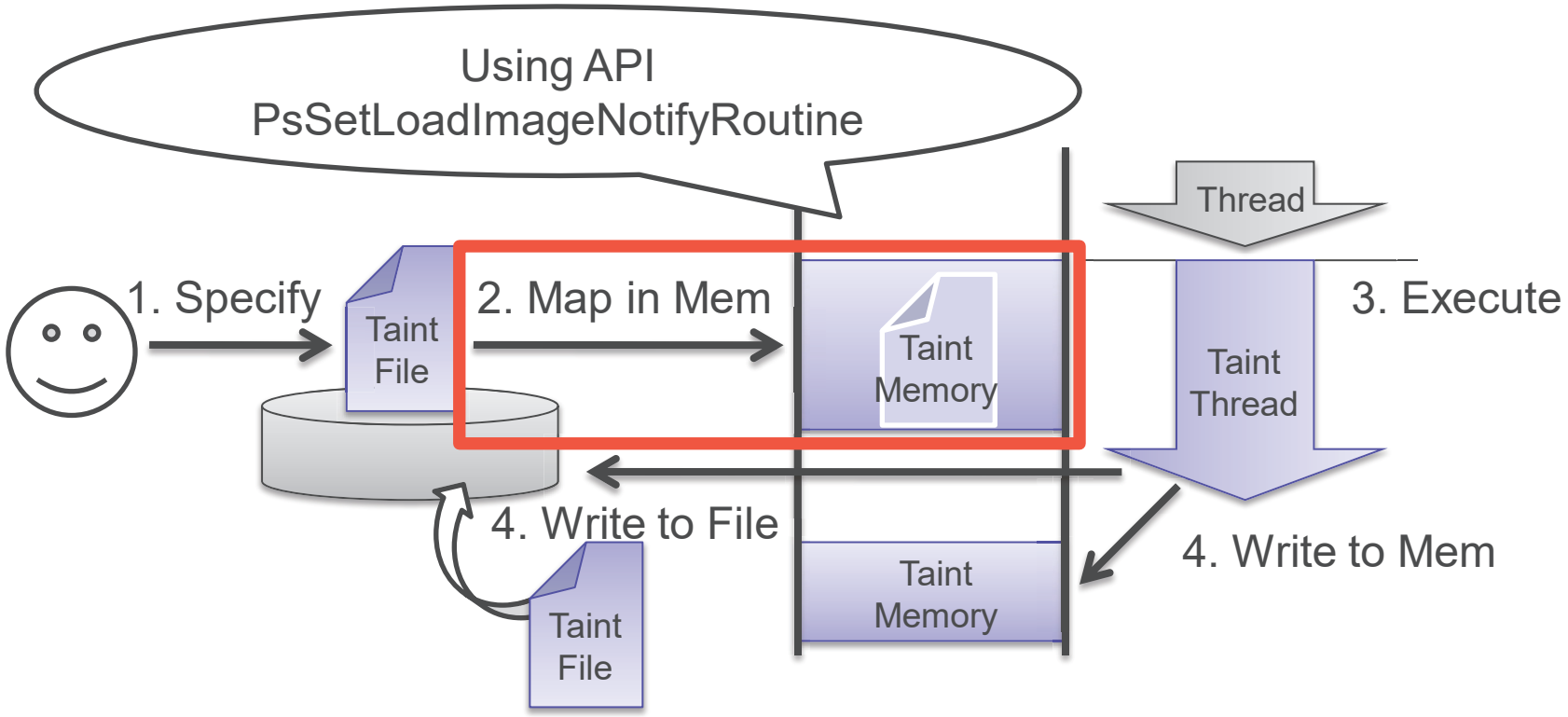


An overview of taint tracing approach of “egg”

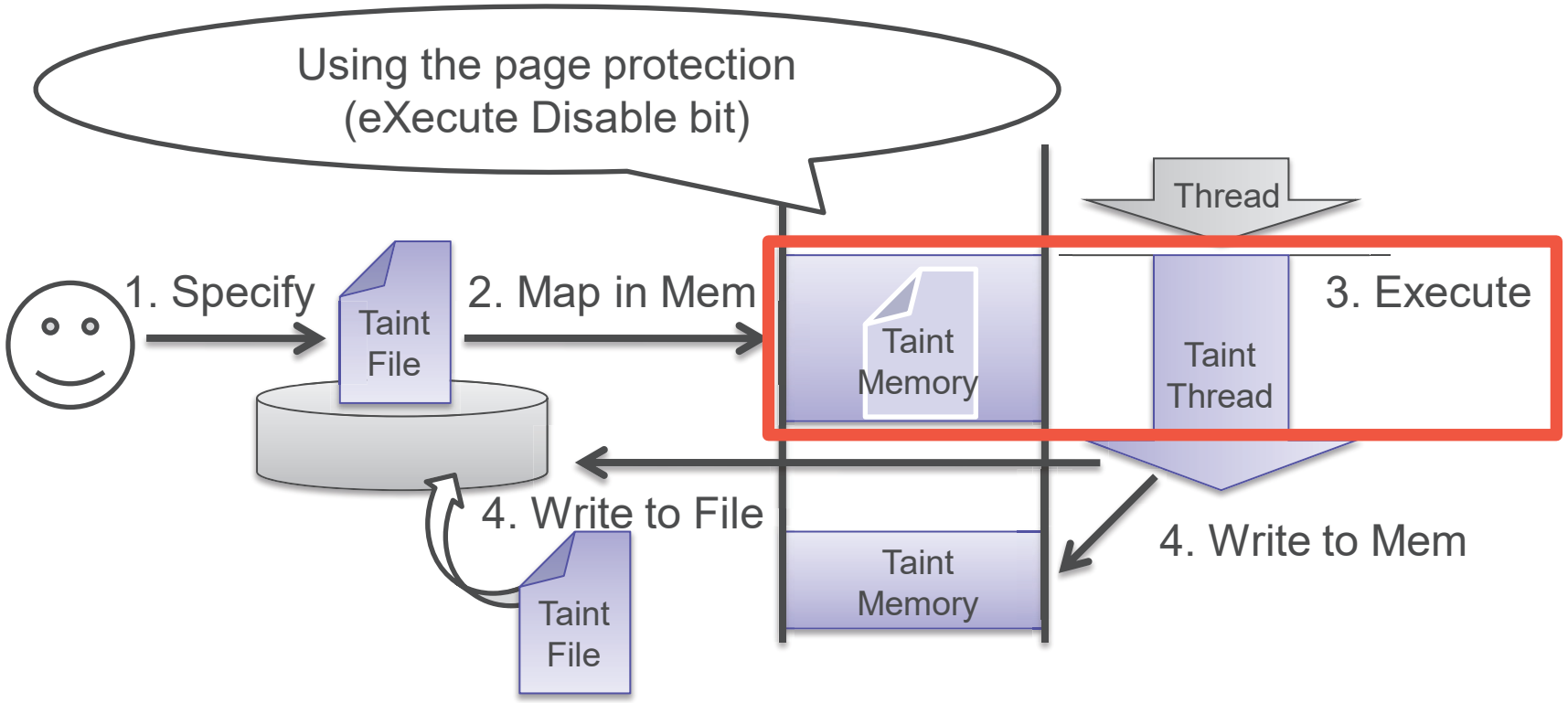
- egg takes a novel approach to implement the taint tracing.
- In case of egg, “Elements” are Files, Virtual memory and Threads.



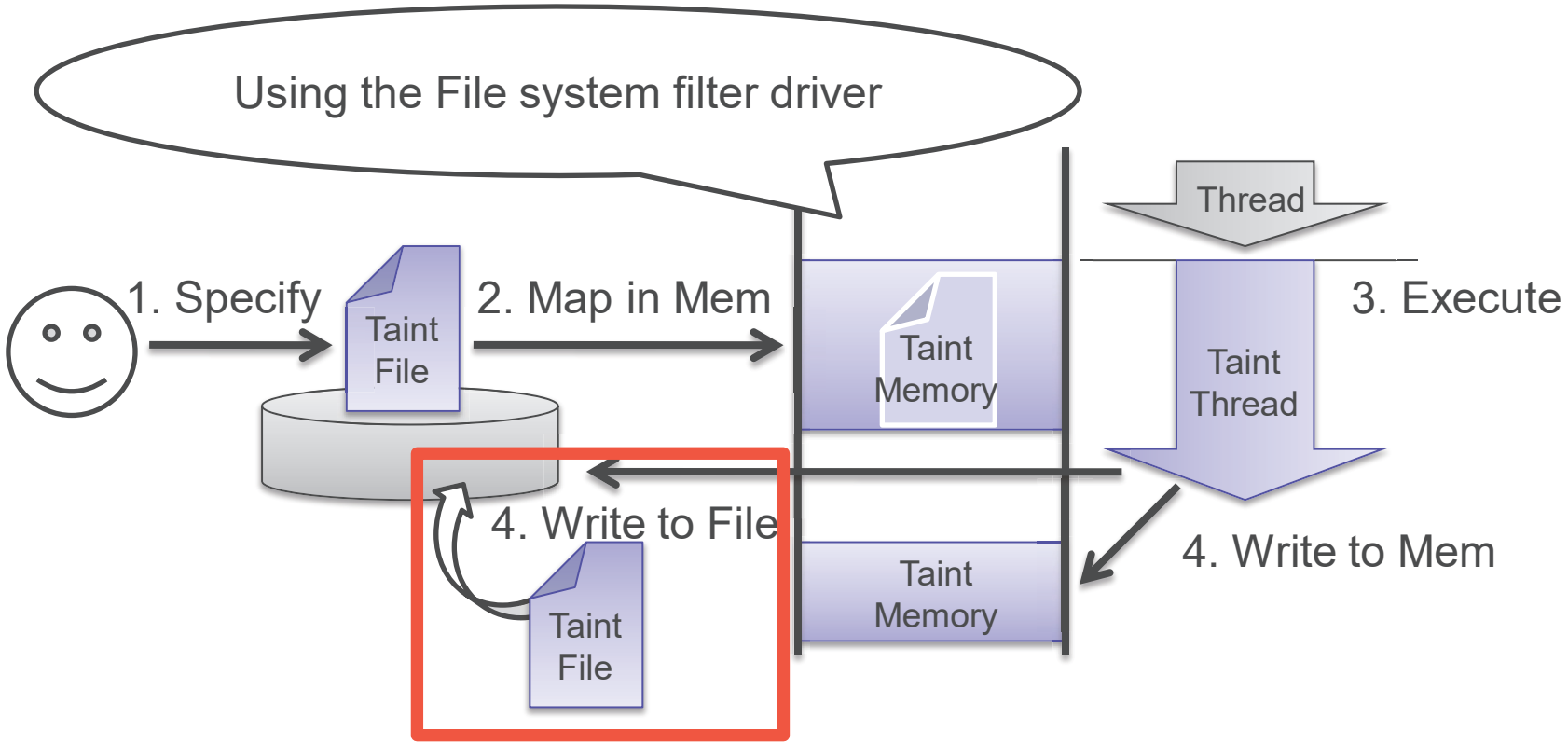
Implementation of taint tracing in ring-0



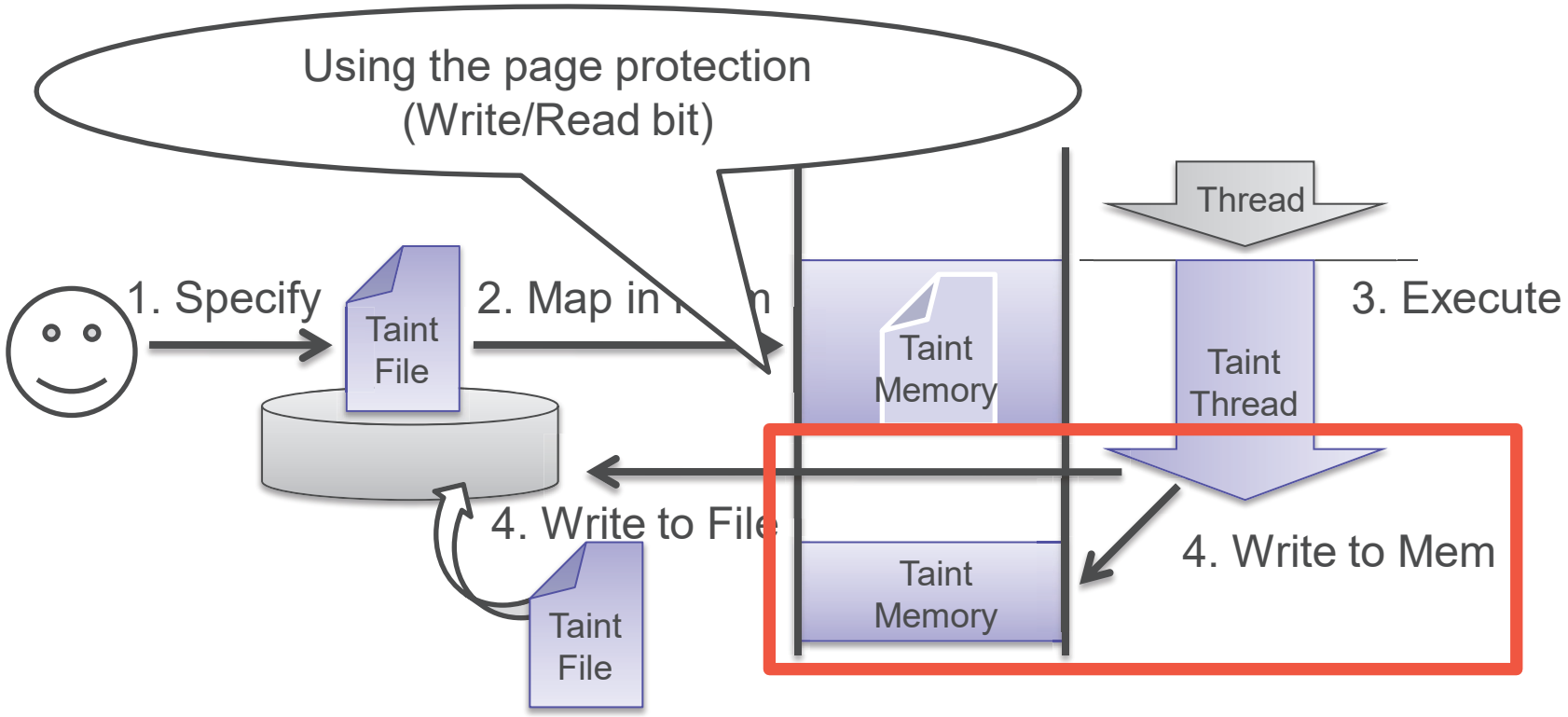
Implementation of taint tracing in ring-0



Implementation of taint tracing in ring-0

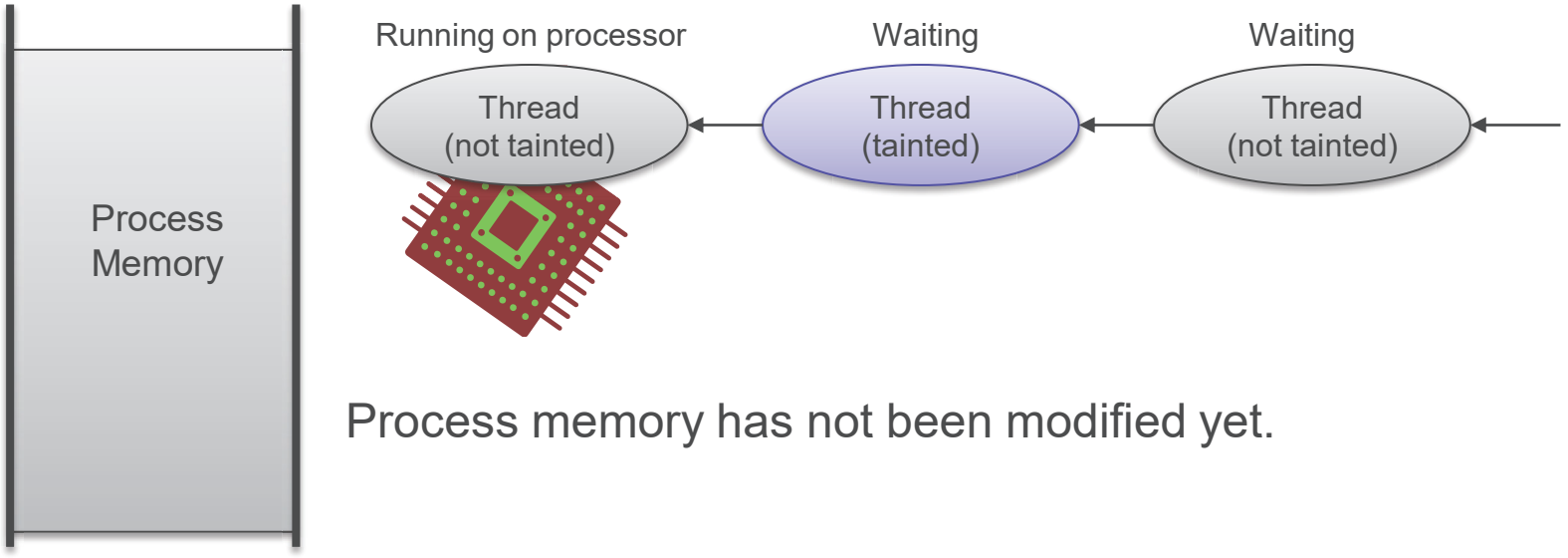


Implementation of taint tracing in ring-0



Implementation of taint tracing in ring-0

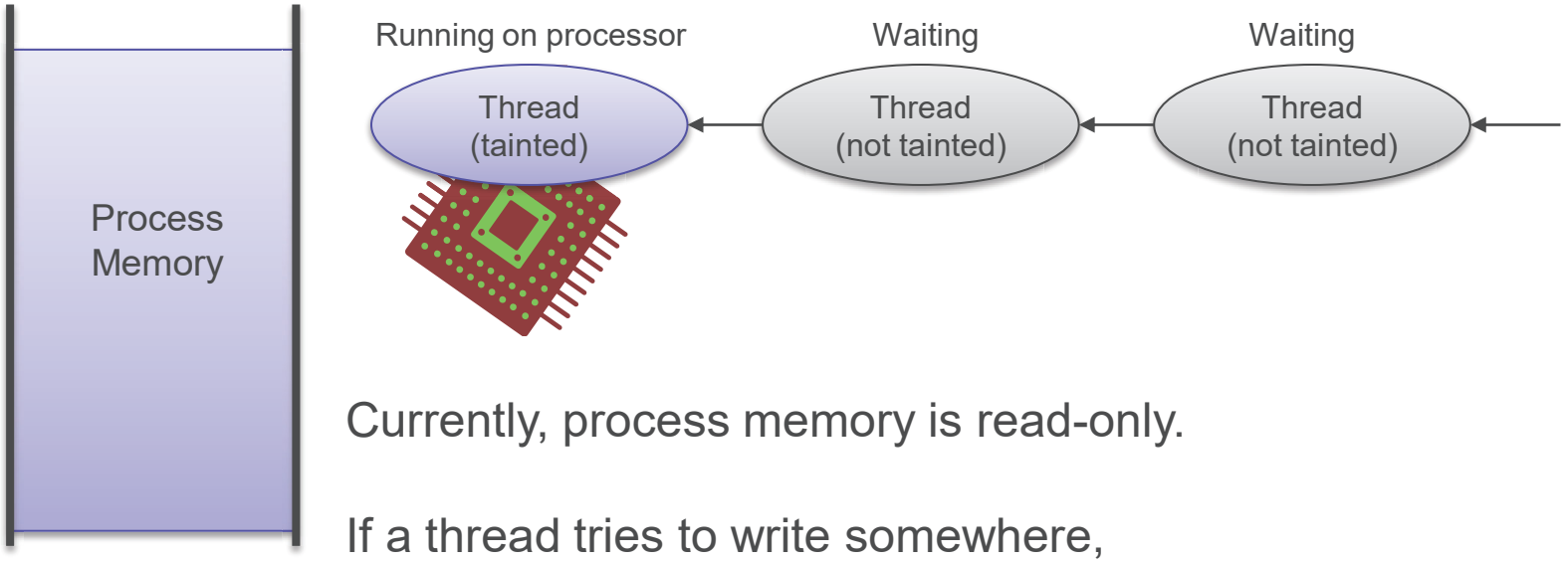
- For thread safety, egg hooks thread switching function (called SwapContext).
- Therefore egg can notice a thread switching.



Process memory has not been modified yet.

Implementation of taint tracing in ring-0

- When taint thread becomes active, egg changes every process memory to read-only.

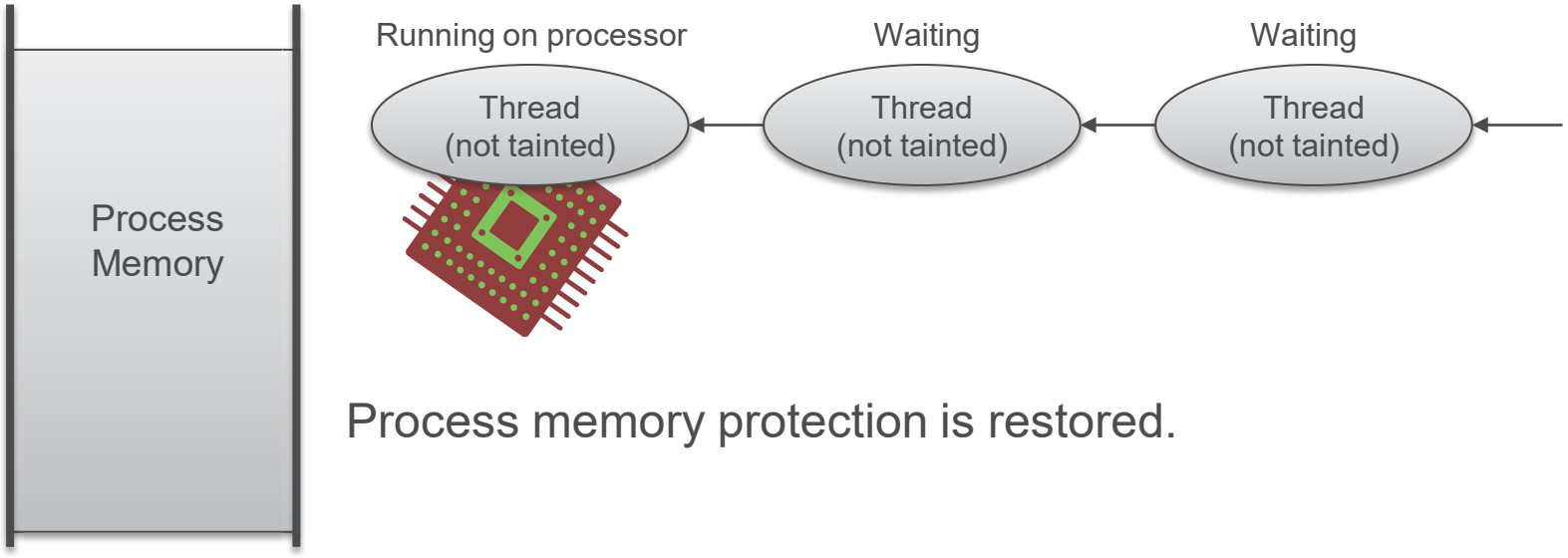


Currently, process memory is read-only.

If a thread tries to write somewhere,
the processor causes an exception.
egg catches this exception as taint event.

Implementation of taint tracing in ring-0

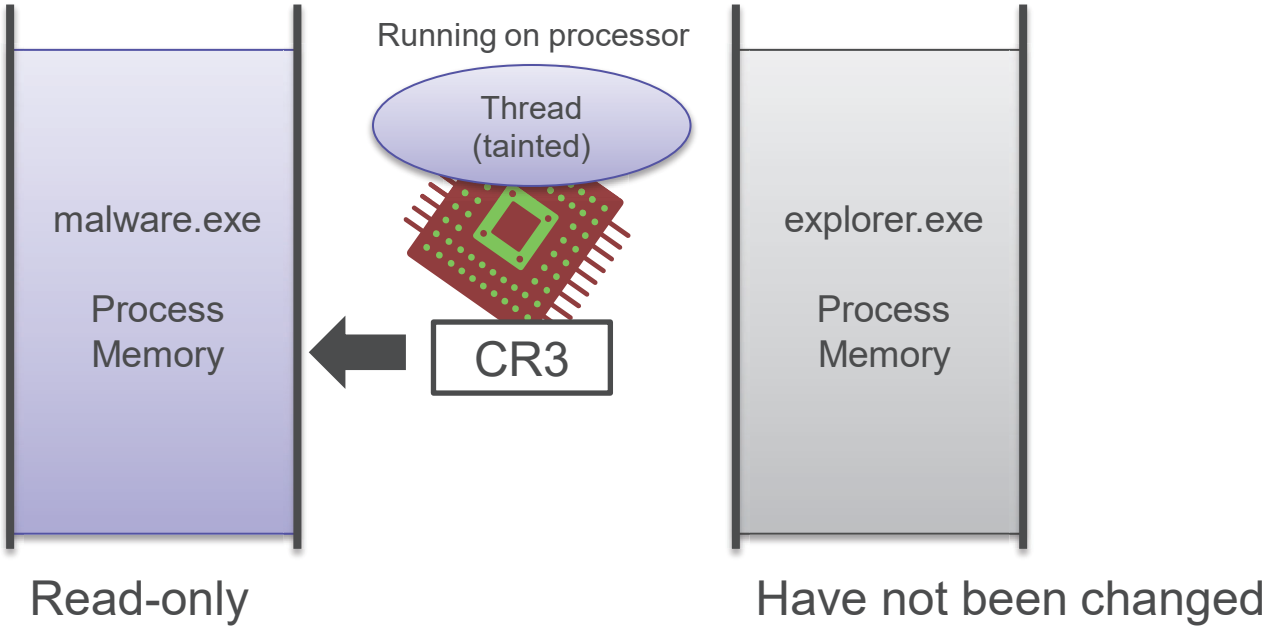
- When taint thread becomes inactive, egg restores every page protection.



Process memory protection is restored.

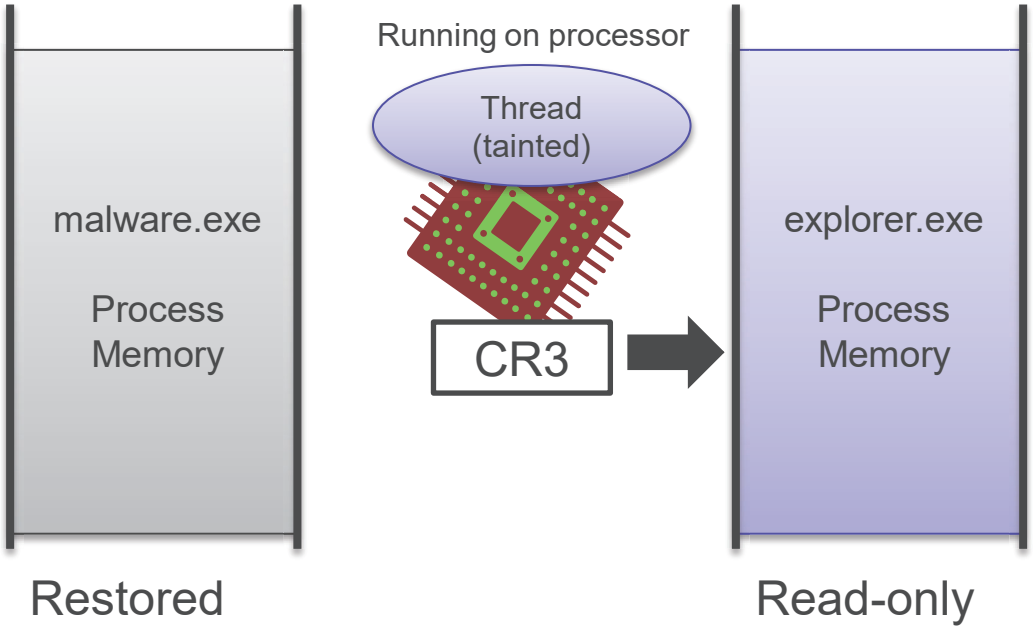
Tracking the cross-process memory operation

- To trace cross-process memory operation, egg hooks context switching function (called KiSwapProcess).
- Therefore egg can notice cross-process memory operation.



Tracking the cross-process memory operation

- When taint thread is running on other process memory, its process memory will be changed to read-only.



egg can trace cross-process memory operation.
(e.g. WriteProcessMemory)

Demonstration of the taint tracing function(movie)

- The sample is the thread injection code.

- Sample malware called “injector.exe” injects to notepad.exe with VirtualAllocEx, WriteProcessMemory and CreateRemoteThread.
- Injected thread calls AllocConsole and WriteConsole in infinite loop.
- egg will trace the injected thread.

Problem of same privilege

- egg has limitation against kernel mode code.
  - egg is visible and breakable from kernel mode malware.

- This limitation is result of trade off for avoiding detection by the VM detection.

Conclusion

Type of system	egg	Traditional	Innovative
Getting useful information	Good	Insufficient	Good
Analyzing a kernel mode code	Better	Insufficient	Good
Analyzing a spreading malware.	Good	Insufficient	Good
Not affected by VM detection techniques	Good	Good	Insufficient

- We can save time by using egg.
- In the future, I will try to improve its stability and usability.

Thank you!

Fourteenforty Research Institute, Inc.

http://www.fourteenforty.jp