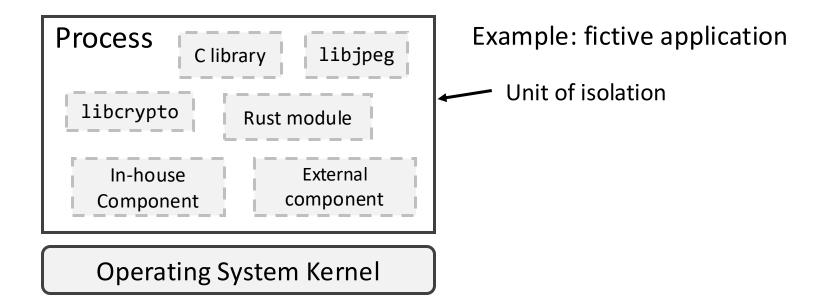


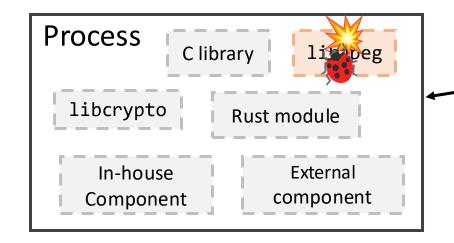
Software Compartmentalization Everywhere - What Will it Take?



Hugo Lefeuvre

Postdoctoral Research Fellow
University of British Columbia
Vancouver (191)





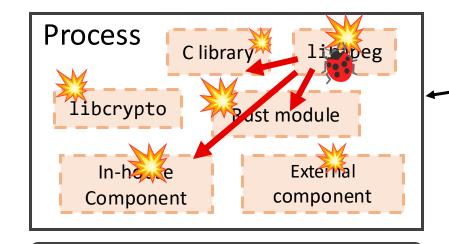
Operating System Kernel

Example: fictive application

Unit of isolation



Attacker exploits bug in one of the libraries



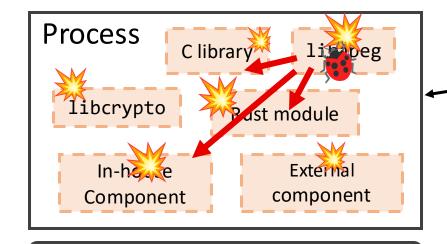
Operating System Kernel

Example: fictive application

Unit of isolation



Spreads trivially because of **ambient trust**



Operating System Kernel

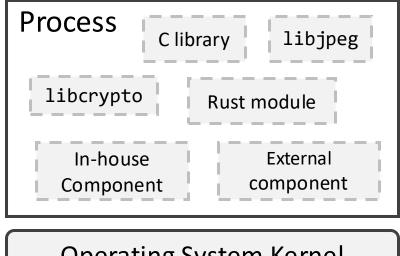
Unit of isolation

Example: fictive application



Spreads trivially because of **ambient trust**

This historical approach is increasingly problematic

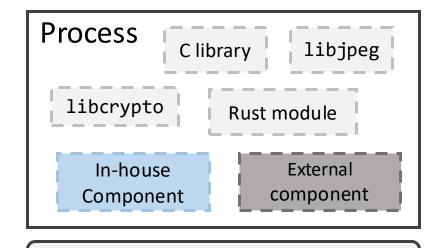


Operating System Kernel

This historical approach is increasingly problematic

Among others...

Modern programs are deeply heterogeneous



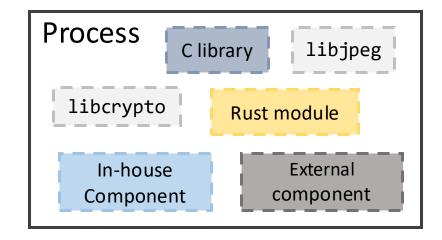
Operating System Kernel

This historical approach is increasingly problematic

Among others...

Modern programs are deeply heterogeneous

Different origins...



Operating System Kernel

This historical approach is increasingly problematic

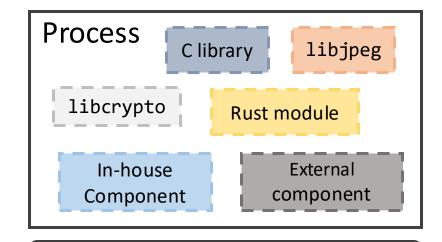
Among others...

Modern programs are deeply heterogeneous

programming languages...

Different

Different origins...



This historical approach is increasingly problematic

Operating System Kernel

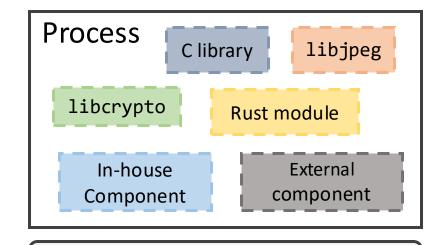
Among others...

Modern programs are deeply heterogeneous

Different programming languages...

Different origins... Diff

Different degrees of risk...



Operating System Kernel

This historical approach is increasingly problematic

Among others...

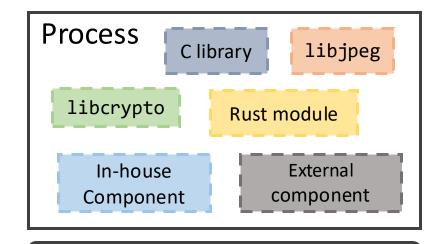
Modern programs are deeply heterogeneous

Different programming languages...

More or less valuable data...

Different origins...

Different degrees of risk...



Operating System Kernel

This historical approach is increasingly problematic

Different

programming

languages...

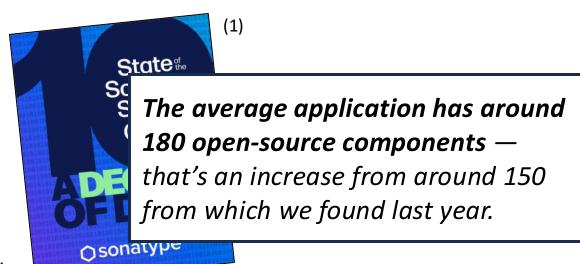
Among others...

Modern programs are deeply heterogeneous

More or less valuable data...

Different origins...

Different degrees of risk...



(1) 10th Annual State of the Software Supply Chain, Sonatype Inc.

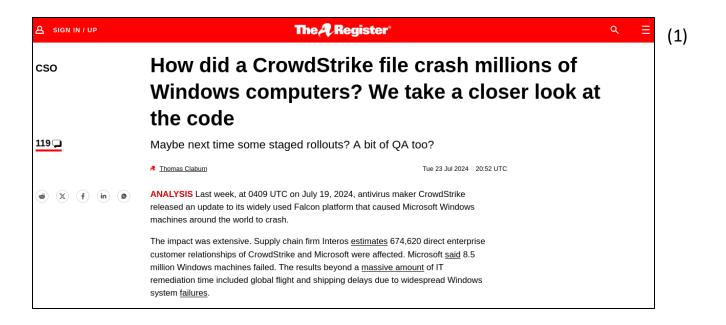


Illustration of this problem:

Last year's Crowdstrike disaster where a bug in an external antivirus plug-in crashes millions of computers



Illustration of this problem:

Last year's Crowdstrike disaster where a bug in an external antivirus plug-in crashes millions of computers

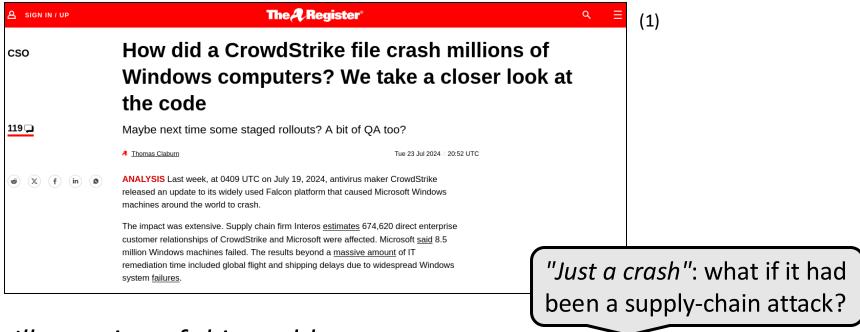


Illustration of this problem:

Last year's Crowdstrike disaster where a bug in an external antivirus plug-in crashes millions of computers

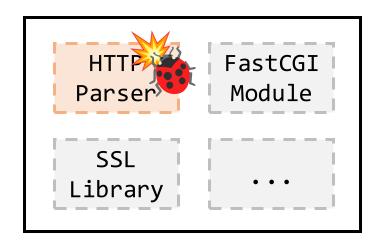
Program = single unit of trust

Software compartmentalization =

- design programs split into distrusting and isolated components
- such that if one of the components is breached, the attacker does
 not compromise the whole program

Software compartmentalization =

- design programs split into distrusting and isolated components
- such that if one of the components is breached, the attacker does not compromise the whole program

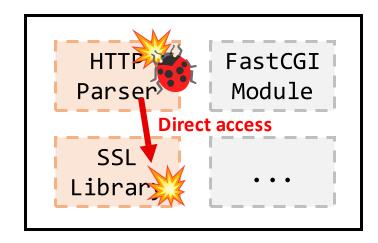


Monolithic

Example: a web server (Apache)

Software compartmentalization =

- design programs split into distrusting and isolated components
- such that if one of the components is breached, the attacker does not compromise the whole program

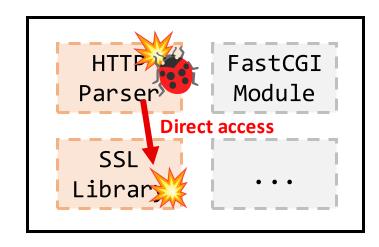


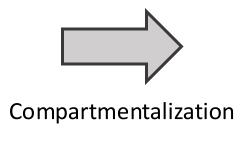
Monolithic

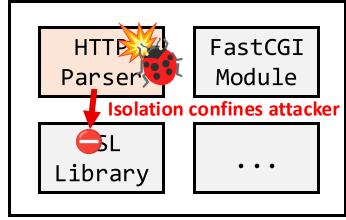
Example: a web server (Apache)

Software compartmentalization =

- design programs split into distrusting and isolated components
- such that if one of the components is breached, the attacker does not compromise the whole program







Monolithic

Compartmentalized

Example: a web server (Apache)

Isolation boundary

Software compartmentalization =

- design programs split into distrusting and isolated components
- such that if one of the components is breached, the attacker does
 not compromise the whole program

Compartmentalization is applied to high-profile software

Server Software

























Area of investment and support

Digital security by design

The digital security by design (DSbD) challenge funds business and researchers to update the foundation of the insecure digital computing infrastructure by creating a

such

not

Budget: £70 million

Duration: From 2020 to 2025

Partners involved: Innovate UK, Engineering and Physical Sc

new, more secure hardware and software ecosystem.

(EPSRC)



Significant research funding in recent years

solated components hed, the attacker does

INDUSTRY EMERGING INNOVATION INDUSTRY NEWS

\$50M

Contracts for innovation: DSbD Advancing **CHERI Tools and software**

| Opportunity status: | Closed |
|---------------------|---|
| Funders: | Innovate UK |
| Co-funders: | Department for Science, Innovation and Technology |
| Funding type: | Other |
| Publication date: | 19 May 2025 |
| Opening date: | 14 May 2025 9:30am UK time |
| Closing date: | 18 June 2025 11:00am UK time |
| | |

See the full opportunity details on the Innovation Funding Service

Organisations can apply for a share of up to £12 million, inclusive of VAT, to work on maturing and enabling the availability of CHERI Tools and Software components for RISC-V embedded devices that implement the CHERI architecture extensions

DARPA to Hold Compartmentalization and Privilege Management (CPM) **Proposers Day**

Full solutions are anticipated to require new hardware support, though software-only solutions that can meet performance expectations are in-scope for the program.

By Homeland Security Today April 1, 2023

(1)

Re: double-free vulnerability in OpenSSH server 9.1 (CVE-2023-25136)

From: Qualys Security Advisory <qsa () qualys com> Date: Mon, 13 Feb 2023 12:02:13 +0000



Hi all,

On Thu, Feb 02, 2023 at 01:02:04PM +0000, Qualys Security Advisory wrote: Exploiting this vulnerability will not be easy: modern memory allocators provide protections against double frees, and the impacted sshd process is unprivileged and heavily sandboxed.

Quick update: we were able to gain arbitrary control of the "rip" register through this bug (i.e., we can jump wherever we want in sshd's address space) on an unpatched installation of OpenBSD 7.2 (which runs OpenSSH 9.1 by default). This is by no means the end of the story: this was only step 1, bypass the malloc and double-free protections. The next steps, which may or may not be feasible at all, are:

- step 2, execute arbitrary code despite the ASLR, NX, and ROP protections (this will probably require an information leak, either through the same bug or through a secondary bug);
- step 3, escape from sshd's sandbox (through a secondary bug, either in the privileged parent process or in the kernel's reduced attack surface).

- OpenSSH is vulnerable to a bug with remote-code execution
- It is exploitable: attackers can break allocator-based mitigations
- Compartmentalization ultimately mitigates the exploit by containing it in an unprivileged process

(1) https://seclists.org/oss-sec/2023/q1/92

(1)

Re: double-free vulnerability in OpenSSH server 9.1 (CVE-2023-25136)

From: Qualys Security Advisory <qsa () qualys com> Date: Mon. 13 Feb 2023 12:02:13 +0000

Hi all,

On Thu, Feb 02, 2023 at 01:02:04PM +0000, Qualys Security Advisory wrote:
Exploiting this vulnerability will not be easy: modern memory allocators
provide protections against double frees, and the impacted sshd process
is unprivileged and heavily sandboxed.

Quick update: we were able to gain arbitrary control of the "rip" register through this bug (i.e., we can jump wherever we want in sshd's address space) on an unpatched installation of OpenBSD 7.2 (which runs OpenBSH 9.1 by default). This is by no means the end of the story: this was only step 1, bypass the malloc and double-free protections. The next steps, which may or may not be feasible at all, are:



- step 2, execute arbitrary code despite the ASLR, NX, and ROP protections (this will probably require an information leak, either through the same bug or through a secondary bug);
- step 3, escape from sshd's sandbox (through a secondary bug, either in the privileged parent process or in the kernel's reduced attack surface).

- OpenSSH is vulnerable to a bug with remote-code execution
- It is exploitable: attackers can break allocator-based mitigations
- Compartmentalization ultimately mitigates the exploit by containing it in an unprivileged process

(1) https://seclists.org/oss-sec/2023/q1/92

(1)

Re: double-free vulnerability in OpenSSH server 9.1 (CVE-2023-25136)

From: Qualys Security Advisory <qsa () qualys com> Date: Mon. 13 Feb 2023 12:02:13 +0000

Hi all,

On Thu, Feb 02, 2023 at 01:02:04PM +0000, Qualys Security Advisory wrote:
Exploiting this vulnerability will not be easy: modern memory allocators
provide protections against double frees, and the impacted sshd process
is unprivileged and heavily sandboxed.

Quick update: we were able to gain arbitrary control of the "rip" register through this bug (i.e., we can jump wherever we want in sshd's address space) on an unpatched installation of OpenBSD 7.2 (which runs OpenSSH 9.1 by default). This is by no means the end of the story: this was only step 1, bypass the malloc and double-free protections. The next steps, which may or may not be feasible at all, are:

- step 2, execute arbitrary code despite the ASLR, NX, and ROP protections (this will probably require an information leak, either through the same bug or through a secondary bug);
- step 3, escape from sshd's sandbox (through a secondary bug, either in the privileged parent process or in the kernel's reduced attack surface).

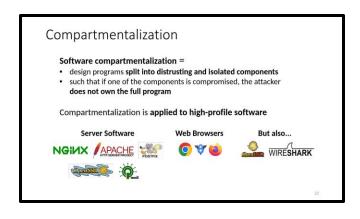


- It is exploitable: attackers can break allocator-based mitigations
- Compartmentalization ultimately mitigates the exploit by containing it in an unprivileged process

(1) https://seclists.org/oss-sec/2023/q1/92

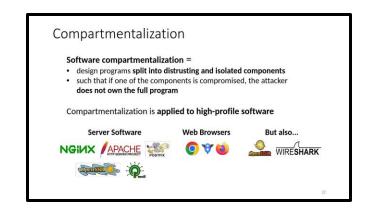
Yet, in 2025, compartmentalization is not a widespread practice

Yet, in 2025, compartmentalization is not a widespread practice



Beyond the popular software I mentioned earlier, compartmentalization is rather rare

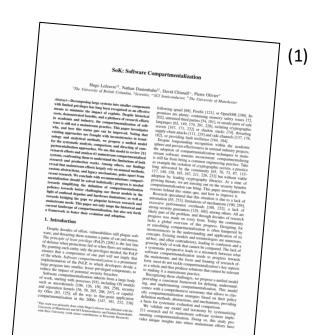
Yet, in 2025, compartmentalization is not a widespread practice



Beyond the popular software I mentioned earlier, compartmentalization is rather rare

"Less than 56 apps out of the 1,520 most popular Debian applications are compartmentalized"

(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025



Why?

Why?

SoK: Software Compartmentalization

Hugo Lefeuvre^{†1}, Nathan Dautenhahn^{‡1}, David Chisnall**, Pierre Olivier* Tugo Leteuvre⁻⁻, Nation Daurenman⁻⁻, David Choshau⁻, Fietre Chiver⁻

The University of British Columbia, ¹Serentite, ²SCI Semiconductor, ²The University of Maschester

Abstract—Decomposing large systems into smaller components Assistance—Accomposing large systems into sunance components with limited privileges has long been recognized as an effective means to minimize the impact of exploits. Despite historical roots, demonstrated benefits, and a plethora of research efforts in academia and industry, the compartmentalization of software is still not a mainstream practice. This paper investigates water is sturn not a mainstream practice. This paper investigates why, and how this status quo can be improved. Noting that existing approaches are fraught with inconsistencies in termiexisting approaches are traught with inconsistencies in termi-nology and analytical methods, we propose a unified model for the systematic analysis, comparison, and directing of compartmentalization approaches. We use this model to review 211 research efforts and analyze 61 mainstream compartmentalized systems, confronting them to understand the limitations of both systems, communing ment to mouresonant in the systems, our findings research and production works. Among others, our findings research and production works. Among others, our innuings reveal that mainstream efforts largely rely on manual methods, reveal that mainstream errors rargery recy on maintai memors, custom abstractions, and legacy mechanisms, poles apart from custom anstractions, and regacy mechanisms, pores apart virun recent research. We conclude with recommendations: compart-mentalization should be solved holistically; progress is needed increaseastion smooth personal description of compartmentalization towards simplifying the definition of compartmentalization policies; towards better challenging our threat models in the policies; towards netter chanlenging our inreat inouels in the light of confused deputies and hardware limitations; as well as towards bridging the gaps we pinpoint between research and towares ortugung the gaps we purpoint between research and mainstream needs. This paper not only maps the historical and todasseem steen. Fins juster not only maps the instorical and current landscape of compartmentalization, but also sets forth a framework to foster their evolution and adoption.

1. Introduction

Despite decades of effort, vulnerabilities still plague soft-Despite decades of effort, vulnerabilities still plague sort-ware, and thwarting them remains a game of cut and mouse. The principle of lexal privilege (Pol. P) [209] is the last line of defense when protections fail or when flaws are unknown. By granting each entity only the privileges needled, the Pol. P courses that a commencia of one new will not inmult that by graning each entity only the privileges needed, the PoLP-ensures that a compromise of one part will not imply that of the whole. Software compartmentalization is a prominent implementation of the PoLP, in which developers divide a

implementation of the PoLP, in which developers divide a large program analite, lesser privileged components, to reduce the imit of potential security breaches. Software consensational on inherits from a large body of work, starting the processes [95]: including OS models such as microbial policy [106, 126, 150, 164, 258], security and amazurion beneath [106, 126, 150, 164, 258]. such as microkernels [100, 120, 150, 164, 258], security and separation kernels [56, 58, 205, 206, 247], or capability OSes [83, 125]; all the way to fine-graphetision compartmentalization in the 2000s [147, 181, 232, 238]

following qmail [68]. Postfix [121], or OpenSSH [198]. Its Iotiowing quait [68], Postitx [121], or OpenSM [198]. Its promises are proving containing memory safety issues [72, 202], untrusted third parties [54, 181], or unsafe parts of safe languages [62, 149, 174, 201, 228], isolating cryptographic secrets [167, 171, 232] or shadow stacks [74], thwarting samply-chain attacks [111, 234] and side-chaonick [137, 136]. supply-chain attacks [111, 235] and side-channels [137, 176,

supply-chain attacks [111, 253] and side-chainnes [137, 176, 182], or providing fault resilience [164, 184]. Despite longstanding recognition within the academic sphere and proven effectiveness in seminal industry projects, sphere and proven effectiveness in seminal industry projects, sphere and proven effectiveness in seminar mutusity projects, the adoption of compartmentalization techniques in main-stream software remains inconsistent: compartmentalizing stream software remains inconsistent: compartmentalizing is still far from being a common engineering practice. Take as example the isolation of cryptome secrets, a practice long advocated by the community [65, 70, 73, 87, 115–117, 140, 158, 165, 167, 211, 226, 227] but without viable adoution by leading creatographs bilaries. At a time of 117, 140, 130, 103, 103, 211, 220, 2321 but without viatore adoption by leading cryptography libraries. At a time of growing threats, we are missing out on the security benefits compartmentalization can bring. This paper investigates the compartmentalization can oring. It is paper investigates reasons behind this status quo, and how to improve it and the status quo, and how to improve it.

reasons seema unis status quo, and now so improve it.

Research speculated that this situation is due to a lack of
automation [65, 253], limitations of mechanisms [190, 259]. excessive performance overheads [188, 232], a lack of strong security guarantees [129, 160], among others. All are strong security guarantees [1/29, 100], among otners. All are likely part of the problem, and through decades of research likely part of the problem, and through decades of research progress was made on every front. Today the community lacks a global overview of this progress. Design for or retrofitting compartmentalization is often hampered by inconsistencies in the understanding and application of the concepts. Esting models and terminologies are numerous ad-hoc, and often consideration of the configuration of a growing body of work that cannot be compared. The lack of a systematic perspective leads to a mismatch between what a systematic perspective leads to a mismatch between what software compartmentalization needs to progress towards the mainstream, and the focus and framing of research efthe manufeam, and the focus and maning of research of forts; most do not tackle compartmentalization's key aspects as a whole and thus produce solutions that cannot be relevant to making it a mainstream practice.

to inazang n a manstream practice.

Recognizing these challenges, we propose a unified model providing a consistent memowork for defining, understanding, and implementing compartmentalization. This model comes with a comprehensive taxonomy that allows to classify compartmentalization strategies based on their policy stry compartmentanzation strategies based on men poney definition methods, abstractions, and mechanisms, providing

We validate our model and taxonomy by systematizing This work say produced while flago Leferore was affixed with the University of Manchester and SCI Sentenchester, and Nombo Danteshabe with Rev University (with miner contributions at Riverside Research).

We validate our model and taxonomy by systematizing 211 research and 61 mainstream software systems imple-menting compartmentalization. Doing so, this study pro-vides unique insights into where mainstream efforts have

- Started by making a fundamental observation: experts do not even agree on what compartmentalization is
 - Define a fundamental model and consistent terminology for compartmentalization
- Propose a taxonomy for evaluating compartmentalization approaches and apply it systematically to a wide set (200+) of research and mainstream efforts
- From this: extract a set of core challenges

- Started by making a fundamental observation: experts do not even agree on what compartmentalization is
 - Define a fundamental model and consistent terminology for compartmentalization
- Propose a taxonomy for evaluating compartmentalization approaches and apply it systematically to a wide set (200+) of research and mainstream efforts
- From this: extract a set of core challenges

- Started by making a fundamental observation: experts do not even agree on what compartmentalization is
 - Define a fundamental model and consistent terminology for compartmentalization
- Propose a taxonomy for evaluating compartmentalization approaches and apply it systematically to a wide set (200+) of research and mainstream efforts
- From this: extract a set of core challenges

- Started by making a fundamental observation: experts do not even agree on what compartmentalization is
 - Define a fundamental model and consistent terminology for compartmentalization
- Propose a taxonomy for evaluating compartmentalization approaches and apply it systematically to a wide set (200+) of research and mainstream efforts
- From this: extract a set of core challenges

My promise for this talk

A journey through twenty years of compartmentalization

- 1. What is software compartmentalization? (slightly more formal)
- 2. A systematic perspective on compartmentalization
- 3. The why: compartmentalization everywhere, what will it take?

My promise for this talk

A journey through twenty years of compartmentalization

- 1. What is software compartmentalization? (slightly more formal)

- 2. A systematic perspective on compartmentalization
- 3. The why: compartmentalization everywhere, what will it take?

Software compartmentalization is...

a software engineering practice

Software compartmentalization is...

 a software engineering practice where developers break down a program into groups of isolated/distrusting components

Software compartmentalization is...

 a software engineering practice where developers break down a program into groups of isolated/distrusting components, each controlling only the resources they precisely require

Software compartmentalization is...

- a software engineering practice where developers break down a program into groups of isolated/distrusting components, each controlling only the resources they precisely require
- an embodiment of the principle of least-privilege

Software compartmentalization is...

- a software engineering practice where developers break down a program into groups of isolated/distrusting components, each controlling only the resources they precisely require
- an embodiment of the principle of least-privilege

Fundamental idea:

 if a component is compromised, attackers are restricted to the permissions of the compromised component

Software compartmentalization is...

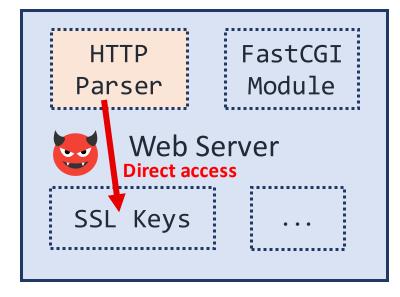
- a software engineering practice where developers break down a program into groups of isolated/distrusting components, each controlling only the resources they precisely require
- an embodiment of the principle of least-privilege

Fundamental idea:

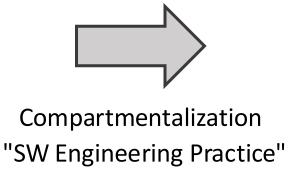
- if a component is compromised, attackers are restricted to the permissions of the compromised component
- to escalate privileges and compromise the rest of the system, attackers must find and exploit additional software vulnerabilities

Going back to our previous example...





Monolithic



HTTP
Parser

Web Server
Isolation prevents leak

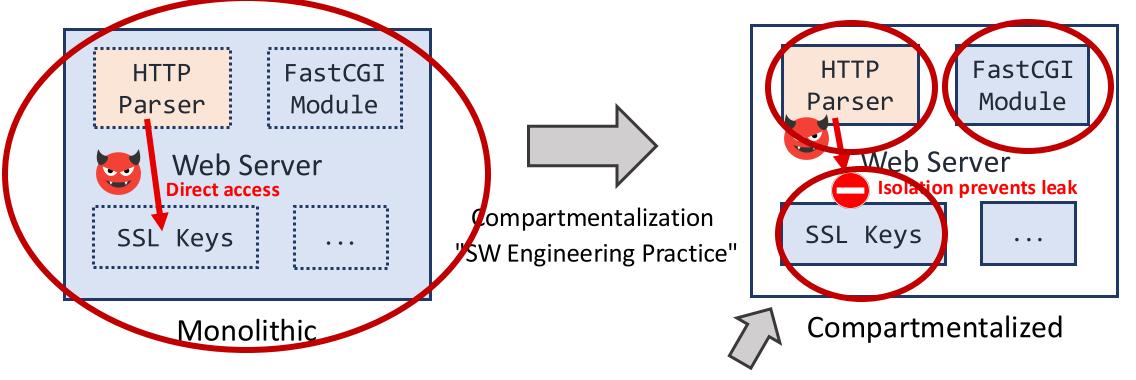
SSL Keys

...

Compartmentalized

Going back to our previous example...





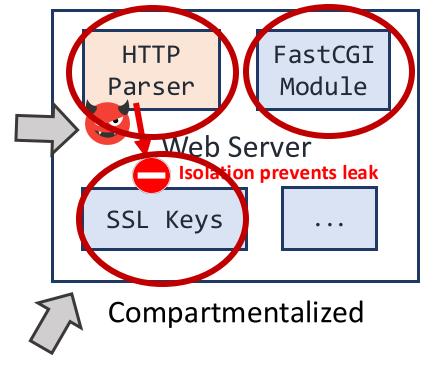
Components now distrust each other

Going back to our previous example...

Protection Domain

Isolation Boundary

Compartmentalizing adds a (potentially very tricky) step on the attacker's path towards taking over the program



Components now distrust each other

Software compartmentalization is applicable to any kind of program

Web Browsers











Microkernels





Software compartmentalization is applicable to any kind of program

Web Browsers













Microkernels





Applications

Software compartmentalization is applicable to any kind of program

Web Browsers











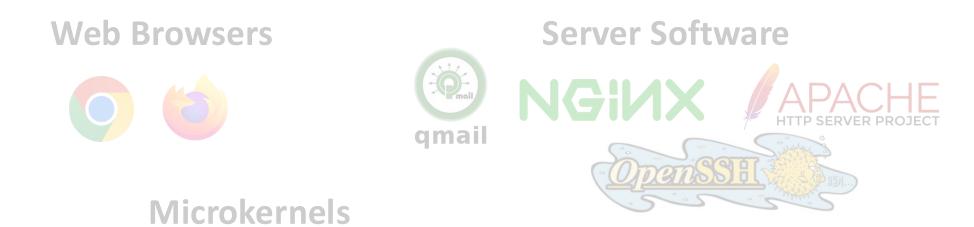
Microkernels





OS kernels

Software compartmentalization is applicable to any kind of program



SELIA Fuchsia

But also: hypervisors, firmware, etc.

Software compartmentalization can be **applied to new as well as existing programs**:

• When applied to existing programs, we talk about retrofitting

Software compartmentalization can be **applied to new as well as existing programs**:

When applied to existing programs, we talk about retrofitting

Some programs which included compartmentalization in their design in the first place (non-exhaustive!):





^{*}Mail transfer agent, one of the first compartmentalized programs to the best of my knowledge

Software compartmentalization can be **applied to new as well as existing programs**:

When applied to existing programs, we talk about retrofitting

Some programs which included compartmentalization in their design in the first place (non-exhaustive!):





...and some where it was retrofitted:



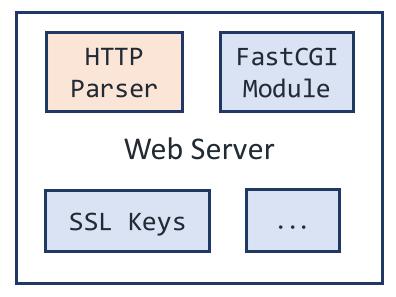


^{*}Mail transfer agent, one of the first compartmentalized programs to the best of my knowledge

Still, software compartmentalization as we refer to here applies within one program, not across programs

Still, software compartmentalization as we refer to here applies within one program, not across programs

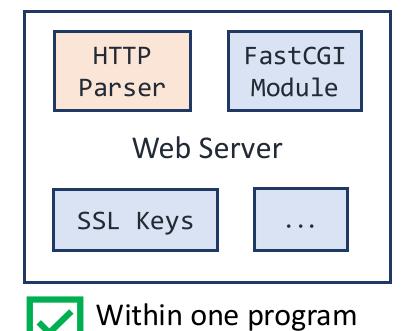
Isolation Boundary





Still, software compartmentalization as we refer to here applies within one program, not across programs

Isolation Boundary



SW Compartmentalization



Still, software compartmentalization as we refer to here applies within

One progran *More general isolation*: share challenges and solutions with software compartmentalization as we define it





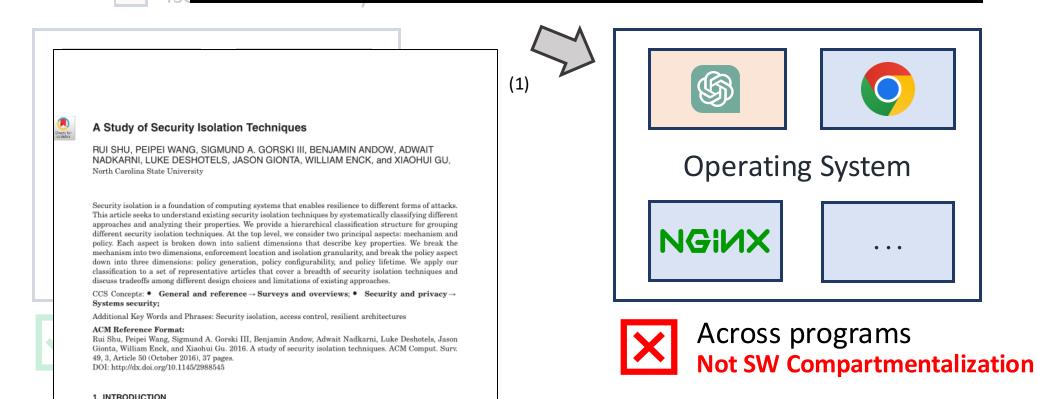




Still, software compartmentalization as we refer to here applies within

one progran

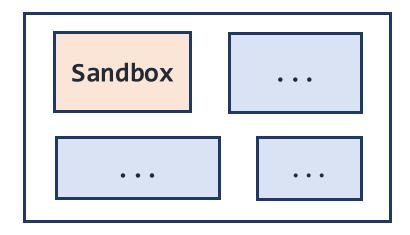
More general isolation: share challenges and solutions with software compartmentalization as we define it



Software compartmentalization can target different trust models

Software compartmentalization can target different trust models

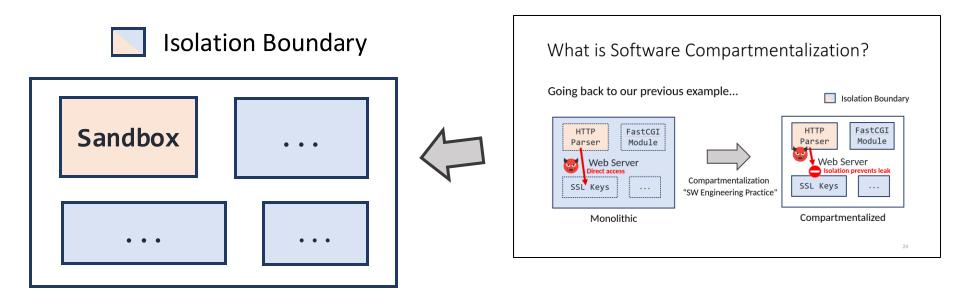
Isolation Boundary



Sandbox: component isolated to protect the rest of the system

All but the sandbox is trusted

Software compartmentalization can target different trust models

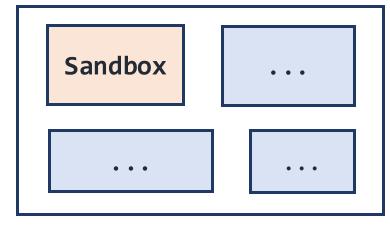


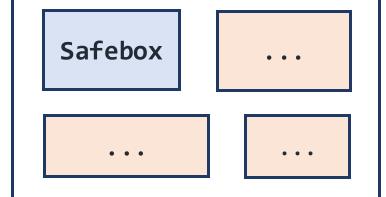
Sandbox: component isolated to protect the rest of the system

All but the sandbox is trusted

Software compartmentalization can target different trust models

Isolation Boundary





Sandbox: component isolated to protect the rest of the system

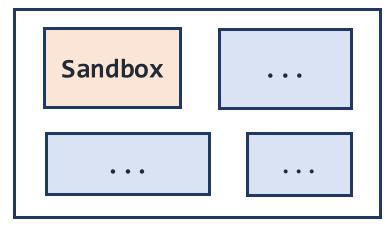
Safebox: component isolated to protect it from others

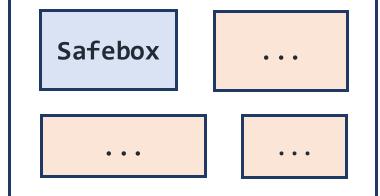
All but the sandbox is trusted

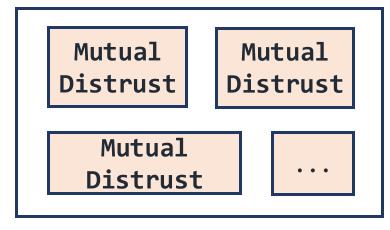
Only the safebox is trusted

Software compartmentalization can target different trust models

Isolation Boundary







Sandbox: component isolated to protect the rest of the system

Safebox: component isolated to protect it from others

Mutual distrust: components distrust each other

All but the sandbox is trusted

Only the safebox is trusted

Software compartmentalization can target different trust models

Isolation Boundary



These trust models are distinct and result in different software architectures





Sandbox: component isolated to protect the rest of the system

Mutual distrust: components distrust each other

All but the sandbox is trusted

Only the safebox is trusted

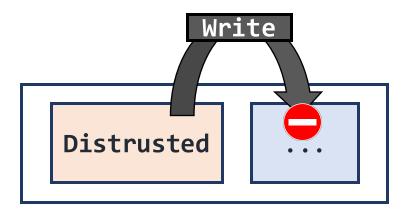
protect it from others

Safebox: component isolated to

Software compartmentalization can target different properties

Software compartmentalization can target different properties

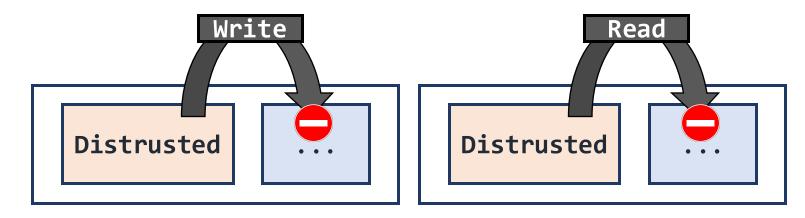
Isolation Boundary



Integrity: compartment cannot alter other compartment's data

Software compartmentalization can target different properties

Isolation Boundary



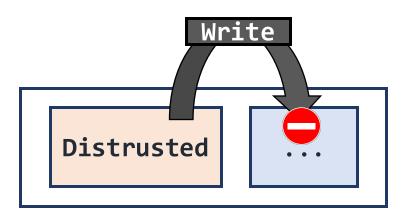
Integrity: compartment cannot alter other compartment's data

Confidentiality: compartment cannot read other compartment's data

Software compartmentalization can target different properties

Read

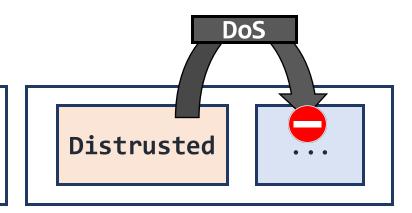
Isolation Boundary



Confidentiality: compartment cannot read other

compartment's data

Distrusted

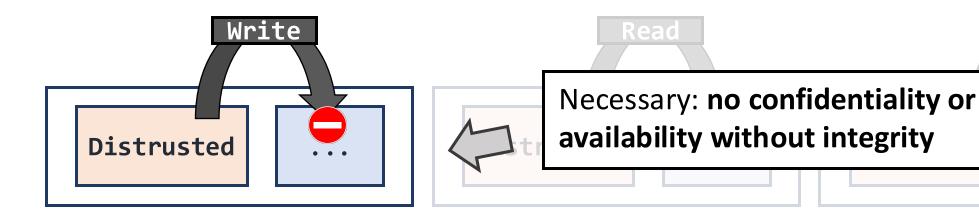


Availability: compartment cannot prevent other compartments from operating

Integrity: compartment cannot alter other compartment's data

Software compartmentalization can target different properties

Isolation Boundary



Integrity: compartment cannot alter other compartment's data

Confidentiality: compartment cannot read other compartment's data

Availability: compartment cannot prevent other compartments from operating

69

My promise for this talk

A journey through twenty years of compartmentalization

- 1. What is software compartmentalization? (slightly more formal)
- 2. A systematic perspective on compartmentalization 4
- 3. The why: compartmentalization everywhere, what will it take?

Take a step back: How do practitioners compartmentalize? What does research say? What are open challenges?

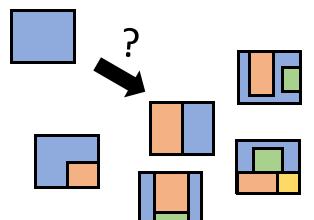
Bigger Picture of Compartmentalization

Propose to view compartmentalization as 3 problems:

- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction
- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism

Bigger Picture of Compartmentalization

Propose to view compartmentalization as 3 problems:



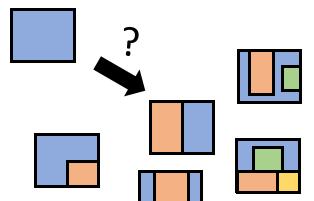
- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction
- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism

- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction

- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism

- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction
- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism





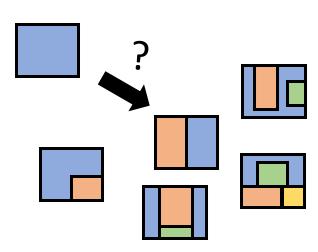
- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction
- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism



Problem #1

How to determine the right policy to enforce?

Developers must define which components are to be separated and which properties should be enforced on the resulting compartments

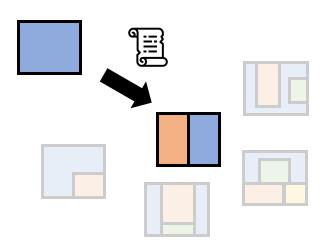


Problem #1

How to determine the right policy to enforce?

Developers must define which components are to be separated and which properties should be enforced on the resulting compartments

The result is a compartmentalization policy



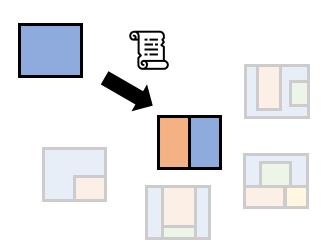
Problem #1

How to determine the right policy to enforce?

Developers must define which components are to be separated and which properties should be enforced on the resulting compartments

The result is a compartmentalization policy

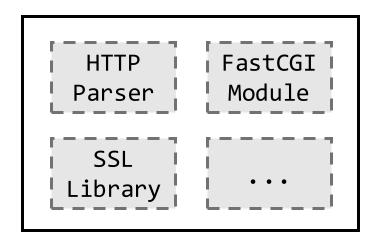
To design policies, developers employ a **Policy Definition Method**



Historically, people have done this manually.

Historically, people have done this manually.

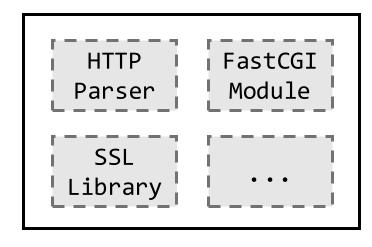
Example: we want to split this C program.



(Fictive monolithic program)

Historically, people have done this **manually**.

Example: we want to split this C program.

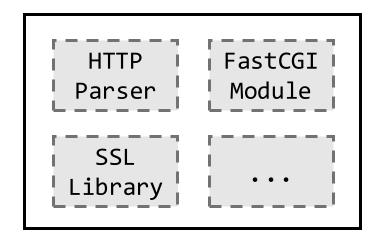


(Fictive monolithic program)

A developer looks attentively at the program...

Historically, people have done this **manually**.

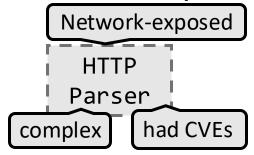
Example: we want to split this C program.



(Fictive monolithic program)

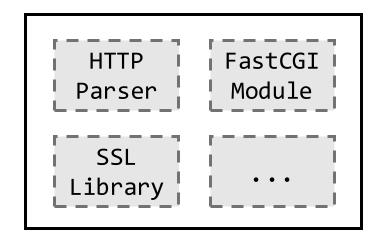
A developer looks attentively at the program...

"I do not trust this component"



Historically, people have done this manually.

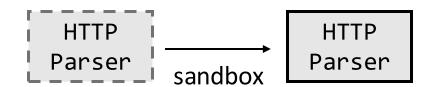
Example: we want to split this C program.



(Fictive monolithic program)

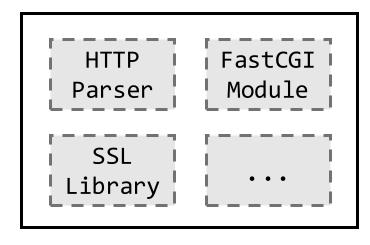
A developer looks attentively at the program...

"I do not trust this component"



Historically, people have done this manually.

Example: we want to split this C program.

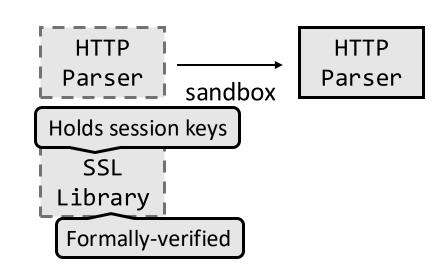


(Fictive monolithic program)

A developer looks attentively at the program...

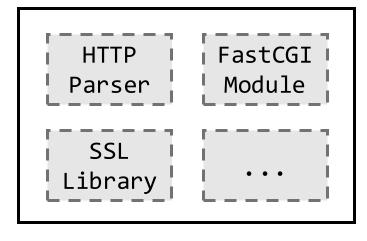
"I do not trust this component"

"I particularly value this component"



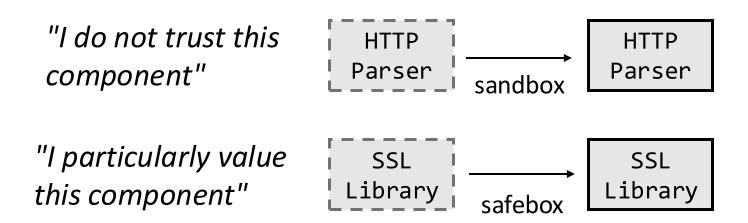
Historically, people have done this manually.

Example: we want to split this C program.



(Fictive monolithic program)

A developer looks attentively at the program...



This is still the most common way to do it today.

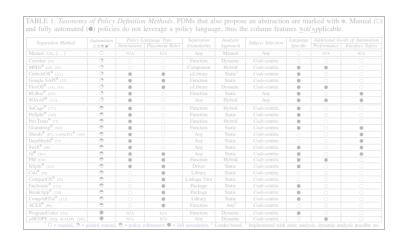
Systematizing Policy Definition Methods

Policy Definition Methods are a very active research area, with goals such as...

Minimize developer effort Maximize soundness Minimize over-privilege

Maximize performance
Maximize security of interfaces

We systematized these works in our SoK





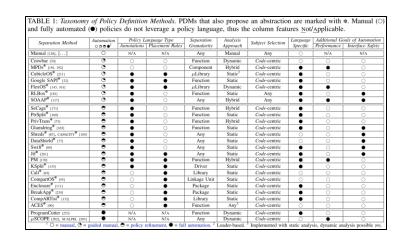
Systematizing Policy Definition Methods

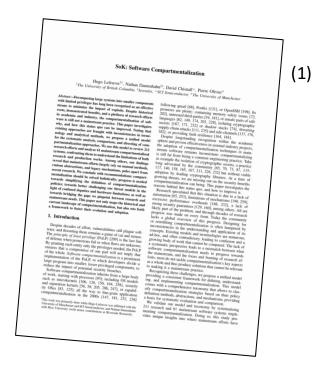
Policy Definition Methods are a very active research area, with goals such as...

Minimize developer effort Maximize soundness Minimize over-privilege

Maximize performance
Maximize security of interfaces

We systematized these works in our SoK





(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

Systematizing Policy Definition Methods

Policy definition methods table from the paper

TABLE 1: *Taxonomy of Policy Definition Methods*. PDMs that also propose an abstraction are marked with *****. Manual (○) and fully automated (●) policies do not leverage a policy language, thus the column features Not/Applicable.

| Policy Definition Method | Automation | Policy L | апдиаде Туре | Separation | Analysis | Subject Selection | Language | Additional Ge | oals of Automation |
|-------------------------------|-----------------------------------|-------------|-----------------|---------------|---------------------|-------------------|----------|---------------|--------------------|
| Policy Definition Method | $\circ \bullet \bullet \bullet^I$ | Annotations | Placement Rules | Granularity | Approach | Subject Selection | Specific | Performance | Interface Safety |
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | • | • | 0 |
| CubicleOS* [211] | • | • | • | μ Library | Static ² | Code-centric | • | 0 | 0 |
| Google SAPI* [22] | O | • | • | Function | Static | Code-centric | • | 0 | 0 |
| FlexOS* [143, 161] | • | • | • | μ Library | Dynamic | Code-centric | • | • | 0 |
| RLBox* [181] | • | • | 0 | Function | Static | Any | • | 0 | • |
| SOAAP* [117] | • | • | 0 | Any | Hybrid | Any | • | • | • |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| Glamdring* [165] | • | • | 0 | Function | Static | Code-centric | • | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| Swift* [89] | • | • | 0 | Any | Static | Code-centric | • | 0 | • |
| Jif* [261] | • | • | • | Any | Static | Code-centric | • | 0 | • |
| PM [170] | • | • | • | Function | Hybrid | Code-centric | • | • | 0 |
| KSplit* [133] | • | • | • | Driver | Static | Code-centric | • | 0 | 0 |
| Cali* [65] | • | 0 | • | Library | Static | Code-centric | 0 | 0 | 0 |
| CompartOS* [55] | • | 0 | • | Linkage Unit | Static | Code-centric | 0 | 0 | 0 |
| Enclosure* [111] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| BreakApp* [235] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| CompARTist* [132] | • | 0 | • | Library | Static | Code-centric | • | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | Dynamic | Code-centric | • | 0 | 0 |
| μSCOPE [202], SCALPEL [203] | • | N/A | N/A | Any | Dynamic | Code-centric | 0 | • | 0 |

Names of the policy definition methods we consider



Characteristics we included in the taxonomy



| Dalian Dafritian Mathad | Automation | Policy L | anguage Type | Separation | Analysis | Cubiant Calantina | Language | Additional Ge | oals of Automation |
|----------------------------------|-----------------------------------|-------------|-----------------|---------------|---------------------|-------------------|----------|---------------|--------------------|
| Policy Definition Method | $\circ \circ \bullet \bullet^{I}$ | Annotations | Placement Rules | Granularity | Approach | Subject Selection | Specific | Performance | Interface Safety |
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | • | • | 0 |
| CubicleOS* [211] | • | • | • | μ Library | Static ² | Code-centric | • | 0 | 0 |
| Google SAPI* [22] | • | • | • | Function | Static | Code-centric | • | 0 | 0 |
| FlexOS* [143, 161] | • | • | • | μ Library | Dynamic | Code-centric | • | • | 0 |
| RLBox* [181] | • | • | 0 | Function | Static | Any | • | 0 | • |
| SOAAP* [117] | • | • | 0 | Any | Hybrid | Any | • | • | • |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| Glamdring* [165] | • | • | 0 | Function | Static | Code-centric | • | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| Swift* [89] | • | • | 0 | Any | Static | Code-centric | • | 0 | • |
| Jif* [261] | • | • | • | Any | Static | Code-centric | • | 0 | • |
| PM [170] | • | • | • | Function | Hybrid | Code-centric | • | • | 0 |
| KSplit* [133] | • | • | • | Driver | Static | Code-centric | • | 0 | 0 |
| Cali* [65] | • | 0 | • | Library | Static | Code-centric | 0 | 0 | 0 |
| CompartOS* [55] | • | 0 | • | Linkage Unit | Static | Code-centric | 0 | 0 | 0 |
| Enclosure* [111] | • | 0 | • | Package | Static | Code-centric | | 0 | 0 |
| BreakApp* [235] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| CompARTist* [132] | • | 0 | • | Library | Static | Code-centric | • | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | Dynamic | Code-centric | • | 0 | 0 |
| μ SCOPE [202], SCALPEL [203] | • | N/A | N/A | Any | Dynamic | Code-centric | 0 | • | 0 |

 $^{^{1}}$ O = manual, O = guided manual, \odot = policy refinement, \odot = full automation. 2 Loader-based. 3 Implemented with static analysis, dynamic analysis possible [90].

Names of the policy definition methods we consider



Characteristics we included in the taxonomy

| Policy Definition Method | Automation $\circ \bullet \bullet^{l}$ | Policy L Annotations | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go | pals of Automation Interface Safety |
|----------------------------------|--|-------------------------|--------------------------------|---------------------------|----------------------|-------------------|----------------------|---------------|-------------------------------------|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowdai [70] | O | 0 | 0 | Function | Dynamic | Code-centric | 0 | | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | • | • | 0 |
| CubicleOS* [211] | • | • | • | μ Library | Static ² | Code-centric | • | 0 | 0 |
| Google SAPI* [22] | • | • | • | Function | Static | Code-centric | • | 0 | 0 |
| FlexOS* [143, 161] | • | • | • | μ Library | Dynamic | Code-centric | • | • | 0 |
| RLBox* [181] | • | • | 0 | Function | Static | Any | • | 0 | • |
| SOAAP* [117] | O | • | 0 | Any | Hybrid | Any | • | • | • |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| Glamdring* [165] | • | • | 0 | Function | Static | Code-centric | • | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| Swift* [89] | • | • | 0 | Any | Static | Code-centric | • | 0 | • |
| Jif* [261] | • | • | • | Any | Static | Code-centric | • | 0 | • |
| PM [170] | • | • | • | Function | Hybrid | Code-centric | • | • | 0 |
| KSplit* [133] | • | • | • | Driver | Static | Code-centric | • | 0 | 0 |
| Cali* [65] | • | 0 | • | Library | Static | Code-centric | 0 | 0 | 0 |
| CompartOS* [55] | • | 0 | • | Linkage Unit | Static | Code-centric | 0 | 0 | 0 |
| Enclosure* [111] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| BreakApp* [235] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| CompARTist* [132] | • | 0 | • | Library | Static | Code-centric | • | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | Dynamic | Code-centric | • | 0 | 0 |
| μ SCOPE [202], SCALPEL [203] | • | N/A | N/A | Any | Dynamic | Code-centric | 0 | • | 0 |



Description of the state of the state analysis, dynamic analysis possible [90].

Description of the state analysis, dynamic analysis possible [90].

Names of the policy definition methods we consider



Characteristics we included in the taxonomy

| Policy Definition Method | Automation | | anguage Type | Separation | Analysis | Subject Selection | Language | Additional Go | oals of Automation |
|----------------------------------|------------|-------------|-----------------|---------------|---------------------|-------------------|----------|---------------|--------------------|
| Toticy Definition Method | | Annotations | Placement Rules | Granularity | Approach | Subject Selection | Specific | Performance | Interface Safety |
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowdar [70] | O | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | • | • | 0 |
| CubicleOS* [211] | • | • | • | μ Library | Static ² | Code-centric | • | 0 | 0 |
| Google SAPI* [22] | • | • | • | Function | Static | Code-centric | • | 0 | 0 |
| FlexOS* [143, 161] | • | • | • | μ Library | Dynamic | Code-centric | • | • | 0 |
| RLBox* [181] | • | • | 0 | Function | Static | Any | • | 0 | • |
| SOAAP* [117] | • | • | 0 | Any | Hybrid | Any | • | • | • |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| Glamdring* [165] | • | • | 0 | Function | Static | Code-centric | • | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| Swift* [89] | • | • | 0 | Any | Static | Code-centric | • | 0 | • |
| Jif* [261] | • | • | • | Any | Static | Code-centric | • | 0 | • |
| PM [170] | • | • | • | Function | Hybrid | Code-centric | • | • | 0 |
| KSplit* [133] | • | • | • | Driver | Static | Code-centric | • | 0 | 0 |
| Cali* [65] | • | 0 | • | Library | Static | Code-centric | 0 | 0 | 0 |
| CompartOS* [55] | • | 0 | • | Linkage Unit | Static | Code-centric | 0 | | 0 |
| Enclosure* [111] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| BreakApp* [235] | • | 0 | • | Package | Static | Code-centric | • | 0 | 0 |
| CompARTist* [132] | • | 0 | • | Library | Static | Code-centric | • | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | Dynamic | Code-centric | • | 0 | 0 |
| μ SCOPE [202], SCALPEL [203] | • | N/A | N/A | Any | Dynamic | Code-centric | 0 | • | 0 |

¹ O = manual, O = guided manual, O = policy refinement, O = full automation. ² Loader-based. ³ Implemented with static analysis, dynamic analysis possible [90].



Entirely manual

| Policy Definition Method | Automation $\bigcirc \bigcirc \bigcirc \bigcirc \bigcirc $ | Policy L Annotations | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go Performance | oals of Automation Interface Safety |
|----------------------------------|--|-------------------------|------------------------------|---------------------------|----------------------|-------------------|----------------------|------------------------------|--|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | O | 0 | 0 | Function | Dynamic | Code-centric | 0 | | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | | | 0 |
| CubicleOS* [211] | • | | | μ Library | Static ² | Code-centric | | 0 | |
| Google SAPI* [22] | • | | | Function | Static | Code-centric | | 0 | |
| FlexOS* [143, 161] | • | | | μ Library | Dynamic | Code-centric | | • | |
| RLBox* [181] | • | | | Function | Static | Any | | 0 | • |
| SOAAP* [117] | • | | | Any | Hybrid | Any | | | |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | | | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | | 0 | 0 |
| PrivTrans* [73] | • | • | | Function | Hybrid | Code-centric | | 0 | |
| Glamdring* [165] | • | | | Function | Static | Code-centric | | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | | | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | | | Any | Static | Code-centric | 0 | 0 | |
| Swift* [89] | • | | | Any | Static | Code-centric | | 0 | |
| Jif* [261] | • | | | Any | Static | Code-centric | | | • |
| PM [170] | • | | | Function | Hybrid | Code-centric | | | |
| KSplit* [133] | • | • | | Driver | Static | Code-centric | | 0 | |
| Cali* [65] | • | 0 | | Library | Static | Code-centric | 0 | 0 | |
| CompartOS* [55] | • | 0 | | Linkage Unit | Static | Code-centric | 0 | | |
| Enclosure* [111] | • | 0 | | Package | Static | Code-centric | | 0 | |
| BreakApp* [235] | • | 0 | | Package | Static | Code-centric | | 0 | |
| CompARTist* [132] | • | 0 | | Library | Static | Code-centric | | 0 | |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | Dynamic | Code-centric | | 0 | 0 |
| μ SCOPE [202], SCALPEL [203] | • | N/A | N/A | Any | Dynamic | Code-centric | 0 | | 0 |

Fully automated

¹ O = manual, O = guided manual, O = policy refinement, O = full automation. ² Loader-based. ³ Implemented with static analysis, dynamic analysis possible [90].

Entirely manual

| Policy Definition Method | Automation $\bigcirc \bigcirc \bigcirc \bigcirc \bullet^I$ | | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go | pals of Automation Interface Safety |
|-------------------------------|--|---------|---------------------------------|---------------------------|----------------------|-------------------|----------------------|---------------|-------------------------------------|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | | | 0 |
| CubicleOS* [211] | • | • | • | μ Library | Static ² | Code-centric | | 0 | 0 |
| Google SAPI* [22] | • | • | • | Function | Static | Code-centric | | 0 | 0 |
| FlexOS* [143, 161] | • | • | • | μ Library | Dynamic | Code-centric | | | |
| RLBox* [181] | • | • | 0 | Function | Static | Any | | 0 | • |
| SOAAP* [117] | O | • | 0 | Any | Hybrid | Any | | | |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | | | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | | 0 | 0 |
| PrivTrans* [73] | • | • | 0 | Function | Hybrid | Code-centric | | 0 | 0 |
| Glamdring* [165] | • | • | 0 | Function | Static | Code-centric | | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | • | 0 | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | • | 0 | Any | Static | Code-centric | | | • |
| Swift* [89] | • | • | 0 | Any | Static | Code-centric | | 0 | |
| Jif* [261] | • | • | • | Any | Static | Code-centric | | | |
| PM [170] | • | • | • | Function | Hybrid | Code-centric | | | 0 |
| KSplit* [133] | • | • | • | Driver | Static | Code-centric | | 0 | 0 |
| Cali* [65] | • | 0 | • | Library | Static | Code-centric | 0 | 0 | 0 |
| CompartOS* [55] | • | 0 | • | Linkage Unit | Static | Code-centric | 0 | 0 | 0 |
| Enclosure* [111] | • | 0 | • | Package | Static | Code-centric | | 0 | 0 |
| BreakApp* [235] | • | 0 | • | Package | Static | Code-centric | | 0 | 0 |
| CompARTist* [132] | • | 0 | • | Library | Static | Code-centric | | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | NIL | NT/A | Eunation | Dynamic | Code-centric | | 0 | 0 |
| μSCOPE [202], SCALPEL [203] | • | They ac | hieve auto | mation | ynamic | Code-centric | 0 | | 0 |

Fully automated

They do

leveraging different kinds of inputs from developers.

er-based. ³ Implemented with static analysis, dynamic analysis possible [90].

Entirely manual

| Policy Definition Method | Automation | Policy L | anguage Type Placement Rules | Separation Granularity | Analysis | Subject Selection | Language | Additional Go | pals of Automation |
|----------------------------------|------------|----------|------------------------------|---------------------------|---------------------|-------------------|----------|---------------|--------------------|
| | | 1 | | Granularity | Approach | | Specific | J | Interface Safety |
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | |
| MPDs* [191, 192] | • | \circ | | Component | Hybrid | Code-centric | | | |
| CubicleOS* [211] | • | | | μ Library | Static ² | Code-centric | | 0 | |
| Google SAPI* [22] | • | | | Function | Static | Code-centric | | | |
| FlexOS* [143, 161] | • | | | μ Library | Dynamic | Code-centric | | | |
| RLBox* [181] | • | | | Function | Static | Any | | | |
| SOAAP* [117] | • | | 0 | Any | Hybrid | Any | | | |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | | 0 | 0 |
| PtrSplit* [169] | • | | | Function | Static | Code-centric | | 0 | |
| PrivTrans* [73] | • | | | Function | Hybrid | Code-centric | | 0 | |
| Glamdring* [165] | • | | | Function | Static | Code-centric | | 0 | |
| Shreds* [87], CAPACITY* [105] | • | | | Any | Static | Code-centric | 0 | 0 | |
| DataShield* [77] | • | | | Any | Static | Code-centric | 0 | 0 | |
| Swift* [89] | • | | | Any | Static | Code-centric | | 0 | |
| Jif* [261] | • | | | Any | Static | Code-centric | | | |
| PM [170] | • | | | Function | Hybrid | Code-centric | | | |
| KSplit* [133] | • | | | Driver | Static | Code-centric | | 0 | 0 |
| Cali* [65] | • | 0 | | Library | Static | Code-centric | 0 | 0 | |
| CompartOS* [55] | • | 0 | | Linkage Unit | Static | Code-centric | 0 | 0 | |
| Enclosure* [111] | • | 0 | | Package | Static | Code-centric | | 0 | |
| BreakApp* [235] | • | 0 | • | Package | Static | Code-centric | | 0 | 0 |
| CompARTist* [132] | • | 0 | | Library | Static | Code-centric | | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | D | Code-centric | | | 0 |
| μ SCOPE [202], SCALPEL [203] | • | N/A | N/A | A The | | 1.00 | | | |

Fully automated

O = M Sanual, O = M Squided manual, O = M Squid

They employ different kinds of analysis techniques (static, dynamic, hybrid)

dynamic analysis possible [90].

Specializing on a particular programming language is often necessary to simplify the problem

Entirely manual

| Policy Definition Method | Automation $\circ \circ \bullet \bullet^{1}$ | | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go Performance | oals of Automation Interface Safety |
|----------------------------------|--|-----|---------------------------------|---------------------------|----------------------|-------------------|----------------------|------------------------------|-------------------------------------|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | 0 | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | 0 |
| MPDs* [191, 192] | • | 0 | | Component | Hybrid | Code-centric | • | • | |
| CubicleOS* [211] | • | | | μ Library | Static ² | Code-centric | • | 0 | |
| Google SAPI* [22] | • | | | Function | Static | Code-centric | • | 0 | |
| FlexOS* [143, 161] | • | | | μ Library | Dynamic | Code-centric | • | | |
| RLBox* [181] | • | | | Function | Static | Any | • | 0 | |
| SOAAP* [117] | • | • | 0 | Any | Hybrid | Any | • | | |
| SeCage* [171] | • | • | 0 | Function | Hybrid | Code-centric | • | 0 | 0 |
| PtrSplit* [169] | • | • | | Function | Static | Code-centric | • | 0 | |
| PrivTrans* [73] | • | | | Function | Hybrid | Code-centric | • | 0 | |
| Glamdring* [165] | • | | | Function | Static | Code-centric | • | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | | | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | • | | | Any | Static | Code-centric | 0 | 0 | • |
| Swift* [89] | • | • | | Any | Static | Code-centric | • | 0 | |
| Jif* [261] | • | | | Any | Static | Code-centric | • | | |
| PM [170] | • | | | Function | Hybrid | Code-centric | • | | |
| KSplit* [133] | • | | • | Driver | Static | Code-centric | • | 0 | 0 |
| Cali* [65] | • | 0 | | Library | Static | Code-centric | 0 | 0 | 0 |
| CompartOS* [55] | • | 0 | | Linkage Unit | Static | Code-centric | 0 | 0 | |
| Enclosure* [111] | • | 0 | | Package | Static | Code-centric | • | 0 | |
| BreakApp* [235] | • | 0 | | Package | Static | Code-centric | • | 0 | 0 |
| CompARTist* [132] | • | 0 | • | Library | Static | Code-centric | • | 0 | 0 |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | • | N/A | N/A | Function | Dynamic | Code-centric | • | 0 | 0 |
| μ SCOPE [202], SCALPEL [203] | • | N/A | N/A | Any | Dynamic | Code-centric | 0 | | 0 |

Fully automated

¹ O = manual, O = guided manual, O = policy refinement, O = full automation. ² Loader-based. ³ Implemented with static analysis, dynamic analysis possible [90].

| Policy Definition Method | Automation | Policy L | anguage Type | Separation | Analysis | Subject Selection | Language | Additional Go | oals of Automation |
|----------------------------------|---------------------------------|-------------|-----------------|---------------|---------------------|-------------------|----------|---------------|--------------------|
| Toncy Definition Memod | $\circ \circ \bullet \bullet^I$ | Annotations | Placement Rules | Granularity | Approach | Subject Selection | Specific | Performance | Interface Safety |
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | | 0 |
| MPDs* [191, 192] | • | 0 | | Component | Hybrid | Code-centric | | • | |
| CubicleOS* [211] | • | | | μ Library | Static ² | Code-centric | | 0 | |
| Google SAPI* [22] | • | | | Function | Static | Code-centric | | 0 | |
| FlexOS* [143, 161] | • | | | μ Library | Dynamic | Code-centric | | | |
| RLBox* [181] | • | | | Function | Static | Any | | 0 | |
| SOAAP* [117] | • | | | Any | Hybrid | Any | | | |
| SeCage* [171] | | | 0 | Function | Hybrid | Code-centric | | | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | | | 0 | Function | Hybrid | Code-centric | | 0 | 0 |
| Glamdring* [165] | • | | | Function | Static | Code-centric | | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | | | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* [77] | | | | Any | Static | Code-centric | 0 | 0 | |
| Swift* [89] | | | | Any | Static | Code-centric | | | |
| Jif* [261] | | | | Any | Static | Code-centric | | | |
| PM [170] | • | | | Function | Hybrid | Code-centric | | | |
| KSplit* [133] | | | | Driver | Static | Code-centric | | 0 | |
| Cali* [65] | | 0 | | Library | Static | Code-centric | 0 | | |
| CompartOS* [55] | | 0 | | Linkage Unit | Static | Code-centric | 0 | | |
| Enclosure* [111] | | 0 | | Package | Static | Code-centric | | | |
| BreakApp* [235] | | 0 | | Package | Static | Code-centric | | | |
| CompARTist* [132] | | 0 | | Library | Static | Code-centric | | | |
| ACES* [90] | • | 0 | | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | | N/A | N/A | Function | Dynamic | Code-centric | | | 0 |
| μ SCOPE [202], SCALPEL [203] | | N/A | N/A | Any | Dynamic | Code-centric | 0 | | |

¹ ○ = manual, ○ = guided manual, ○ = policy refinement, ○ = full automation. ² Loader-based. ³ Implemented with static analysis, dynamic analysis possible [90].

| Policy Definition Method | Automation | | anguage Type | Separation | Analysis | Subject Selection | Language | | oals of Automation |
|----------------------------------|-----------------------------------|-------------|-----------------|---------------|---------------------|-------------------|----------|-------------|--------------------|
| Tottey Definition Memod | $\circ \bullet \bullet \bullet^I$ | Annotations | Placement Rules | Granularity | Approach | Subject Selection | Specific | Performance | Interface Safety |
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | 0 |
| MPDs* [191, 192] | • | | | Component | Hybrid | Code-centric | | • | |
| CubicleOS* [211] | • | | | μ Library | Static ² | Code-centric | | 0 | |
| Google SAPI* [22] | • | | • | Function | Static | Code-centric | | 0 | |
| FlexOS* [143, 161] | emi-aut | omatad | +00 | μ Library | Dynamic | Code-centric | | | |
| RLBox* [181] | emi-aut | omateu | 1001 | Function | Static | Any | | 0 | • |
| SOAAP* [117] | or safek | oxina co | ode | Any | Hybrid | Any | | | • |
| SeCage* [171] | | | | Function | Hybrid | Code-centric | | | 0 |
| PtrSplit* [169] | 0 | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | | | 0 | Function | Hybrid | Code-centric | | 0 | 0 |
| Glamdring* [165] | | | | Function | Static | Code-centric | | 0 | • |
| Shreds* [87], CAPACITY* [105] | • | | | Any | Static | Code-centric | | | • |
| DataShield* [77] | | | | Any | Static | Code-centric | | 0 | • |
| Swift* [89] | | | | Any | Static | Code-centric | | 0 | • |
| Jif* [261] | • | | | Any | Static | Code-centric | | 0 | • |
| PM [170] | • | | | Function | Hybrid | Code-centric | | | |
| KSplit* [133] | | | | Driver | Static | Code-centric | | | |
| Cali* [65] | | | | Library | Static | Code-centric | | 0 | |
| CompartOS* [55] | | | | Linkage Unit | Static | Code-centric | | | |
| Enclosure* [111] | | | | Package | Static | Code-centric | | 0 | |
| BreakApp* [235] | | 0 | | Package | Static | Code-centric | | | |
| CompARTist* [132] | | | | Library | Static | Code-centric | | 0 | |
| ACES* [90] | • | 0 | | Function | Any ³ | Code-centric | 0 | | 0 |
| ProgramCutter [253] | | N/A | N/A | Function | Dynamic | Code-centric | | | 0 |
| μ SCOPE [202], SCALPEL [203] | | N/A | N/A | Any | Dynamic | Code-centric | 0 | | 0 |

 $^{^{1}}$ \bigcirc = manual, \bigcirc = guided manual, \bigcirc = policy refinement, \bigcirc = full automation. 2 Loader-based. 3 Implemented with static analysis, dynamic analysis possible [90].

| Policy Definition Method | Automation $\bigcirc \bigcirc \bigcirc \bigcirc \bullet^I$ | Policy L. Annotations | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go Performance | pals of Automation Interface Safety |
|-----------------------------|--|-----------------------|------------------------------|---------------------------|----------------------|-------------------|----------------------|------------------------------|-------------------------------------|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | | N/A | N/A |
| Crowbar [70] | • | | 0 | Function | Dynamic | Code-centric | 0 | | |
| MPDs* [191, 192] | 0 | 0 | 0 | Component | Hybrid | Code-centric | | | 0 |
| CubicleOS* [211] | • | • | | μ Library | Static ² | Code-centric | | 0 | 0 |
| Google SAPI* [22] | • | | • | Function | Static | Code-centric | | 0 | 0 |
| FlexOS* [143, 161] | | t - d | + | μ Library | Dynamic | Code-centric | | | 0 |
| RLBox* [181] | emi-aut | omated | 1001 | Function | Static | Any | | 0 | • |
| SOAAP* [117] | or safek | oxing co | ode | Any | Hybrid | Any | • | | • |
| SeCage* [171] | | | | Function | Hybrid | Code-centric | | | 0 |
| PtrSplit* [169] | | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | • | | 0 | Function | Hybrid | Code-centric | | 0 | 0 |
| Glamdring* [Apporto | to value | ble data | | Function | Static | Code-centric | | 0 | • |
| Shreds* [87], Allilott | ile valua | ible date | a , | Any | Static | Code-centric | | 0 | • |
| DataShield* PtrSplit | then a | utomati | cally | Any | Static | Code-centric | | 0 | • |
| Swift* [89] | | | | Any | Static | Code-centric | | 0 | • |
| Jif* [261] cuts the | e progra | am to | • | Any | Static | Code-centric | | 0 | • |
| PM [170] safebox | , thom | | | Function | Hybrid | Code-centric | | | |
| Kopiit [155] | k tilelli | | | Driver | Static | Code-centric | | | |
| Cali* [65] | | 0 | | Library | Static | Code-centric | | | |
| CompartOS* [55] | | 0 | | Linkage Unit | Static | Code-centric | | | |
| Enclosure* [111] | | 0 | | Package | Static | Code-centric | | | |
| BreakApp* [235] | • | 0 | | Package | Static | Code-centric | | 0 | |
| CompARTist* [132] | | 0 | | Library | Static | Code-centric | | 0 | |
| ACES* [90] | • | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | | N/A | N/A | Function | Dynamic | Code-centric | | 0 | 0 |
| μSCOPE [202], SCALPEL [203] | | N/A | N/A | Any | Dynamic | Code-centric | 0 | | 0 |

implemented with static analysis, dynamic analysis possible [50]

| Policy Definition Method | Automation $\bigcirc \bigcirc \bigcirc \bigcirc \bullet^{I}$ | Policy L. Annotations | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go Performance | pals of Automation Interface Safety |
|-----------------------------|--|-----------------------|------------------------------|---------------------------|----------------------|-------------------|----------------------|------------------------------|-------------------------------------|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | | | | Function | Dynamic | Code-centric | | | 0 |
| MPDs* [191, 192] | • | 0 | 0 | Component | Hybrid | Code-centric | | | 0 |
| CubicleOS* [211] | • | | | Α | | Sode-centric | | | 0 |
| Google SAPI* [22] | • | | T | he progra | ım is the | en ode-centric | | 0 | 0 |
| Floy 0 C* 1142 1611 | | | | | | | | • | 0 |
| RLBox* [181] | emi-aut | omated | tool | split at a | rbitrary | Any | | | • |
| SOAAP* [117] | or safek | oxing co | ode fu | inction b | oundarie | es Any | | | • |
| SeCage* [171] | | Toxing of | | ron | Hybrid | Code-centric | | | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | • | | | Function | Hybrid | Code-centric | | 0 | 0 |
| Glamdring* (Apport | yta valus | blo date | | Function | Static | Code-centric | | 0 | • |
| Shreds* [87], Annous | ile valud | able data | a, | Any | Static | Code-centric | 0 | 0 | • |
| DataShield* PtrSplit | t then a | utomati | cally | Any | Static | Code-centric | 0 | 0 | • |
| Swift* [89] | | | | Any | Static | Code-centric | | 0 | • |
| Jif* [261] cuts th | e progra | am to | | Any | Static | Code-centric | | 0 | • |
| PM [170] safebox | y thom | | | Function | Hybrid | Code-centric | | | |
| Kopiit [155] | x them | | | Driver | Static | Code-centric | | | |
| Cali* [65] | | 0 | | Library | Static | Code-centric | 0 | 0 | |
| CompartOS* [55] | | 0 | | Linkage Unit | Static | Code-centric | 0 | | |
| Enclosure* [111] | | 0 | | Package | Static | Code-centric | | | |
| BreakApp* [235] | | 0 | | Package | Static | Code-centric | | 0 | |
| CompARTist* [132] | | 0 | | Library | Static | Code-centric | | 0 | |
| ACES* [90] | | 0 | • | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | | N/A | N/A | Function | Dynamic | Code-centric | | | 0 |
| μSCOPE [202], SCALPEL [203] | | N/A | N/A | Any | Dynamic | Code-centric | 0 | | 0 |

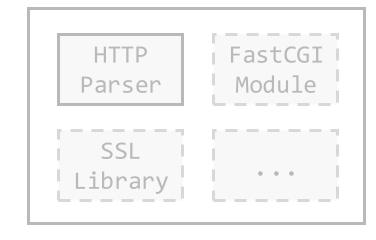
⁹⁹

| Policy Definition Method | Automation $\bigcirc \bigcirc \bigcirc \bigcirc \bullet^{I}$ | Policy Lands | anguage Type Placement Rules | Separation Granularity | Analysis Approach | Subject Selection | Language Specific | Additional Go Performance | oals of Automation Interface Safety |
|--|--|--------------|---------------------------------|------------------------------------|----------------------|-------------------|----------------------|------------------------------|-------------------------------------|
| Manual [128], [] | 0 | N/A | N/A | Any | Manual | Any | 0 | N/A | N/A |
| Crowbar [70] | • | 0 | 0 | Function | Dynamic | Code-centric | 0 | 0 | 0 |
| MPDs* [191, 192] | • | 0 | | Component | Hybrid | Code-centric | | | |
| CubicleOS* [211] | • | | • (] | | : | Sode-centric | | | |
| Google SAPI* [22] | G | | • 11 | ne progra | im is the | en pode-centric | | 0 | |
| FlexOS* [143, 161] | omi aut | omated | tool | split at a | rhitrary | ode-centric | | | |
| KLD0X [181] | emi-aut | omateu | | • | • | Any | | 0 | |
| SOAAP* [117] | or safek | oxing co | ode $igsqcup f\iota$ | inction be | oundarie | S Any | | | • |
| SeCage* [171] | | - <u> </u> | | aon | Hybrid | Code-centric | | 0 | 0 |
| PtrSplit* [169] | • | • | 0 | Function | Static | Code-centric | • | 0 | 0 |
| PrivTrans* [73] | | | 0 | Function | | Code-centric | | 0 | 0 |
| Glamdring* Annotate valuable data, The split is done statically by | | | | | | | 0 | | |
| Shieus [87], | | | | The split is dolle statically by | | | | 0 | |
| DataShield* PtrSplit then automatically | | | | the tool, i.e., just by looking at | | | | | |
| Swift* [89] | | | | the code (vs. a dynamic tool | | | | 0 | • |
| Jif* [261] cuts the program to | | | | tne c | ode (vs. | a dynamic | tooi | 0 | • |
| PM [170] KSplit* (122) safebox them | | | | that would run the program) | | | | | 0 |
| Kopiit [155] | X tileiii | | | triat | would it | an the prog | <u> </u> | | <u> </u> |
| Cali* [65] | 0 | 0 | • | Library | Static | Code-centric | 0 | 0 | <u> </u> |
| CompartOS* [55] | 0 | 0 | | Linkage Unit | Static | Code-centric | 0 | 0 | <u> </u> |
| Enclosure* [111] | 0 | 0 | | Package | Static | Code-centric | | 0 | 0 |
| BreakApp* [235] | 0 | 0 | | Package | Static | Code-centric | | 0 | 0 |
| CompARTist* [132] | 0 | 0 | | Library | Static | Code-centric | | 0 | 0 |
| ACES* [90] | | 0 | | Function | Any ³ | Code-centric | 0 | 0 | 0 |
| ProgramCutter [253] | | N/A | N/A | Function | Dynamic | Code-centric | | 0 | 0 |
| μ SCOPE [202], SCALPEL [203] | | N/A | N/A | Any | Dynamic | Code-centric | 0 | | |

Automated Policy Definition Methods trade off security and/or performance for developer effort.

Take our example: *our split cuts a hot path*. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

Automated methods struggle to do that since the semantics of programs fundamentally cannot be captured automatically.



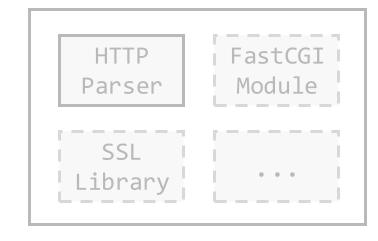
(Fictive monolithic program)



Automated Policy Definition Methods trade off security and/or performance for developer effort.

Take our example: *our split cuts a hot path*. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

Automated methods struggle to do that since the semantics of programs fundamentally cannot be captured automatically.



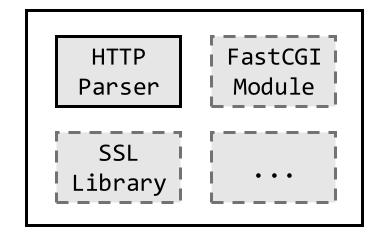
(Fictive monolithic program)



Automated Policy Definition Methods trade off security and/or performance for developer effort.

Take our example: *our split cuts a hot path*. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

Automated methods struggle to do that since the semantics of programs fundamentally cannot be captured automatically.



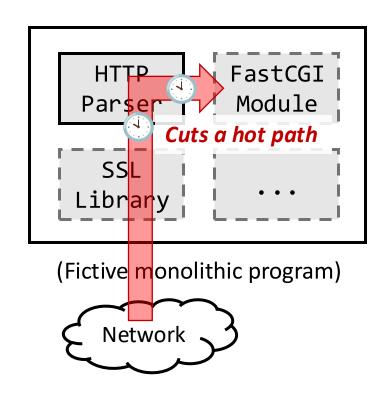
(Fictive monolithic program)



Automated Policy Definition Methods trade off security and/or performance for developer effort.

Take our example: *our split cuts a hot path*. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

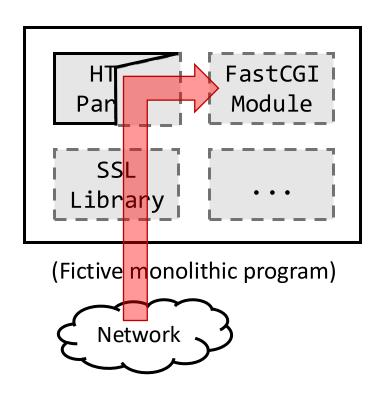
Automated methods struggle to do that since the semantics of programs fundamentally cannot be captured automatically.



Automated Policy Definition Methods trade off security and/or performance for developer effort.

Take our example: our split cuts a hot path. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

Automated methods struggle to do that since the semantics of programs fundamentally cannot be captured automatically.



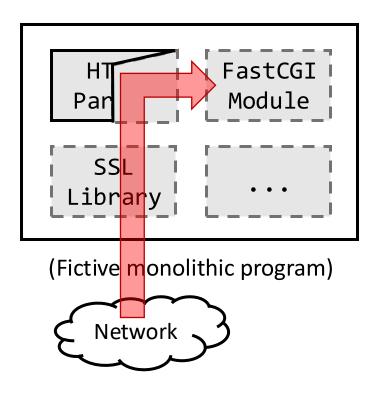
Automated Policy Definition Methods trade off security and/or performance for developer effort.

Take our example: our split cuts a hot path. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

Automated methods struggle to do that since the semantics of programs fundamentally cannot be captured automatically.

Rice's theorem: "non-trivial serior programs for the serior programs fundamentally cannot be captured automatically.

Rice's theorem: "non-trivial semantic properties of programs are undecidable".



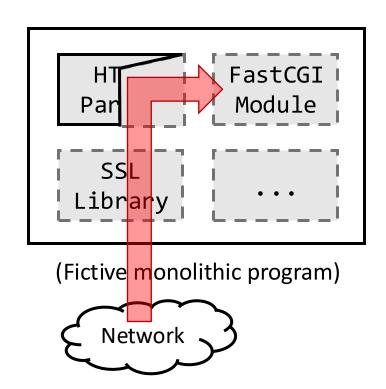
Automated Policy Definition Methods trade off security and/or performance for developer effort.

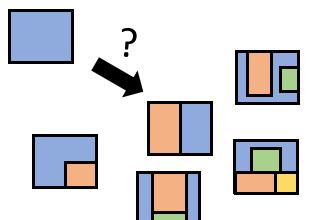
Take our example: our split cuts a hot path. We can spend more developer time to look attentively and refine our cut to avoid the critical code paths.

Automated methods struggle to do that since the semantics of programs fundamentally cannot be

captured automatically.

Rice's theorem: "non-trivial semantic properties of programs are undecidable".





- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction
- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism



How to implement policies?

Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

How to implement policies?

Having defined a policy, we need to express it in the program.

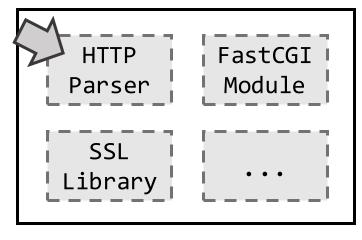
Developers perform this using programming abstractions.

Historically, people have done this with processes (=the process abstraction).

How to implement policies?

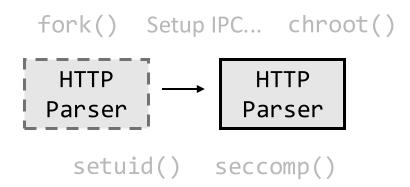
Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

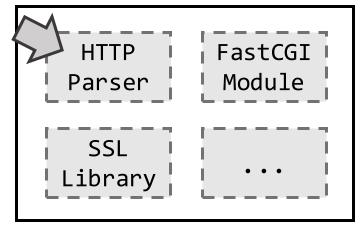
Example: assume we sandbox the HTTP parser.



How to implement policies?

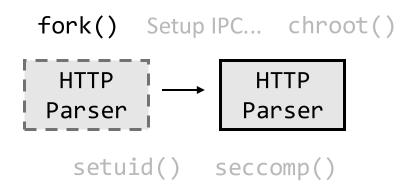
Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

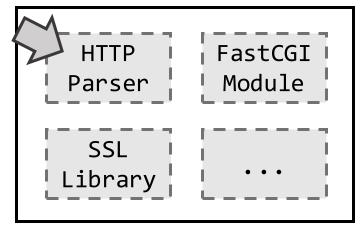
Example: assume we sandbox the HTTP parser.



How to implement policies?

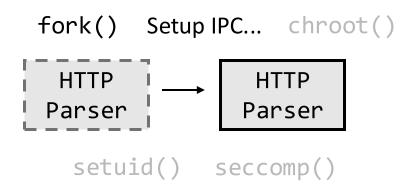
Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

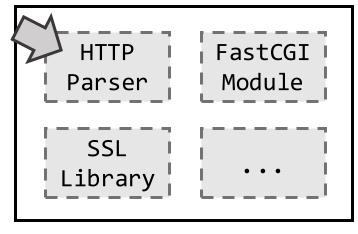
Example: assume we sandbox the HTTP parser.



How to implement policies?

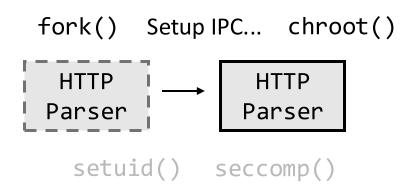
Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

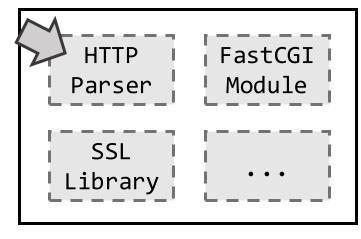
Example: assume we sandbox the HTTP parser.



How to implement policies?

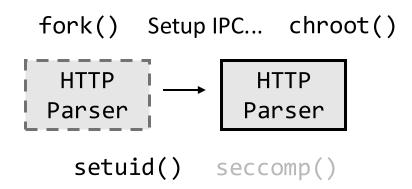
Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

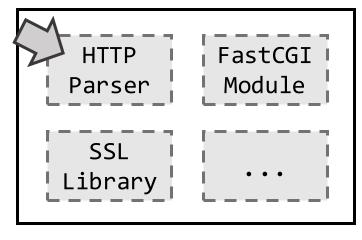
Example: assume we sandbox the HTTP parser.



How to implement policies?

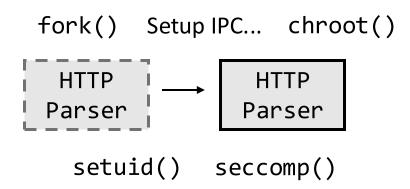
Having defined a policy, we need to **express it in the program**. Developers perform this using **programming abstractions**.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

Example: assume we sandbox the HTTP parser.

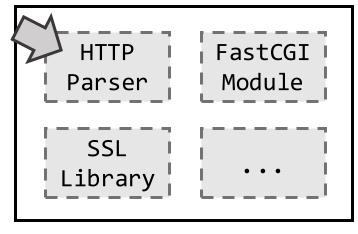


How to implement policies?

Having defined a policy, we need to express it in the program.

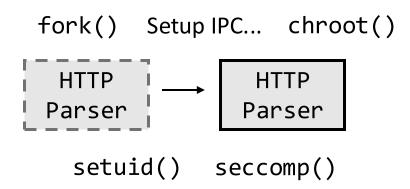
Developers perform this using programming abstractions.

Historically, people have done this with processes (=the process abstraction).



(Fictive monolithic program)

Example: assume we sandbox the HTTP parser.

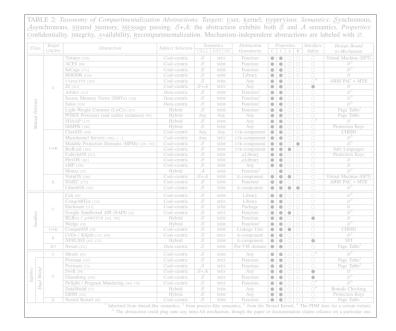


Compartmentalization Abstractions

Programming abstractions for compartmentalization are also a very active research area:

Minimize developer effort (intuitive, easy to use)
Leverage domain-specific knowledge (threat model, deployment)
Maximize security properties that can be achieved

Making the most of a specific enforcement mechanism
Support generic mechanisms





(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

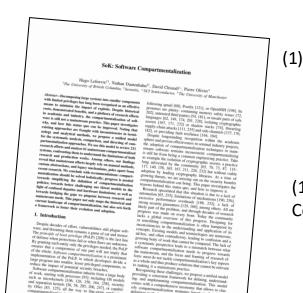
Compartmentalization Abstractions

Programming abstractions for compartmentalization are also a very active research area:

Minimize developer effort (intuitive, easy to use)
Leverage domain-specific knowledge (threat model, deployment)
Maximize security properties that can be achieved

Making the most of a specific enforcement mechanism
Support generic mechanisms

| | Target | | | I Sam | antics | Abstraction | | Prope | neti o | | Interface | Design Bound |
|-----------------|--|--|-------------------|-------|--------|-----------------------|---|-------|--------|---|----------------|-------------------------|
| Class | U/K/HV | Abstraction | Subject Selection | | ASSIGN | Granularity | | I | | | Safety | to Mechanism |
| | Ť | Virtines [242] | Code-centric | S | MES | Function | • | • | 0 | 0 | 0 | Virtual Machine (El |
| | | ACES [90] | Code-centric | S | SHM | Function | • | • | 0 | 0 | 0 | ø, |
| | | SeCage [171] | Code-centric | S | SHM | Function | • | • | 0 | 0 | 0 | ø, |
| | | HODOR [124] | Code-centric | S | SHM | Library | • | • | 0 | 0 | 0 | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | • | • | 0 | 0 | O ⁴ | ARM PAC + MT |
| | U | Jif [261] | Code-centric | S+A | MES | Any | • | • | 0 | 0 | • | Ø |
| | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | • | • | | 0 | 0 | a, |
| 2- | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | • | • | | 0 | 0 | øs. |
| Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | • | • | 0 | 0 | 0 | ø, |
| isi(| | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | • | • | 0 | 0 | 0 | Page Table ² |
| 7 | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | • | • | 0 | 0 | 0 | Page Table |
| Tag | | SOAAP [117] | Hybrid | 8 | SHM | Any | • | • | 0 | 0 | 0* | Ø |
| Mu | | libMPK [190] | Hybrid | S | SHM | Any | • | • | 0 | 0 | 0 | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | • | • | 0 | 0 | 0 | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | • | • | 0 | 0 | 0 | ø, |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | • | • | 0 | • | 0 | ø, |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | • | • | • | 0 | 0 | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μLibrary | • | • | 0 | 0 | 0 | Protection Keys |
| | | FlexOS [161] | Code-centric | S | SHM | μLibrary | | • | 0 | 0 | 0 | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | • | • | 0 | 0 | 0 | Ø |
| | | Monza [35] | Hybrid | A | SHM | Function ^a | 0 | • | | 0 | 0 | Ø ₂ |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | • | • | • | 0 | 0 | Virtual Machine (E |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | • | | 0 | 0 | ARM PAC + MT |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | • | • | • | • | 0 | ø, |
| | | Cali [65] | Code-centric | S | SHM | Library | • | • | | 0 | 0 | Ø ⁵ |
| | | CompARTist [132] | Code-centric | S | MES | Library | • | • | | 0 | 0 | ø, |
| | U | Enclosure [111] | Code-centric | S | SHM | Package | • | • | 0 | 0 | 0 | Ø |
| oχ | - | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | • | • | • | 0 | 0 | ø, |
| Sandbox | | RLBox / µSWITCH [181, 195] | Hybrid | S | SHM | Function | • | • | 0 | 0 | • | Ø |
| San | | Wedge [70] | Hybrid | S | SHM | Function ³ | | • | | 0 | 0 | ø, |
| | U+K | CompartOS [55] | Code-centric | S | SHM | Linkage Unit | • | • | ٠ | 0 | 0 | CHERI |
| | К | LVDs / KSplit [133, 185] | Code-centric | S | MES | K-component | | • | 0 | 0 | 0 | ø, |
| | | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | | • | 0 | 0 | • | SFI |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | • | • | | 0 | 0 | Page Table ³ |
| | U | Shreds [87] | Code-centric | S | SHM | Any | • | • | | 0 | 04 | ø ⁵ |
| | | Privman (147) | Code-centric | S | MES | Function | • | • | 0 | 0 | 0 | Page Table ² |
| | . | Privtrans [73] | Code-centric | S | MES | Function | • | • | | 0 | 0 | Page Table ² |
| Safebox | Ē. | Swift [89] | Code-centric | S+A | MES | Any | • | • | 0 | 0 | • | Ø |
| afe | Ĕ U | Glamdring [165] | Code-centric | S | MES | Function | | • | 0 | 0 | • | ø, |
| S | TO SECTION SEC | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | • | • | | 0 | 04 | Ø3 |
| 1 | 5 | DataShield [77] | Hybrid | S | SHM | Any | • | • | | 0 | O ⁴ | Bounds Checkin |
| | | ERIM [232] | Hybrid | S | SHM | Any | | • | 0 | 0 | 0 | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | 8 | SHM | Function | • | • | 0 | 0 | 0 | Page Table |



(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

| Class | Target | Abstraction | Subject Selection | I | iantics | Abstraction | P | rop | ertie | s | Interface | Design Bound |
|-----------------------|--------|--|-------------------|---------------|---------|-----------------------|---|--------------|-------|---|----------------|-------------------------|
| Citiss | U/K/HV | nostraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R | Safety | to Mechanism |
| | | Virtines [242] | Code-centric | S | MES | Function | • | • | 0 | 0 | 0 | Virtual Machine (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | • | • | 0 | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | • | • | 0 | 0 | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | \mathcal{S} | SHM | Library | • | • | 0 | 0 | 0 | Ø |
| | | CAPACITY [105] | Code-centric | \mathcal{S} | SHM | Any | | lacktriangle | 0 | 0 | O ⁴ | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | • | • | 0 | 0 | • | Ø |
| | | Arbiter [241] | Data-centric | \mathcal{S} | SHM | Function ¹ | | • | 0 | 0 | 0 | Ø ⁵ |
| st | | Secure Memory Views (SMVs) [128] | Data-centric | \mathcal{S} | SHM | Function ¹ | | lacktriangle | 0 | 0 | 0 | Ø ⁵ |
| Mutual Distrust | | Salus [226] | Data-centric | \mathcal{S} | SHM | Function ¹ | | • | 0 | 0 | 0 | Ø ⁵ |
| Disi | | Light-Weight Contexts (LwCs) [167] | Hybrid | \mathcal{S} | SHM | Function ¹ | • | • | 0 | 0 | 0 | Page Table ² |
| l la | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | • | • | 0 | 0 | 0 | Page Table |
| tuc | | SOAAP [117] | Hybrid | \mathcal{S} | SHM | Any | • | • | 0 | 0 | O ⁴ | Ø |
| M_{l} | | libMPK [190] | Hybrid | \mathcal{S} | SHM | Any | | • | 0 | 0 | 0 | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | • | • | 0 | 0 | 0 | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | • | 0 | 0 | 0 | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | \mathcal{S} | SHM | U/K-component | | lacktriangle | 0 | • | 0 | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | \mathcal{S} | SHM | U/K-component | | • | • | 0 | 0 | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | • | • | 0 | 0 | 0 | Protection Keys |
| Í I | | FlexOS [161] | Code-centric | \mathcal{S} | SHM | μ Library | | • | 0 | 0 | 0 | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | | • | 0 | 0 | 0 | Ø |
| Í I | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | 0 | • | 0 | 0 | 0 | Ø ⁵ |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | | • | • | 0 | 0 | Virtual Machine (EPT) |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | • | 0 | 0 | 0 | ARM PAC + MTE |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | | • | • | • | 0 | Ø ⁵ |
| | | Cali [65] | Code-centric | \mathcal{S} | SHM | Library | | lacktriangle | 0 | 0 | 0 | Ø ⁵ |
| | | CompARTist [132] | Code-centric | S | MES | Library | | • | 0 | 0 | 0 | Ø ⁵ |
| Í I | U | Enclosure [111] | Code-centric | \mathcal{S} | SHM | Package | • | • | 0 | 0 | 0 | Ø |
| ox | | Google Sandboxed API (SAPI) [22] | Code-centric | \mathcal{S} | MES | Function | • | lacktriangle | • | 0 | 0 | Ø ⁵ |
| Sandbox | | RLBox / μSWITCH [181, 195] | Hybrid | \mathcal{S} | SHM | Function | | • | 0 | 0 | • | Ø |
| Sar | | Wedge [70] | Hybrid | S | SHM | Function ¹ | | lacktriangle | 0 | 0 | 0 | Ø ⁵ |
| [| U+K | CompartOS [55] | Code-centric | \mathcal{S} | SHM | Linkage Unit | | • | • | 0 | 0 | CHERI |
| [| K | LVDs / KSplit [133, 185] | Code-centric | \mathcal{S} | MES | K-component | | lacktriangle | 0 | 0 | 0 | Ø ⁵ |
| 1 1 | K | XFI/LXFI [107, 172] | Hybrid | \mathcal{S} | SHM | K-component | | • | 0 | 0 | • | SFI |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | | • | 0 | 0 | 0 | Page Table ³ |
| | U | Shreds [87] | Code-centric | S | SHM | Any | • | • | 0 | 0 | \bigcirc^4 | Ø ⁵ |
| | | Privman [147] | Code-centric | S | MES | Function | • | • | 0 | 0 | 0 | Page Table ² |
| | | Privtrans [73] | Code-centric | \mathcal{S} | MES | Function | • | • | 0 | 0 | 0 | Page Table ² |
| box | | Swift [89] | Code-centric | S+A | MES | Any | • | • | 0 | 0 | • | Ø |
| Safebox al World | U | Glamdring [165] | Code-centric | S | MES | Function | • | • | 0 | 0 | • | Ø ⁵ |
| Safebox Dual World | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | • | • | Ō | 0 | O ⁴ | Ø ⁵ |
| $ D_{l} $ | | DataShield [77] | Hybrid | S | SHM | Any | • | • | Ō | Ō | 04 | Bounds Checking |
| | | ERIM [232] | Hybrid | S | SHM | Any | | • | Ŏ | Ŏ | Ö | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | S | SHM | Function | | | Ŏ | Ŏ | Ō | Page Table |

¹ Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent).

⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

Names of the abstractions we consider

Characteristics we included in the taxonomy

| Class | Target | Abatuastian | Cubicat Calactica | Sen | iantics | Abstraction | I | rop | ertie | s | Interface | Desig ad |
|-----------------------|--------|--|-------------------|---------------|---------|-----------------------|---|-----|-------|---|----------------|-------------------------|
| Class | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R | Safety | to sm |
| | | Virtines | Code-centric | S | MES | Function | | • | 0 | 0 | 0 | Virtual (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | | • | 0 | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | • | Ō | Ŏ | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | S | SHM | Library | Ŏ | • | Ŏ | Ŏ | Ö | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | • | Ō | Ō | 04 | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | | • | Ō | Ō | | Ø |
| | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | • | 0 | 0 | 0 | Ø ⁵ |
| 4 | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | | • | 0 | 0 | 0 | Ø ⁵ |
| Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | | • | 0 | 0 | 0 | Ø ⁵ |
| ist | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | | • | Ō | Ŏ | 0 | Page Table ² |
| 7 1 | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | Ŏ | • | Ŏ | Ŏ | Ö | Page Table |
| tua | | SOAAP [117] | Hybrid | S | SHM | Any | | • | 0 | Ō | O ⁴ | Ø |
| M_{U} | | libMPK [190] | Hybrid | S | SHM | Any | Ŏ | • | Ŏ | Ŏ | Ö | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | • | Ō | 0 | 0 | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | • | 0 | 0 | 0 | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | • | 0 | | 0 | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | | • | • | 0 | 0 | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | | • | 0 | 0 | 0 | Protection Keys |
| | | FlexOS [161] | Code-centric | S | SHM | μ Library | | • | 0 | 0 | 0 | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | | • | 0 | 0 | 0 | Ø |
| | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | 0 | • | 0 | 0 | 0 | \varnothing^5 |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | | • | | 0 | 0 | Virtual Machine (EPT) |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | • | 0 | 0 | 0 | ARM PAC + MTE |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | | • | | | 0 | Ø ⁵ |
| | | Cali [65] | Code-centric | S | SHM | Library | | • | 0 | | 0 | \varnothing^5 |
| | | CompARTist [132] | Code-centric | S | MES | Library | | • | 0 | 0 | 0 | Ø ⁵ |
| | U | Enclosure [111] | Code-centric | S | SHM | Package | | • | 0 | Ō | 0 | Ø |
| x | | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | | • | | 0 | 0 | Ø ⁵ |
| Sandbox | | RLBox / µSWITCH [181, 195] | Hybrid | S | SHM | Function | | • | Ō | Ŏ | • | Ø |
| an | | Wedge [70] | Hybrid | S | SHM | Function ¹ | | • | 0 | 0 | 0 | Ø ⁵ |
| \ \sigma_2 | U+K | CompartOS [55] | Code-centric | S | SHM | Linkage Unit | | • | • | Ō | Ō | CHERI |
| | | LVDs / KSplit [133, 185] | Code-centric | S | MES | K-component | | • | 0 | 0 | 0 | Ø ⁵ |
| | K | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | | • | 0 | 0 | • | SFI |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | | • | 0 | 0 | 0 | Page Table ³ |
| | U | Shreds [87] | Code-centric | S | SHM | Any | | • | 0 | | O ⁴ | Ø ⁵ |
| | | Privman [147] | Code-centric | S | MES | Function | Ŏ | • | 0 | 0 | 0 | Page Table ² |
| | | Privtrans [73] | Code-centric | S | MES | Function | | • | Ō | Ō | 0 | Page Table ² |
| ox rld | | Swift [89] | Code-centric | S+A | MES | Any | Ŏ | • | Ō | Ŏ | • | Ø |
| Safebox al World | U | Glamdring [165] | Code-centric | S | MES | Function | | • | Ŏ | Ŏ | • | Ø ⁵ |
| Safebox Dual World | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | Ŏ | • | ŏ | Ö | O ⁴ | Ø ⁵ |
| D_{u} | | DataShield [77] | Hybrid | S | SHM | Any | | Ť | 0 | 0 | O ⁴ | Bounds Checking |
| | | ERIM [232] | Hybrid | S | SHM | Any | | ÷ | 0 | Ö | 0 | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | S | SHM | Function | | ÷ | 0 | 0 | 0 | Page Table |
| | | ¹ Inherited from thread-like | | | | | | _ | | | | |

¹ Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent). ⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

Names of the abstractions we consider

Characteristics we included in the taxonomy

| Class | Target | Abstraction | Cubicat Calcation | Sen | nantics | Abstraction | F | rop | ertie | s | Interface | Desig ad |
|-----------------------|--------|--|-------------------|---------------|---------|-----------------------|---|-----|----------------------|---|----------------|-------------------------|
| Class | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R | Safety | to sm |
| | | Virtines | Code-centric | S | MES | Function | | • | 0 | 0 | 0 | Virtual (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | | • | 0 | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | • | Ō | Ŏ | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | S | SHM | Library | Ŏ | • | Ŏ | Ŏ | Ö | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | • | Ō | Ō | 04 | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | Ŏ | • | Ŏ | Ŏ | • | Ø |
| | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | • | Ō | Ō | 0 | Ø ⁵ |
| £. | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | | • | 0 | Ö | 0 | Ø ⁵ |
| Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | Ŏ | • | 0 | Ŏ | 0 | Ø ⁵ |
| isti | | Light Weight C | Hybrid | S | SHM | Function ¹ | Ŏ | • | 0 | Ŏ | 0 | Page Table ² |
| <i>Q 1</i> | | POSIX Processes (and earlier instances) [93] | | Any | Any | Any | ŏ | Ť | | ŏ | 0 | Page Table |
| na | | News | Hybrid | S | SHM | Any | ă | • | 0 | 0 | O ⁴ | Ø |
| Иш | | libMPK [190] | Hybrid | S | SHM | Any | ŏ | Ť | 0 | ŏ | 0 | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | Ŏ | Ť | ŏ | ŏ | Ö | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | Ŏ | • | 0 | Ō | 0 | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | • | 0 | ŏ | 0 | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | H | • | $\overline{\bullet}$ | | 0 | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μLibrary | ŏ | Ť | 0 | ŏ | 0 | Protection Keys |
| | | FlexOS [161] | Code-centric | S | SHM | μLibrary | Ŏ | • | Ŏ | Ŏ | Ö | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | Ŏ | • | Ŏ | Ŏ | Ö | Ø |
| | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | Ō | • | Ō | 0 | 0 | Ø ⁵ |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | | • | | 0 | 0 | Virtual Machine (EPT) |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | • | Ō | Ō | 0 | ARM PAC + MTE |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | | • | • | | 0 | Ø ⁵ |
| | Ì | Cali [65] | Code-centric | S | SHM | Library | | • | 0 | | 0 | \varnothing^5 |
| | | CompARTist [132] | Code-centric | S | MES | Library | | • | Ō | Ō | 0 | Ø ⁵ |
| | | Enclosure [111] | Code-centric | S | SHM | Package | Ŏ | Ť | Ö | ŏ | 0 | Ø |
| 8 | U | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | Ŏ | • | • | Ŏ | 0 | Ø ⁵ |
| Sandbox | | RLBox / µSWITCH [181, 195] | Hybrid | S | SHM | Function | Ŏ | • | 0 | Ŏ | • | Ø |
| ana | | Wedge [70] | Hybrid | S | SHM | Function ¹ | | • | Ō | Ŏ | 0 | Ø ⁵ |
| S | U+K | CompartOS [55] | Code-centric | S | SHM | Linkage Unit | Ŏ | Ŏ | Ŏ | Ŏ | Ö | CHERI |
| | | LVDs / KSplit [133, 185] | Code-centric | S | MES | K-component | | • | | Ō | 0 | Ø ⁵ |
| | K | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | Ŏ | • | Ō | Ŏ | • | SFI |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | | • | 0 | 0 | 0 | Page Table ³ |
| | U | Shreds [87] | Code-centric | S | SHM | Any | | • | 0 | 0 | O ⁴ | Ø ⁵ |
| | | Privman [147] | Code-centric | S | MES | Function | Ŏ | • | 0 | 0 | 0 | Page Table ² |
| | | Privtrans [73] | Code-centric | S | MES | Function | | • | Ō | Ō | 0 | Page Table ² |
| ox | | Swift [89] | Code-centric | S+A | MES | Any | Ŏ | • | Ō | Ŏ | • | Ø |
| Safebox al World | U | Glamdring [165] | Code-centric | S | MES | Function | | • | 0 | Ō | • | Ø ⁵ |
| Safebox Dual World | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | | • | ŏ | Ö | O ⁴ | Ø ⁵ |
| D_{u} | | DataShield [77] | Hybrid | S | SHM | Any | | Ť | 0 | 0 | O ⁴ | Bounds Checking |
| | | ERIM [232] | Hybrid | S | SHM | Any | | ŏ | | ŏ | 0 | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | 8 | SHM | Function | ŏ | Ť | 0 | Ö | 0 | Page Table |
| | | ¹ Inherited from thread-lik | | | | | | _ | | | | |

¹ Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent). ⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

This talk: present a few interesting aspects, full discussion in the paper \bigcirc



| Cl | Target | Al-to-st-or | Cultinat Calantina | Sen | nantics | Abstraction | P | roper | ies | Interface | Design Bound |
|-----------------------|--------|--|--------------------|---------------------------|---------|-----------------------|-------|-------|-----|------------------|----------------------------|
| Class | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | .] | R Safety | to Mechanism |
| | Ì | Virtines [242] | Code-centric | S | MES | Function | • | • (| | | Virtual Machine (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | | • | _ | | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | • | _ | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | \mathcal{S} | SHM | Library | | • | _ | | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | • (| | O 04 | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | • | • (| |) • | Ø |
| | | Arbiter [241] | Data-centric | \mathcal{S} | SHM | Function ¹ | • | • (|) (| 0 | Ø ⁵ |
| | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | • | • (|) (| 0 | Ø ⁵ |
| Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | • | • (| | 0 | Ø ⁵ |
|)isi | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | • | • (|) (| | Page Table ² |
| I II | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | • | • (| | | Page Table |
| ıtuc | | SOAAP [117] | Hybrid | S | SHM | Any | • | • (|) (|) O ⁴ | Ø |
| M_{ν} | | libMPK [190] | Hybrid | \mathcal{S} | SHM | Any | • | • |) (| | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | • (| _ | | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | • (| | | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | \mathcal{S} | SHM | U/K-component | | • | | | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | \mathcal{S} | SHM | U/K-component | | • | | | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | | • | | 0 | Protection Keys |
| | | FlexOS [161] | Code-centric | S | SHM | μ Library | _ | • (| | 0 | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | | • (| _ | | Ø |
| | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | _ | • (| | | Ø ⁵ |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | • | • • | _ | 0 | Virtual Machine (EPT) |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | • | _ | 0 | ARM PAC + MTE |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | | • • | | | Ø ⁵ |
| | | Cali [65] | Code-centric | \mathcal{S} | SHM | Library | | • | | | \varnothing^5 |
| | | CompARTist [132] | Code-centric | \mathcal{S} | MES | Library | | • | | | Ø ⁵ |
| | U | Enclosure [111] | Code-centric | S | SHM | Package | • | • (|) (| 0 | Ø |
| ox | | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | | • | | 0 | Ø ⁵ |
| Sandbox | | RLBox / μSWITCH [181, 195] | Hybrid | \mathcal{S} | SHM | Function | | • | _ |) • | Ø |
| Sai | | Wedge [70] | Hybrid | S | SHM | Function ¹ | | • (| | | Ø ⁵ |
| | U+K | CompartOS [55] | Code-centric | S | SHM | Linkage Unit | | • • | _ | | CHERI |
| | K | LVDs / KSplit [133, 185] | Code-centric | \mathcal{S} | MES | K-component | | • (| | 0 | Ø ⁵ |
| | | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | - | • (| _ |) • | SFI |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | | • (| | 0 | Page Table ³ |
| | U | Shreds [87] | Code-centric | \mathcal{S} | SHM | Any | • | • (| | O 04 | Ø ⁵ |
| | | Privman [147] | Code-centric | S | MES | Function | • | • (|) (| 0 | Page Table ² |
| . ~ | . | Privtrans [73] | Code-centric | \mathcal{S} | MES | Function | • | • (|) (|) 0 | Page Table ² |
| $ c_{0} $ | | Swift [89] | Code-centric | $\mathcal{S}+\mathcal{A}$ | MES | Any | • | • |) (|) • | Ø |
| Safebox | U | Glamdring [165] | Code-centric | S | MES | Function | • | • |) (|) • | Ø ⁵ |
| Safebox Dual World | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | • | • (| | O 04 | Ø ⁵ |
| ~ | | DataShield [77] | Hybrid | S | SHM | Any | | • (| | O O4 | Bounds Checking |
| | | ERIM [232] | Hybrid | S | SHM | Any | • | • | | 0 | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | S | SHM | Function | | • (| | 0 | Page Table |
| | | 1 Inharitad from throad lile | 2 c | | 121 | 3 6 4 3 | т , 1 | 1.77 | 1 4 | TI DDA | door (to a contain autont) |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent). ⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

Abstractions often specialize on a particular threat model

| | C1 | Targe | t At | 0.1: . 0.1 .: | Sen | nantics | Abstraction | Pro | pertie | 2S | Interface | Design Pound | , h are |
|----------|-----------------|------------|--|-------------------|---------------|---------|-----------------------|-----|---------|----|----------------|-------------------------|---|
| | Clas | S U/K/H | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | CI | A | R | Safety | to Me | - which ar |
| Ì | | İ | Virtines [242] | Code-centric | 8 | MES | Function | • • | | | 0 | Virtual | es, which are heric but also kier to use |
| | | | ACES [90] | Code-centric | S | SHM | Function | • • | 10 | 0 | 0 | orocci | ric but are |
| | | | SeCage [171] | Code-centric | S | SHM | Function | • • | 0 | | 0 | V3. Pur del | neric' .ce |
| | | | HODOR [124] | Code-centric | S | SHM | Library | • • | | Ŏ | Ŏ | aulte 9° | ·ar to use |
| | | | CAPACITY [105] | Code-centric | S | SHM | Any | • • | _ | Ō | 04 | +rick | (lel co |
| | | U | Jif [261] | Code-centric | S+A | MES | Any | 0 0 | | Ŏ | • | Uno | |
| | | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | • • | 10 | 0 | 0 | Ø ⁵ | 1 |
| | | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | • • | _ | _ | 0 | Ø ⁵ | |
| | Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | • • | +- | Ō | 0 | Ø ⁵ | |
| | istı | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | • • | \perp | 0 | 0 | Page Table ² | 1 |
| | q_{l} | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | • • | _ | _ | 0 | Page Table | 1 |
| | tua | | SOAAP [117] | Hybrid | S | SHM | Any | • • | _ | Ŏ | 04 | Ø | 1 |
| | Ииї | | libMPK [190] | Hybrid | 8 | SHM | Any | | | tŏ | Ö | Protection Keys | 1 |
| | | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | \perp | tŏ | Ö | CHERI | 1 |
| | | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | _ | Ō | 0 | Ø ⁵ | |
| | | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | _ | _ | 0 | Ø ⁵ | |
| | | U+K | | Code-centric | 8 | SHM | U/K-component | | _ | _ | Ö | Safe Languages | 1 |
| | | | CubicleOS [211] | Code-centric | S | SHM | μLibrary | 0 0 | _ | Ŏ | Ö | Protection Keys | |
| | | | FlexOS [161] | Code-centric | S | SHM | μLibrary | • • | ĬŎ | Ŏ | Ö | Ø | 1 |
| | | | xMP [196] | Code-centric | S | SHM | Any | • • | _ | Ō | Ō | Ø | 1 |
| | | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | 0 | 10 | 0 | 0 | Ø ⁵ | 1 |
| | | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | • • | _ | _ | Ō | Virtual Machine (EPT) | 1 |
| | | K | HAKC [173] | Code-centric | S | SHM | Function | • • | 0 | 0 | 0 | ARM PAC + MTE | |
| | | | LibrettOS [187] | Code-centric | S | SHM | K-component | • • | • | • | 0 | Ø ⁵ | |
| Ì | | Ì | Cali [65] | Code-centric | S | SHM | Library | | To | То | 0 | Ø ⁵ | |
| | | | CompARTist [132] | Code-centric | S | MES | Library | | _ | | 0 | Ø ⁵ | 1 |
| | | U | Enclosure [111] | Code-centric | S | SHM | Package | 0 0 | _ | _ | 0 | Ø | 1 |
| | X | U | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | • | • | Ŏ | Ō | Ø ⁵ | 1 |
| | ▼ Sandbox | | RLBox / µSWITCH [181, 195] | Hybrid | S | SHM | Function | 0 0 | 10 | Ō | • | Ø | |
| <u> </u> | an | | Wedge [70] | Hybrid | S | SHM | Function ¹ | • | _ | _ | 0 | Ø ⁵ | What is Software Co |
| | S | U+K | | Code-centric | S | SHM | Linkage Unit | 0 0 | _ | | Ö | CHERI | What is software co |
| | | | LVDs / KSplit [133, 185] | Code-centric | S | MES | K-component | • • | 10 | _ | 0 | Ø ⁵ | Software compartmentalization |
| | | K | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | 0 0 | _ | | • | SFI | Isolation Boundary |
| | | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | • • | _ | 0 | 0 | Page Table ³ | |
| Ì | | U | Shreds [87] | Code-centric | S | SHM | Any | | To | 0 | O ⁴ | Ø ⁵ | Sa |
| | Г | | Privman [147] | Code-centric | S | MES | Function | 0 0 | _ | Ŏ | 0 | Page Table ² | Times growth |
| | | | Privtrans [73] | Code-centric | S | MES | Function | | _ | 0 | 0 | Page Table ² | Sandbox: component isolated to Safebox: c |
| | xo. | rld | Swift [89] | Code-centric | S+A | MES | Any | | | | • | Ø | protect the rest of the system protect it |
| | Safebox | Dual World | Glamdring [165] | Code-centric | S | MES | Function | | _ | _ | • | Ø ⁵ | All but the sandbox is trusted Only the s |
| | Sa | al | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | | _ | 6 | O ⁴ | Ø ⁵ | Illustrativités adalpartia anguarki templupantan kahi kitan stiliananang, tra [2] titan iran akkipa |
| | | Cu | DataShield [77] | Hybrid | S | SHM | Any | | + | 0 | 04 | Bounds Checking | 1 |
| | | | ERIM [232] | Hybrid | 8 | SHM | Any | | _ | _ | 0 | Protection Keys | 1 |
| | | | Nested Vermal rous | Cada contrio | 0 | SHM | Emetion | | 10 | 10 | 0 | Dogo Toble | 1 |

For sandboxing

(libraries, drivers)

For **safeboxing**

privileged code)

Nested Kernel [94]

(crypto keys,

What is Software Compartmentalization? Software compartmentalization can target different trust models

Only the safebox is trusted

SHM Function Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent). ⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

A lot of interest in abstractions for kernel compartmentalization

| Class | Target | Abstraction | Subject Selection | | nantics | Abstraction | | operti | _ | Interface | Design Bound | |
|--------------------|---------------|---|-------------------|---------------|---------|-----------------------|----------|----------|---|----------------|-------------------------|-----------------|
| | U/K/HV | | | | ASSIGN | | _ | | R | Safety | to Mechanism | |
| | | Virtines [242] | Code-centric | S | MES | Function | | | _ | 0 | Virtual Machine (EPT) | |
| | | ACES [90] | Code-centric | S | SHM | Function | • • | | 0 | 0 | Ø ⁵ | |
| | | SeCage [171] | Code-centric | S | SHM | Function | • (| | 0 | 0 | Ø ⁵ | |
| | | HODOR [124] | Code-centric | S | SHM | Library | • • | | 0 | 0 | Ø | |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | • (| | 0 | 04 | ARM PAC + MTE | |
| | U | Jif [261] | Code-centric | S+A | MES | Any | • (| | 0 | • | Ø | |
| | | Arbiter [241] | Data-centric | \mathcal{S} | SHM | Function ¹ | • (| | 0 | 0 | Ø ⁵ | |
| | 0 100 10 | | Data-centric | \mathcal{S} | SHM | Function ¹ | • | | 0 | 0 | Ø ⁵ | |
| | omp | artmentalize | Data-centric | S | SHM | Function ¹ | • (| | 0 | 0 | Ø ⁵ | |
| | • | 167] | Hybrid | S | SHM | Function ¹ | • (| | 0 | 0 | Page Table ² | |
| us | erlar | nd and kernel stances) [9 | Hybrid | Any | Any | Any | • (| 0 | 0 | 0 | Page Table | |
| | | | Hybrid | S | SHM | Any | • (| | 0 | O ⁴ | Ø | |
| | t d | ogether | Hybrid | S | SHM | Any | • (| | 0 | 0 | Protection Keys | |
| | | ogether | Code-centric | Any | Any | U/K-component | | 0 | 0 | 0 | CHERI | |
| , | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | • | | 0 | 0 | Ø ⁵ | |
| | - | Mutable Protection Domains (MPDs) [191, | 192] Code-centric | S | SHM | U/K-component | • (| | • | 0 | Ø ⁵ | |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | • | | 0 | 0 | Safe Languages | |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | • | | 0 | 0 | Protection Keys | |
| | | FlexOS [161] | Code-centric | S | SHM | μ Library | • | 0 | 0 | 0 | Ø | |
| | | xMP [196] | Code-centric | S | SHM | Any | | | 0 | 0 | Ø | |
| | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | 0 | | 0 | 0 | Ø ⁵ | |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | | 9 | 0 | 0 | Virtual Machine (EPT) | |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | | 0 | 0 | ARM PAC + MTE | |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | • | • | • | 0 | Ø ⁵ | |
| | | [65] | Code-centric | S | SHM | Library | • (| | 0 | 0 | \varnothing^5 | |
| nei | only. | pARTist [132] | Code-centric | \mathcal{S} | MES | Library | • | | 0 | 0 | This are | se as a revival |
| | • | osure [111] | Code-centric | S | SHM | Package | • | | 0 | 0 | inis car | ne as a revival |
| ŝ | | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | • | • | 0 | 0 | | |
| Sandbox | | RLBox / μSWITCH [181, 195] | Hybrid | S | SHM | Function | • (| | 0 | • | researc | h in microkerne |
| Sar | | Wedge [70] | Hybrid | S | SHM | Function ¹ | • | | 0 | 0 | R | |
| -1 | U+K | CompartOS [55] | Code-centric | S | SHM | Linkage Unit | | 9 | 0 | 0 | CHERI | |
| | K | LVDs / KSplit [133, 185] | Code-centric | S | MES | K-component | | | 0 | 0 | \varnothing^5 | |
| | K | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | • | | 0 | • | SFI | |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | | | 0 | 0 | Page Table ³ | |
| | | | Code-centric | S | SHM | Any | • (| | 0 | O ⁴ | Ø ⁵ | |
| | r hy | pervisors ———— | Code-centric | S | MES | Function | • (| | 0 | 0 | Page Table ² | |
| 1 | , , ,, | PC1 413013 | Code-centric | S | MES | Function | • | | 0 | 0 | Page Table ² | |
| 00 0 | | Switt [09] | Code-centric | S+A | MES | Any | • (|) | 0 | • | Ø | |
| Safeb _o | U | Glamdring [165] | Code-centric | \mathcal{S} | MES | Function | • | | 0 | • | Ø ⁵ | |
| Safebo Dual Wor | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | • | | 0 | O ⁴ | Ø ⁵ | |
| ã | [] | DataShield [77] | Hybrid | S | SHM | Any | • | | 0 | O ⁴ | Bounds Checking | |
| | | ERIM [232] | Hybrid | 8 | SHM | Any | • (| | 0 | 0 | Protection Keys | |
| | K | Nested Kernel [94] | Code-centric | S | SHM | Function | A | | 0 | 0 | Page Table | |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent). ⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

A lot of interest in rethinking communication and sharing between domains

| Class | Target U/K/HV | Abstraction | Subject Selection | Sen CALL | nantics ASSIGN | Abstraction Granularity | Pro | pertie A | R | Interface Safety | Design Bound to Mechanism |
|--|------------------|--|--------------------|---------------|-------------------|--------------------------------|----------|-------------|-------------|---------------------|------------------------------|
| | | Virtines [242] | Code-centric | S | MES | Function | | | | 0 | Virtual Machine (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | | <u> </u> | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | _ | 0 | 0 | ~ Ø ⁵ |
| | | HODOR [124] | Code-centric | S | SHM | Library | | | 0 | 0 | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | <u> </u> | 0 | O ⁴ | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | | 0 | 0 | | Ø |
| | 0 | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | | 0 | 0 | ~ Ø ⁵ |
| | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | | | | 0 | Ø ⁵ |
| ust | | Salus [226] | Data oo | | | | | | 1 () 1 | | Ø ⁵ |
| Mutual Distrust | | Light-Weight Contexts (LwCs) [167] | Hybr Le | ver | age s | hared m | em | or | v t | n | Page Table ² |
| P | | POSIX Processes (and earlier instances) [93] | Hybr | | 48C 3 | ···a··c·a··· | | ٠. | , - | | Page Table |
| жаІ | | SOAAP [117] | Hybr SD | haad | unc | haring b | otv | VO | an | | rage Table |
| [uti | | libMPK [190] | Hybr SP | JEEU | ups | ilai ilig k | JELV | VC | - 11 | | Protection Keys |
| 1 2 | | CheriOS [108] | | | : /. | | | | | : ~\ | CHERI |
| | | Microkernel Servers [106], [] | Code-ce CC | oma | ins (v | s. mess | age | pa | 155 | ing) | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | _ `_ | | T - T - | Ť | | | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | 8 | SHM | U/K-component U/K-component | • • | | | 0 | Safe Languages |
| | OIR | CubicleOS [211] | Code-centric | S | SHM | μLibrary | | | | 0 | Protection Keys |
| | | FlexOS [161] | Code-centric | S | SHM | μ Library | | _ | 0 | 0 | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | | | 0 | 0 | Ø |
| | | Monza [35] | Hybrid | \mathcal{A} | SHM | Function ¹ | | _ | 0 | 0 | Ø ⁵ |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | | | 0 | 0 | Virtual Machine (EPT) |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | | 0 | 0 | ARM PAC + MTE |
| | IX. | LibrettOS [187] | Code-centric | S | SHM | K-component | • | | | 0 | Ø ⁵ |
| | | Cali [65] | Code-centric | S | SHM | Library | | | | 0 | Ø ⁵ |
| | | CompARTist [132] | Code-centric | S | MES | Library | | 1 - | 0 | 0 | Ø ⁵ |
| | | Enclosure [111] | Code-centric | S | SHM | Package | | 0 | ŏ | 0 | Ø |
| 8 | U | Google Sandber 1 + Dr. (G + Dr.) | | | | nction | | | Ŏ | 0 | Ø ⁵ |
| Sandbox | | DID / | | | | tion | | | ŏ | • | Ø |
| ana | | Wedge [70] Exposing com | municati | on | | tion ¹ | • • | 0 | Ŏ | 0 | Ø ⁵ |
| \ \cdot \cdo | U+K | CompartOS | | | | e Unit | • | | Ŏ | Ö | CHERI |
| | | LVDs / KSp between dom | ains as fu | unct | ion | onent | • | 0 | Ō | 0 | Ø ⁵ |
| | K | XFI/LXFI [1 | | | | onent | • | 0 | Ŏ | • | SFI |
| | HV | Nexen [221] calls (vs. IPCs) | to be mo | ore i | ntuit | domain | • • | _ | Ō | 0 | Page Table ³ |
| | U | Shreds [87] | to be me | ,, с , | rreare | ny | • • | | | O ⁴ | \varnothing^5 |
| | | Privman [147] | Code-centric | S | MES | Function | • • | 0 | 0 | 0 | Page Table ² |
| | | Privtrans [73] | Code-centric | S | MES | Function | • • | | 0 | 0 | Page Table ² |
| 2000 | | Swift [89] | Code-centric | S+A | MES | Any | • • | | 0 | • | Ø |
| Safebox Dual World | U | Glamdring [165] | Code-centric | S | MES | Function | • • | 0 | 0 | • | Ø ⁵ |
| Se | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | • • | 0 | 0 | O ⁴ | Ø ⁵ |
| D_{l} | | DataShield [77] | Hybrid | S | SHM | Any | • | 0 | Ō | O ⁴ | Bounds Checking |
| | | ERIM [232] | Hybrid | S | SHM | Any | • • | _ | Ŏ | 0 | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | S | SHM | Function | • • | _ | Ŏ | 0 | Page Table |
| | | 1 Inherited from thread-like | a compation 2 from | . n=0000 | lilra como | ntice 3 from the | Jactad I | Zarnal | 4 T | ha DDM d | pas (to a cartain avtant) |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent). ⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

Interest in abstractions that compose with different isolation mechanisms

| Target | |
|--|----------|
| ACES 100 Code-centric S SHM Function | |
| Secage (171) | |
| Secage (171) | |
| HODÖR 124 Code-centric S SHM | |
| 1 | |
| Arbiter [241] | |
| Secure Memory Views (SMVs) | |
| Salus 226 Data-centric S SHM Function O O O Check Che | |
| CherioS [108] | |
| CherioS [108] | |
| CherioS [108] | ons that |
| CherioS [108] | |
| CherioS [108] | e |
| Microkernel Servers [106], [] Code-centric Any MES U/K-component | C |
| U+K Mutable Protection Domains (MPDs) [191, 192] Code-centric S SHM U/K-component O O O Safe Languages | |
| U+K | |
| $ \begin{array}{ c c c c c c c c } \hline CubicleOS [211] & Code-centric & S & SHM & \mu Library & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline FlexOS [161] & Code-centric & S & SHM & \mu Library & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline xMP [196] & Code-centric & S & SHM & Any & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline xMP [196] & Code-centric & S & SHM & Any & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline Monza [35] & Hybrid & A & SHM & Function & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline WirtuOS [186] & Code-centric & S & SHM & Function & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline WirtuOS [186] & Code-centric & S & SHM & Function & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline LibrettOS [187] & Code-centric & S & SHM & Function & \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline & & & & & & & & & & & & & & & & & &$ | |
| FlexOS [161] | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | |
| Monza [35] | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | |
| $ \begin{array}{ c c c c c c c c c c c c c c c c c c c$ | |
| $ \begin{array}{ c c c c c c c c c c c c c c c c c c c$ | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | |
| | |
| $ \begin{array}{ c c c c c c c c c c c c c c c c c c c$ | |
| $U+K$ CompartOS [55] Code-centric S SHM Linkage Unit \bullet | |
| $U+K$ CompartOS [55] Code-centric S SHM Linkage Unit \bullet | |
| $U+K$ CompartOS [55] Code-centric S SHM Linkage Unit \bullet | |
| LVDs / VCnlit ress ress | |
| | |
| XFI/LXFI [107, 172] Hybrid S SHM K-component • • • • • SFI | |
| HVNexen [221]Data-centric S SHMPer-VM domain \bullet \bullet \bigcirc \bigcirc Page Table ³ | |
| U Shreds [87] $Code$ -centric S SHM Any \bullet \bullet \bigcirc | |
| Privman [147] $Code$ -centric S MES Function \bullet \bullet \bigcirc \bigcirc Page Table ² | |
| Privtrans [73] Code-centric S MES Function • • • • • Page Table ² | |
| $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ | |
| Swift [89] Code-centric S+A MES Any Glamdring [165] Code-centric S MES Function PtrSplit / Program Mandering (160 170) Code centric S MES Function O O O O O O O O O O O O O O O O O O O | |
| PtrSplit / Program Mandering [169, 170] | |
| DataShield [77] Hybrid S SHM Any \bullet O O O Bounds Checking | |
| ERIM [232] Hybrid S SHM Any • • O O Protection Keys | |
| K Nested Kernel [94] Code-centric S SHM Function • • • • • Page Table The price of from three dillocamenting 2 from process like companies 3 from the Nested Kernel 4 The PDM does (to a contain extent) | |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent).

⁵ The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

| Class | Target | Abstraction | Subject Selection | Sen | nantics | Abstraction | 1 | Prop | ertie | 25 | Interface | Design Bound |
|-----------------------|--------|--|-------------------|------|---------|-----------------------|---|------|-------|----|----------------|-------------------------|
| Ciass | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R | Safety | to Mechanism |
| | | Virtines [242] | Code-centric | S | MES | Function | | | 0 | | 0 | Virtual Machine (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | | | 0 | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | | 0 | 0 | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | S | SHM | Library | | | 0 | 0 | 0 | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | | 0 | 0 | O ⁴ | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | | | 0 | 0 | | Ø |
| | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | | | 0 | | Ø ⁵ |
| 19 | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | | | 0 | | | \varnothing^5 |
| Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | | | 0 | 0 | | Ø ⁵ |
|)isi | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | | | | 0 | | Page Table ² |
| I II | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | | | 0 | 0 | | Page Table |
| ıtαc | | SOAAP [117] | Hybrid | S | SHM | Any | | | 0 | 0 | O ⁴ | Ø |
| M_{l} | | libMPK [190] | Hybrid | S | SHM | Any | | | | 0 | | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | | | | | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | | 0 | | | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | | | | | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | | | | 0 | | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | | | 0 | 0 | 0 | Protection Keys |
| | | FlexOS [161] | Code-centric | S | SHM | μ Library | | | 0 | 0 | 0 | Ø |
| | | xMP [196] | Code-centric | S | SHM | Any | | | 0 | 0 | 0 | Ø |
| | | Monza [35] | Hybrid | A | SHM | Function ¹ | 0 | | 0 | 0 | 0 | Ø ⁵ |
| | | VirtuOS [186] | Code-centric | S+A | SHM | K-component | | | | 0 | 0 | Virtual Machine (EPT) |
| | K | HAKC [173] | Code-centric | S | SHM | Function | | | 0 | 0 | 0 | ARM PAC + MTE |
| | | LibrettOS [187] | Code-centric | S | SHM | K-component | | | | | 0 | Ø ⁵ |
| | | Cali [65] | Code-centric | S | SHM | Library | | | | 0 | | \varnothing^5 |
| | | CompARTist [132] | Code-centric | S | MES | Library | | | 0 | 0 | | \varnothing^5 |
| | U | Enclosure [111] | Code-centric | S | SHM | Package | | | | 0 | | Ø |
| ox | | Google Sandboxed API (SAPI) [22] | Code-centric | S | MES | Function | | | | 0 | 0 | \varnothing^5 |
| Sandbox | | RLBox / μSWITCH [181, 195] | Hybrid | S | SHM | Function | | | 0 | 0 | • | Ø |
| Sai | | Wedge [70] | Hybrid | S | SHM | Function ¹ | | | 0 | 0 | 0 | Ø ⁵ |
| | U+K | CompartOS [55] | Code-centric | S | SHM | Linkage Unit | | | | 0 | 0 | CHERI |
| | K | LVDs / KSplit [133, 185] | Code-centric | S | MES | K-component | | | 0 | 0 | 0 | Ø ⁵ |
| | | XFI/LXFI [107, 172] | Hybrid | S | SHM | K-component | | • | 0 | 0 | | SFI |
| | HV | Nexen [221] | Data-centric | S | SHM | Per-VM domain | | | | | 0 | Page Table ³ |
| _ | U | Shreds [87] | Code-centric | S | SHM | Any | | | 0 | 0 | O ⁴ | \varnothing^5 |
| | | Privman [147] | Code-centric | S | MES | Function | | | 0 | 0 | | Page Table ² |
| ء ا ء | | Privtrans [73] | Code-centric | S | MES | Function | | | | 0 | | Page Table ² |
| bo. | | Swift [89] | Code-centric | S+A | MES | Any | | | | 0 | | Ø |
| Safebox Dual World | U | Glamdring [165] | Code-centric | S | MES | Function | | | 0 | 0 | | Ø ⁵ |
| s s | | PtrSplit / Program Mandering [169, 170] | Code-centric | S | MES | Function | | | | 0 | O ⁴ | Ø ⁵ |
| P | | DataShield [77] | Hybrid | S | SHM | Any | | | 0 | 0 | O ⁴ | Bounds Checking |
| | | ERIM [232] | Hybrid | S | SHM | Any | | | 0 | 0 | 0 | Protection Keys |
| | K | Nested Kernel [94] | Code-centric | S | SHM | Function | | | 0 | 0 | | Page Table |

| Cla | <i>T</i> | Target | Abstraction | Subject Selection | Sen | ıantics | Abstraction | P_{I} | oper | ies | | Interface | Design Bound |
|---------------------|------------|--------------------|--|--|---|--|--|---|------|-----|----------|----------------|---|
| Cia | SS U | /K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | 1 | R | Safety | to Mechanism |
| | | | Virtines [242] | Code-centric | S | MES | Function | | • (|) İ | οĪ | 0 | Virtual Machine (El |
| | | | ACES [90] | Code-centric | S | SHM | Function | | | | 0 | 0 | Ø ⁵ |
| | | | SeCage [171] | Code-centric | S | SHM | Function | | | 5 | 0 | 0 | Ø ⁵ |
| | | | HODOR [124] | Code-centric | S | SHM | Library | _ | | 5 | ŏ | 0 | Ø |
| | | | CAPACITY [105] | Code-centric | S | SHM | Any | • | | | | O ⁴ | ARM PAC + MT |
| | | U | Jif [261] | Code-centric | S+A | MES | Any | | | | ŏ | | Ø |
| | | C | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | | 5 | 0 | 0 | Ø ⁵ |
| + | . | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | • | • | | | 0 | Ø ⁵ |
| Mutual Distrust | | | Salus [226] | Data-centric | S | SHM | Function ¹ | | | | 0 | 0 | Ø ⁵ |
| iet. | | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | • | | | | 0 | Page Table ² |
| 1 / | <u> </u> | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | | | 5 | õ | Ö | Page Table |
| tito | | | SOAAP [117] | Hybrid | S | SHM | Any | | | | 0 | O ⁴ | Ø |
| 7 | | | libMPK [190] | Hybrid | S | SHM | Any | | | 5 | ŏ | 0 | Protection Key |
| | · | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | | | | 0 | CHERI |
| | | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | • | | | | 0 | Ø ⁵ |
| | | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | • | | | | 0 | Ø ⁵ |
| | 1 | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | | 0 (| | 0 | 0 | Safe Language |
| | | | CubicleOS [211] | Code-centric | S | SHM | μLibrary | | | 5 | | 0 | Protection Key |
| | | | | Code-centric | S | SHM | μ Library | | | | 0 | 0 | Ø |
| | | | | | | | | | | | | | ~ |
| tr: | cti | ion | cnacializad | Code-centric | S | SHM | Any | | | | \circ | | Ø |
| tra | acti | ion | specialized | Code-centric Hybrid | S | SHM SHM | Any Function ¹ | _ | | | <u> </u> | 0 | Ø ⁵ |
| | | | · · · · · · · · · · · · · · · · · · · | | A S+A | | , | 0 | _ | | 0 | | Ø ⁵ |
| | | | specialized ng untrusted | Hybrid | A S+A S | SHM | Function ¹ | 0 | | | 0 | 0 | Ø ⁵ Virtual Machine (I ARM PAC + M |
| an | db | oxir | ng untrusted | Hybrid Code-centric | A S+A | SHM SHM | Function ¹ K-component | 0 | • (| | 0 | 0 | Ø ⁵ Virtual Machine (I |
| an | db | oxir | · · · · · · · · · · · · · · · · · · · | Hybrid Code-centric Code-centric | A S+A S | SHM SHM SHM | Function ¹ K-component Function K-component | 0 | • (| | 0 | 0 0 | Ø ⁵ Virtual Machine (I ARM PAC + M |
| an | db | oxir | ng untrusted | Hybrid Code-centric Code-centric Code-centric | A S+A S | SHM SHM SHM SHM | Function K-component Function K-component Library | 0 | 0 0 | | | 0 | Virtual Machine (I ARM PAC + M' |
| an | db | oxir ch a | ng untrusted | Hybrid Code-centric Code-centric Code-centric Code-centric | A S+A S S | SHM SHM SHM SHM | Function ¹ K-component Function K-component | 0 | | | | 0 0 0 | Virtual Machine (I ARM PAC + M' |
| an e \ | dbo (su | oxir | ng untrusted as libraries) | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric | A S+A S S S | SHM SHM SHM SHM SHM | Function K-component Function K-component Library Library | 00000000000 | | | | 0 0 0 | Virtual Machine (I ARM PAC + M ² |
| an e \ | dbo (su | oxir ch a | ng untrusted as libraries) Enclosure [111] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | A S+A S S S | SHM SHM SHM SHM SHM SHM SHM | Function K-component Function K-component Library Library Package | 0 | | | | 0 0 0 | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' |
| an | dbo (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | A S+A S S S S | SHM SHM SHM SHM SHM MES SHM MES | Function K-component Function K-component Library Library Package Function | | | | 0 | 0 0 0 0 0 0 0 | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø |
| an e \ | dbo (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid | A S+A S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM | Function K-component Function K-component Library Library Package Function Function | | | | 0 | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø |
| an e \ | dbo (su | oxir ch a | ng untrusted as libraries) Comp. Reflection Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid | A S+A S S S S S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM SHM | Function K-component Function K-component Library Library Package Function Function | | | | 0 | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' CHERI |
| an e \ | dbo (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric | A S+A S S S S S S S S S S S S S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM MES SHM SHM | Function K-component Function K-component Library Library Package Function Function Linkage Unit | | | | 0 | | Virtual Machine (I ARM PAC + M Ø |
| an e \ | dbo (su | oxir ch a | ng untrusted as libraries) Comp. Reflection Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric | A S+A S S S S S S S S S S S S S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM MES SHM SHM SHM SHM | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component | | | | 0 | | Virtual Machine (I ARM PAC + M Ø ⁵ Ø ⁵ Ø Ø Ø CHERI |
| an e \ | dbo (su | oxir ch a | ng untrusted as libraries) Comp. Ref. Bet [172] Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Hybrid Code-centric Hybrid | A S+A S S S S S S S S S S S S S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM SHM SHM SHM SHM | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component | | | | 0 | | Virtual Machine (ARM PAC + M |
| an e \ | dbo (su | U U U+K K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / μswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Data-centric | A S+A S S S S S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM SHM SHM SHM SHM SHM MES SHM | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component F-component F-component F-component F-component F-component F-component F-component | | | | 0 | | Virtual Machine (I ARM PAC + M Ø |
| e vangoox | db (su | U U U+K K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / μSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Code-centric Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | A S+A S S S S S S S S S | SHM SHM SHM SHM SHM SHM MES SHM MES SHM SHM SHM SHM SHM SHM SHM SHM SHM | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component F-component F-component F-component F-component F-component F-component F-component F-component | | | | 0 | | Virtual Machine (I ARM PAC + M Ø's Ø's Ø's Ø's Ø's Ø's Ø's CHERI Ø's SFI Page Table's |
| e vangoox | db (su | U U U+K K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | A S+A S S S S S S S S S S S | SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM SHM SHM SHM SHM MES SHM SHM SHM SHM SHM SHM | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component F-component | | | | 0 | | Virtual Machine (I ARM PAC + M Ø |
| e vangoox | db (su | U U U+K K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Code-centric | A S+A S S S S S S S S S S S S | SHM SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM SHM SHM SHM MES SHM MES SHM SHM MES SHM SHM SHM SHM SHM SHM SHM SHM SHM SH | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component Fer-VM domain Any Function Function Function | | | | 0 | | Virtual Machine (1) ARM PAC + M Ø |
| e vangoox | db (su | U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric | A S+A S S S S S S S S S S S S S S S S S | SHM SHM SHM SHM SHM SHM MES SHM MES SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM MES SHM SHM SHM SHM SHM SHM SHM SHM SHM SH | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component Per-VM domain Any Function Function Function Any Function | | | | 0 | | Virtual Machine (I ARM PAC + M Ø |
| e vandbox | dbo (su | U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric A S+A S S S S S S S S S S S S S S S S S | SHM SHM SHM SHM SHM SHM MES SHM MES SHM SHM SHM SHM SHM MES SHM MES SHM SHM SHM SHM SHM SHM SHM SHM SHM SH | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component F-component F-component F-component Per-VM domain Any Function Function Function Function Function Function Function Function Function Function | | | | 0 | | Virtual Machine (I ARM PAC + M Ø |
| e vandbox | db (su | U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric | A S+A S S S S S S S S S S S S S S S S S | SHM SHM SHM SHM SHM SHM MES SHM MES SHM MES SHM SHM SHM MES SHM MES SHM MES SHM SHM SHM SHM SHM SHM SHM SHM SHM SH | Function K-component Function K-component Library Library Package Function Function Linkage Unit K-component K-component Per-VM domain Any Function Function Function Any Function | | | | 0 | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' Page Table' Page Table Ø Ø' O' O' O' O' O' O' O' O' |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent).

| Cla | | Target | Abstraction | Subject Selection | | antics | Abstraction | 1 | Prop | ertie | S | Interface | Design Bound |
|-----------------|-----------|--------------------------|--|--|--------------------------------------|--|--|-----------|----------|----------|-------|----------------|--|
| Cia | 53 U | /K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R | Safety | to Mechanism |
| | | | Virtines [242] | Code-centric | S | MES | Function | | | 0 | 0 | 0 | Virtual Machine (El |
| | | | ACES [90] | Code-centric | S | SHM | Function | | | | | | \varnothing^5 |
| | | | SeCage [171] | Code-centric | S | SHM | Function | | | | | | Ø ⁵ |
| | | | HODOR [124] | Code-centric | S | SHM | Library | | | | | | Ø |
| | | | CAPACITY [105] | Code-centric | S | SHM | Any | | | | | 04 | ARM PAC + MT |
| | | U | Jif [261] | Code-centric | S+A | MES | Any | | | | | | Ø |
| | | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | | | | | Ø ⁵ |
| 19 | | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function 1 | | | | | | Ø ⁵ |
| Mutual Distrust | | | Salus [226] | Data-centric | S | SHM | Function ¹ | | | | | | Ø ⁵ |
| Disi | | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | | | | | | Page Table ² |
| I Ji | İ | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | | | 0 | 0 | | Page Table |
| tuc | | | SOAAP [117] | Hybrid | S | SHM | Any | | | | | O ⁴ | Ø |
| M_{ν} | | | libMPK [190] | Hybrid | S | SHM | Any | | | | | | Protection Keys |
| | | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | | | | | CHERI |
| | | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | | | | | Ø ⁵ |
| | | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | | | | | Ø ⁵ |
| | | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | | | | | | Safe Languages |
| | | | CubicleOS [211] | Code-centric | S | SHM | μ Library | | | | | | Protection Keys |
| | | | | Code-centric | S | SHM | μ Library | | | | | | Ø |
| | | | | | | | | | | | | | |
| ra | act | ion | specialized | Code-centric | S | SHM | Anv | | | | | 0 | Ø |
| ra | act | ion | specialized | Code-centric | s | SHW | Anv | | ct r | <u> </u> | \ D : | , ° | Ø ⁵ |
| | | | • | Code-centric | s nent | SHW | ss-comp | ar | rtr | ne | su. | t | Ø ⁵ Virtual Machine (E |
| | | | specialized ng untrusted | Implem | | s cro | - | | | | | t | Ø ⁵ Virtual Machine (E ARM PAC + MT |
| n | db | oxir | ng untrusted | Implem | | s cro | ess-comp | | | | | t | |
| n | db | oxir | • | Implem comm | nuni | s cro catio | n with f | un | ct | | | t | Ø ⁵ Virtual Machine (E ARM PAC + MT |
| n | db | oxir | ng untrusted | Implem comm | nuni | s cro catio | n with f | un | ct | | | t | Virtual Machine (E ARM PAC + MT |
| in e (| db (su | oxir | ng untrusted as libraries) Enclosure [111] | Implem comm | nuni | s cro catio | - | un | ct | | | t p | Virtual Machine (E ARM PAC + MT |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] | Implem comm | nuni Ils, v | s cro catio | n with for h is intui | un | ct | io | | t | Virtual Machine (E ARM PAC + MT |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] | Implem comm | nuni Ils, v | s cro catio | n with fo | un tiv | ct ⁄e | io | | t | Virtual Machine (E ARM PAC + MT |
| n | db (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] | Code-centric Implem comm ca Code-centric Hybrid Hybrid | lls, | catio catio which | n with for is intuing the function Function Function | un tiv | ct ⁄e | io | n | | Virtual Machine (E ARM PAC + M7 Ø |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Somp Helde [115] Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] | Code-centric Implem comm ca Code-centric Hybrid Hybrid Code-centric | Ils, | cs cro catio which | n with for is intui | un tiv | ct ve | io | n | | Virtual Machine (E ARM PAC + MT Ø' Ø' Ø' Ø' Ø' Ø' O' O' O' O' |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Comparative (195) Enclosure (111) Google Sandboxed API (SAPI) (22) RLBox / µswitch (181, 195) Wedge (70) CompartOS (55) LVDs / KSplit (133, 185) | Code-centric Implem comm ca Code-centric Hybrid Hybrid Code-centric Code-centric | Ils, | catio catio which MES SHM SHM SHM MES | Function Function Function Function Function Linkage Unit K-component | tiv | ct /e | io | n | | Virtual Machine (FARM PAC + MTAC) |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Somp Helde [115] Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] | Code-centric Implem comm ca Code-centric Hybrid Hybrid Code-centric | Ils, | catio which | Function Function Function Function Linkage Unit K-component K-component | un tiv | ct /e | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Comparative (195) Enclosure (111) Google Sandboxed API (SAPI) (22) RLBox / µswitch (181, 195) Wedge (70) CompartOS (55) LVDs / KSplit (133, 185) | Code-centric Implem comm ca Code-centric Hybrid Hybrid Code-centric Code-centric | Ils, | catio catio which MES SHM SHM SHM MES | Function Function Function Function Function Linkage Unit K-component | tiv | ct /e | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' CHERI |
| in e (| db (su | oxir ch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] | Code-centric Implem Comm Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid | Ils, | catio which MES SHM SHM SHM MES SHM | Function Function Function Function Linkage Unit K-component K-component | un tiv | ct /e | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI |
| in e (| db (su | U U U+K K HV | Ing untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / μswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] | Code-centric Implem Comm Cat Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Data-centric | Ils, | CS Cro catio which MES SHM SHM MES SHM SHM SHM | Function Function Function Function Linkage Unit K-component K-component Per-VM domain | tiv | ct /e | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' |
| sandbox (| db | U U U+K K HV | Ing untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / μSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] | Code-centric Implem Comm Cat Code-centric Hybrid Hybrid Code-centric Code-centric Hybrid Data-centric Code-centric | Ils, | CS Cro catio which MES SHM SHM MES SHM SHM SHM SHM | Function Function Function Function Function Linkage Unit K-component K-component Per-VM domain Any | tiv | etve | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' |
| sandbox (| db | U U U+K K HV | Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] | Code-centric Implem Comm Cat Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric Code-centric | Ils, | CS Cro catio which MES SHM SHM MES SHM SHM SHM SHM SHM SHM SHM | Function Function Function Function Function Linkage Unit K-component K-component Per-VM domain Any Function | un tiv | ct re | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' Page Table' |
| sandbox (| db | U U U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] | Code-centric Implem Comm Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric Code-centric Code-centric Code-centric Code-centric | Ils, | CS Cro catio which MES SHM SHM MES SHM SHM SHM SHM SHM SHM SHM SHM MES MES | Function Function Function Function Function Function Linkage Unit K-component K-component Per-VM domain Any Function Function | un | ct re | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' Page Table' Page Table' |
| sandbox (| db | U U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Code-centric Implem Comm Ca Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | s s s s s s s | CS Cro Catio Which MES SHM SHM MES SHM SHM SHM SHM MES MES MES MES MES | Function Function Function Function Function Function Linkage Unit K-component K-component Per-VM domain Any Function Function Any Function Function | un tiv | et/e | io | n | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' Page Table' Page Table Page Table Page Table |
| sandbox (| db (su | U U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Code-centric Implem Comm Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric | s s s s s s s s | CS Cro Catio Which MES SHM SHM MES SHM SHM MES SHM SHM MES MES MES MES MES MES | Function Function Function Function Function Function Linkage Unit K-component K-component Per-VM domain Any Function Function Any Function Function Function Function | un tiv | etve | io | n | | Virtual Machine (E ARM PAC + MT Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' Page Table' Page Table' Page Table' Ø' Ø' O' O' O' O' O' O' O' O |
| sandbox (| db | U U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, 185] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Code-centric Implem Comm Ca Code-centric Hybrid Hybrid Code-centric Hybrid Data-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | s s s s s s s | CS Cro Catio Which MES SHM SHM MES SHM SHM SHM SHM MES MES MES MES MES | Function Function Function Function Function Function Linkage Unit K-component K-component Per-VM domain Any Function Function Any Function Function | un tiv | et/e | io | n | | Virtual Machine (FARM PAC + MTAC) STATE OF STAT |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent).

| Class | Target | Abstraction | Subject Selection | Sen | ıantics | Abstraction | 1 | Proper | ties | Inte | rface | Design Bound |
|-----------------|----------------|---|--|-----------------|--|--|-----------|-----------|------|---------|---|--|
| Ciass | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R Sa | fety | to Mechanism |
| | | Virtines [242] | Code-centric | S | MES | Function | | | οТ | 0 0 | | Virtual Machine (EF |
| | | ACES [90] | Code-centric | S | SHM | Function | | | 0 | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | | ō | | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | S | SHM | Library | | | Ō | 0 | 0 | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | | ot | 0 | O ⁴ | ARM PAC + MT |
| | U | Jif [261] | Code-centric | S+A | MES | Any | | | | 0 1 | | Ø |
| | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | | | 0 (| | Ø ⁵ |
| <i>t</i> : | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | | | | 0 (| | \varnothing^5 |
| Mutual Distrust | | Salus [226] | Data-centric | S | SHM | Function ¹ | | | | 0 | | Ø ⁵ |
|)isi | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | | | | 0 0 | | Page Table ² |
| ıl I | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | | | 0 | | | Page Table |
| ıtuc | | SOAAP [117] | Hybrid | S | SHM | Any | | | | 0 0 | O ⁴ | Ø |
| M_{l} | | libMPK [190] | Hybrid | S | SHM | Any | | | | 0 0 | | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | | | 0 | | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | | | 0 | | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | | | | | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | | _ | • | | | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | | | | | | Protection Keys |
| | | | Code-centric | S | SHM | μ Library | | | | \circ | | Ø |
| | | | 0 1 | 0 | | A | | | | | | |
| trac | ction | specialized | Code-centric | S | MHZ | Anv | | | | | 0 | Ø |
| | | specialized | Code-centric | s nant | ts cro | Anv ASS-COMP | 121 | tm | 10 | nt |)) | Ø ⁵ |
| | | | Code-centric | nent | ts cro | ss-comp | ar | tm | ne | nt |))) | Ø ⁵ Virtual Machine (E |
| and | boxii | ng untrusted | Implen | | | | | | | | 0 | Ø ⁵ Virtual Machine (E ARM PAC + M7 |
| and | boxii | ng untrusted | Implen | | | ss-comp | | | | |)))) | Virtual Machine (E ARM PAC + MT |
| and | boxii | | Implen comm | nuni | catio | n with f | un | cti | | | 0 | Virtual Machine (E ARM PAC + M7 |
| and | boxii | ng untrusted as libraries) | Implen comm | nuni | catio | | un | cti | | | | Virtual Machine (F ARM PAC + M7 |
| and e (s | boxii | ng untrusted as libraries) | Implen comm | nuni | catio | n with for the second terms of the second term | un | cti ⁄e | 01 | | | Virtual Machine (E ARM PAC + M7 |
| and e (s | boxii uch | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] | Implen comm | nuni Ils, | catio whicl | n with for his intui | un | cti ⁄e | 10 | 1 | | Virtual Machine (E ARM PAC + M7 Ø ⁵ Ø ⁵ Ø ⁵ Ø ⁵ Ø ⁵ Ø ⁵ |
| and e (s | boxii uch | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] | Implen comm | nuni | catio | n with for is intui | un tiv | cti ⁄e | 10 | 1 | | Virtual Machine (E ARM PAC + M7 Ø |
| and | boxii uch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] | Implen comm | nuni Ils, | catio whicl | n with for his intui | un | cti ⁄e | 10 | 1 |) | Virtual Machine (E ARM PAC + M7 Ø |
| and e (s | boxii uch | ng untrusted as libraries) Comp. Retrot [105] Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] | Implen comn ca Code-centric Hybrid Hybrid | nuni Ils, | catio whicl | n with for is intuing the function function function function | un tiv | cti ⁄e | 10 | |)) | Virtual Machine (F ARM PAC + MT Ø'S Ø'S Ø'S Ø'S Ø'S Ø'S Ø'S OCHERI |
| and e (s | boxii uch a | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133] Builds on sl | Implen comn ca Code-centric Hybrid Hybrid | nuni Ils, | catio whicl | n with for is intuing the function function function function | un tiv | cti ⁄e | 10 | 1 |))) | Virtual Machine (F ARM PAC + MT Ø's Ø's Ø's Ø's Ø's Ø's CHERI |
| and e (s | uch o | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Builds on sl | Implen communication communica | nuni Ills, v | which | r with for is intuing Function Function Function Speed | un tiv | cti ⁄e | 10 | | | Virtual Machine (E ARM PAC + MT Ø's Ø's Ø's Ø's Ø's Ø's Ø's CHERI Ø's SFI |
| and e (s | UU+K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133 XFI/LXFI [107, 172] Nexen [221] Builds on sl | Implen communication communica | nuni Ills, v | which | r with for is intuing Function Function Function Speed | un tiv | cti ⁄e | 10 | | | Virtual Machine (E ARM PAC + MT |
| and e (s | uch o | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133 XFI/LXFI [107, 172] Nexen [221] Shreds [87] | Implen communication communica | emo | whick whick which which was seen to the whole with the whole with the whole which will be a seen to the whol | Function Function Function Function Speed Sharing | tiv | cti ⁄e | 10 | | | Virtual Machine (E ARM PAC + MT Ø |
| and e (s | UU+K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133 XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric | emo | whick MES SHM ry to data S | Function Function Function Function Speed Sharing Function | un tiv | cti ⁄e | 10 | | | Virtual Machine (E ARM PAC + MT |
| e (s | UU+K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / Wedge [70] CompartOS [55] LVDs / KSplit [133] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric Code-centric | emo | whick whick MES SHM ry to data S MES MES | Function Function Function Function Function Function Function Function Function Function Function Function | un | cti /e | 10 | | | Virtual Machine (F ARM PAC + MT Ø'S Ø'S Ø'S Ø'S Ø'S Ø'S O'S CHERI Ø'S SFI Page Table'S |
| e (s | UU+KKK | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133 XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric | emo | mes mes mes mes mes mes | Function Function Function Function Function Function Function Function Function Function Function Function Function Any | tiv | cti /e | 10 | | | Virtual Machine (I ARM PAC + M' Ø' Ø' Ø' Ø' Ø' Ø' Ø' Ø' CHERI Ø' SFI Page Table' Page Table' Page Table Ø' |
| e (s | UU+K | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | emo | mes mes mes mes mes mes mes | Function Function Function Function Function Function Function Any Function Function Function Function Function Function Function Function Function Function Function Function Function | tiv | cti re | 10 | | | Virtual Machine (I ARM PAC + MT Ø's Ø's Ø's Ø's Ø's Ø's Ø's CHERI Ø's SFI Page Table's Page Table Page Table Ø's Page Table's |
| e (s | UU+KKK | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | emo | mes mes mes mes mes mes mes mes | Function Function Function Function Function Function Function Any Function Function Function Function Function Function Function Function Function Function Function Function Function | un | cti /e | 10 | | | Virtual Machine (FARM PAC + MTAC) |
| e (s | UU+KKK | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] DataShield [77] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Hybrid | emo | mes mes mes mes mes mes mes mes mes | Function Function Function Function Function Function Function Any Function Function Any Function Function Any Function Any Function Any Function Any | un | cti re | 10 | | | Virtual Machine (E ARM PAC + MT |
| e (s | UU+KKK | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Code-centric Implen comm ca Code-centric Hybrid Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | emo | mes mes mes mes mes mes mes mes | Function Function Function Function Function Function Function Any Function Function Function Function Function Function Function Function Function Function Function Function Function | un | cti /e | 10 | | | Virtual Machine (E ARM PAC + MT |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent).

The abstraction could plug onto any intra-AS mechanism, though the paper or documentation claims reliance on a particular one.

(1) Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Renderer, USENIX Security 2020

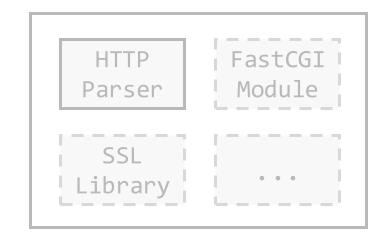
| Class | Target | Abstraction | Subject Selection | Sen | nantics | Abstraction | P | rop | erties | s | Interface | Design Bound |
|-----------------|----------------|---|--|---------------------------|--|---|-----------------|--------------|--------|---|----------------|---|
| Class | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A | R | Safety | to Mechanism |
| | | Virtines [242] | Code-centric | S | MES | Function | | | | 0 | 0 | Virtual Machine (EPT) |
| | | ACES [90] | Code-centric | S | SHM | Function | | | | 0 | 0 | Ø ⁵ |
| | | SeCage [171] | Code-centric | S | SHM | Function | | | | | 0 | Ø ⁵ |
| | | HODOR [124] | Code-centric | S | SHM | Library | | • | Ŏ | Ŏ | 0 | Ø |
| | | CAPACITY [105] | Code-centric | S | SHM | Any | | | | | 04 | ARM PAC + MTE |
| | U | Jif [261] | Code-centric | S+A | MES | Any | | | | 0 | | Ø |
| | | Arbiter [241] | Data-centric | S | SHM | Function ¹ | | | | 0 | | Ø ⁵ |
| <i>t</i> | | Secure Memory Views (SMVs) [128] | Data-centric | S | SHM | Function ¹ | | | 0 | 0 | 0 | Ø ⁵ |
| rus | | Salus [226] | Data-centric | S | SHM | Function ¹ | | | 0 | 0 | | Ø ⁵ |
|)ist | | Light-Weight Contexts (LwCs) [167] | Hybrid | S | SHM | Function ¹ | | | 0 | 0 | | Page Table ² |
| ון ד | | POSIX Processes (and earlier instances) [93] | Hybrid | Any | Any | Any | | | | 0 | 0 | Page Table |
| Mutual Distrust | | SOAAP [117] | Hybrid | S | SHM | Any | | | 0 | 0 | O ⁴ | Ø |
| M_{ν} | | libMPK [190] | Hybrid | S | SHM | Any | | | | | 0 | Protection Keys |
| | | CheriOS [108] | Code-centric | Any | Any | U/K-component | | | 0 | 0 | | CHERI |
| | | Microkernel Servers [106], [] | Code-centric | Any | MES | U/K-component | | | | 0 | | Ø ⁵ |
| | | Mutable Protection Domains (MPDs) [191, 192] | Code-centric | S | SHM | U/K-component | | | | | | Ø ⁵ |
| | U+K | RedLeaf [184] | Code-centric | S | SHM | U/K-component | | | | 0 | | Safe Languages |
| | | CubicleOS [211] | Code-centric | S | SHM | μ Library | | | 0 | 0 | | Protection Keys |
| | | | Code-centric | S | SHM | μ Library | | | 0 | 0 | 0 | Ø |
| | | | 0 1 | | | | | | | | | |
| rac | tion | specialized | Code-centric | S | SHM | Anv | | <u>.</u> | | | | Ø Ø 5 |
| nd | boxii | ng untrusted | • | | | ss-comp | | | | | | |
| nd | boxii | ng untrusted | • | | | ss-comp | | | | | | Ø ⁵ Virtual Machine (EPT) ARM PAC + MTE |
| nd | boxii | • | comn | nun | icatio | n with f | un | ct | | | | Ø ^S Virtual Machine (EPT) ARM PAC + MTE Ø ^S |
| nd | boxii uch | ng untrusted | comn | nun | icatio | • | un | ct | | | | Virtual Machine (EPT) ARM PAC + MTE |
| ind e (s | boxii | ng untrusted as libraries) | comn | nun | icatio | n with f | un | ct | | | | Virtual Machine (EPT) ARM PAC + MTE |
| ind e (s | boxii uch | ng untrusted as libraries) | comn | nun | icatio | n with f | un | ct e | io | | | Virtual Machine (EPT) ARM PAC + MTE |
| nd | boxii uch | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] | comn ca | nuni Ills, | icatio whicl | n with f | un tiv | ct e | io | n | | Virtual Machine (EPT) ARM PAC + MTE |
| ind e (s | boxii uch | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] | Code-centric Hybrid Hybrid | nuni | whicl | n with f | un tiv | ct e | io | | | Virtual Machine (EPT) ARM PAC + MTE Ø ⁵ Ø ⁵ Ø ⁵ Ø ⁵ Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø |
| ind e (s | uch o | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [135, XFI/LXFI [107, 172]] Builds on sh | comn ca Code-centric Hybrid Hybrid | nuni Ills, | which which which we have a second with the se | n with fin is intuing Function Function Engage | un tiv | ct e | io | | | Virtual Machine (EPT) ARM PAC + MTE |
| ind e (s | boxii uch i | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [135, XFI/LXFI [107, 172]] Builds on sh | comn ca Code-centric Hybrid Hybrid | nuni Ills, | which which which we have a second with the se | n with fin is intuing Function Function Engage | un tiv | ct e | io | | | Virtual Machine (EPT) ARM PAC + MTE Ø ⁵ Ø ⁵ Ø ⁵ Ø ⁵ Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø |
| ind e (s | uch o | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Builds on sh | comn ca Code-centric Hybrid Hybrid | nuni Ills, | which which which we have a second with the se | n with fin is intuing Function Function Engage | un tiv | ct e | io | | | Virtual Machine (EPT) ARM PAC + MTE |
| ind e (s | U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133] XFI/LXFI [107, 172] Nexen [221] Up cross-co | comn ca Code-centric Hybrid Hybrid | nuni Ills, | which which which we have a second with the se | n with fin is intuing Function Function Engage | un tiv | ct e c | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |
| sandpox (S | U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Nexen [221] Shreds [87] Builds on shup cross-co | comn Code-centric Hybrid Hybrid nared me mpartme | nuni ills, s emo | which which we see the control of th | r with fin is intuing Function Function Enforces | un tiv | ct | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |
| sandpox (S | U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133 XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] | Code-centric Hybrid Hybrid nared me | nuni ills, s emo | which which wes shm ry to data s | Function Function Function European European Enforce Sharing Function | tiv es | ct | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |
| sandpox (S | U+K K HV | ng untrusted as libraries) Comparted [111] Google Sandboxed API (SAPI) [22] RLBox / \(\mu\)SWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] | Code-centric Hybrid Hybrid Hybrid Mared me | emo | which which MES SHM ry to data S MES MES | Function Function Function Function Eurocion Eurocion Function Function Function Function Function Function | un tiv es | ct | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |
| sandpox (S | U U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / \(\mu\)SWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] | Comn Comn Code-centric Hybrid Hybrid The code-centric Code-centric Code-centric Code-centric | emo | mes shm ry to data s | Function Function Function Function Function Function Function Function Function Function Function Any | un tiv es | ct | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |
| sandpox (S | U U+K K HV | ng untrusted as libraries) Comparation [132] Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Comn Comn Code-centric Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric | emo | mes mes mes mes mes mes mes | Function Function Function Function Function Function Function Function Function Any Function | un tiv es | ct | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |
| sandpox Sandpox | U U+K K HV | ng untrusted as libraries) Enclosure [111] Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [135] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Comn Comn Code-centric Hybrid Hybrid Code-centric Code-centric Code-centric Code-centric Code-centric Code-centric | emo | mes mes mes mes mes mes mes mes mes | Function Function Function Function Function Function Function Function Function Function Function Function Function Function Function Function Function Function | un tiv es | ct | io | | denti | Virtual Machine (EPT) ARM PAC + MTE |

Inherited from thread-like semantics, ² from process-like semantics, ³ from the Nested Kernel, ⁴ The PDM does (to a certain extent).

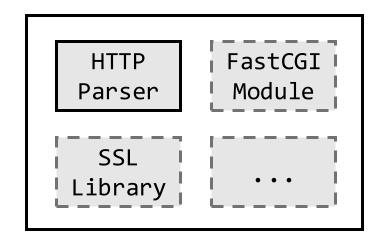
| Virtimes 1241 Code-centric Call. Ras Safety to Mechanism | | Target | Abstraction | Subject Selection | Ser | nantics | Abstraction | 1 | Prop | erties | Interface | Design Bound |
|--|----------|--------------|--|---|---------------|---|--|-----|------|------------------|-----------|---|
| ACES per ACES Per Code-centric S SIM Function O O O O O O O O O | Class | U/K/HV | Abstraction | Subject Selection | CALL | ASSIGN | Granularity | С | I | A I | Safety | to Mechanism |
| Secage (191) HODOR (134) Code-centric S SIMM Function U AFRICAPACITY (105) Code-centric S SIMM Library U AFRICAPACITY (105) Code-centric S SIMM Any Any O O O O O O O O O O O O O O O O O O O | | | Virtines [242] | Code-centric | S | MES | Function | | | 0 (| | Virtual Machine (EPT |
| HODOR 1284 Code-centric S SHM Library O O O O AMP PACE HITE Jif 1881 Secure Memory Views (SMVs) 1281 Data-centric S SHM Function O O O O O O O O O O O O O O O O O O O | | | ACES [90] | Code-centric | S | SHM | Function | | | 0 0 | | Ø ⁵ |
| HODOR 1284 Code-centric S SHM Library O O O O AMP PACE HITE Jif 1881 Secure Memory Views (SMVs) 1281 Data-centric S SHM Function O O O O O O O O O O O O O O O O O O O | | | SeCage [171] | Code-centric | S | SHM | Function | | | 0 0 | | Ø ⁵ |
| U Hr Part Arbiter Part Data-centric S S S S S Function S S S S S S S S S | | | | Code-centric | | SHM | Library | | | 0 | | Ø |
| Arbiter [141] Secure Memory Views (SMVs) [128] Data-centric Secure M | | | CAPACITY [105] | Code-centric | S | SHM | Any | | | 0 0 | 04 | ARM PAC + MTE |
| Arbiter [241] Secure Memory Views (SMVs) [128] Data-centric S SHM Function Function Data-centric S SHM Function Data-ce | | II | | Code-centric | S+A | MES | Any | | | 0 | | Ø |
| Secure Memory Views (SMVs) [128] Salus 128] Salus 128] Data-centric S SHM Function O O O O O O O O O O O O O O O O O O O | | C | | Data-centric | | SHM | | _ | | 0 | | Ø ⁵ |
| Salus 1290 Light-Weight Contexts (LwCs) 11671 POSIX Processes (and earlier instances) 1931 Hybrid S SHM Function | _ | | | Data-centric | | SHM | Function ¹ | - | | 0 0 | | Ø ⁵ |
| Light-Weight Contexts (LwCs) [167] Hybrid S SHM Punction POSIX Processes (and earlier instances) [93] Hybrid Any Any Any Any Any Any Any Any Any Any | Distrust | | * \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ | Data-centric | | SHM | Function ¹ | _ | | 0 | | Ø ⁵ |
| POSIX Processes (and earlier instances) [93] Hybrid Any Any Any Any Any Any BOAAP [117] Hybrid S SHM Any O O O Page Table SOAAP [117] Hybrid S SHM Any O O O O O O O O O O O O O O O O O O O | isti | | | Hybrid | | SHM | Function ¹ | - | - | | | Page Table ² |
| SOAAP [117] Hybrid S SIM Any O O O O O O O O O O O O O O O O O O O | q_1 | | | J | | | | _ | | 0 | | |
| TibMPK [199] | Mutual | | , , , , , , , , , , , , , , , , , , , | | | - | | - | | 0 | 04 | |
| Cherios 1108 Whiterokernel Servers 1106], [] Code-centric Any Any U/K-component O O O O | Mu. | | | | | | | _ | _ | ŏ l d | | |
| Microkernel Servers (106), [] Middable Protection Domains (MPDs) (191, 192) Code-centric S SHM U/K-component O O O O O O O O O O O O O O O O O O O | ~ | | | | | | V | _ | | ŏ | | - V |
| U+K RedLeaf [184] Code-centric S SHM U/K-component U/K-compo | | | | Code-centric | | | . A | - | _ | 0 | | Ø ⁵ |
| U+K RedLeaf [184] Code-centric S SHM U/K-component Discrete Discre | | | | | | | 1 | - | | | | Ø ⁵ |
| CubicleOS (211) Code-centric S SHM µLibrary | | U+K | | | | | A. | - | | | | Safe Languages |
| Ction specialized aboxing untrusted such as libraries) U Enclosure [111] Code-centric S SHM Function Calls, which is intuitive Enforceme Comparto SHM Function Calls, which is intuitive Enforceme Comparto Compart | | | CubicleOS [211] | Code-centric | S | SHM | μLibrary | _ | | 0 | | Protection Keys |
| Implements cross-compartment communication with function calls, which is intuitive enforcement. | | | | Code-centric | S | SHM | μLibrary | | | 0 | | Ø |
| Implements cross-compartment communication with function calls, which is intuitive U | ract | tion | cnacializad | Code-centric | S | SHM | Any | | | | | Ø |
| Calls, which is intuitive Enclosure [111] Google Sandboxed API (SAPI) [22] Code-centric RLBox / μswitch [181, 195] Wedge [70] U+κ CompartOS [55] κ LVDs / KSplit [133 XFI/LXFI [107, 172] HV Nexen [221] HV Nexen [221] Privman [147] Privman [147] Privtrans [73] Swift [89] U Glamdring [165] V Glamdring [165] PtrSplit / Program Mandering [169, 170] Code-centric S MES Function O Page Table² Pustion Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Swift [89] Code-centric S MES Function O Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Page Table² Swift [89] Code-centric S MES Function O O O O O O O O O O O O O O O O O O O | | | | • | | | • | | | | | ADM DAC . MTE |
| RLBox / µSWITCH [181, 195] | <u>`</u> | _ | ') | l ca | ills, | whicl | n is intui | tiv | /e | | | enforceme |
| Wedge [70] U+K CompartOS [55] K LVDs / KSplit [133] AFI/LXFI [107, 172] HV Nexen [221] U Shreds [87] Privman [147] Privtrans [73] Swift [89] U Glamdring [165] PtrSplit / Program Mandering [169, 170] DataShield [77] Hybrid S Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Enforces confidentiality and int Page Table ³ MES Function Page Table ² Privtrans [73] Swift [89] Code-centric S MES Function Glamdring [165] PtrSplit / Program Mandering [169, 170] Code-centric S MES Function Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any Hybrid S SHM Any | | U | | | | | | | | | | |
| U+K CompartOS 55 | yoq | U | Google Sandboxed API (SAPI) [22] | Code-centric | | MES | Function | | | | | |
| Builds on shared memory to Enforces confidentiality and Interest of the Intere | ndbox | U | Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] | Code-centric Hybrid | | MES | Function | | | | | Ø |
| U Shreds [87] | Sandbox | U | Google Sandboxed API (SAPI) [22] RLBox / | Code-centric Hybrid | | MES | Function Function | • | • | 0 (| | α ⁵ |
| V Shreds [87] | Sandbox | | Google Sandboxed API (SAPI) [22] RLBox / \(\mu \)SWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133] Ruilds on sk | Code-centric Hybrid Hybrid | S | MES SHM | Function Function | • | • | 0 (| | α ⁵ |
| Privman [147] | Sandbox | K | Google Sandboxed API (SAPI) [22] RLBox / \(\mu \)SWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Builds on sh | Code-centric Hybrid Hybrid nared me | s s emo | shm ry to | Function Function Europian Enforc | • | • | 0 (| | iality and in |
| Privtrans [73] Code-centric S MES Function O O O Page Table ² | Sandbox | K HV | Google Sandboxed API (SAPI) [22] RLBox / µSWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172] Nexen [221] Builds on shape of the compart o | Code-centric Hybrid Hybrid nared me | s s emo | shm ry to | Function Function Europian Enforc | • | C | 0 (| | iality and in |
| Swift [89] Code-centric S+A MES Any | Sandbox | K HV | Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133] XFI/LXFI [107, 172] Nexen [221] Shreds [87] Google Sandboxed API (SAPI) [22] Builds on shape of the same of | Hybrid Hybrid Mared me mpartme | emo ent | ry to | Function Function Emption Emption Emption Sharing | es | C | 0 (| ident | iality and in |
| Datashield [//] Shiw Ally O O O Bounds Checking | | K HV | Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133 XFI/LXFI [107, 172] Nexen [221] Shreds [87] Privman [147] | Hybrid Hybrid Hybrid nared me mpartme | emo | ry to | Function Function Emerical Enforce Sharing Function | es | C | 0 (| ident | Page Table S |
| Datashield [//] Shiw Ally O O O Bounds Checking | | K HV | Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Nexen [221] Shreds [87] Privman [147] Privtrans [73] | Hybrid Hybrid Hybrid Mared me mpartme Code-centric Code-centric | emo ent | ry to | Function Function Eurorition Eurorition Enforce Sharing Function Function Function | es | C | 0 (| ident | Page Table ³ Page Table ³ Page Table ² Page Table ² |
| Datashield [//] Shiri Ally O O O Doulld's Checking | | K HV U | Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] | Mybrid Hybrid Hybrid Mared me mpartme Code-centric Code-centric Code-centric | emo ent | ry to data: MES MES MES MES | Function Function Eurorition Eurorition Eurorition Eurorition Eurorition Function Function Function Any | es | C | 0 (| ident | Page Table ³ Page Table ³ Page Table ² Page Table ² Page Table ² |
| Datashield [//] Shiri Ally O O O Doulld's Checking | | K HV U | Google Sandboxed API (SAPI) [22] RLBox / µswitch [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] | Mybrid Hybrid Hybrid Hybrid Mared me mpartme Code-centric Code-centric Code-centric Code-centric | emo ent | ry to data: MES MES MES MES MES MES | Function Function Eurorical Enforce Sharing Function Function Any Function Function Function Function Function | es | • C(| 0 (| ident | Page Table ³ Page Table ³ Page Table ² Page Table ² Page Table ² Ø Ø ³ |
| | | K HV U | Google Sandboxed API (SAPI) [22] RLBox / \(\mu \)SWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Code-centric Hybrid Hybrid Hybrid Mared me mpartme Code-centric Code-centric Code-centric Code-centric Code-centric | emo ent | mes shm ry to data: Mes Mes Mes Mes Mes Mes Mes Mes Mes Mes | Function Function Eurorical Enforc Sharing Function Function Any Function Function Function Function Function Function | es | • C(| 0 (| ident | Page Table ³ Page Table ³ Page Table ² Page Table ² Page Table ² Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø |
| K Nested Kernel [94] Code-centric S SHM Function Page Table | | K HV U | Google Sandboxed API (SAPI) [22] RLBox / \(\mu \)SWITCH [181, 195] Wedge [70] CompartOS [55] LVDs / KSplit [133, XFI/LXFI [107, 172]] Nexen [221] Shreds [87] Privman [147] Privtrans [73] Swift [89] Glamdring [165] PtrSplit / Program Mandering [169, 170] | Code-centric Hybrid Hybrid Hybrid Mared me mpartme Code-centric Code-centric Code-centric Code-centric Code-centric | emo ent | mes shm ry to data: Mes Mes Mes Mes Mes Mes Mes Mes Mes Mes | Function Function Eurorical Enforc Sharing Function Function Any Function Function Function Function Function Function | es | • C(| 0 (| ident | Page Table ³ Page Table ³ Page Table ² Page Table ² Page Table ² Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø |

¹³³

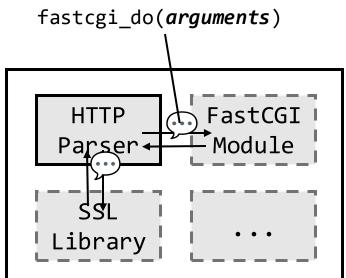
Hardening compartment interfaces is key to obtaining strong security properties with compartmentalization



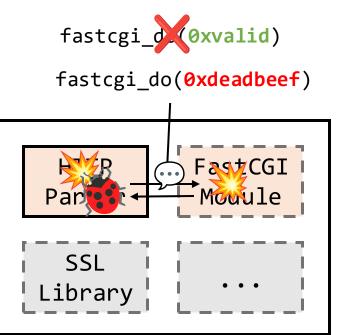
Hardening compartment interfaces is key to obtaining strong security properties with compartmentalization



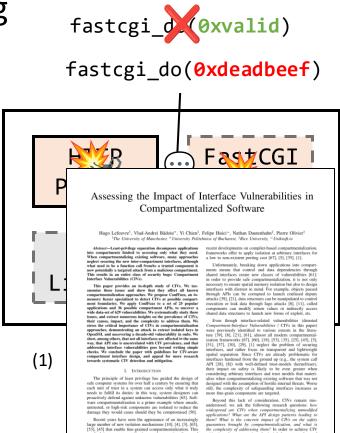
Hardening compartment interfaces is key to obtaining strong security properties with compartmentalization



Hardening compartment interfaces is key to obtaining strong security properties with compartmentalization



Hardening compartment interfaces is key to obtaining strong security properties with compartmentalization



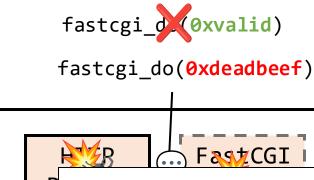
Open Problem in Compartmentalization

| Class | Target | Abstraction | Subject Selection | Coll. | Alst Sign | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collection | Collecti

es is key to obtaining npartmentalization

iplemented our ed to ick surface.

Most abstractions consider the **hardening of compartment interfaces** an orthogonal problem.



Assessing the Impact of Interface Vulnerabilities in Compartmentalized Software

Abstract—Least-privilege separation decomposes application compartments limited to accessing only what they ne When compartmentalizing existing software, many approach neglect securing the new inter-compartment interfaces, although the second of the security of the security of the second of the second of the security of the second of the security of the security bags: Compartment interface Waternalities (CIV).

This paper provides an indepth study of CTN. We taxmonitie them is some and show that they affect all knows compartmentalization approaches. We propose Conflict, as inmore than the confliction of the confliction of the confliction and the possible of the confliction and the possible of the compartment AFIs is necessary applications and 30 possible compartment AFIs is necessary applications and 30 possible compartment AFIs is necessary their cases, impact, and the complexity to address them. We there is a superconfliction of the confliction of the confliction proposedies, demonstrating an attack to extent industrial type in such. We opposed, and uneverting a decide-old vulnerability in such. We approached, demonstrating an attack to extent industrial type in such. We approached, the confliction of the confliction of the confliction of the specific of the confliction of the confliction of the confliction of the confliction of the confliction of the confliction of the specific of the confliction of the confliction of the confliction of the confliction of the confliction of the confliction of the confliction of the specific of the confliction of the confliction of the confliction of the specific of the confliction of the confliction of the confliction of the confliction of the specific of the confliction of the confliction of the confliction of the specific of the confliction of the confliction of the confliction of the specific of the confliction of t

I. INTRODUCTION

The principle of least privilege has guided the design sale computer systems for over half a century by ensuring the each unit of trust in a system can access only what it medical to fallill its duties: in this way, system designers or proactively defend against unknown vulnerabilities [65]. Soft was comparamentalization is a grime example where unsafts of the components are isolated to reduce the untrusted, or high-tisk components are isolated to reduce the datament the would cause should they be commonised [50].

Recent years have ease the gaycamese of an increasing large number of new isolation mechanisms [10, 14, 13, 16, [53], [45] that enable fine-grained compartmentalization. The resulted in compartmentalization works targeting finer a finer granularities, such as liberaise [67], [60], [91, [42], [15, 16], [16], [17], [18

Network and Distributed System Security (NDSS) Symp 27 February - 3 March 2023, San Diego, CA, USA ISBN 1-89150-83-5 https://dx.doi.org/10.14722/rdss.2023.24117 recent developments on compiler-based compartmentalization, frameworks offer to apply isolation at arbitrary interfaces for a low to non-existent porting cost [67], [5], [35], [1].

Unfortunately, breaking down applications into comparments inscens that control and data dependencies fromps, barred interfaces create new clauses of vulnerabilities (61) in in order to provide sale compartmentalization, it is not only necessary to ensure spatial memory isolation but also to design interfaces with directivat in mind. For example, objects passed through. APIs can be corrupted to launch confused deputy artacks (59), [21], data structures can be manipulated to control execution or leak data through lago attacks [31, [11], called components can modify entire values or indirectly accordces.

Even though interface-celled vulnerabilities (denoted Comparament Interface Vulnerabilities (CIVs) in this paper were previously identified to various actions in the literature proviously identified to various actions in the literature of the lit

Beyond this lack of consideration, CIVs remain mis understood, we sak the following research questions: how widespread are CIVs when compartmentalizing amoughing applications? What are the API design patterns remodifie, when What is the concerts impact of CIVs on the side? when the complexity of addressing them, In order to achieve the complexity of addressing them, In order to achieve the complexity of addressing them, In order to achieve for formalize and quantify the problem.

This paper provides an in-depth study of CIVs. We taxon minden from the above the transverse and systematize existin efforts to address them, highlighting categories that now the control of the control

Open Problem in Compartmentalization

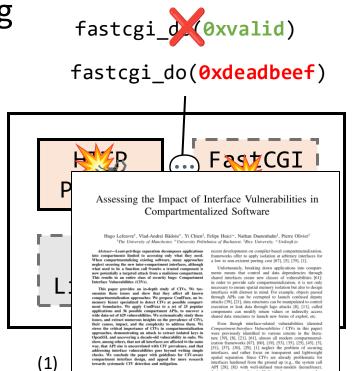
Abstra

| Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Control | Contro

Most abstractions consider the **hardening of compartment interfaces** an orthogonal problem.

We stress that abstractions can and should facilitate the hardening of compartment interfaces.

How can abstractions that facilitate the implementation of secure domain interfaces?



Open Problem in Compartmentalization

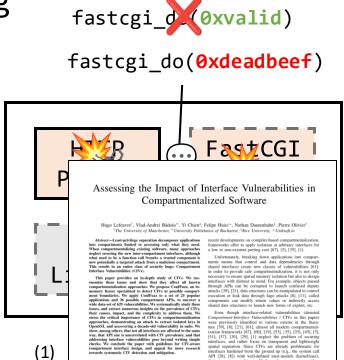
Abstra Hardening strong sec Example: a boundarie ed to ick surface. communic

s is key to obtaining npartmentalization iplemented our

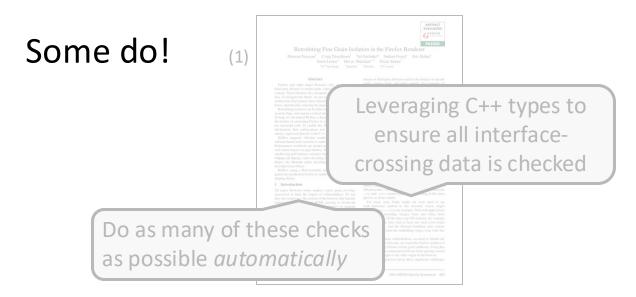
Most abstractions consider the hardening of compartment interfaces an orthogonal problem.

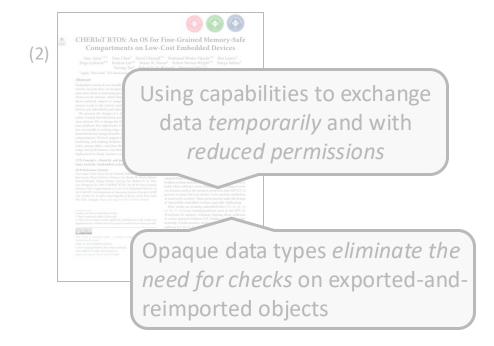
We stress that abstractions can and should facilitate the hardening of compartment interfaces.

> How can abstractions that facilitate the implementation of secure domain interfaces?



Most abstractions consider the **hardening of compartment interfaces** an orthogonal problem.



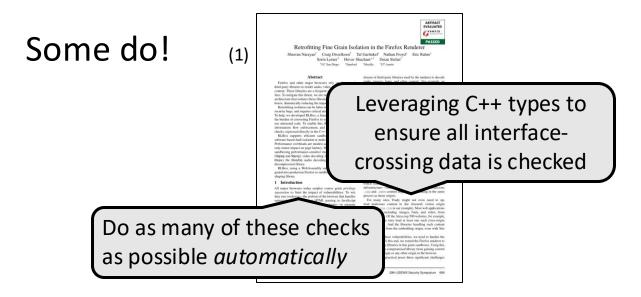


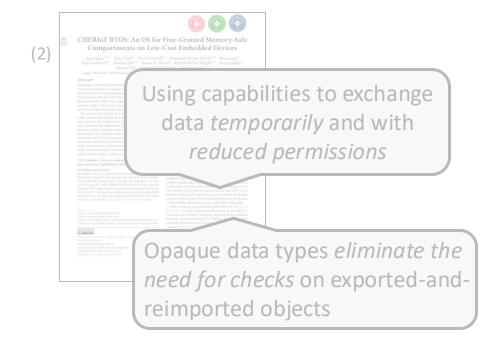
More work needed in that direction.

⁽²⁾ Amar et al., CHERIOT RTOS: An OS for Fine-Grained Memory-Safe Compartments on Low-Cost Embedded Devices, SOSP 2025

⁽¹⁾ Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Renderer, USENIX Security 2020

Most abstractions consider the **hardening of compartment interfaces** an orthogonal problem.

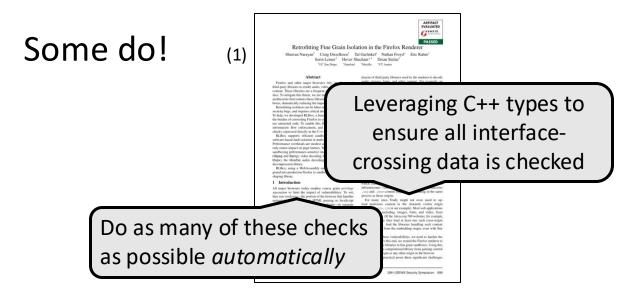


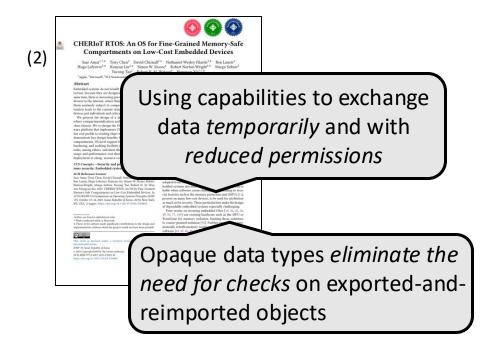


More work needed in that direction.

⁽²⁾ Amar et al., CHERIOT RTOS: An OS for Fine-Grained Memory-Safe Compartments on Low-Cost Embedded Devices, SOSP 2025

Most abstractions consider the **hardening of compartment interfaces** an orthogonal problem.





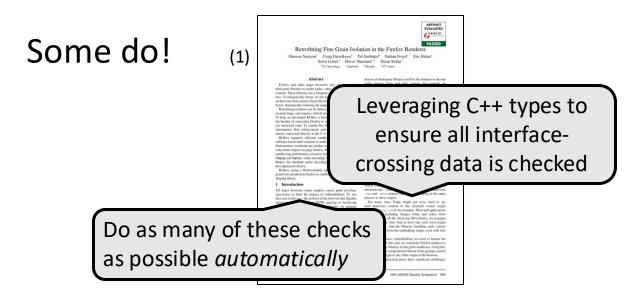
More work needed in that direction.

⁽²⁾ Amar et al., CHERIOT RTOS: An OS for Fine-Grained Memory-Safe Compartments on Low-Cost Embedded Devices, SOSP 2025

⁽¹⁾ Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Renderer, USENIX Security 2020

Open Problem in Compartmentalization Abstractions

Most abstractions consider the **hardening of compartment interfaces** an orthogonal problem.



(2)

CHERIOT RTOS: An OS for Fine-Grained Memory-Safe Compartments on Low-Cost Embedded Devices

Say Aram's Tory Cheric Justice Cherical Street Cherical Stree

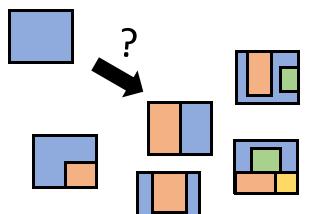
More work needed in that direction.

⁽²⁾ Amar et al., CHERIOT RTOS: An OS for Fine-Grained Memory-Safe Compartments on Low-Cost Embedded Devices, SOSP 2025

⁽¹⁾ Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Renderer, USENIX Security 2020

Bigger Picture of Compartmentalization

Propose to view compartmentalization as 3 problems:



- 1. How to determine the right policy to enforce?
 - Done with a policy definition method
- 2. How to integrate the notion of compartmentalization policies in software / programming models / idioms?
 - Done with a compartmentalization abstraction
- 3. How to enforce policies at runtime?
 - Done with a compartmentalization mechanism



How to enforce policies?

Finally, we must **enforce the policy at runtime**.

This is achieved with an **enforcement mechanism**.

How to enforce policies?

Finally, we must **enforce the policy at runtime**.

This is achieved with an **enforcement mechanism**.

Page tables (via the MMU) are the historical enforcement mechanism.

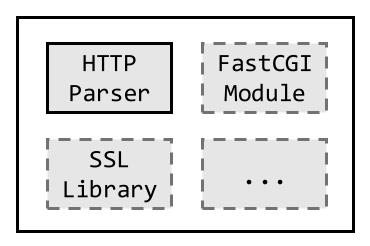
How to enforce policies?

Finally, we must enforce the policy at runtime.

This is achieved with an enforcement mechanism.

Page tables (via the MMU) are the historical enforcement mechanism.

Example from earlier: assume we implemented the sandbox with processes.



(Fictive monolithic program)

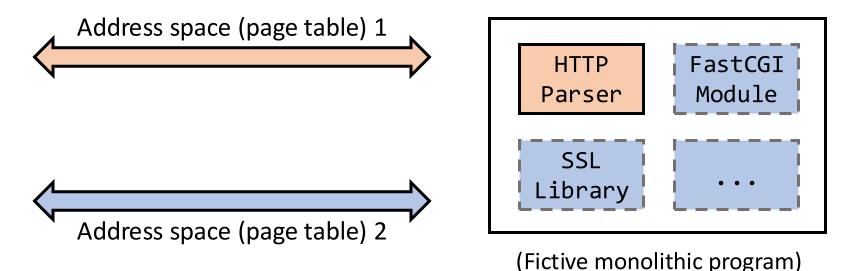
How to enforce policies?

Finally, we must enforce the policy at runtime.

This is achieved with an enforcement mechanism.

Page tables (via the MMU) are the historical enforcement mechanism.

Example from earlier: assume we implemented the sandbox with processes.



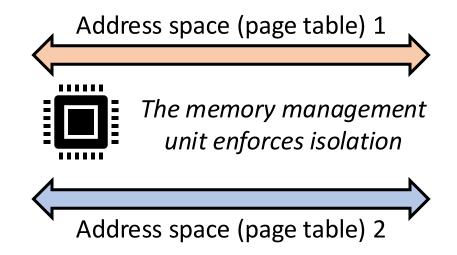
How to enforce policies?

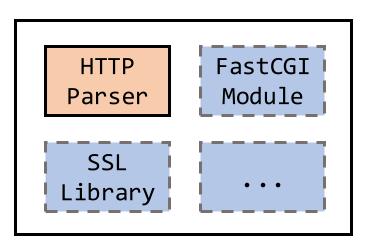
Finally, we must **enforce the policy at runtime**.

This is achieved with an enforcement mechanism.

Page tables (via the MMU) are the historical enforcement mechanism.

Example from earlier: assume we implemented the sandbox with processes.





(Fictive monolithic program)

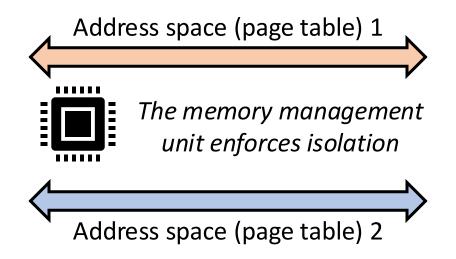
How to enforce policies?

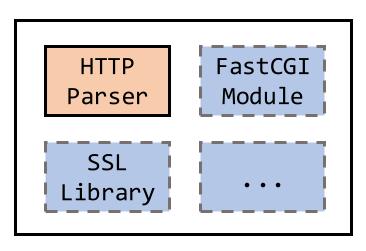
Finally, we must **enforce the policy at runtime**.

This is achieved with an enforcement mechanism.

Page tables (via the MMU) are the historical enforcement mechanism.

Example from earlier: assume we implemented the sandbox with processes.





(Fictive monolithic program)

This is still the most common way to do it today.

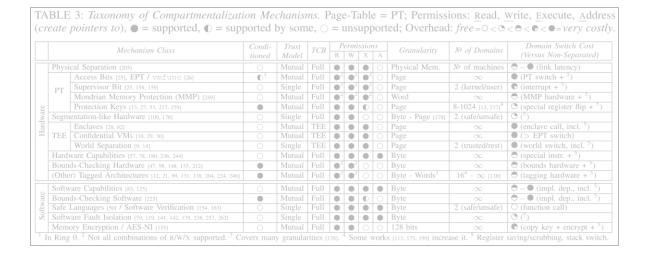
Enforcement Mechanisms

Enforcement mechanisms (for compartmentalization) are also a hot area:

Maximize performance

Minimize hardware cost and complexity

Enable for stronger security properties





(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

(1)

Enforcement Mechanisms

Enforcement mechanisms (for compartmentalization) are also a hot area:

Maximize performance

Minimize hardware cost and complexity

Enable for stronger security properties

| | | | Condi- | Trust | man | Pe | ermi | ssio | ıs | | 14 CD 1 | Domain Switch Cost |
|-------|-------|---|----------------|--------|------|-----------------|-------------|-------------|----|---------------------------|-------------------------------|---|
| | | Mechanism Class | tioned | Model | TCB | R | W | X | Α | Granularity | № of Domains | (Versus Non-Separated) |
| П | Physi | cal Separation [205] | 0 | Mutual | Full | • | • | • | 0 | Physical Mem. | № of machines | → – ● (link latency) |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}_1 | Mutual | Full | \bullet^2 | \bullet^2 | \bullet^2 | 0 | Page | ∞ | ● (PT switch + ⁵) |
| | РТ | Supervisor Bit [25, 158, 159] | 0 | Single | Full | $ \bullet^2 $ | \bullet^2 | \bullet^2 | 0 | Page | 2 (kernel/user) | (interrupt + ⁵) |
| | • • | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | $ \bullet^2 $ | \bullet^2 | lacksquare | 0 | Word | ∞ | (MMP hardware + 5) |
| are | | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | • | • | • | 0 | Page | 8-1024 [13, 217] ⁴ | (special register flip + 3) |
| | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | • | • | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | (3) |
| Hardw | | Enclaves [28, 92] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (enclave call, incl. 5) |
| Ξ | TEE | | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | ● (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | • | • | • | 0 | Page | 2 (trusted/rest) | (world switch, incl. ⁵) |
| [| Hardy | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | (special instr. + 5) |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | \bullet^2 | \bullet^2 | 0 | 0 | Byte | ∞ | (bounds hardware + 5) |
| | (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | \bullet^2 | \bullet^2 | 0 | 0 | Byte - Words ³ | 16 ⁴ - ∞ [138] | (tagging hardware + 5) |
| | Softw | vare Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| /are | | ds-Checking Software [225] | • | Mutual | Full | • | • | 0 | 0 | Byte | ∞ | → (impl. dep., incl. 5) |
| €[| | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | • | • | • | • | Byte | 2 (safe/unsafe) | O (function call) |
| S | Softw | vare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | • | • | • | • | Byte | ∞ | (°) |
| | Memo | ory Encryption / AES-NI [155] | 0 | Mutual | Full | • | • | 0 | 0 | 128 bits | ∞ | |



(1) Lefeuvre et al., SoK: Software

Compartmentalization, S&P 2025

(1)

| | | Mechanism Class | Condi- | Trust | TCB | P | ermi | ssioi | ns | Granularity | № of Domains | Domain Switch Cost |
|---------------|-------|---|----------------|--------|------|----------------|----------------|----------------|----|---------------------------|-------------------------------|---|
| | | meenanism Ciass | tioned | Model | TCD | R | W | X | A | Granularly | N- Of Bomains | (Versus Non-Separated) |
| | Physi | cal Separation [205] | 0 | Mutual | Full | • | • | • | 0 | Physical Mem. | № of machines | ● – ● (link latency) |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}_1 | Mutual | Full | \bullet^2 | $lacksquare^2$ | $lacksquare^2$ | 0 | Page | ∞ | • (PT switch + ⁵) |
| | PT | Supervisor Bit [25, 158, 159] | 0 | Single | Full | $lacksquare^2$ | $lacksquare^2$ | $lacksquare^2$ | 0 | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| | * * | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | $lacksquare^2$ | 0 | Word | ∞ | (MMP hardware + ⁵) |
| are | | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | • | • | 0 | 0 | Page | 8-1024 [13, 217] ⁴ | (special register flip + 5) |
| ≥ | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | • | • | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | ⊙ (³) |
| ard | | Enclaves [28, 92] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| H | TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | • | • | | 0 | Page | ∞ | • (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | • | • | • | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | • | • | | • | Byte | ∞ | \bigcirc (special instr. + 5) |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | 0 | 0 | Byte | ∞ | (bounds hardware + 5) |
| | (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) |
| | Softw | vare Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| /are | Boun | ds-Checking Software [225] | • | Mutual | Full | • | • | • | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| -ftw | | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | • | • | | • | Byte | 2 (safe/unsafe) | O (function call) |
| $ S_{\rm o} $ | Softw | vare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | • | • | • | • | Byte | ∞ | (5) |
| | Mem | ory Encryption / AES-NI [155] | 0 | Mutual | Full | • | • | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

Names of the mechanisms we consider

Characteristics we included in the taxonomy

| | W 1 : Cl | Condi- | Trust | TCD | P | Permi | ssio | ns | C 1 ' | No. C.D. | Domain Switch Cost |
|--------|---|---|--|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|-------------------------------|---|
| | Mechanism Class | tioned | Model | ICB | R | W | X | A | Granularity | Nº of Di | (Versus Non-Separated) |
| Physic | cal Separation [205] | 0 | Mutual | Full | | | | 0 | Physical Mem. | № of machines | ● – ● (link latency) |
| | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}^1 | Mutual | Full | $lacksquare^2$ | •2 | | 0 | Page | ∞ | • (PT switch + ⁵) |
| РТ | Supervisor Bit [25, 158, 159] | 0 | Single | Full | •2 | •2 | O ² | | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| • • | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | •2 | •2 | • ² | | Word | ∞ | (MMP hardware + 5) |
| | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | | • | • | 0 | Page | 8-1024 [13, 217] ⁴ | (special register flip + 5) |
| Segm | entation-like Hardware [109, 178] | 0 | Single | Full | | | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| | Enclaves [28, 92] | 0 | Mutual | TEE | | | | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | | | | 0 | Page | ∞ | (> EPT switch) |
| | World Separation [9, 14] | 0 | Single | TEE | | | | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| Hardy | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | | Byte | ∞ | (special instr. + ⁵) |
| Bound | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | lacksquare | $lacksquare^2$ | 0 | 0 | Byte | ∞ | (bounds hardware + ⁵) |
| (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | lacksquare | $lacksquare^2$ | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) |
| Softw | vare Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| | | • | Mutual | Full | | | • | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| | | 0 | Single | Full | | | | | Byte | 2 (safe/unsafe) | O (function call) |
| Softw | rare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | | | | | Byte | ∞ | O (5) |
| Memo | ory Encryption / AES-NI [155] | 0 | Mutual | Full | | | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |
| | PT Segm TEE Hardy Bound (Other Softw Bound Safe Softw Memory | PT Supervisor Bit [25, 158, 159] Mondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] Segmentation-like Hardware [109, 178] Enclaves [28, 92] Confidential VMs [10, 29, 30] World Separation [9, 14] Hardware Capabilities [57, 78, 180, 236, 244] Bounds-Checking Hardware [47, 98, 148, 155, 212] (Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] Software Capabilities [83, 125] Bounds-Checking Software [225] Safe Languages [50] / Software Verification [154, 163] Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] Memory Encryption / AES-NI [155] | Physical Separation [205] Access Bits [25], EPT / vmfunc [26] Supervisor Bit [25, 158, 159] Mondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] Segmentation-like Hardware [109, 178] Enclaves [28, 92] Confidential VMs [10, 29, 30] World Separation [9, 14] Hardware Capabilities [57, 78, 180, 236, 244] Bounds-Checking Hardware [47, 98, 148, 155, 212] (Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] Software Capabilities [83, 125] Bounds-Checking Software [225] Safe Languages [50] / Software Verification [154, 163] Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] Memory Encryption / AES-NI [155] | Physical Separation [205] | Physical Separation [205] |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

Names of the mechanisms we consider

Characteristics we included in the taxonomy

| | | Condi- | Trust | | 1 | Perm | issio | ns | | | Domain Switch Cost |
|--------|---|---|---|--------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|-------------------------------|---|
| | Mechanism Class | tioned | Model | TCB | R | W | X | A | Granularity | № of D | (Versus Non-Separated) |
| Physi | cal Separation [205] | 0 | Mutual | Full | | | | 0 | Physical Mem. | № of machines | Ō – ● (link latency) |
| | Access Bits [25], EPT / vmfunc [26] | $lue{\mathbb{O}}^{1}$ | Mutual | Full | | | • | 0 | Page | ∞ | • (PT switch + ⁵) |
| РТ | Supervisor Bit [25, 158, 159] | 0 | Single | Full | | | | | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | | | | | Word | ∞ | (MMP hardware + ⁵) |
| | Protection Keys [13, 27, 53, 217, 259] | | Mutual | Full | | | • | 0 | Page | 8-1024 [13, 217] ⁴ | (special register flip + 5) |
| Segm | entation-like Hardware [109, 178] | 0 | Single | Full | • | | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| | Enclaves [28, 92] | 0 | Mutual | TEE | | | | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | | | | 0 | Page | ∞ | (> EPT switch) |
| | World Separation [9, 14] | 0 | Single | TEE | | | | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| Hardy | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | | Byte | ∞ | \bigcirc (special instr. + 5) |
| Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | | | 0 | 0 | Byte | ∞ | (bounds hardware + ⁵) |
| (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | | | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) |
| Softw | rare Capabilities [83, 125] | 0 | Mutual | Full | • | • | | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| Boun | ds-Checking Software [225] | • | Mutual | Full | | | • | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| Safe 1 | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | | | | | Byte | 2 (safe/unsafe) | O (function call) |
| Softw | rare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | | | | | Byte | ∞ | (5) |
| Memo | ory Encryption / AES-NI [155] | 0 | Mutual | Full | | | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |
| | PT Segm TEE Hardy Bound (Othe Softw Bound Safe Softw Memo | PT Mondrian Memory Protection (MMP) [249] | Physical Separation [205] Access Bits [25], EPT / vmfunc [26] PT Supervisor Bit [25, 158, 159] Mondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] Segmentation-like Hardware [109, 178] Enclaves [28, 92] TEE Confidential VMs [10, 29, 30] World Separation [9, 14] Hardware Capabilities [57, 78, 180, 236, 244] Bounds-Checking Hardware [47, 98, 148, 155, 212] (Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] Software Capabilities [83, 125] Bounds-Checking Software [225] Safe Languages [50] / Software Verification [154, 163] Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] Memory Encryption / AES-NI [155] | Physical Separation 1205 | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation (205) | Physical Separation 1265 |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

Again, a few interesting aspects 🙂



| | | Mechanism Class | Condi- tioned | Trust Model | ТСВ | R | ermi W | | ns A | Granularity | № of Domains | Domain Switch Cost (Versus Non-Separated) |
|----------|-------|---|------------------|----------------|------|----------------|----------------|----------------|---------|---------------------------|-------------------------------|---|
| | Physi | cal Separation [205] | 0 | Mutual | Full | • | • | • | 0 | Physical Mem. | № of machines | O – ● (link latency) |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}^1 | Mutual | Full | \bullet^2 | \bullet^2 | •2 | | Page | ∞ | • (PT switch + ⁵) |
| | PT | Supervisor Bit [25, 158, 159] | 0 | Single | Full | \bullet^2 | \bullet^2 | $lacksquare^2$ | | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| | 11 | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | \bullet^2 | \bullet^2 | $lacksquare^2$ | | Word | ∞ | (MMP hardware + 5) |
| le l | | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | • | • | 0 | 0 | Page | 8-1024 [13, 217] ⁴ | • (special register flip + ⁵) |
| Hardware | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | • | • | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| ard | | Enclaves [28, 92] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| Ξ | TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | • | • | • | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | Hardy | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc (special instr. + 5) |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | \bullet^2 | \bullet^2 | 0 | 0 | Byte | ∞ | (bounds hardware + 5) |
| | (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) |
| | Softw | vare Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| /are | Boun | ds-Checking Software [225] | • | Mutual | Full | • | • | 0 | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| Softw | Safe | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | • | • | • | • | Byte | 2 (safe/unsafe) | O (function call) |
| So | Softw | rare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | • | • | • | • | Byte | ∞ | (5) |
| | | ory Encryption / AES-NI [155] | 0 | Mutual | Full | • | • | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

Hardware and software are both popular research areas

| | | | | Mechanism Class | Condi- tioned | Trust Model | ТСВ | R | ermi W | ssion | is A | Granularity | № of Domains | Domain Switch Cost (Versus Non-Separated) |
|--------|------|-----------------|--------|---|------------------|----------------|-----------|----------------|----------------|----------------|------|---------------------------|---------------------------------|---|
| | Ī | | Physic | cal Separation [205] | 0 | Mutual | Full | • | • | • | 0 | Physical Mem. | № of machines | ● – ● (link latency) |
| 1 | | | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}^1 | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | $lacksquare^2$ | 0 | Page | ∞ | \bullet (PT switch + 5) |
| Intel | | | PT [| Supervisor Bit [25, 158, 159] | 0 | Single | Full | lacksquare | $lacksquare^2$ | $lacksquare^2$ | 0 | Page | 2 (kernel/user) | \bullet (interrupt + 5) |
| | | 7 | \ | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | lacksquare | $lacksquare^2$ | $lacksquare^2$ | 0 | Word | ∞ | (MMP hardware + ⁵) |
| MPK | | 7/ | ١ | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | | | 0 | 0 | Page | 8-1024 [13, 217] ⁴ | \circ (special register flip + 5) |
| | - 1/ | WI | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | | | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | (⁵) |
| | | Hardw | | Enclaves [28, 92] | 0 | Mutual | TEE | | | | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| | | 天 | TEE [| Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | | | | 0 | Page | 8 | • (> EPT switch) |
| CHERI | | ~/ | | World Separation [9, 14] | 0 | Single | TEE | | • | • | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| CITEIN | | | | vare Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | | Byte | ∞ | \odot (special instr. + 5) |
| | 1 | | Bound | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | lacksquare | $lacksquare^2$ | 0 | 0 | Byte | ∞ | \odot (bounds hardware + 5) |
| | l | | (Other | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | lacksquare | $lacksquare^2$ | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | \odot (tagging hardware + 5) |
| | | | Softw | are Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| | | /are | Bound | ds-Checking Software [225] | • | Mutual | Full | • | • | 0 | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| | | ftw | Safe I | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | • | • | • | • | Byte | 2 (safe/unsafe) | O (function call) |
| | | Soft | Softw | are Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | | • | • | | Byte | ∞ | (5) |
| | | | Memo | pry prion / AES-NI [155] | 0 | Mutual | Full | • | • | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |
| | | ¹ In | Ring | 0 all combinations of R/W/X supported. ³ Co | overs man | y granulai | rities [1 | 38]. | 4 So | me v | vork | s [113, 173, 190] incre | ase it. ⁵ Register s | saving/scrubbing, stack switch. |

WebAssembly

A lot of interest in enforcing memory isolation at a byte granularity (vs. a page)

| | | Mechanism Class | Condi- | Trust | TCB | P | ermi | | ıs | Granularity | № of Domains | Domain Switch Cost |
|---------|--------|--|----------------|-----------|-----------|----------------|----------------|--------------|----|---------------------------|-------------------------------|---|
| | | Meentanii Vii | tioned | Model | 102 | R | W | X | A | | tr of 2 oments | (Versus Non-Separated) |
| | Physi | cal Separation [205] | 0 | Mutual | Full | | | • | 0 | Physical Mem. | № of machines | ● – ● (link latency) |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}^1 | Mutual | Full | \bullet^2 | \bullet^2 | \bullet^2 | 0 | Page | ∞ | \bullet (PT switch + 5) |
| | PT | Supervisor Bit [25, 158, 159] | 0 | Single | Full | lacksquare | lacksquare | \bullet^2 | 0 | Page | 2 (kernel/user) | \bullet (interrupt + 5) |
| | | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | lacksquare | $lacksquare^2$ | \bullet^2 | 0 | Word | ∞ | (MMP hardware + ⁵) |
| le l | | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | | • | lacktriangle | 0 | Page | 8-1024 [13, 217] ⁴ | • (special register flip + ⁵) |
| ardware | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | | • | \bigcirc | 0 | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| arc | | Enclaves [28, 92] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| H | TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | • | • | • | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | (special instr. + ⁵) |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | lacksquare | $lacksquare^2$ | 0 | 0 | Byte | ∞ | (bounds hardware + ⁵) |
| | (Othe | er) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) |
| | Softw | vare Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| /are | Boun | ds-Checking Software [225] | • | Mutual | Full | • | • | • | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| Softwa | Safe | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | • | • | • | • | Byte | 2 (safe/unsafe) | O (function call) |
| So | Softv | vare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | • | • | • | • | Byte | ∞ | (5) |
| | Mem | ory Encryption / AES-NI [155] | 0 | Mutual | Full | • | • | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |
| 1] | n Ring | 0. 2 Not all combinations of R/W/X supported. 3 Co | overs man | y granula | rities [1 | 38]. ' | [‡] ? | | | | | rubbing, stack switch. |

Very widely varying – 128 bits!

| | | Mechanism Class | Condi- | Trust | TCB | P | ermi | ssioi | ıs | Granularity | № of Domains | Domain Switch Cost |
|------|-------|--|----------------|--------|------|----------------|----------------|----------------|----|---------------------------|-------------------------------|---|
| | | mechanism Ciass | tioned | Model | TCD | R | W | X | A | Granularity | N= 0j Domains | (Versus Non-Separated) |
| | Physi | cal Separation [205] | 0 | Mutual | Full | • | • | • | 0 | Physical Mem. | № of machines | Ō – ● (link latency) |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}_1 | Mutual | Full | \bullet^2 | $lacksquare^2$ | $lacksquare^2$ | 0 | Page | ∞ | • (PT switch + ⁵) |
| | PT | Supervisor Bit [25, 158, 159] | 0 | Single | Full | $lacksquare^2$ | $lacksquare^2$ | $lacksquare^2$ | 0 | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| | * * | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | $lacksquare^2$ | 0 | Word | ∞ | (MMP hardware + ⁵) |
| are | | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | • | • | • | 0 | Page | 8-1024 [13, 217] ⁴ | (special register flip + ⁵) |
| ≽ | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | | • | 0 | 0 | Byte - Page [178] | 2 (safe/unsafe) | ⊙ (⁵) |
| ard | | Enclaves [28, 92] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (enclave call, incl. ⁵) |
| Ξ | TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | • | • | • | 0 | Page | ∞ | • (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | | • | • | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | • | • | | Byte | ∞ | \bigcirc (special instr. + 5) |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | 0 | 0 | Byte | ∞ | (bounds hardware + 5) |
| | (Othe | er) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | $lacksquare^2$ | $lacksquare^2$ | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + ⁵) |
| | Softw | vare Capabilities [83, 125] | 0 | Mutual | Full | • | • | • | • | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| /are | Boun | ds-Checking Software [225] | • | Mutual | Full | • | • | • | 0 | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| -ftw | | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | • | • | • | • | Byte | 2 (safe/unsafe) | O (function call) |
| So | Softw | vare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | • | • | • | | Byte | ∞ | O (⁵) |
| | Mem | ory Encryption / AES-NI [155] | 0 | Mutual | Full | • | • | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

Nearly all works aim at cutting the cost of domain switches

| | Machanism Class | Condi- | Trust | TCR | P | ermi | ssioi | rs | Granularity | No of Domains | Domain Switch Cost |
|--------|---|--|---------------------------|---|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|-------------------------------|---|
| | mechanism Ciass | tioned | Model | TCD | R | W | X | A | Granularity | Nº Of Domains | (Versus Non-Separated) |
| Physi | cal Separation [205] | | Mutual | Full | | | | | Physical Mem. | № of machines | ● – • (link latency) |
| | Access Bits [25], EPT / vmfunc [26] | \mathbb{O}^1 | Mutual | Full | o ² | 2 | 2 | | Page | ∞ | • (PT switch + ⁵) |
| РТ | Supervisor Bit [25, 158, 159] | | Single | Full | O ² | 0 ² | 0 ² | | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| 1 1 | Mondrian Memory Protection (MMP) [249] | | Mutual | Full | o ² | 0 ² | o ² | | Word | ∞ | (MMP hardware + 5) |
| | Protection Keys [13, 27, 53, 217, 259] | | Mutual | Full | | | • | | Page | 8-1024 [13, 217] ⁴ | (special register flip + 5) |
| Segm | entation-like Hardware [109, 178] | | Single | Full | | | | | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| | Enclaves [28, 92] | | Mutual | TEE | | | | | Page | ∞ | • (enclave call, incl. ⁵) |
| TEE | Confidential VMs [10, 29, 30] | | Mutual | TEE | | | | | Page | ∞ | (> EPT switch) |
| | World Separation [9, 14] | | Single | TEE | | | | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| Hardy | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | • | | Byte | ∞ | (special instr. + ⁵) |
| Bound | ds-Checking Hardware [47, 98, 148, 155, 212] | | Mutual | Full | •2 | • ² | \bigcirc | 0 | Byte | ∞ | (bounds hardware + 5) |
| (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | | Mutual | Full | • ² | 0 ² | | | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) |
| Softw | vare Capabilities [83, 125] | \circ | Mutual | Full | | | | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| Boun | ds-Checking Software [225] | | Mutual | Full | | | • | 0 | Byte | ∞ | ○ – ● (impl. dep., incl. ⁵) |
| Safe 1 | Languages [50] / Software Verification [154, 163] | \bigcirc | Single | Full | | | | | Byte | 2 (safe/unsafe) | O (function call) |
| Softw | rare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | | Single | Full | | | | | Byte | ∞ | (°) |
| Memo | ory Encryption / AES-NI [155] | | Mutual | Full | | | | | 128 bits | ∞ | • (copy key + encrypt + ⁵) |
| | Segm TEE Hardy Bound (Other Softw Bound Safe I | Supervisor Bit [25, 158, 159] Mondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] Segmentation-like Hardware [109, 178] Enclaves [28, 92] Confidential VMs [10, 29, 30] World Separation [9, 14] Hardware Capabilities [57, 78, 180, 236, 244] Bounds-Checking Hardware [47, 98, 148, 155, 212] (Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] Software Capabilities [83, 125] Bounds-Checking Software [225] Safe Languages [50] / Software Verification [154, 163] Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] Memory Encryption / AES-NI [155] | Physical Separation [205] | Natural Physical Separation [205] Mutual Access Bits [25], EPT / vmfunc [26] Mutual Supervisor Bit [25, 158, 159] Single Mondrian Memory Protection (MMP) [249] Mutual Protection Keys [13, 27, 53, 217, 259] Mutual Protection Like Hardware [109, 178] Single Enclaves [28, 92] Mutual Confidential VMs [10, 29, 30] Mutual World Separation [9, 14] Single Hardware Capabilities [57, 78, 180, 236, 244] Mutual Bounds-Checking Hardware [47, 98, 148, 155, 212] Mutual (Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] Mutual Software Capabilities [83, 125] Mutual Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] Single Memory Encryption / AES-NI [155] Mutual | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] | Physical Separation [205] |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

| | | Mechanism Class | Condi- | Trust | TCB | P | ermi | ssioi | ıs | Granularity | № of Domains | Domain Switch Cost |
|----------|--------------|---|----------------|--------|------|-----------------------|-----------------------|----------|----|---------------------------|-----------------------|---|
| | | weenunism Cluss | tioned | Model | TCB | R | W | X | A | - <i>Отаништиу</i> | N- of Domains | (Versus Non-Separated) |
| Hardwa | <i>ire</i> n | nechanism that | | Mutual | Full | | | | | Physical Mem. | № of machines | ● – • (link latency) |
| | _ | func [26] | \mathbb{O}^1 | Mutual | Full | 0 ² | • ² | 2 | | Page | ∞ | • (PT switch + ⁵) |
| comes a | as ai | n ISA extension | | Single | Full | 0 ² | 0 ² | 2 | | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| | | Monarian Memory Protection (MMP) [249] | | Mutual | Full | • ² | 0 ² | 2 | | Word | ∞ | (MMP hardware + 5) |
| le l | | Protection Keys [13, 27, 53, 217, 259] | | Mutual | Full | | | 0 | | Page | 8-1024 [13, 217]4 | (special register flip + 5) |
| Hardware | Segn | mentation-like Hardware [109, 178] | | Single | Full | | | | | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| ard | | Enclaves [28, 92] | 0 | Mutual | TEE | | | | | Page | ∞ | • (enclave call, incl. 5) |
| = | TEE | Confidential VMs [10, 29, 30] | | Mutual | TEE | | | | | Page | ∞ | (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | | | | 0 | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | Hard | dware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | | Byte | ∞ | (special instr. + ⁵) |
| | Bou | nds-Checking Hardware [47, 98, 148, 155, 212] | | Mutual | Full | •2 | •2 | | 0 | Byte | ∞ | \bigcirc (bounds hardware + 5) |
| | (Oth | ner) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | | Mutual | Full | • ² | 0 ² | | | Byte - Words ³ | $16^4 - \infty$ [138] | \bigcirc (tagging hardware + 5) |
| | Soft | ware Capabilities [83, 125] | | Mutual | Full | | | | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| 'are | Bou | nds-Checking Software [225] | | Mutual | Full | | | 0 | 0 | Byte | ∞ | ○ – ● (impl. dep., incl. ⁵) |
| ft | Safe | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | | | | | Byte | 2 (safe/unsafe) | O (function call) |
| So | Soft | ware Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | | Single | Full | | | | | Byte | ∞ | O (5) |
| | Men | nory Encryption / AES-NI [155] | | Mutual | Full | | | | | 128 bits | ∞ | • (copy key + encrypt + 5 |

¹ In Ring 0. ² Not all combinations of R/W/X supported. ³ Covers many granularities [138]. ⁴ Some works [113, 173, 190] increase it. ⁵ Register saving/scrubbing, stack switch.

| | Mechanism Class | Condi- tioned | Trust Model | ТСВ | R | ermi. W | ssioi X | ns A | Granularity | № of Domains | Domain Switch Cost (Versus Non-Separated) |
|-----------------|--|------------------|----------------|----------|-----------------------|-----------------------|-----------------------|---------|---------------------------|-------------------------------|---|
| Hardwar | re mechanism that | | Mutual | Full | | | | | Physical Mem. | № of machines | ● – ● (link latency) |
| | func [26] | \mathbb{O}^1 | Mutual | Full | •2 | 0 ² | 0 ² | 0 | Page | ∞ | • (PT switch + ⁵) |
| comes a | s an ISA extension | 0 | Single | Eu11 | 2 | 2 | 2 | | Dago | 2 (kornol/user) | • (interrupt + ⁵) |
| | Mondrian Memory Protection (MMP) [249] | 0 | M | CH | FR | l ex | xte | nd | s pointers v | with | (MMP hardware + ⁵) |
| E | Protection Keys [13, 27, 53, 217, 259] | | M | | | | | | • | 71 ' | (special register flip + 5) |
| Hardware | Segmentation-like Hardware [109, 178] | 0 | Si | | οοι | าทด | ds : | an | d permissio | n e) | O (⁵) |
| ard | Enclaves [28, 92] | 0 | M | for | ~ ~ | ٠i٠ | <u></u> | 1+ ; | · | ossival | • (enclave call, incl. ⁵) |
| # | TEE Confidential VMs [10, 29, 30] | 0 | M | HOH | Ha | ιιΟ | Π. | ILI | s very <i>expr</i> | essive! | (> EPT switch) |
| | World Separation [9, 14] | 0 | Single | TEE | | _ | | | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | Hardware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | • | • | • | | Byte | ∞ | (special instr. + ⁵) |
| [[| Bounds-Checking Hardware [47, 98, 148, 155, 212] | | Mutual | Full | O ² | •2 | 0 | 0 | Byte | ∞ | (bounds hardware + 5) |
| | (Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | | Mutual | Full | 2 | • ² | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + ⁵) |
| | Software Capabilities [83, 125] | | Mutual | Full | | | | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| /are | Bounds-Checking Software [225] | | Mutual | Full | | | 0 | 0 | Byte | ∞ | ○ – ● (impl. dep., incl. ⁵) |
| lft. | Safe Languages [50] / Software Verification [154, 163] | 0 | Single | Full | | | | | Byte | 2 (safe/unsafe) | O (function call) |
| So | Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | | | | | Byte | ∞ | (5) |
| | Memory Encryption / AES-NI [155] | 0 | Mutual | Full | | | 0 | 0 | 128 bits | ∞ | \bullet (copy key + encrypt + 5) |
| ¹ Ir | n Ring 0. ² Not all combinations of R/W/X supported. ³ C | overs man | y granula | rities [| [38]. | Soi | me v | vork | s [113, 173, 190] incre | ase it. ⁵ Register | saving/scrubbing, stack switch |

| nechanism that n ISA extension Mondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] mentation-like Hardware [109, 178] | | Single | Full Full | • • • • • • • • • • • • • • • • • • • | • • • • • • • • • • • • • • • • • • • | | Physical Mem. Page | № of machines ∞ | |
|--|---|--|---|---|--|--|--|--|---|
| func [26] Nondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] nentation-like Hardware [109, 178] | 0 | Single | Eu11 | 2 | 2 2 2 | $\binom{2}{2}$ | Page | | (PT switch + ⁵) |
| Protection Keys [13, 27, 53, 217, 259] mentation-like Hardware [109, 178] | 0 | Sillota | | 2 | 2 | 2 | | | |
| Protection Keys [13, 27, 53, 217, 259] nentation-like Hardware [109, 178] | | M | $C \sqcup I$ | | | | Dage | 2 (kernel/user) | (interrupt + ⁵) |
| nentation-like Hardware [109, 178] | | 3.4 | | -RI | ext | end | ds pointers | with | (MMP hardware + ⁵) |
| | | IVI | | | | | • | 7] | (special register flip + 5) |
| Englaving rag on | | Si | b | ou | nds | an | d permission | e) | O (5) |
| Enclaves [28, 92] | | M in | forn | ^ ~ + | ion | 1+ | ic vory over | occival | • (enclave call, incl. ⁵) |
| Confidential VMs [10, 29, 30] | | М | 10111 | Iat | 1011 | . IL | is very <i>expr</i> | essive! | (> EPT switch) |
| World Separation [9, 14] | | Single | TEE | | | | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| Iware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | Byte | ∞ | \bigcirc (special instr. + 5) |
| nds-Checking Hardware [47, 98, 148, 155, 212] | | Mutual | Full | \square^2 | \bullet^2 | | But | $-\infty$ | \bigcirc (bounds hardware + 5) |
| er) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | | It c | an l | വല | IISA | d ta | enforce | - ∞ [138] | \bigcirc (tagging hardware + 5) |
| ware Capabilities [83, 125] | | M | | | | | | $_{\perp}$ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| nds-Checking Software [225] | | M ISO | latio | on a | at a | by | te granuları | ity, ∞ | ● – ● (impl. dep., incl. ⁵) |
| Languages [50] / Software Verification [154, 163] | | C: | | | | • | _ | (fo/manfa) | O (function call) |
| ware Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Sin | opp | OSE | ea t | o e | nure pages | ∞ | O (5) |
| nory Encryption / AES-NI [155] | 0 | Mutual | Full | | | | 128 bits | ∞ | • (copy key + encrypt + |
| 1 e X | ware Capabilities [57, 78, 180, 236, 244] Ids-Checking Hardware [47, 98, 148, 155, 212] er) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] vare Capabilities [83, 125] Ids-Checking Software [225] Languages [50] / Software Verification [154, 163] vare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] lory Encryption / AES-NI [155] | ware Capabilities [57, 78, 180, 236, 244] olds-Checking Hardware [47, 98, 148, 155, 212] er) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] ovare Capabilities [83, 125] olds-Checking Software [225] Languages [50] / Software Verification [154, 163] ovare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] ory Encryption / AES-NI [155] | ware Capabilities [57, 78, 180, 236, 244] | ware Capabilities [57, 78, 180, 236, 244] | ware Capabilities [57, 78, 180, 236, 244] olds-Checking Hardware [47, 98, 148, 155, 212] or) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] order Capabilities [83, 125] o | ware Capabilities [57, 78, 180, 236, 244] ods-Checking Hardware [47, 98, 148, 155, 212] or) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] over Capabilities [83, 125] ods-Checking Software [225] ods-Checking Software [225] Languages [50] / Software Verification [154, 163] over Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] ory Encryption / AES-NI [155] Mutual Full ods Mutual Full ods Opposed to the standard properties of the standard properties | ware Capabilities [57, 78, 180, 236, 244] order) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] order) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] order) Capabilities [83, 125] order) Ca | ware Capabilities [57, 78, 180, 236, 244] order Order | ware Capabilities [57, 78, 180, 236, 244] |

| | Mechanism Class | Condi- tioned | Trust Model | ТСВ | | missi W 2 | | - Granularity | № of Domains | Domain Switch Cost (Versus Non-Separated) | |
|----------|--|------------------|---------------------|---|--------------------------------|--------------|--------------------|---|---|---|--|
| Hardwai | | Mutual Mutual | | O ² | D ² C | | Physical Mem. Page | № of machines | ○ - ● (link latency) ● (PT switch + ⁵) | | |
| | comes as an ISA extension Mondrian Memory Protection (MMP) [249] Protection Keys [13, 27, 53, 217, 259] | | | | | | | ls pointers | with n4 | (interrupt + ⁵) (MMP hardware + ⁵) | |
| Hardware | Segmentation-like Hardware [109, 178] Enclaves [28, 92] TEE Confidential VMs [10, 29, 30] | 0 | Sin M II | | | | | d permissions s very <i>expr</i> | | It comes with a very low domain-switching overhead. | |
| | World Separation [9, 14] Hardware Capabilities [57, 78, 180, 236, 244] Bounds-Checking Hardware [47, 98, 148, 155, 212] | 0 | Single Mutual | Full | 2 | 2 | | Page Byte | 2 (trusted/rest) ∞ | (world) (special instr. + 5) (bounds hardware + 5) | |
| | Other) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] Software Capabilities [83, 125] | | M | | | | | enforce | - ∞ [138] ∞ | | |
| Software | Bounds-Checking Software [225] Safe Languages [50] / Software Verification [154, 163] Software Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | C: | solation at a <i>byte grant</i> as opposed to entire pa | | _ | £-/ | O − ● (impl. dep., incl. ⁵) O (function call) O (⁵) | | | |
| 1 II | Memory Encryption / AES-NI [155] n Ring 0. ² Not all combinations of R/W/X supported. ³ C | 0 | Mutual y granula | | 138]. | Some | o o o | 128 bits 13, 173, 190] incre | ∞ | ● (copy key + encrypt + ⁵) saving/scrubbing, stack switch. | |

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we formalize them in the SoK)

Some (popular!) mechanisms do not fulfill all these properties

| | Mechanism Class | | Condi- | | | | | |
|---------|---------------------------|--|----------------|--|--|--|--|--|
| | | Mechanism Class | | | | | | |
| | Physical Separation [205] | | | | | | | |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}_1 | | | | | |
| | PT | Supervisor Bit [25, 158, 159] Mondrian Memory Protection (MMP) [249] | | | | | | |
| | 1 1 | | | | | | | |
| re | | Protection Keys [13, 27, 53, 217, 259] | | | | | | |
| ardware | Segm | Segmentation-like Hardware [109, 178] | | | | | | |
| | | Enclaves [28, 92] | | | | | | |
| I | TEE | Confidential VMs [10, 29, 30] | | | | | | |
| | | World Separation [9, 14] | | | | | | |
| | Hardy | ware Capabilities [57, 78, 180, 236, 244] | | | | | | |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | | | | | | |
| | (Othe | er) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | | | | | | |
| - | Softw | vare Capabilities [83, 125] | 0 | | | | | |
| /are | Boun | ds-Checking Software [225] | | | | | | |
| ftw | Safe | Languages [50] / Software Verification [154, 163] | | | | | | |
| So | Softw | ware Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | | | | | | |
| | Mem | ory Encryption / AES-NI [155] | | | | | | |
| | | | | | | | | |

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we formalize them in the SoK)

Some (popular!) mechanisms do not fulfill all these properties

| | Mechanism Class | | Condi- | Trust | TCB | P | ermi | ssioi | 18 | Granularity | № of Domains | Domain Switch Co |
|----------|---|---|----------------|--------|------|----|------|-------|----|---------------------------|-------------------------------|---|
| | | | | Model | 1 CD | R | W | Х | А | Granularity | N= Of Domains | (Versus Non-Separated) |
| | Physical Separation [205] | | 0 | Mutual | Full | | | | 0 | Physical Mem. | № of machines | |
| | | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}^1 | Mutual | Full | | •2 | | | Page | ∞ | • (PT switch + ⁵) |
| | РТ | Supervisor Bit [25, 158, 159] | 0 | Single | Full | •2 | •2 | •2 | | Page | 2 (kernel/user) | • (interrupt + ⁵) |
| | • • | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | | | | | Word | ∞ | (MMP hardware + ⁵) |
| le. | | Protection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | | | • | | Page | 8-1024 [13, 217] ⁴ | (special register flip + 5) |
| Hardware | Segm | entation-like Hardware [109, 178] | 0 | Single | Full | | | | | Byte - Page [178] | 2 (safe/unsafe) | (5) |
| arc | | Enclaves [28, 92] | 0 | Mutual | TEE | | | | | Page | ∞ | • (enclave call, incl. ⁵) |
| Ξ | TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | | | | | Page | ∞ | (> EPT switch) |
| | | World Separation [9, 14] | 0 | Single | TEE | | | | | Page | 2 (trusted/rest) | • (world switch, incl. ⁵) |
| | Hardy | ware Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | | Byte | ∞ | (special instr. + ⁵) |
| | Boun | ds-Checking Hardware [47, 98, 148, 155, 212] | • | Mutual | Full | •2 | •2 | | | Byte | ∞ | (bounds hardware + 5) |
| | (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | | Mutual | Full | | •2 | 0 | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + ⁵) |
| 4) | Softw | rare Capabilities [83, 125] | 0 | Mutual | Full | | | | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) |
| /are | | ds-Checking Software [225] | • | Mutual | Full | | | • | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) |
| Softw | | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | | | | | Byte | 2 (safe/unsafe) | O (function call) |
| So | Softw | rare Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | | | | | Byte | ∞ | (3) |
| | Mem | ory Encryption / AES-NI [155] | 0 | Mutual | Full | | | | | 128 bits | ∞ | (copy key + encrypt + 5) |
| 1 I | In Ring 0. 2 Not all combinations of R/W/X supported. 3 Covers many granularities [138], 4 Some works [113, 173, 190] increase it. 5 Register saving/scrubbing, stack switch. | | | | | | | | | | | |

How to fairly evaluate the security and performance characteristics of isolation mechanisms?

We call them conditioned.

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we formalize them in the SoK)

Some (popular!) mechanisms do not fulfill all these properties

| | | | | | | | | | | | | | | We call them conditioned. |
|---------|------------------|--------|---|------------------|----------------|----------|----------------|-----------------------|-------------|------|---------------------------|-----------------------|---|---------------------------|
| | | | Mechanism Class | Condi- tioned | Trust Model | TCB | | ermis W | | | Granularity | № of Domains | Domain Switch Co (Versus Non-Separated) | we can them conditioned. |
| | | Physic | cal Separation [205] | 0 | Mutual | Full | | | | 0 | Physical Mem. | № of machines | ● – • (link latency) | |
| Intel | | , | Access Bits [25], EPT / vmfunc [26] | \mathbf{O}_1 | Mutual | Full | \bullet^2 | 2 | \bullet^2 | | Page | ∞ | • (PT switch + 5) | |
| IIICCI | | ъπ | Supervisor Bit [25, 158, 159] | 0 | Single | Full | •2 | 0 ² | 2 | | Page | 2 (kernel/user) | • (interrupt + ⁵) | |
| | // | | Mondrian Memory Protection (MMP) [249] | 0 | Mutual | Full | $lacksquare^2$ | \bullet^2 | \bullet^2 | | Word | ∞ | (MMP hardware + 5) | |
| MPK | | | rotection Keys [13, 27, 53, 217, 259] | • | Mutual | Full | | | • | | Page | 8-1024 [13, 217] | (special register flip + 5) | |
| 1411 17 | lwa [| enl | entation-like Hardware [109, 178] | 0 | Single | Full | | | \circ | | Byte - Page [178] | 2 (safe/unsafe) | (5) | |
| | arc | | Enclaves [28, 92] | 0 | Mutual | TEE | | | | | Page | ∞ | • (enclave call, incl. ⁵) | |
| | = | TEE | Confidential VMs [10, 29, 30] | 0 | Mutual | TEE | | | | | Page | ∞ | (> EPT switch) | |
| | | | World Separation [9, 14] | 0 | Single | TEE | | | | | Page | 2 (trusted/rest) | (world switch, incl. 5) | |
| | | Hardy | vare Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | Full | | | | | Byte | ∞ | (special instr. + ⁵) | |
| | | Bound | ds-Checking Hardware [47, 98, 148, 155, 212] | | Mutual | Full | | \bullet^2 | 0 | | Byte | ∞ | (bounds hardware + 5) | |
| | | (Othe | r) Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | lacksquare | \bullet^2 | 0 | | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) | |
| | | Softw | are Capabilities [83, 125] | 0 | Mutual | Full | | | | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. 5) | |
| | /are | | ls-Checking Software [225] | • | Mutual | Full | | | • | | Byte | ∞ | ○ – ● (impl. dep., incl. ⁵) | |
| |]£[| Safe I | Languages [50] / Software Verification [154, 163] | 0 | Single | Full | | | | | Byte | 2 (safe/unsafe) | (function call) | |
| | % [| Softw | are Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | | | | | Byte | ∞ | (°) | |
| | | Memo | ory Encryption / AES-NI [155] | 0 | Mutual | Full | | | \circ | | 128 bits | ∞ | | |
| | 1 T ₁ | Ring | 0. 2 Not all combinations of R/W/V supported 3 Co | ware mon | v aranula | ritioe H | 201 4 | Son | 00 117 | orke | 1112 172 1001 incre | aca it 5 Pagister | caving/combbing etack exvitch | |

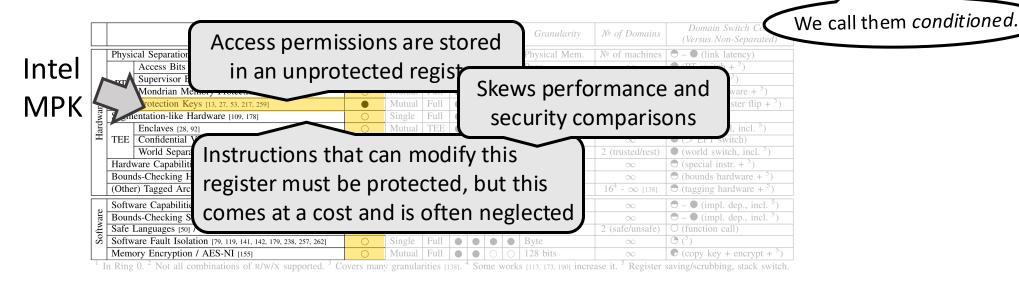
Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we formalize them in the SoK)

Some (popular!) mechanisms do not fulfill all these properties

| | | | | | | | | | Granularity | № of Domains | | e call them conditioned. |
|---------|-----------|---|-------------------|------------|----------|-----------------------|---------------------|------|---------------------------|-----------------------|---|--------------------------|
| | Physics | Access permis | ssions are stored | | | | | | Physical Mem. | № of machines | (Versus Non-Separated) ○ - ● (link latency) | |
| Intel | | Access Bits in an unprod | act | ad ra | aσio | tΔi | r | | Page | ∞ | • (PT switch + ⁵) | |
| IIICCI | | Supervisor h | | | | | | | | 2 (kernel/user) | • (interrupt + ⁵) | |
| | / ~ \ | Mondrian Memory Trotects | | mataa | 1 011 | - | - | _ | Word | ∞ | (MMP hardware + 5) | |
| MPK | | rotection Keys [13, 27, 53, 217, 259] | • | Mutual | | | | | Page | 8-1024 [13, 217]4 | (special register flip + 5) | |
| 1411 17 | ž Sme | ntation-like Hardware [109, 178] | 0 | Single | | 0 0 | | | Byte - Page [178] | 2 (safe/unsafe) | (3) | |
| | | Enclaves [28, 92] | 0 | Mutual | | | | | Page | ∞ | • (enclave call, incl. ⁵) | |
| | _ I | Confidential VMs [10, 29, 30] | 0 | Mutual | | | | 0 | Page | ∞ | (> EPT switch) | |
| | | World Separation [9, 14] | 0 | Single | TEE | • • | | | Page | 2 (trusted/rest) | (world switch, incl. ⁵) | |
| | | rare Capabilities [57, 78, 180, 236, 244] | 0 | Mutual | | 0 0 | | | Byte | ∞ | (special instr. + ⁵) | |
| | Bound | s-Checking Hardware [47, 98, 148, 155, 212] | | Mutual | Full | 0 ² | | | Byte | ∞ | (bounds hardware + 5) | |
| | (Other) | Tagged Architectures [12, 21, 99, 131, 138, 204, 224, 246] | • | Mutual | Full | 0 ² | $ ^2 \bigcirc $ | 0 | Byte - Words ³ | $16^4 - \infty$ [138] | (tagging hardware + 5) | |
| | 4) | are Capabilities [83, 125] | 0 | Mutual | Full | • | | | Byte | ∞ | \bigcirc – \bigcirc (impl. dep., incl. ⁵) | |
| | | s-Checking Software [225] | | Mutual | Full | • | | | Byte | ∞ | ● – ● (impl. dep., incl. ⁵) | |
| | ~ — | anguages [50] / Software Verification [154, 163] | 0 | Single | Full | • | | | Byte | 2 (safe/unsafe) | O (function call) | |
| | | are Fault Isolation [79, 119, 141, 142, 179, 238, 257, 262] | 0 | Single | Full | 0 0 | | | Byte | ∞ | (3) | |
| | Memor | ry Encryption / AES-NI [155] | 0 | Mutual | Full | 0 0 | | | 128 bits | ∞ | \bullet (copy key + encrypt + 5) | |
| | In Ring (| 0. 2 Not all combinations of R/W/X supported. 3 Co | vers man | v granulai | ities 11 | 381. ⁴ S | ome v | vork | s [113, 173, 190] incre | ase it. 5 Register | saving/scrubbing, stack switch. | |

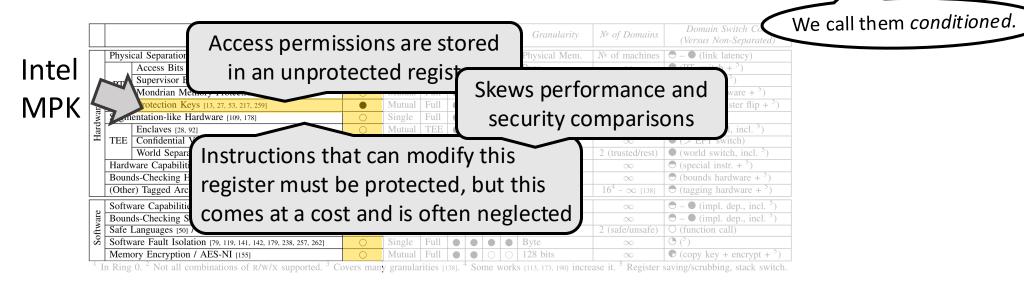
Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we formalize them in the SoK)

Some (popular!) mechanisms do not fulfill all these properties



Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we formalize them in the SoK)

Some (popular!) mechanisms do not fulfill all these properties



My promise for this talk

A journey through twenty years of compartmentalization

- 1. What is software compartmentalization? (slightly more formal)
- 2. A systematic perspective on compartmentalization
- 3. The why: compartmentalization everywhere, what will it take?



What are the remaining obstacles? Where to go from there?

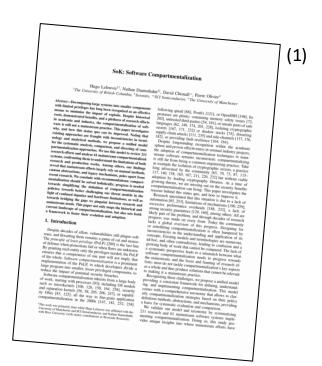
"Out of the 13 groups of compartmentalized applications we constituted, **8 are exclusively authored by academics or security professionals**"



(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

"Out of the 13 groups of compartmentalized applications we constituted, **8 are exclusively authored by academics or security professionals**"

We barely teach it at the university, not part of common "software design patterns"



(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

"Out of the 13 groups of compartmentalized applications we constituted, **8 are exclusively authored by academics or security professionals**"

We barely teach it at the university, not part of common "software design patterns"



Yes, but...

(1) Lefeuvre et al., SoK: Software Compartmentalization, S&P 2025

Potential obstacle #2: Performance?

Potential obstacle #2: Performance?

HTTP FastCGI Parser Module

Compartmentalization has an impact on performance

- Crossing protection domains or IPC is not free (among others)
- This cost varies based on where crossings are in the control flow

This makes compartmentalization more complex and limits the

extent to which one can compartmentalize

Critical stance: in most cases, this is not a problem.

- The community is overly focused on performance
- The cost is reasonable in most cases if the compartmentalization is done correctly
- In fact, many performance-sensitive programs are compartmentalized!



Potential obstacle #2: Performance?

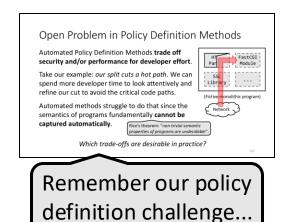
Compartmentalization has an impact on performance

- Crossing protection domains or IPC is not free (among others)
- This cost varies based on *where* crossings are in the control flow This makes compartmentalization more complex and limits the

extent to which one can compartmentalize

Critical stance: in most cases, this is not a problem.

- The community is overly focused on performance
- The cost is reasonable in most cases if the compartmentalization is done correctly
- In fact, many performance-sensitive programs are compartmentalized!



fastcgi do()

FastCGI

Module

HTTP

Parser

Potential obstacle #2: Performance?

Compartmentalization has an impact on performance

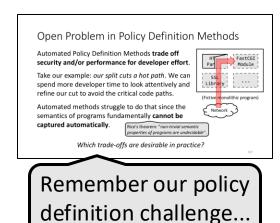
- Crossing protection domains or IPC is not free (among others)
- This cost varies based on where crossings are in the control flow

This makes compartmentalization more complex and limits the

extent to which one can compartmentalize

Critical stance: in most cases, this is not a problem.

- The community is overly focused on performance
- The cost is reasonable in most cases if the compartmentalization is done correctly
- In fact, many performance-sensitive programs are compartmentalized!



fastcgi do()

FastCGI

Module

HTTP

Parser

Potential obstacle #2: Performance?

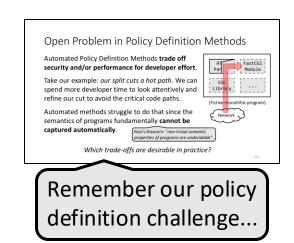
Compartmentalization has an impact on performance

- Crossing protection domains or IPC is not free (among others)
- This cost varies based on *where* crossings are in the control flow This makes compartmentalization more complex and limits the

extent to which one can compartmentalize

Critical stance: in most cases, this is not a problem.

- The community is overly focused on performance
- The cost is reasonable in most cases if the compartmentalization is done correctly
- In fact, many performance-sensitive programs are compartmentalized!



fastcgi do()

.FastCGI

Module

HTTP

Parser

Potential obstacle #2: Performance?

fastcgi_do()

HTTP FastCGI
Parser Module

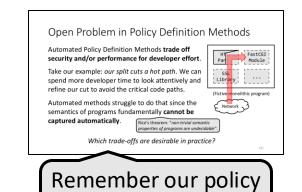
Compartmentalization has an impact on performance

- Crossing protection domains or IPC is not free (among others)
- This cost varies based on *where* crossings are in the control flow This makes compartmentalization more complex and limits the

extent to which one can compartmentalize

Critical stance: in most cases, this is not a problem.

- The community is overly focused on performance
- The cost is reasonable in most cases if the compartmentalization is done correctly
- In fact, many performance-sensitive programs are compartmentalized!

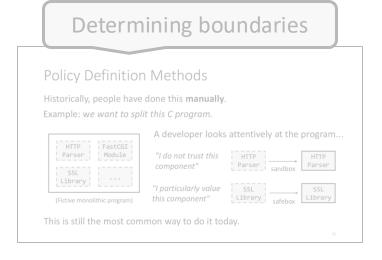


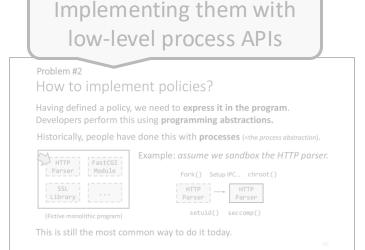
definition challenge..

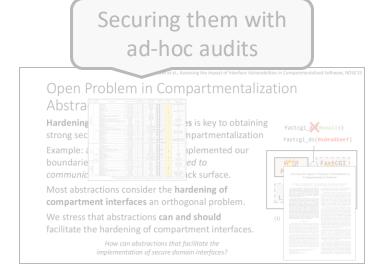
Yes, but...

The historical approach to compartmentalizing software is too complex.

Going back to our example:

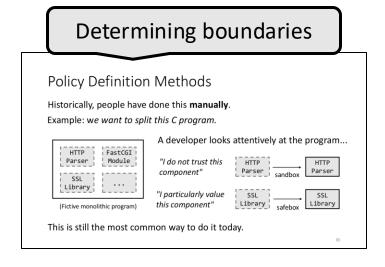


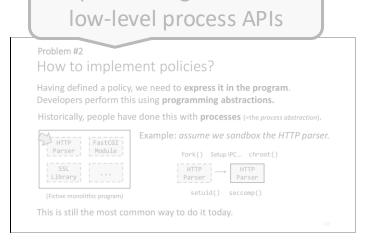




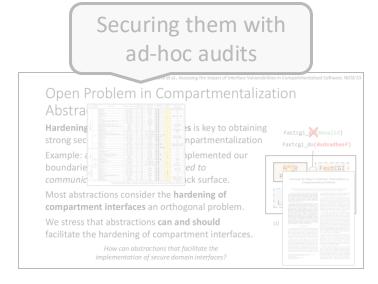
The historical approach to compartmentalizing software is too complex.

Going back to our example:





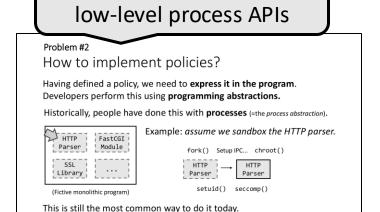
Implementing them with



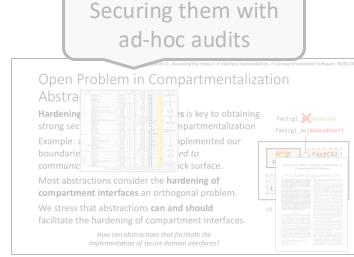
The historical approach to compartmentalizing software is too complex.

Going back to our example:

Policy Definition Methods Historically, people have done this manually. Example: we want to split this C program. A developer looks attentively at the program... "I do not trust this component" "I parsen would be component" "I particularly value this component" This is still the most common way to do it today.

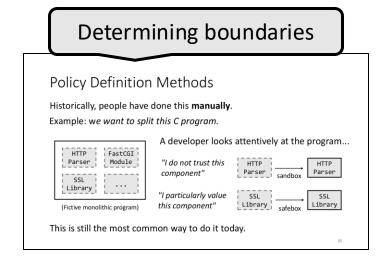


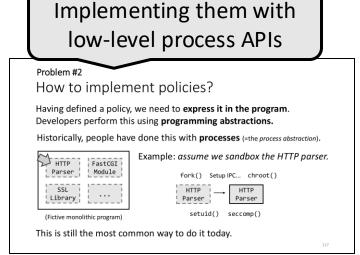
Implementing them with

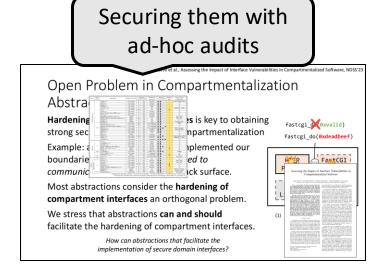


The historical approach to compartmentalizing software is too complex.

Going back to our example:



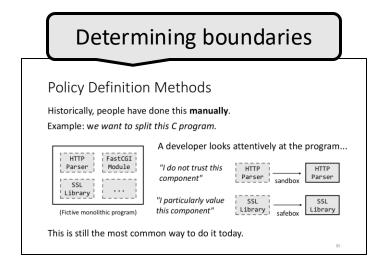


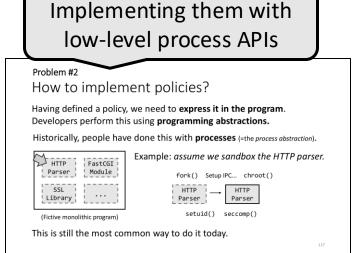


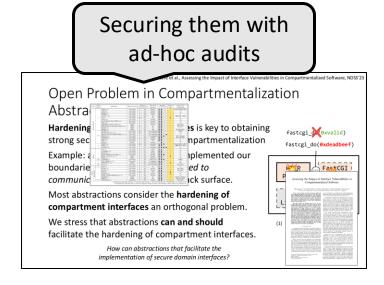
This approach is ad-hoc, costly in developer time, and requires a lot of expertise.

The historical approach to compartmentalizing software is too complex.

Going back to our example:





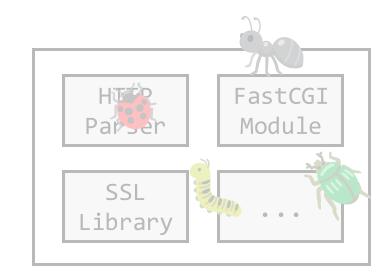


This approach is ad-hoc, costly in developer time, and requires a lot of expertise.

The historical approach to compartmentalizing software is too complex.

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

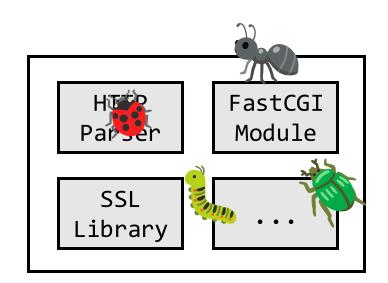
- There will be bugs to fix.
- Bugs now occur across security boundaries.
- This impacts the debugging experience but the debugging ecosystem is not there.



The historical approach to compartmentalizing software is too complex.

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

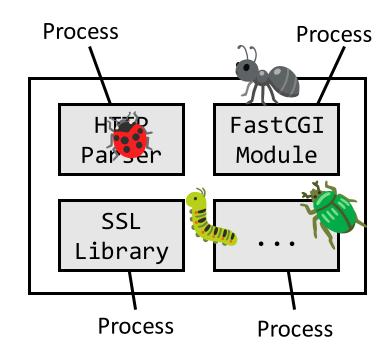
- There will be bugs to fix.
- Bugs now occur across security boundaries.
- This impacts the debugging experience but the debugging ecosystem is not there.



The historical approach to compartmentalizing software is too complex.

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

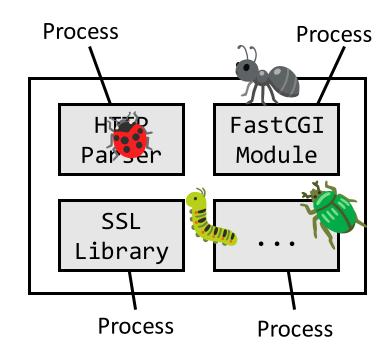
- There will be bugs to fix.
- Bugs now occur across security boundaries.
- This impacts the debugging experience but the debugging ecosystem is not there.



The historical approach to compartmentalizing software is too complex.

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

- There will be bugs to fix.
- Bugs now occur across security boundaries.
- This impacts the debugging experience but the **debugging ecosystem** is not there.

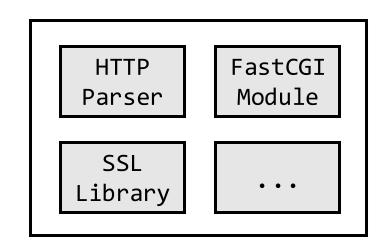


The historical approach to compartmentalizing software is too complex.

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

Consider maintenance!

- Developers will need to maintain the partitioned program.
- Over time, the partitioning strategy may need to change.
- Again, there is historically **no tooling** to handle that safely and avoid regressions.

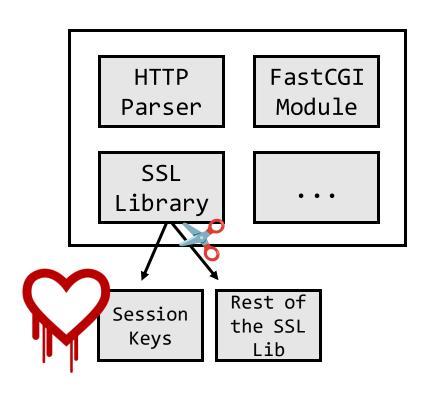


The historical approach to compartmentalizing software is too complex.

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

Consider maintenance!

- Developers will need to maintain the partitioned program.
- Over time, the partitioning strategy may need to change.
- Again, there is historically **no tooling** to handle that safely and avoid regressions.

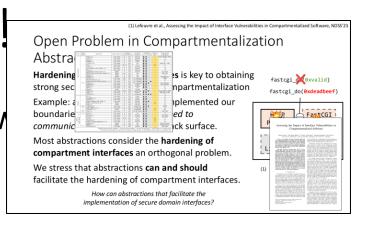


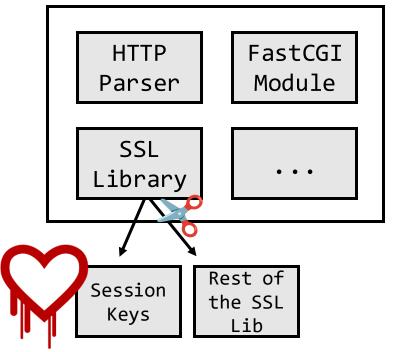
The historical approach to compartmentalizing softv

Worse yet, it considers compartmentalization as a one-off operation, which it isn't.

Consider maintenance!

- Developers will need to maintain the partitioned program.
- Over time, the partitioning strategy may need to change.
- Again, there is historically no tooling to handle that safely and avoid regressions.





Potential obstacle #4: The lack of a systematic approach?

Potential obstacle #4: The lack of a systematic approach?









The historical approach to compartmentalization does not scale:

- Historically, compartmentalization is deeply custom to each program
- Compartmentalize program A, then start all over again with B

Can we make compartmentalization efforts reusable?

- Can we compartmentalize component *C*, and reuse that component compartmentalized everywhere?
- Analogy to the "software crisis" from the 1960s:
 - The answer to "we are not producing enough software" is to build software from reusable components(1)

Potential obstacle #4: The lack of a systematic approach?









The historical approach to compartmentalization does not scale:

- Historically, compartmentalization is deeply custom to each program
- Compartmentalize program A, then start all over again with B

Can we make compartmentalization efforts reusable?

- Can we compartmentalize component C, and reuse that component compartmentalized everywhere?
- Analogy to the "software crisis" from the 1960s:
 - The answer to "we are not producing enough software" is to build software from reusable components⁽¹⁾

Potential obstacle #4: The lack of a systematic approach?









The historical approach to compartmentalization does not scale:

- Historically, compartmentalization is deeply custom to each program
- Compartmentalize program A, then start all over again with B

Can we make compartmentalization efforts reusable?

- Can we compartmentalize component C, and reuse that component compartmentalized everywhere?
- Analogy to the "software crisis" from the 1960s:
 - The answer to "we are not producing enough software" is to build software from reusable components⁽¹⁾

A golden age

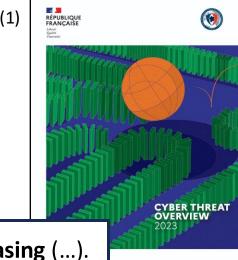
More than ever we need more secure software



The level of cyber threat keeps on increasing (...). Today, ANSSI assesses that attackers linked to China, Russia and cybercrime pose the greatest threat to the most critical networks and to the French ecosystem in a systemic way.

*emphasis mine

- More than ever we need more secure software
- Compartmentalization absolutely fits the needs



(2)

The level of **cyber threat keeps on increasing** (...).

Secure Hardware Foundation: Incorporate architectural features that **enable fine-grained memory protection**, such as those described by Capability Hardware Enhanced RISC Instructions (CHERI)

esses that attackers linked to d cybercrime pose the greatest st critical networks and to the m in a systemic way.

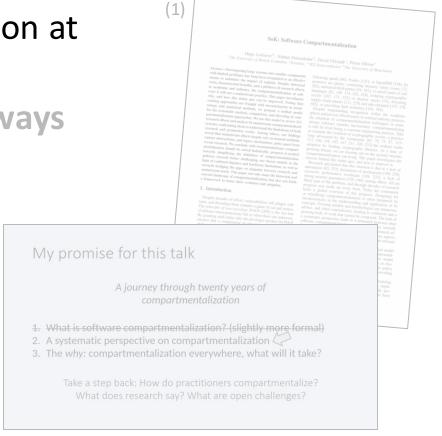
*emphasis mine

Co-signed by, among others, UK, NL, DE, NO

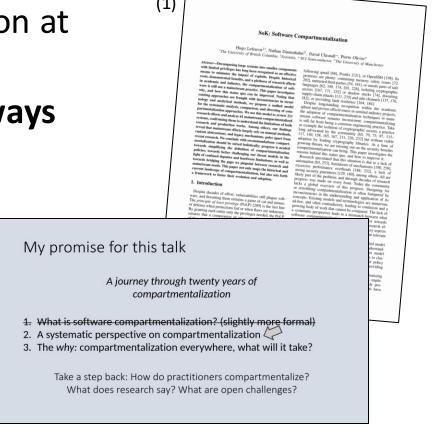
⁽²⁾ https://www.cisa.gov/sites/default/files/2023-10/SecureByDesign_1025_508c.pdf

⁽¹⁾ https://www.cert.ssi.gouv.fr/uploads/CERTFR-2024-CTI-002.pdf

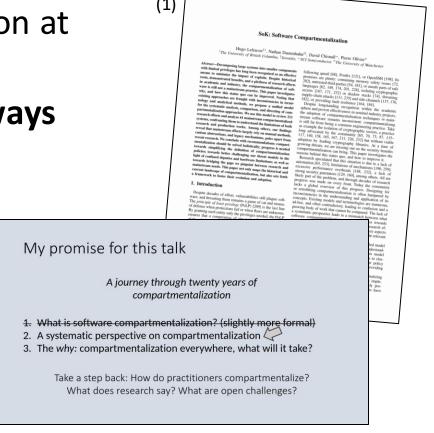
- The challenges to push compartmentalization at scale are **not new**:
 - Research has explored them in many ways and come up with solutions that work (second part of this talk!)
 - These advances are just asking to be developed and mainstreamed



- The challenges to push compartmentalization at scale are not new:
 - Research has explored them in many ways and come up with solutions that work (second part of this talk!)
 - These advances are just asking to be developed and mainstreamed



- The challenges to push compartmentalization at scale are **not new**:
 - Research has explored them in many ways and come up with solutions that work (second part of this talk!)
 - These advances are just asking to be developed and mainstreamed



The missing frameworks are emerging

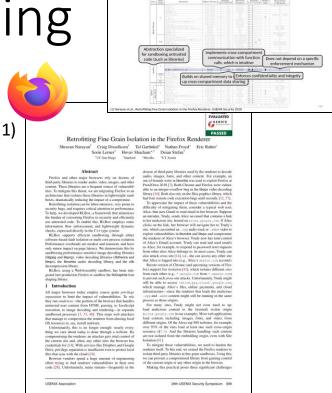
Take RLBox, discussed earlier in the talk

- Productized in Firefox for library isolation
- You can use it: https://github.com/PLSysSec/rlbox

Recently: increasing interest in coming together to create industry standards for compartmentalization

The Open Robust Compartmentalization Alliance
was just accepted as Linux Foundation
project: https://github.com/ORCA-LF/governance

Aiming to foster the adoption and standardization of compartmentalization practices



(1) Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Render, USENIX Security 2020

The missing frameworks are emerging

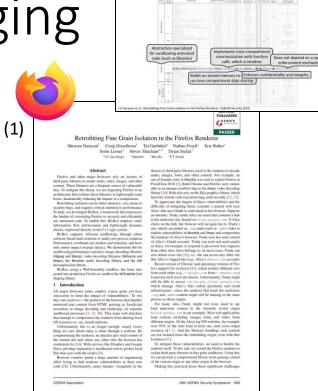
Take RLBox, discussed earlier in the talk

- Productized in Firefox for library isolation
- You can use it: https://github.com/PLSysSec/rlbox

Recently: increasing interest in coming together to create industry standards for compartmentalization

 The Open Robust Compartmentalization Alliance was just accepted as Linux Foundation project: https://github.com/ORCA-LF/governance

Aiming to foster the adoption and standardization of compartmentalization practices



(1) Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Render, USENIX Security 2020

The missing frameworks are emerging

Take RLBox, discussed earlier in the talk

- Productized in Firefox for library isolation
- You can use it: https://github.com/PLSysSec/rlbox

Recently: increasing interest in coming together to create industry standards for compartmentalization

The Open Robust Compartmentalization Alliance
was just accepted as Linux Foundation
project: https://github.com/ORCA-LF/governance

(1) Narayan et al., Retrofitting Fine Grain Isolation in the Firefox Render, USENIX Security 2020

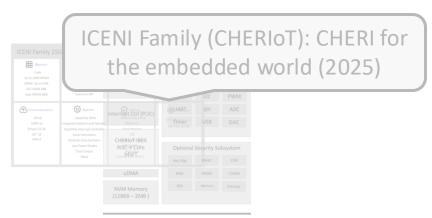
can send me a message.

The hardware ecosystem is emerging

Take CHERI, discussed earlier as well.

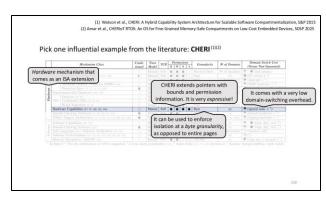
It is not a research prototype. You can use it!

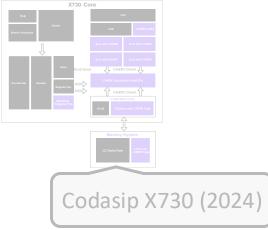




Recently: creation of the *CHERI Alliance* to support industry standards and foster adoption

• Join the CHERI Alliance! https://cheri-alliance.org/



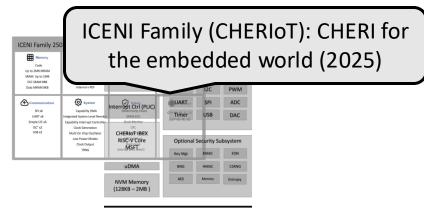


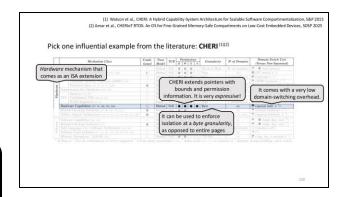
The hardware ecosystem is emerging

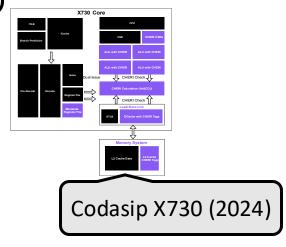
Take CHERI, discussed earlier as well.

It is not a research prototype. You can use it!









Recently: creation of the *CHERI Alliance* to support industry standards and foster adoption

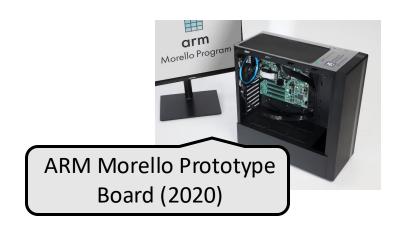
• Join the CHERI Alliance! https://cheri-alliance.org/

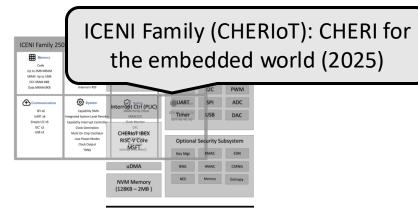
The hardware ecosystem is emerging

Take CHERI, discussed earlier as well.

It is not a research prototype. You can use it!

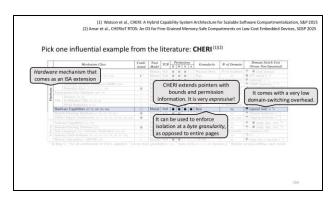
industry standards and foster adoption

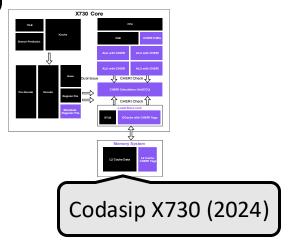




Recently: creation of the CHERI Alliance to support

Join the CHERI Alliance! https://cheri-alliance.org/





Software compartmentalization everywhere: what will it take?

- Open Problem in Enforcement Mechanisms

 Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (see farmise them is the 501

 Some (popular) in Mechanisms do not fulfill all these properties

 Open Problem in Compartmentalization (see farmise them is the 501

 Open Problem in Compartmentalization partmentalization (see farmise the farmise the farmise the farmise the farmise the farmise the farmise the farmise the farmise the farmise of compartment interfaces. In ordinary and farmise the farmise of compartment interfaces an orthogonal problem.

 We stress that abstractions consider the hardening of compartment interfaces. In ordinary the farmise the farmise of compartment interfaces. In ordinary the farmise the partmentalization of consequent interfaces. In ordinary the farmise the partmentalization of process may be farmise the partmentalization of process may be farmise the partmentalization of process may be farmise the partmentalization of process may be farmise the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmed to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmised to the partmentalization of process may be farmed to the partmentalization of process may be farme
- 1. An effort on industry-grade compartmentalization frameworks to lower the bar to compartmentalize software
- 2. An effort on using these frameworks to compartmentalize vulnerable components in a re-usable fashion: can we do this under the umbrella of Linux distributions?
- 3. Supporting the push towards **emerging security-oriented hardware**: integrate CHERI in your hardware products!

Open Problem in Enforcement Mechanisms

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (some must be them to be foot of the foot of

- 1. An effort on industry-grade compartmentalization frameworks to lower the bar to compartmentalize software
- 2. An effort on using these frameworks to compartmentalize vulnerable components in a re-usable fashion: can we do this under the umbrella of Linux distributions?
- 3. Supporting the push towards **emerging security-oriented hardware**: integrate CHERI in your hardware products!

Open Problem in Enforcement Mechanisms

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization (we have been the food of some (popular!) mechanisms do not fulfill all these properties.

Open Problem in Compartmentalization

Abstra

Hardening strong see Fland of the food

- 1. An effort on **industry-grade compartmentalization frameworks** to lower the bar to compartmentalize software
- 2. An effort on using these frameworks to compartmentalize vulnerable components in a re-usable fashion: can we do this under the umbrella of Linux distributions?
- 3. Supporting the push towards **emerging security-oriented hardware**: integrate CHERI in your hardware products!

Open Problem in Enforcement Mechanisms

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization to servation the funding mechanisms do not fulfill all these properties.

Open Problem in Compartmentalization

Abstra

Hardening strong sec

Example:

boundarie

Most abstractions consider the hardening of compartment interfaces an orthogonal problem.

We stress that abstractions can and should facilitate the hardening of compartment interfaces an orthogonal problem.

We stress that abstractions seemed to the hardening of compartment interfaces an orthogonal problem.

We were an observation to secun demonstration effects and the properties of the properties o

- 1. An effort on **industry-grade compartmentalization frameworks** to lower the bar to compartmentalize software
- 2. An effort on using these frameworks to compartmentalize vulnerable components in a re-usable fashion: can we do this under the umbrella of Linux distributions?
- 3. Supporting the push towards **emerging security-oriented hardware**: integrate CHERI in your hardware products!

Open Problem in Enforcement Mechanisms

Mechanisms must fulfill several properties to be suitable to enforce compartmentalization who translate them in the fold of the fulfill of the ful

- 1. An effort on **industry-grade compartmentalization frameworks** to lower the bar to compartmentalize software
- 2. An effort on using these frameworks to compartmentalize vulnerable components in a re-usable fashion: can we do this under the umbrella of Linux distributions?
- 3. Supporting the push towards **emerging security-oriented hardware**: integrate CHERI in your hardware products!



- 1. Compartmentalization is a key practice to **improve the trustworthiness of software**
- Compartmentalization, in its historical form, is hindered by its complexity
- 3. We live at a **golden age to solve this problem**. A call to the community: we can make software more trustworthy!



- Compartmentalization is a key practice to improve the trustworthiness of software
- Compartmentalization, in its historical form, is hindered by its complexity
- 3. We live at a **golden age to solve this problem**. A call to the community: we can make software more trustworthy!



- Compartmentalization is a key practice to improve the trustworthiness of software
- Compartmentalization, in its historical form, is hindered by its complexity
- 3. We live at a **golden age to solve this problem**. A call to the community: we can make software more trustworthy!



- 1. Compartmentalization is a key practice to **improve the trustworthiness of software**
- Compartmentalization, in its historical form, is hindered by its complexity
- 3. We live at a **golden age to solve this problem**. A call to the community: we can make software more trustworthy!



Sponsors:





Collaborators:















Key Takeaways

- Compartmentalization is a key practice to improve the trustworthiness of software
- Compartmentalization, in its historical form, is hindered by its complexity
- We live at a golden age to solve this problem. A call to the community: we can make software more trustworthy!

Questions?

Reach out: hugo.lefeuvre@ubc.ca