Hochschule Worms
University of Applied Sciences

NETWORK INFORMATION HIDING

CH. 3: INTRODUCTION TO GENERIC COUNTERMEASURES

Prof. Dr. Steffen Wendzel
Worms University of Applied Sciences

https://www.wendzel.de (EN) | https://www.hs-worms.de/wendzel/ (DE)

Online Class: https://github.com/cdpxe/Network-Covert-Channels-A-University-level-Course/

Shared Resource Matrix (SRM) Methodology [1,2]

- General approach to detect covert storage channels
- Can be applied at different steps of SDL
  - Covert channels can be detected within textual specifications of a software
  - … but also in source code
- The idea was later improved by McHugh, but we focus only on the original version introduced by Kemmerer.
- General assumption: A system is described by “operations” and “attributes”.

[1] Kemmerer, R. A.: Shared resource matrix methodology: An approach to identifying storage and timing channels, ACM Transactions on Computer Systems (TOCS), Vol. 1(3), pp. 256-277, ACM, 1983.
[2] Bishop, M.: Computer Security, Art and Science, Addison-Wesley Professional, 2003, Chapter 17.

Shared Resource Matrix (SRM) Methodology [1,2]

- Goal of the SRM is to determine whether an Operation X can modify (M) an attribute A under the condition that an Operation Y (w/ Y≠X) can read (R) attribute A.

- **Example:** Let us assume that:

Attr. / Op.	Read	Write	Delete	Create
Existence of file	R	R	R, M	R, M
File owner	-	-	R	M
File name	R	R	R	M
File size	R	M	M	M

[1] Kemmerer, R. A.: Shared resource matrix methodology: An approach to identifying storage and timing channels, ACM Transactions on Computer Systems (TOCS), Vol. 1(3), pp. 256-277, ACM, 1983.
[2] Bishop, M.: Computer Security, Art and Science, Addison-Wesley Professional, 2003, Chapter 17.

Shared Resource Matrix (SRM) Methodology [1,2]

- Problems:
  - Some „covert channels“ can be false positives (e.g. if two operations could build an (R,M) pair but cannot be called by processes of different security levels).
  - The SRM supports no sequences of operations but a sequence of *n* operations may lead to an **indirect recognition** of a modified attribute [2].
  - Kemmerer states that all storage and timing channels can be detected using the SRM. However, Bishop stated that this is wrong (see above).

[1] Kemmerer, R. A.: Shared resource matrix methodology: An approach to identifying storage and timing channels, ACM Transactions on Computer Systems (TOCS), Vol. 1(3), pp. 256-277, ACM, 1983.
[2] Bishop, M.: Computer Security, Art and Science, Addison-Wesley Professional, 2003, Chapter 17.

Covert Flow Trees [1]

- Code-level Detection for Covert Channels.
  (We will only discuss fundamental aspects here.)

Code:

```
Procedure IncreaseTemp()
{
     Heat(cur_temp=internal_temp);
     internal_temp += 1;
}

Procedure CheckTemp()
{
     if (internal_tmp < soft_limit)
          return internal_temp;
     else
          return 999; # error code
}
```

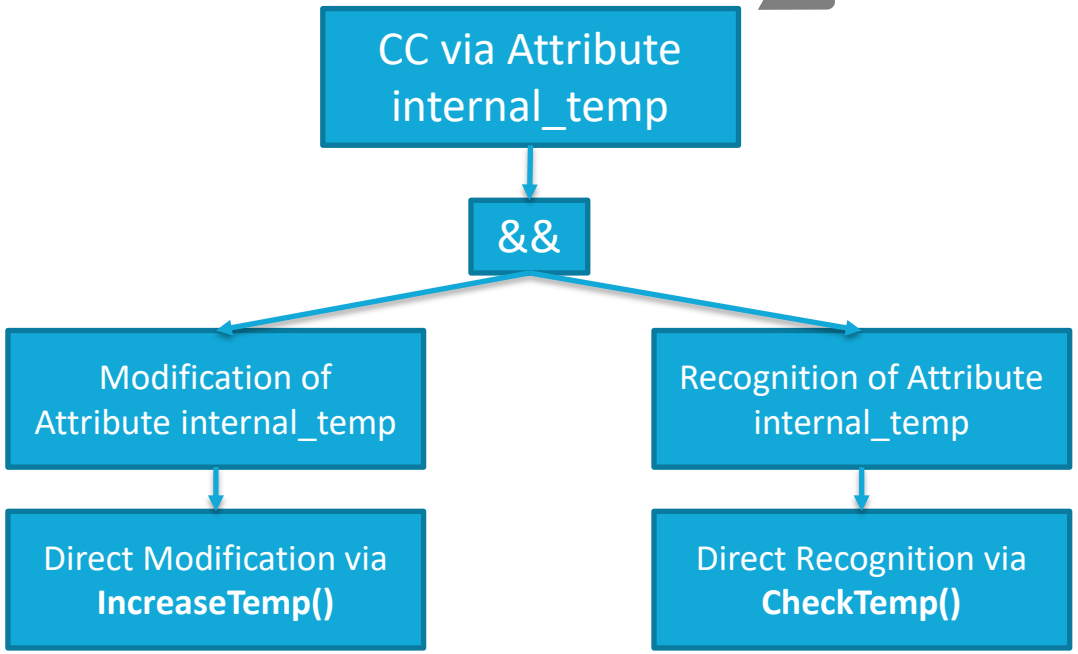
	IncreaseTemp	CheckTemp
Reference	cur_temp, internal_temp	internal_temp soft_limit
Modify	internal_temp	-
Return	-	internal_temp

[1] Kemmerer, R., Porras, P.: Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels, Trans. Software Engineering, IEEE, 1991.

Covert Flow Trees [1]

	IncreaseTemp	CheckTemp
Reference	cur_temp, internal_temp	internal_temp soft_limit
Modify	internal_temp	-
Return	-	internal_temp

- Building a simplified CFT:

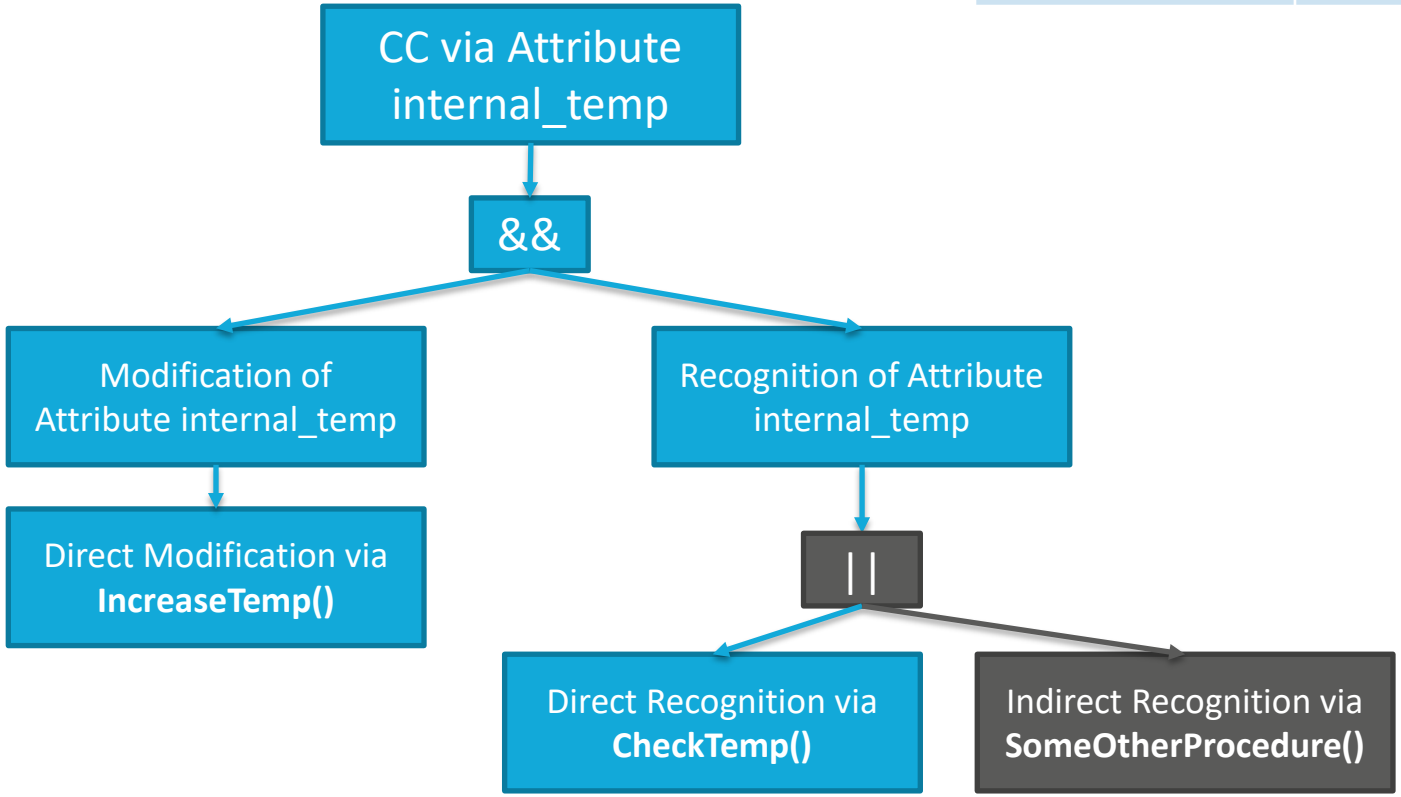


[1] Kemmerer, R., Porras, P.: Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels, Trans. Software Engineering, IEEE, 1991.

Covert Flow Trees [1]

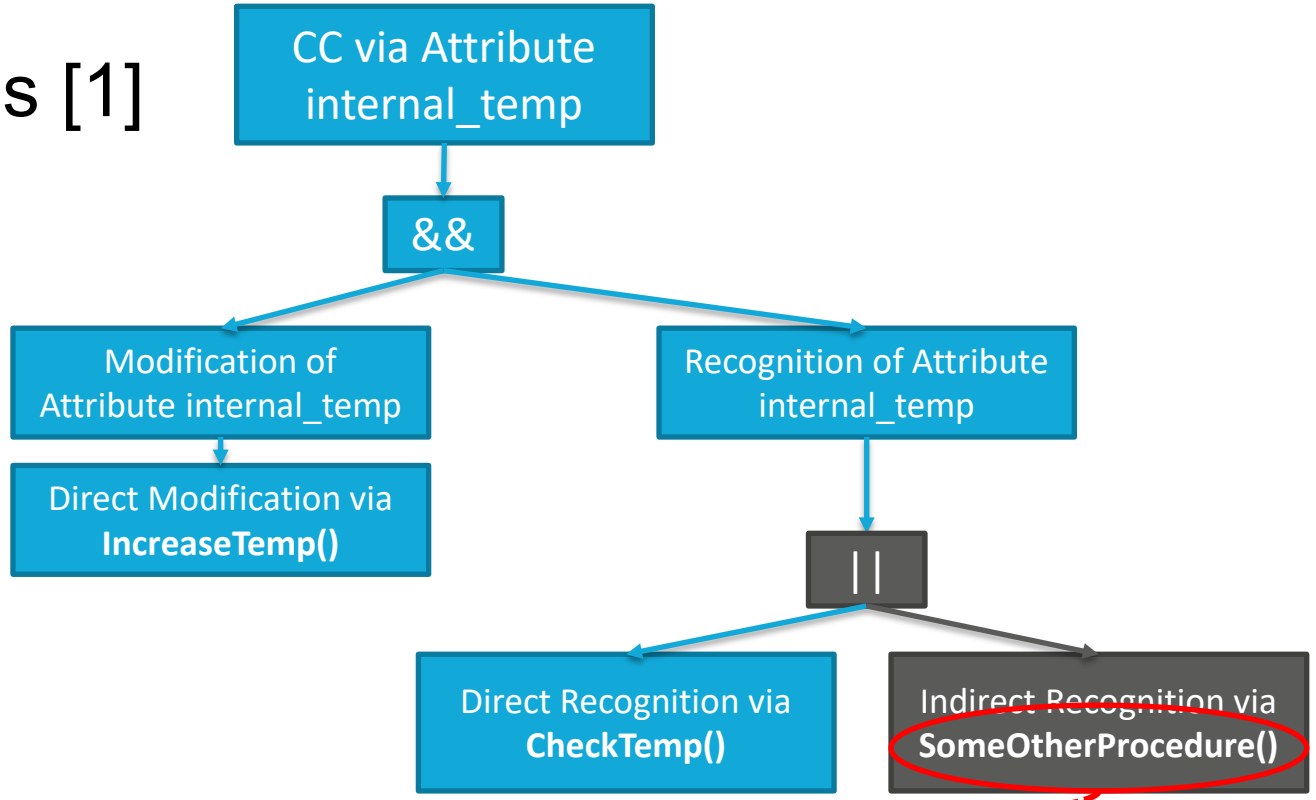
	IncreaseTemp	CheckTemp
Reference	cur_temp, internal_temp	internal_temp soft_limit
Modify	internal_temp	-
Return	-	internal_temp

- Indirect channels can also be detected



[1] Kemmerer, R., Porras, P.: Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels, Trans. Software Engineering, IEEE, 1991.

Covert Flow Trees [1]



Generation of CFT Lists:

They contain sequences of operations that represent a potential covert channel.

- List 1: Operations capable of modifying an attribute
- List 2: Operations capable of reading an attribute

- List 1: (`IncreaseTemp()`)
- List 2: (`CheckTemp()`, `SomeOtherProcedure()`)

Finally, one combines both lists to determine the potential covert channel‘s flows:

- `IncreaseTemp()` → `CheckTemp()`
- `IncreaseTemp()` → `SomeOtherProcedure()`

[1] Kemmerer, R., Porras, P.: Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels, Trans. Software Engineering, IEEE, 1991.

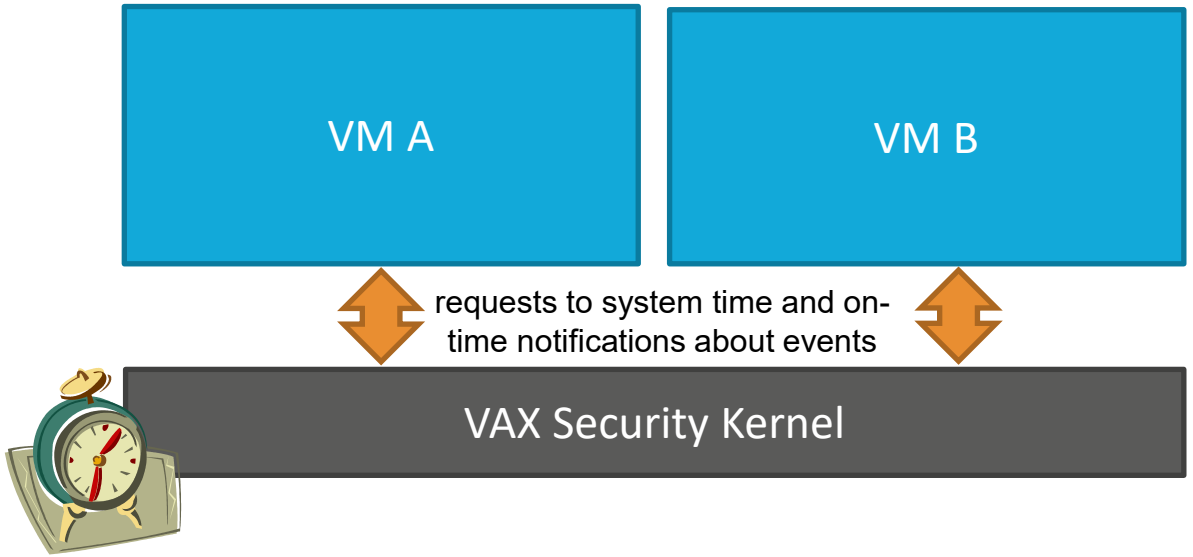
Covert Flow Trees [1,2]

- Discussion:
  - CFTs can only be applied at the source code level (drawback in comparison to the SRM)
  - Nobody has published work on timing channel detection; so far, CFTs can only be applied to detect storage channels
  - Visual representation of flows and automatic CFT generation supported by tools
  - Support for indirect information flows

[1] Kemmerer, R., Porras, P.: Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels, Trans. Software Engineering, IEEE, 1991.
[2] Bishop, M.: Computer Security, Art and Science, Addison-Wesley Professional, 2003, Chapter 17.

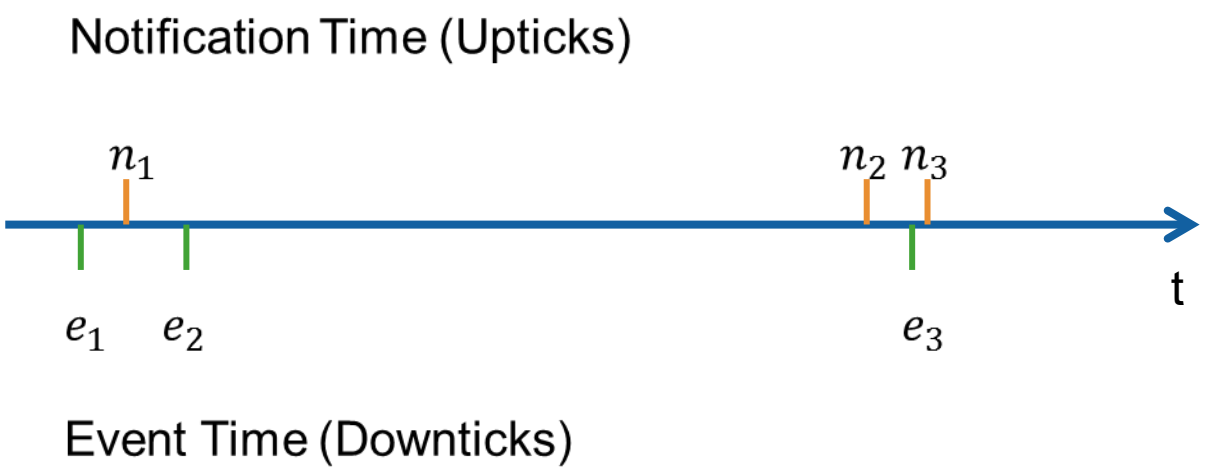
Fuzzy Time [1]

- Approach by W.-M. Hu to limit the channel capacity of covert timing channels between virtual machines; already in 1991 (VAX security kernel).
- The more precise a time measurement is, the higher is the channel capacity (finer distinction of elapsed time possible).
- No detection or prevention of timing channels.



[1] Hu, W.-M.: Reducing Timing Channels with Fuzzy Time, Symp. Security and Privacy, IEEE, 1991.

Fuzzy Time [1]



[1] Hu, W.-M.: Reducing Timing Channels with Fuzzy Time, Symp. Security and Privacy, IEEE, 1991.

Spurious Processes Approach [1]

- Originally designed for databases, however, here explained for filesystem-utilizing storage channels

- **Basic idea:** Introduce a „spurious process“ (SP) into all potentially covert communications between two regular processes of an operating environment.
  - limits capacity of covert storage channels
  - SP introduced on context switch if a shared object is accessed by two processes without previous access of SP to the same object. SP has the same permissions as P2.

- Example: Two Processes in MLS system; unique filenames; P2 calls Create().

P1‘s behavior	**P1 creates file**		**P1 does *not* create file**	
SP‘s behavior	Create()	Create() + Remove()	Create()	Create() + Remove()
Result	File exists	File exists because in write-down, the SP lacks rights	File exists	File does not exist
P2 receives	1 (unsure, whether P1 or SP created file)			0 (sure)

Fadlalla, Y. A. H.: Approaches to Resolving Covert Storage Channels in Multilevel Secure Systems, PhD thesis, University of New Brunswick, 1996.