The Container Security in Healthcare Data Exchange System

Bachelor's degree graduation project

Chih-Hsuan Yang

National Sun Yat-sen University
Advisor: Chun-I Fan

May 28, 2021

Outline

Outcome


健康資料在HL7-FHIR架構下，加速資料交換互通，增進加值應用
新世代電子病歷
交換應用
資料標準
產官學合作
跨部會資源
資料交換平台
推動
堆疊
參與
共享
重點作法
發展交換平台
試作疫苗護照等應用
重點作法
新舊標準平行轉移
部內資料交換電子病歷化
重點作法
發展醫療院所資料轉換工具
建構FHIR人力資源
重點作法
協同經濟部補助聯測及產品服務等資源

2017衛生福利部電子病歷資訊安全檢查表

Change password?

19	具特殊權限公用程式之使用		**目的：** 應限制及嚴密控制可能篡越系統及應用控制措施之公用程式的使用。 **合格項目：** 特權的公用程式應造冊，每次抽查時，未限制不得超過3件。 **[註]**未有使用特權的公用程式者，可自選本條免評。	☐免評 ☐合格 ☐不合格	☐免評 ☐合格 ☐不合格

Parallel permission

19	具特殊權限公用程式之使用		**目的：** 應限制及嚴密控制可能篡越系統及應用控制措施之公用程式的使用。 **合格項目：** 特權的公用程式應造冊，每次抽查時，未限制不得超過3件。 **[註]**未有使用特權的公用程式者，可自選本條免評。	☐免評 ☐合格 ☐不合格	☐免評 ☐合格 ☐不合格

Without encryption?

六、密碼學					
21	密碼控制措施		**目的：** 資訊系統設定加解密演算機制有否符合院內規定，且機密資訊應加密儲存。 **合格項目：** 訂有加解密機制之規範與落實執行。 **[註]**醫院之資訊系統若無設定加解密機制，可自選本條免評。	☐免評 ☐合格 ☐不合格	☐免評 ☐合格 ☐不合格

2017衛生福利部電子病歷資訊安全檢查表

FTP? SFTP?

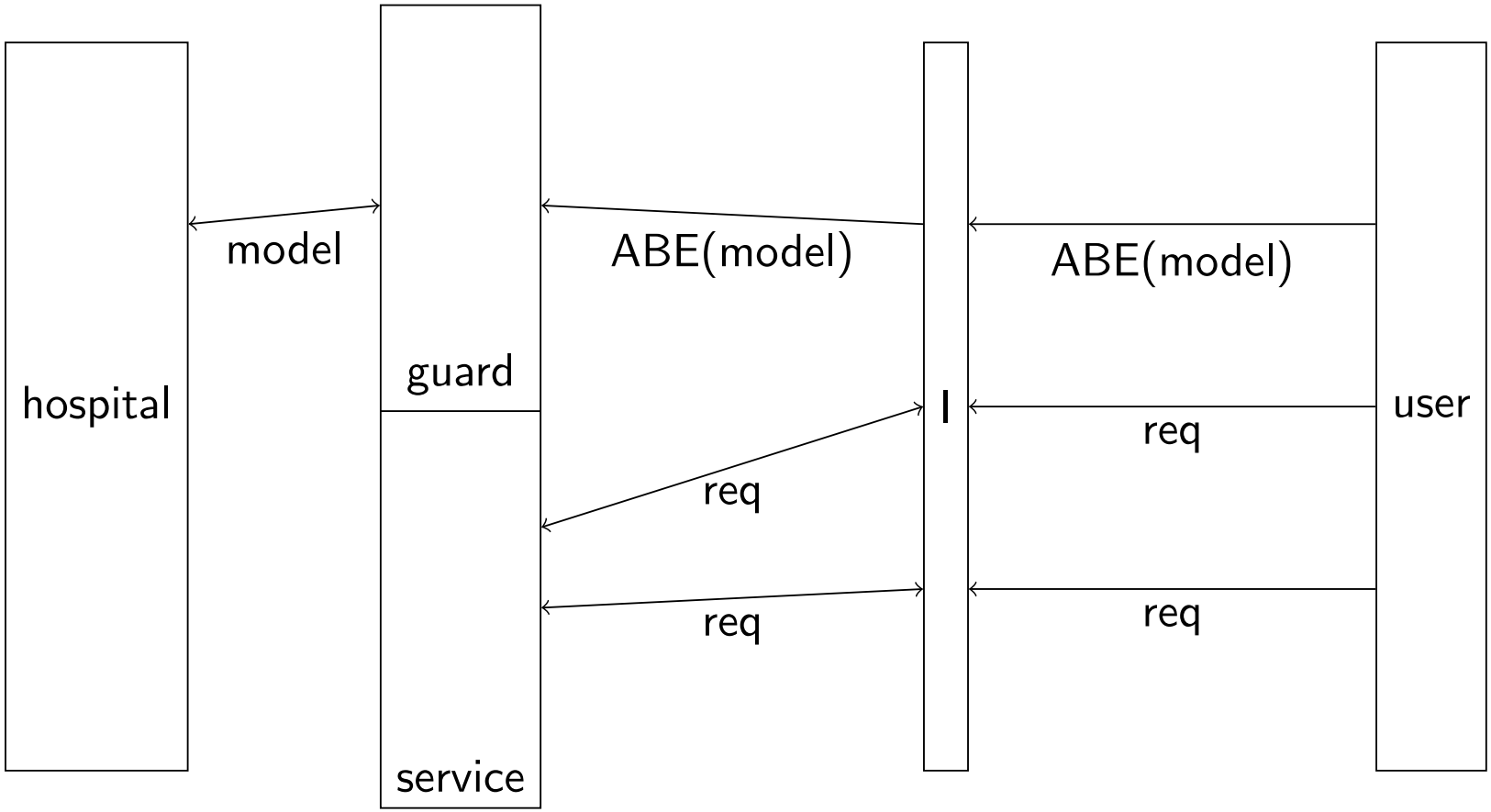
36	資訊傳送政策/程序與協議		**目的：** 應訂有資訊傳送協議(內外部)、政策、程序及控制措施，以保護經由使用所有型式通訊設施或電子(例如電子郵件、即時通訊或 FTP 資料傳檔等)等資訊傳送。 **合格項目** 訂有相關規範與落實執行。	☐合格 ☐不合格	☐合格 ☐不合格

Without certificate?

項次	項目	必要	評量項目	自評結果	檢查結果
22	金鑰		**目的：** 金鑰管理(如軟體憑證等)須符合院內規定。 **合格項目：** 訂有金鑰管理之規範與落實執行。 **[註]**醫院若無軟體憑證等金鑰，可自選本條免評。	☐免評 ☐合格 ☐不合格	☐免評 ☐合格 ☐不合格

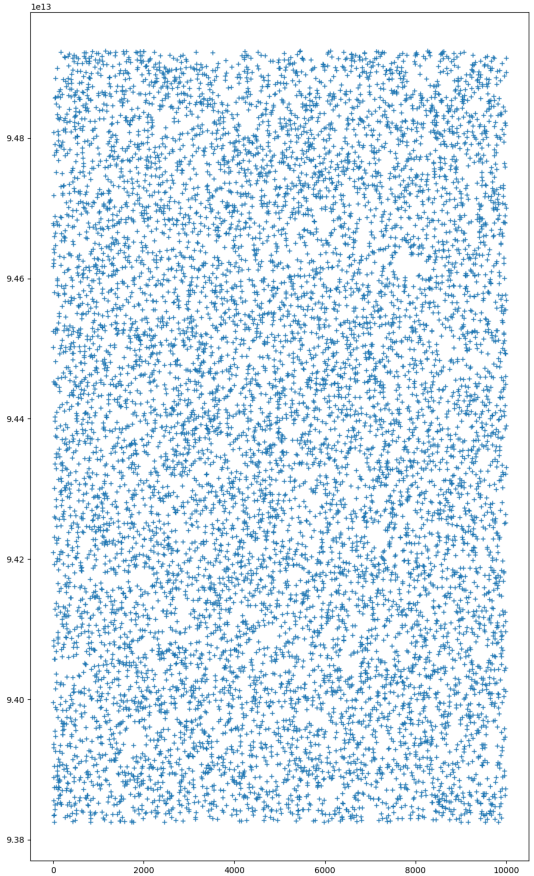
嚴重懷疑沒有經過資安專家審核

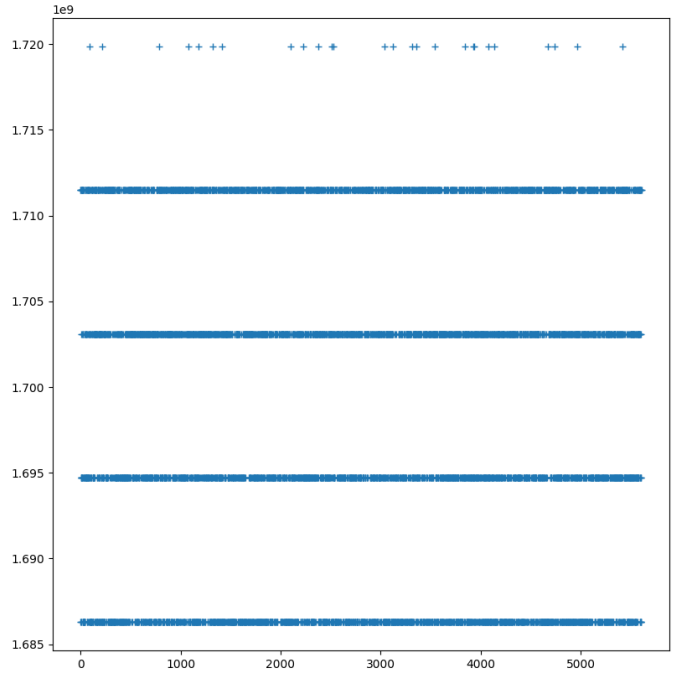
Attribute Based Homomorphic Encryption



Fix the previous test

Outcome - (k)ASLR





The User Mode Linux

The relationship between UML processes and the corresponding host processes for each mode follows from this. Figure 9.3 shows these relationships.

`tt` mode really only exists on x86 hosts. The x86_64 and S/390 ports were made after `skas0` mode was implemented, and they both use that rather than `tt` mode. Because of this, in the following discussion about `tt` mode, I will talk exclusively about x86. Also, the discussion about address space sizes and constraints on UML physical memory sizes are confined to x86, since this issue affects only 32-bit hosts.

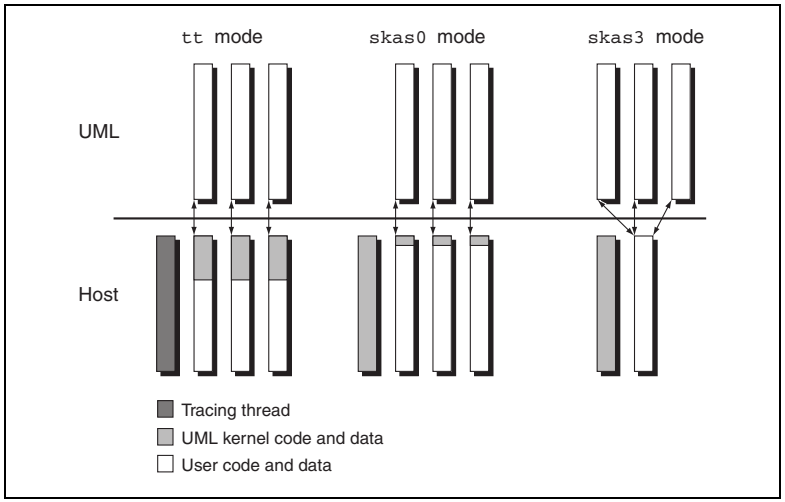
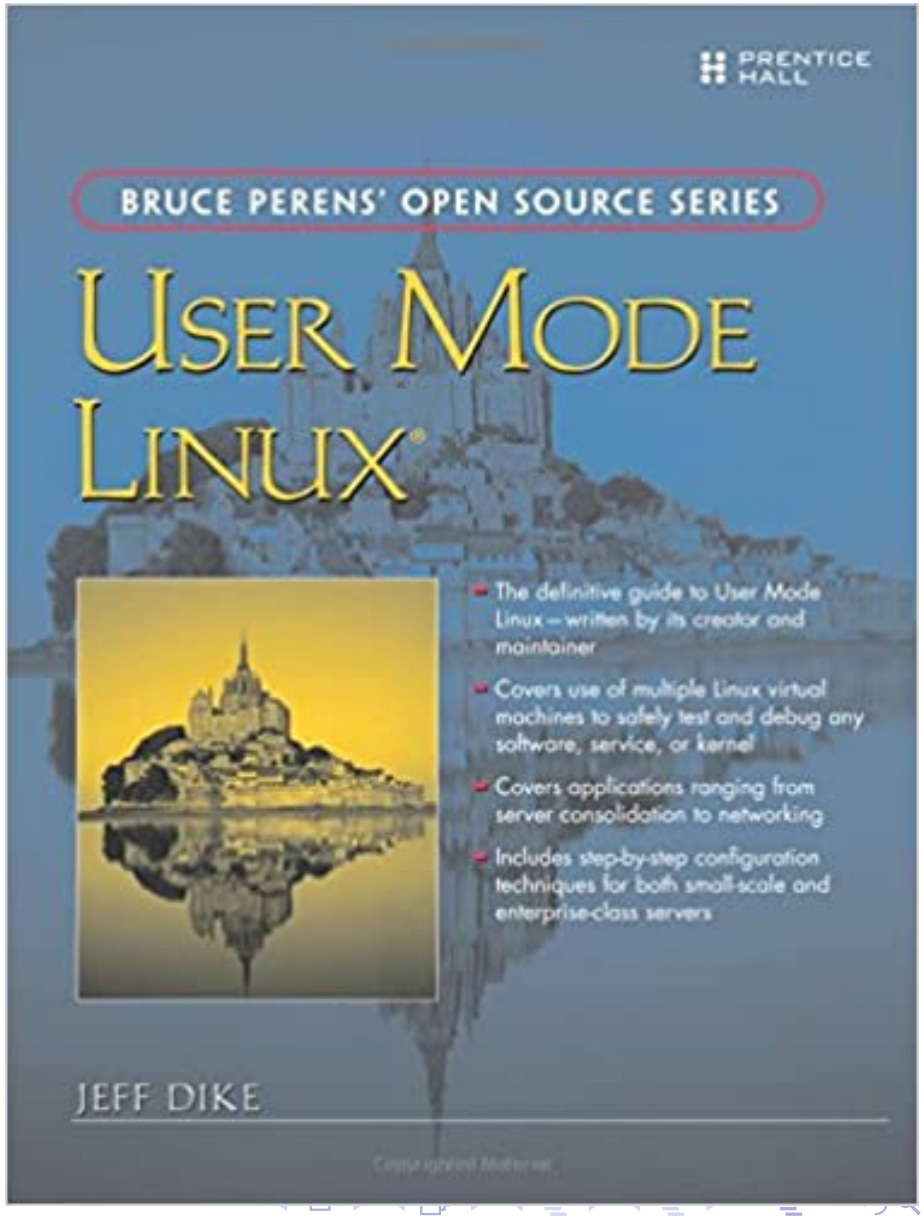


Figure 9.3 Comparison of the three UML execution modes. `tt` mode has a separate host thread (the tracing thread), which controls the execution of the other threads. Processes and threads within the UML instance have corresponding threads on the host. Each such host process has the UML kernel mapped into the top of its address space. In `skas3` mode, there is no separate tracing thread—this role is performed by the kernel thread. There is a single process on the host in which all UML processes run. `skas0` mode is a hybrid of `tt` mode and `skas3` mode. Like `skas3` mode, there is no tracing thread and there is a separate kernel thread in which the UML kernel lives. Like `tt` mode, each UML process has a corresponding host process.



Study

static __latent_entropy struct task_struct *copy_process

- fork.c L1851-2399
- 549

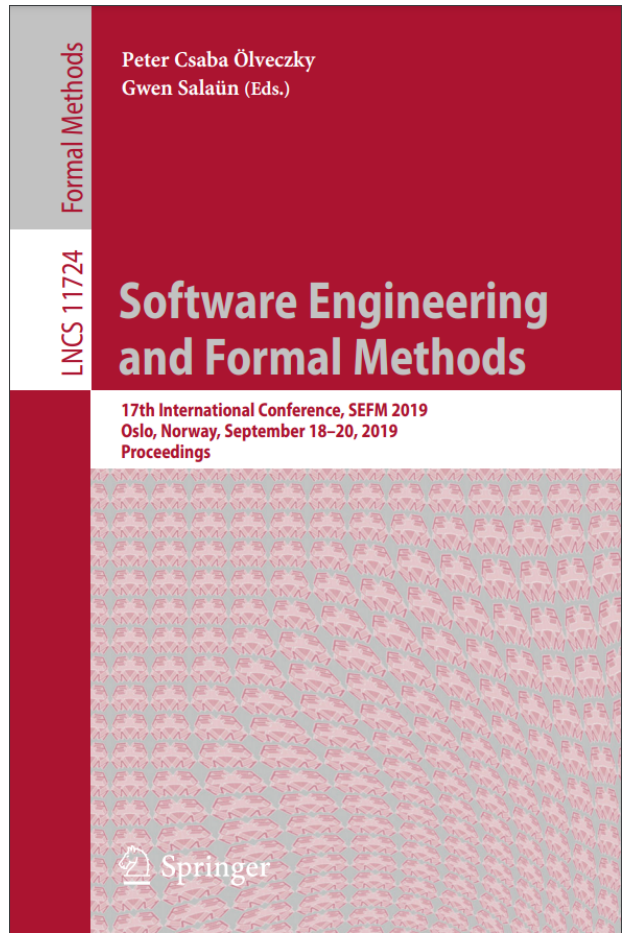
task_struct

- sched.h L649-1401
- 753

Formal verification

Formal verification

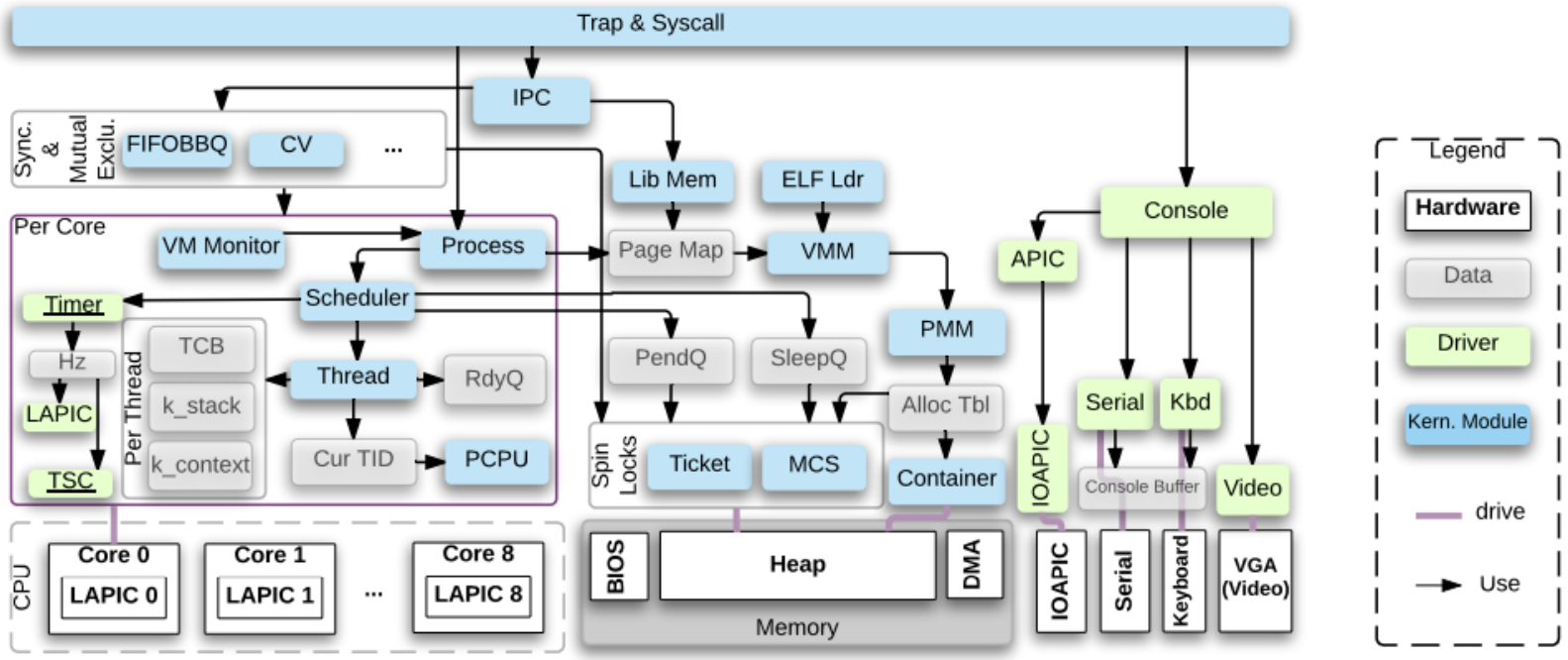
- Abstract Interpretation
- Formal Model Checking
- Theory Prover



[1]

Formal verification

System architecture for the sample kernel [2]



Formal verification

The contextual refinement property about the s kernel can be stated as:

$$\forall P, [[K\&P]]_{x86mc} \subseteq [[P]]_s$$

Formal verification

L: each layer interface
A: an active thread set
$EC(L, A)$: set of valid environment contexts
$\prod_{L(A)}(P, \epsilon)$: thread-modular machine
The semantics for a concurrent layer machine L is then:

$$[[P]]_{L(A)} = \{\prod_{L(A)}(P, \epsilon) | \epsilon \in EC(L, A)\}$$

λ-calculus

Formal Small-step Verification of a Call-by-value Lambda Calculus Machine [3]

Stack

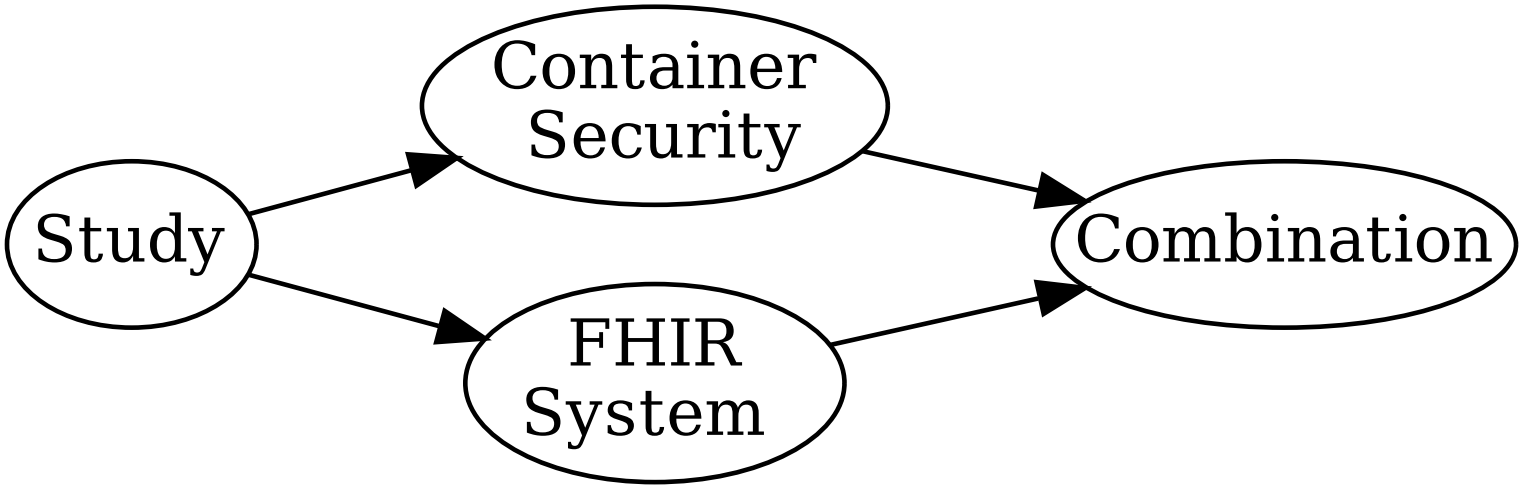
$$(T, V) \gg \sigma := \mathit{closed}(T, V) \wedge (\delta_0 @ T, \delta_1 @ V) = \sigma$$

Heap

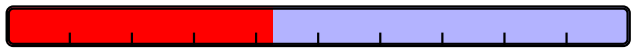
$$(T, V, H) \gg (\dot{T}, \dot{V}) = T \gg_H \dot{T} \wedge V \gg_H \dot{V}$$

[1, 2, 3]

Working flow


Container
Security
Study
Combination
FHIR
System

Time bar

- 專題競賽暨成果展:  $\frac{113}{265} \sim 0.4264$

References

References I

[1] Marjan Sirjani Alessandro Cimatti, ed. *Software Engineering and Formal Methods*. Springer, Cham, 2017. DOI: https://doi.org/10.1007/978-3-319-66197-1.

[2] Ronghui Gu et al. "CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels". In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. OSDI'16. Savannah, GA, USA: USENIX Association, 2016, 653–669. ISBN: 9781931971331.

[3] Fabian Kunze, Gert Smolka, and Yannick Forster. "Formal Small-Step Verification of a Call-by-Value Lambda Calculus Machine". In: *Lecture Notes in Computer Science* (2018), 264–283. ISSN: 1611-3349. DOI: 10.1007/978-3-030-02768-1_15. URL: http://dx.doi.org/10.1007/978-3-030-02768-1_15.