The 8th ACM SIGOPS Asia-Pacific Workshop on Systems (APSYS)

Building a Security OS With Software Defined Infrastructure

Guofei Gu (Texas A&M University)
Hongxin Hu (Clemson University)
Eric Keller (University of Colorado, Boulder)
Zhiqiang Lin (University of Texas at Dallas)
Donald Porter (University of North Carolina at Chapel Hill)

September 2nd, 2017

Outline

Outline

Current Approach for Security

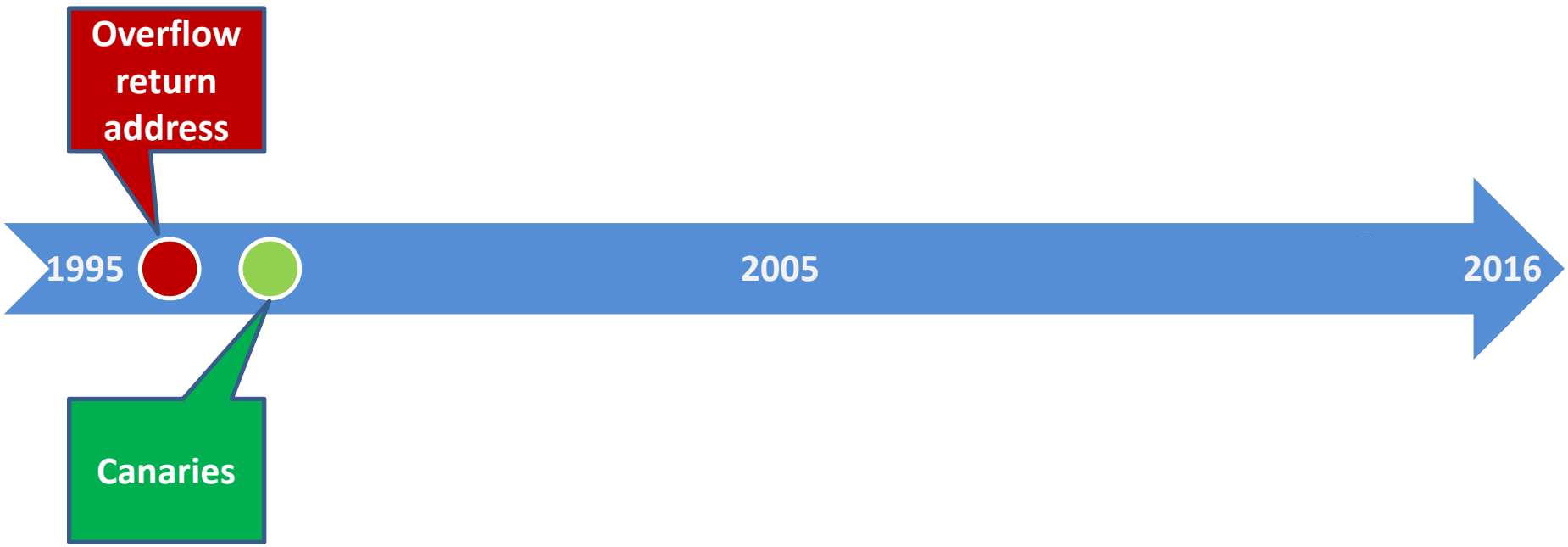
Security as Reactive Add-ons

Current Approach for Security

Security as Reactive Add-ons

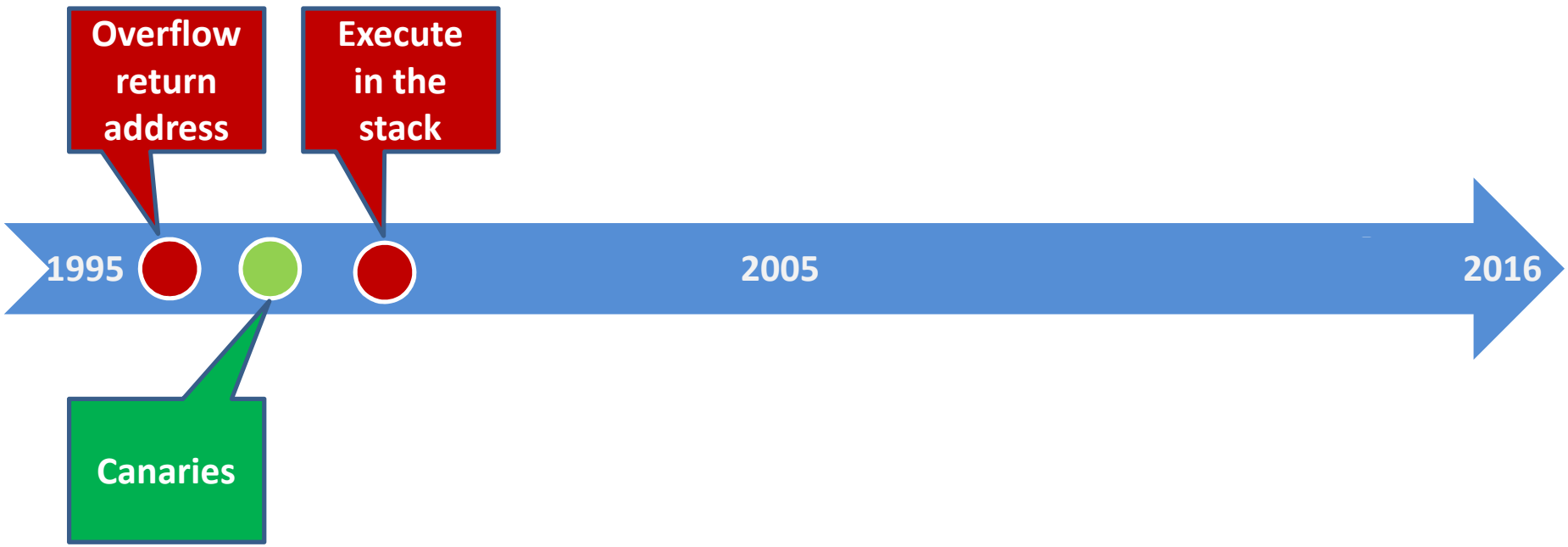
Current Approach for Security

Security as Reactive Add-ons



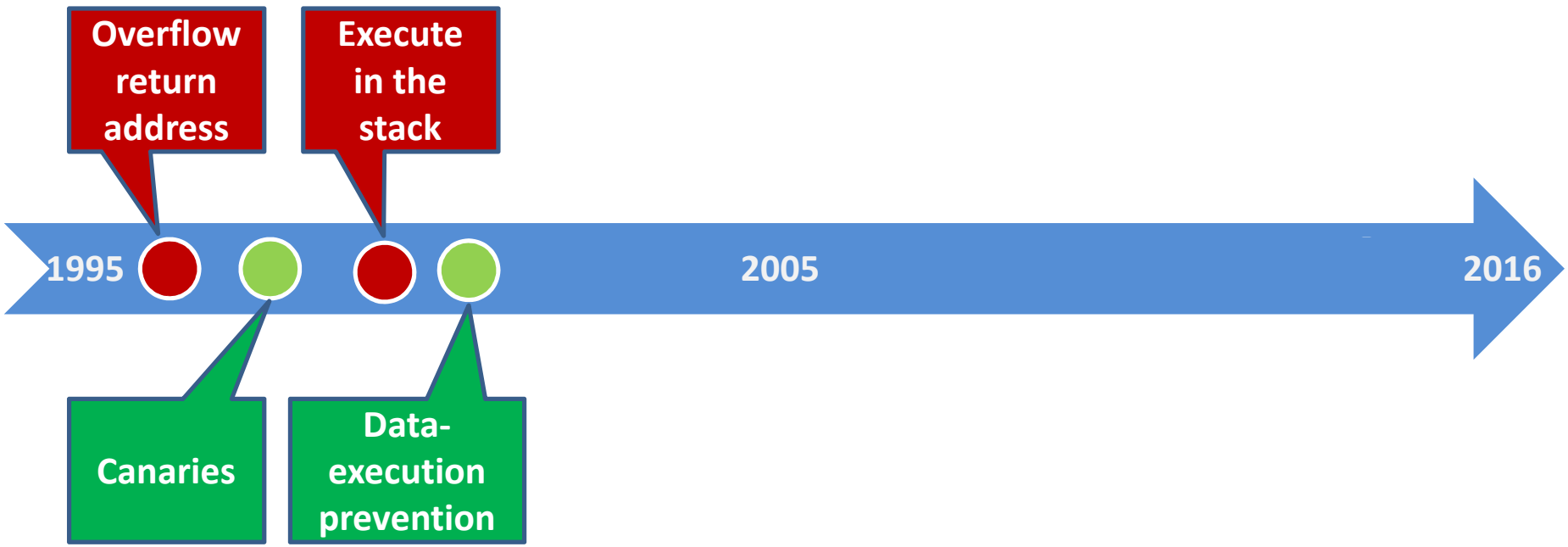
Current Approach for Security

Security as Reactive Add-ons



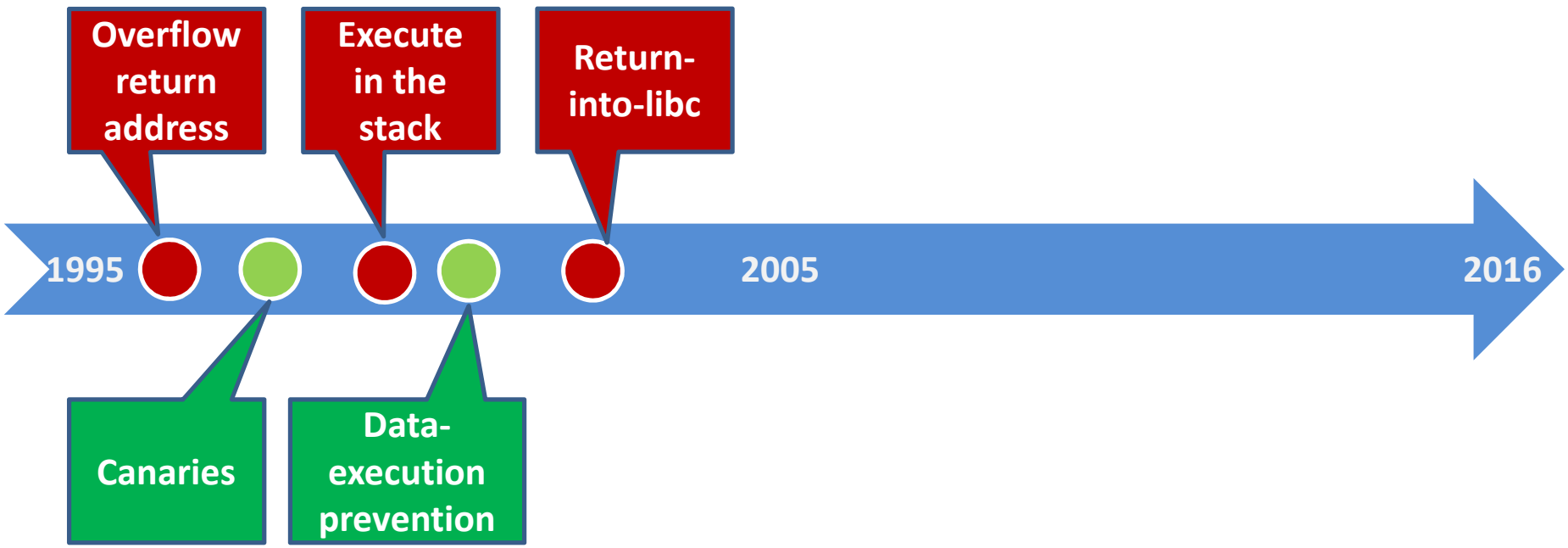
Current Approach for Security

Security as Reactive Add-ons



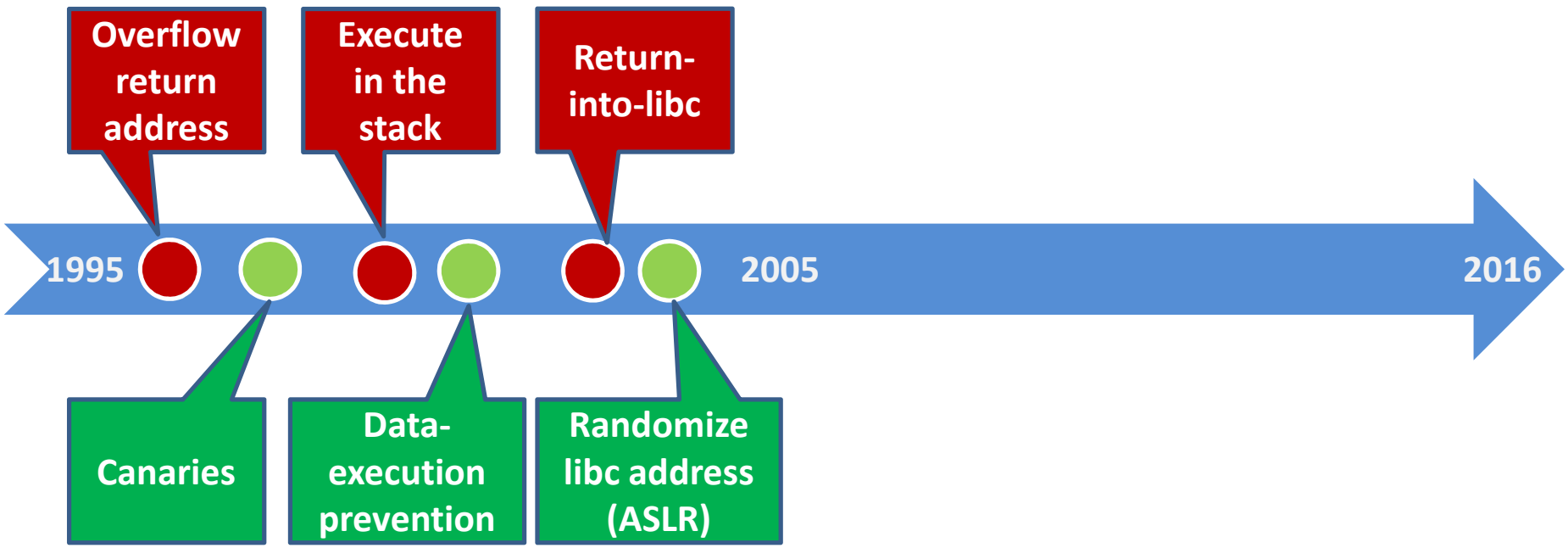
Current Approach for Security

Security as Reactive Add-ons



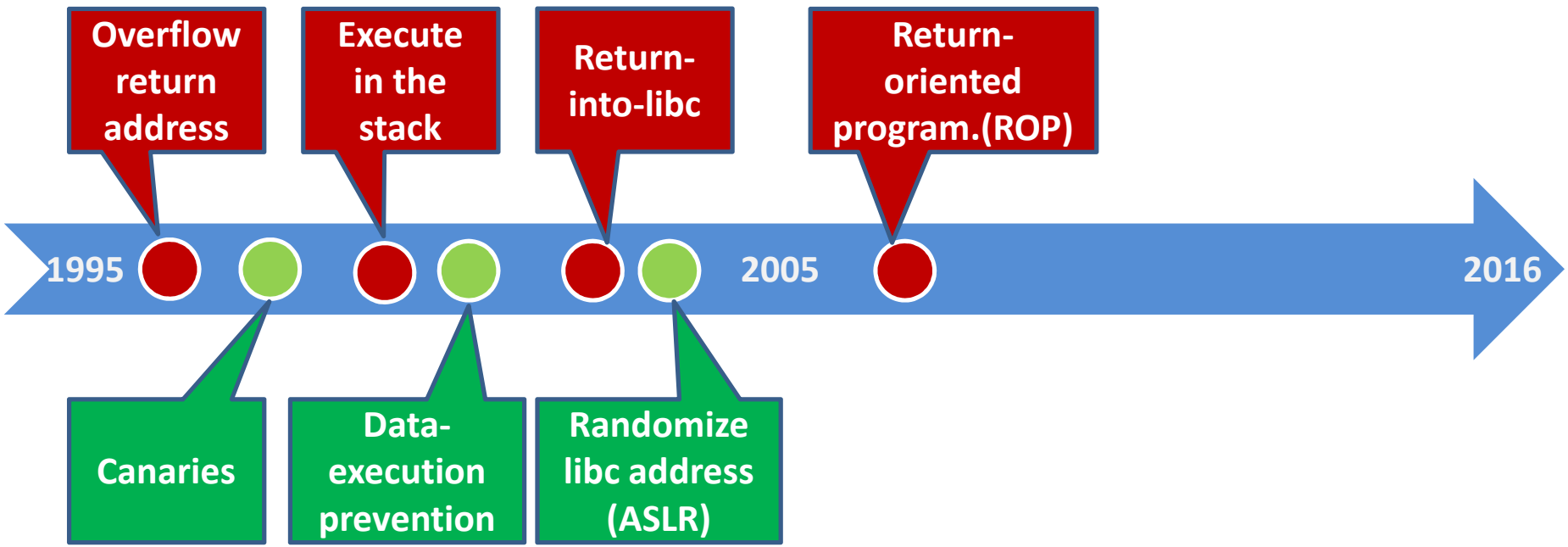
Current Approach for Security

Security as Reactive Add-ons



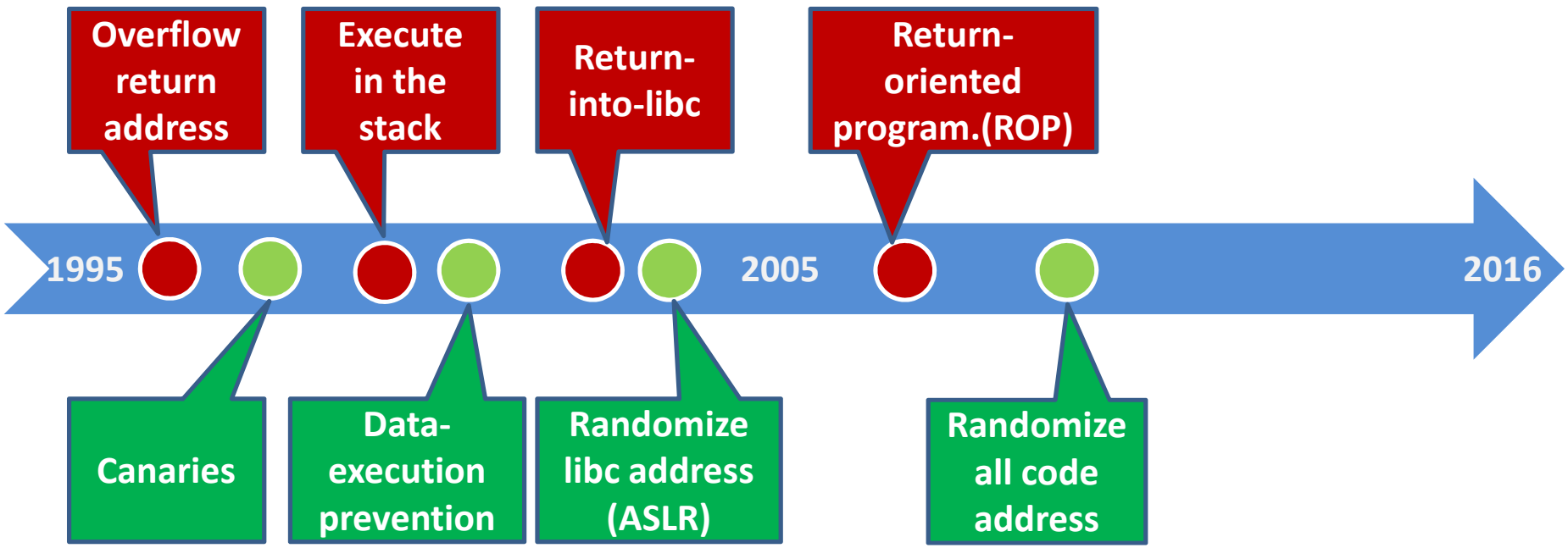
Current Approach for Security

Security as Reactive Add-ons



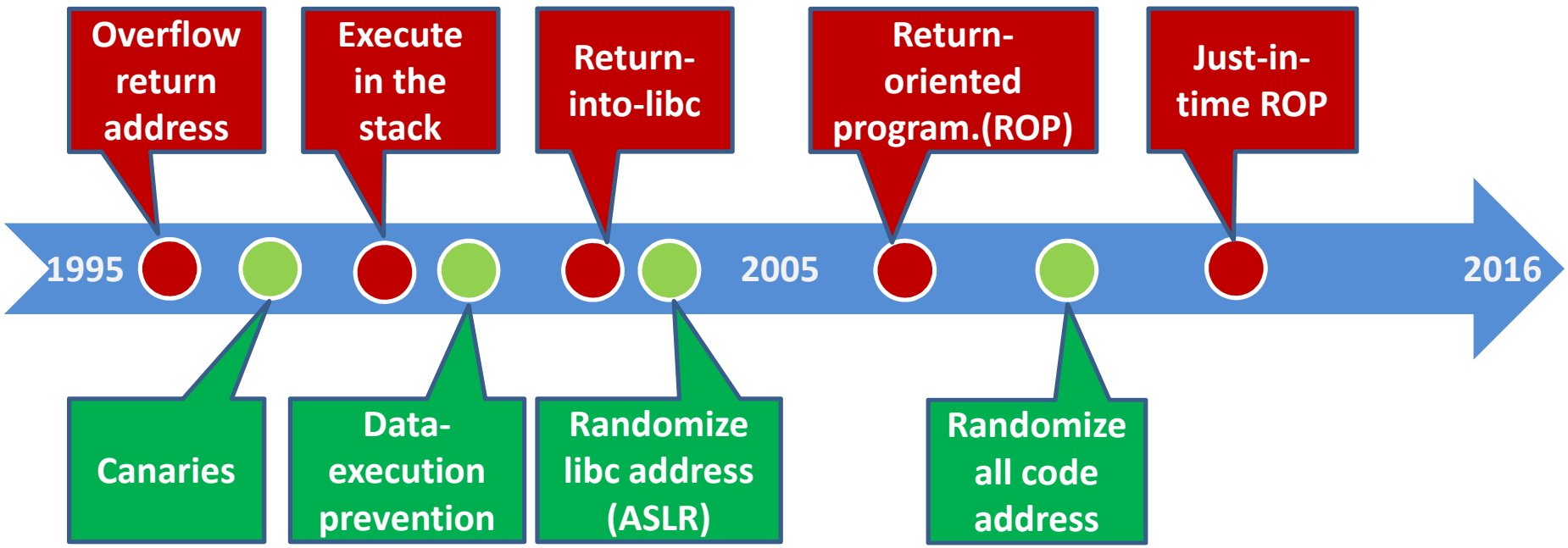
Current Approach for Security

Security as Reactive Add-ons



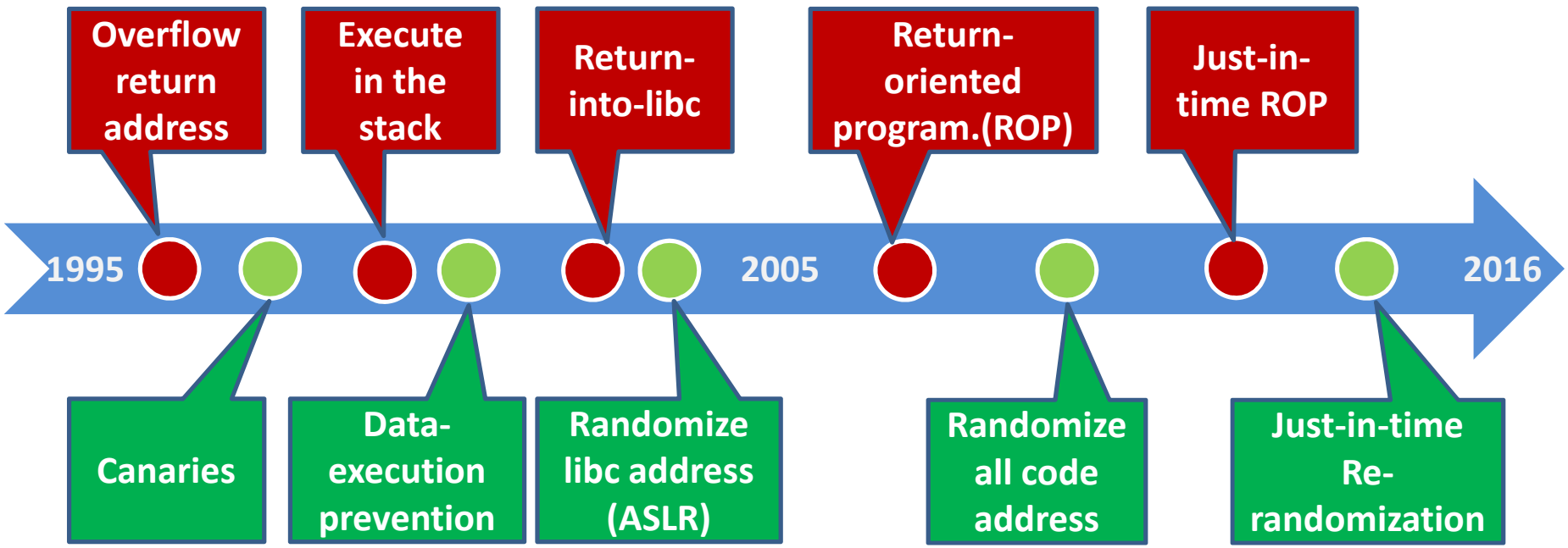
Current Approach for Security

Security as Reactive Add-ons



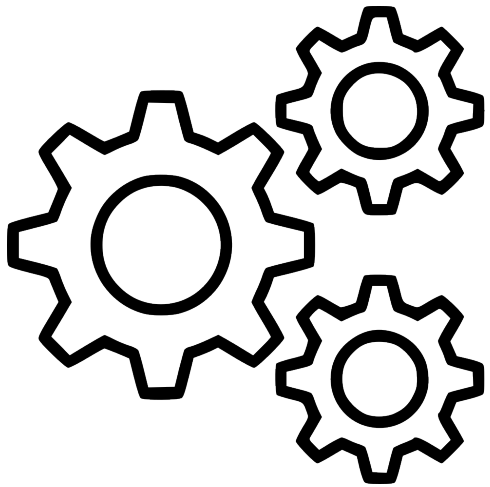
Current Approach for Security

Security as Reactive Add-ons

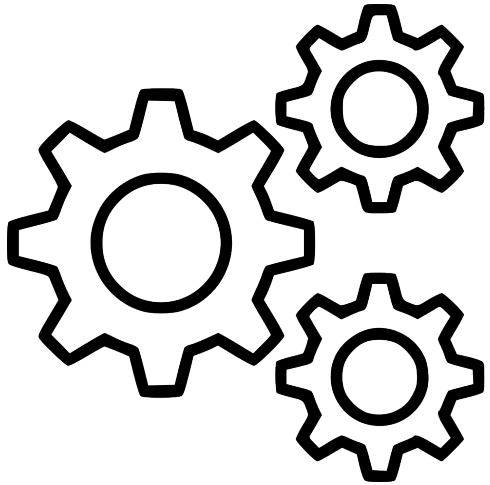


Consequences from Security as Reactive Add-ons

Consequences from Security as Reactive Add-ons



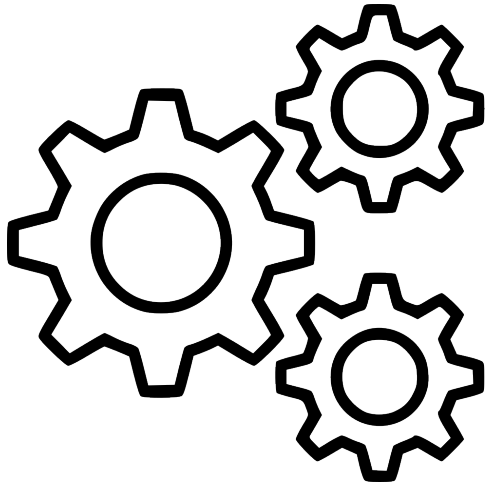
Consequences from Security as Reactive Add-ons



Security Mechanism

1. Coupled with legacy systems
2. Fragmented
3. Hard to configure
4. Hard to reason about

Consequences from Security as Reactive Add-ons



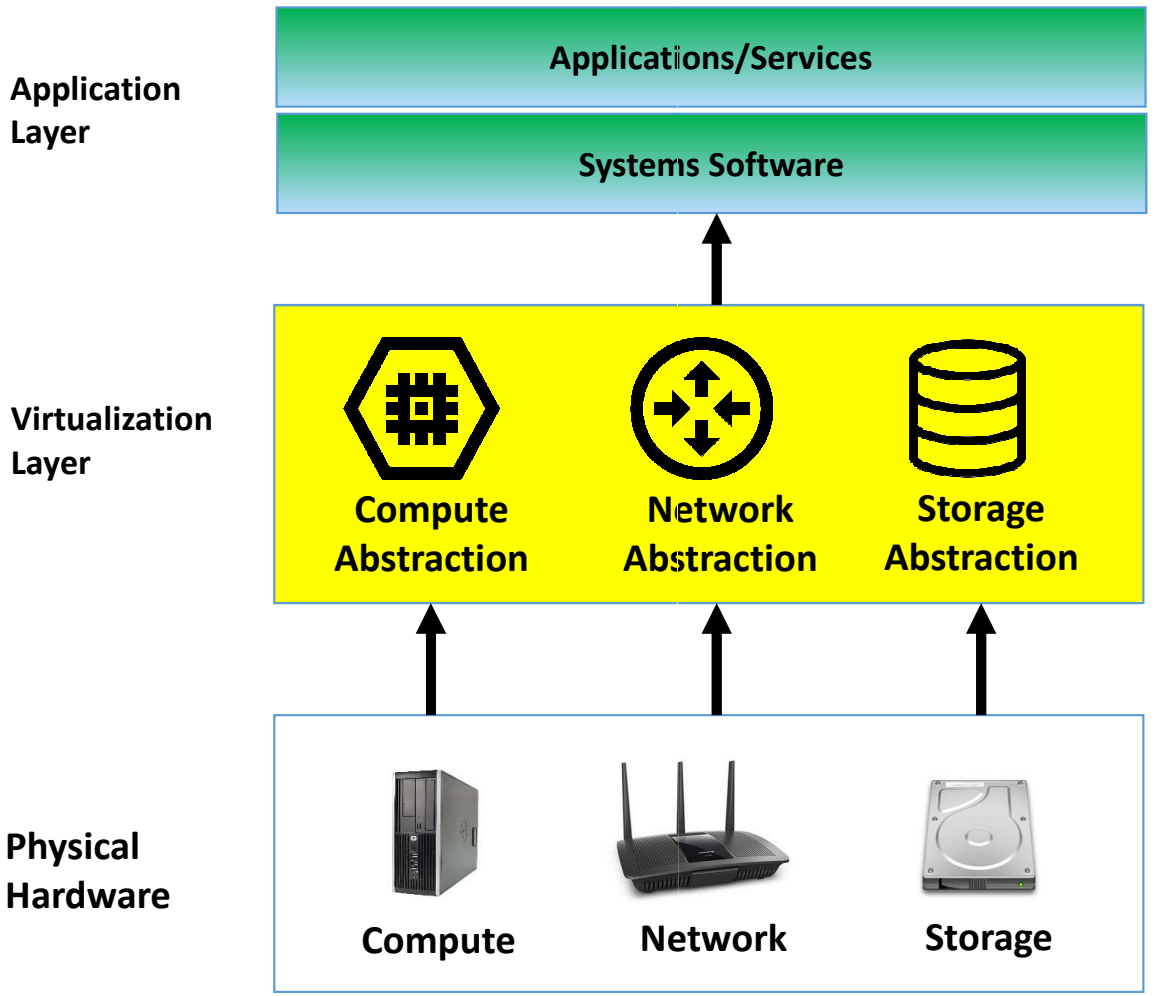
Security Mechanism

1. Coupled with legacy systems
2. Fragmented
3. Hard to configure
4. Hard to reason about

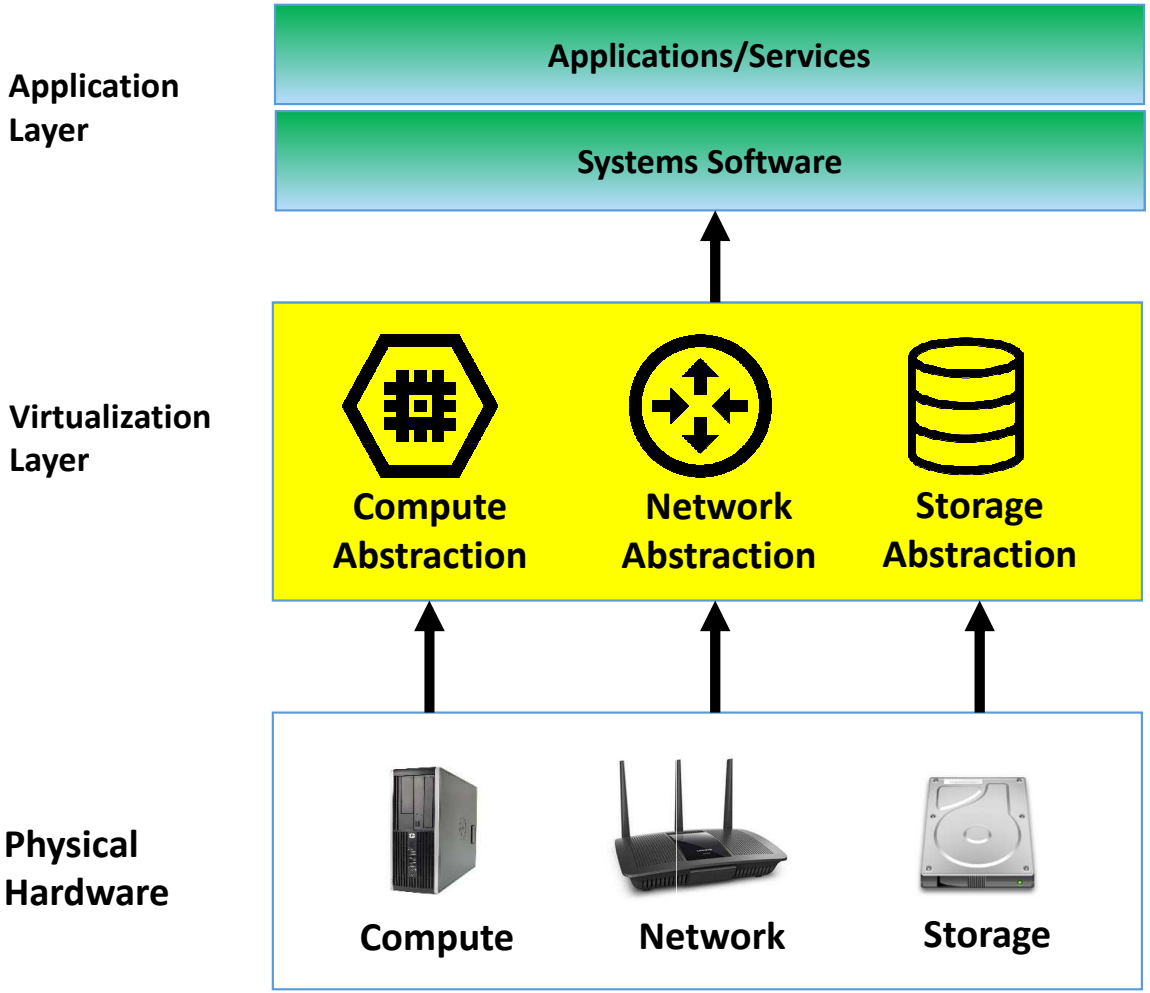
Security Policy

1. Specified w/ low-level artifacts (e.g., processes, IP addrs, port nums)
2. Hard to debug
3. Hard to be consistent

Software Defined Infrastructure (SDI)



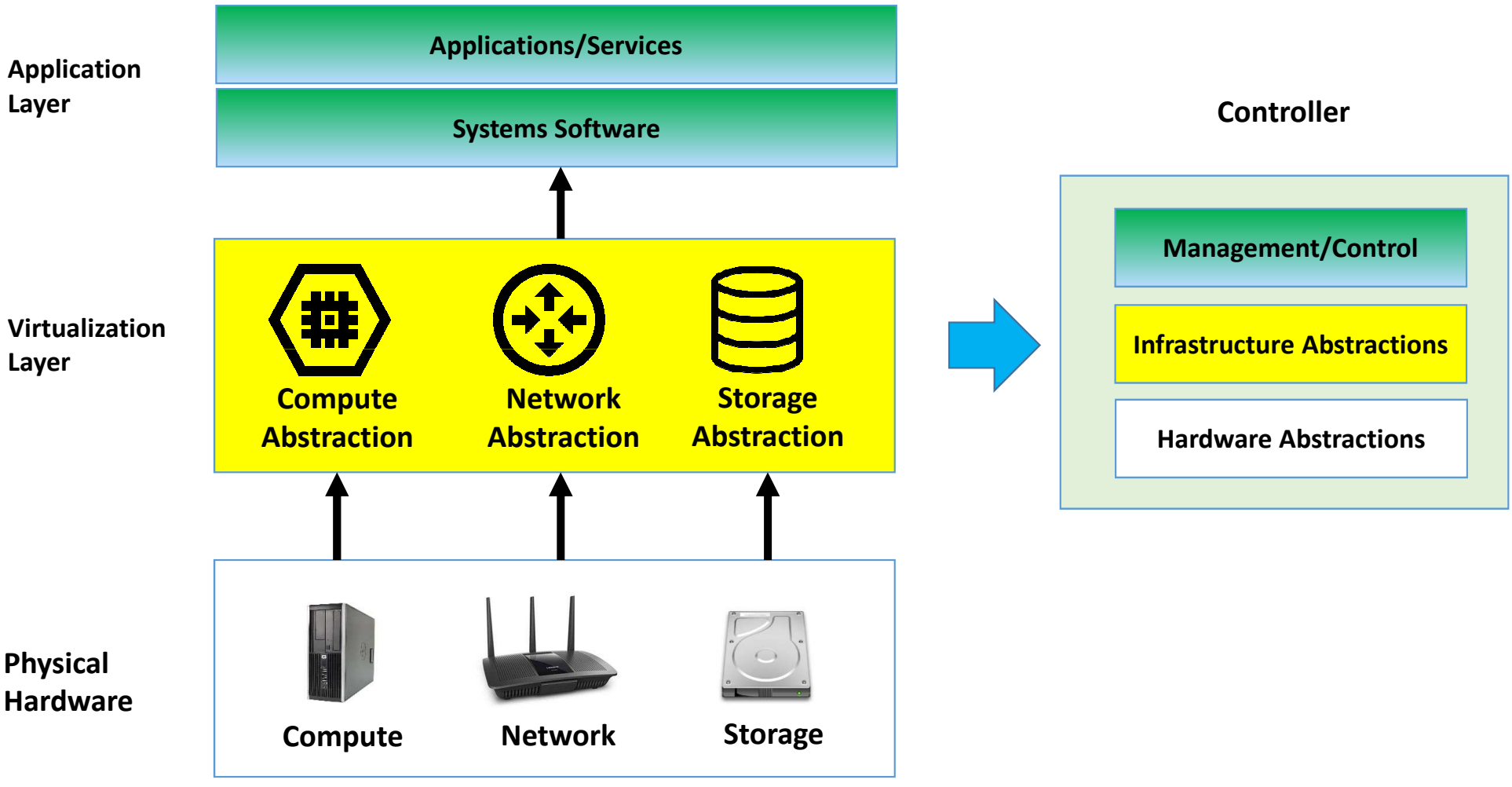
Software Defined Infrastructure (SDI)



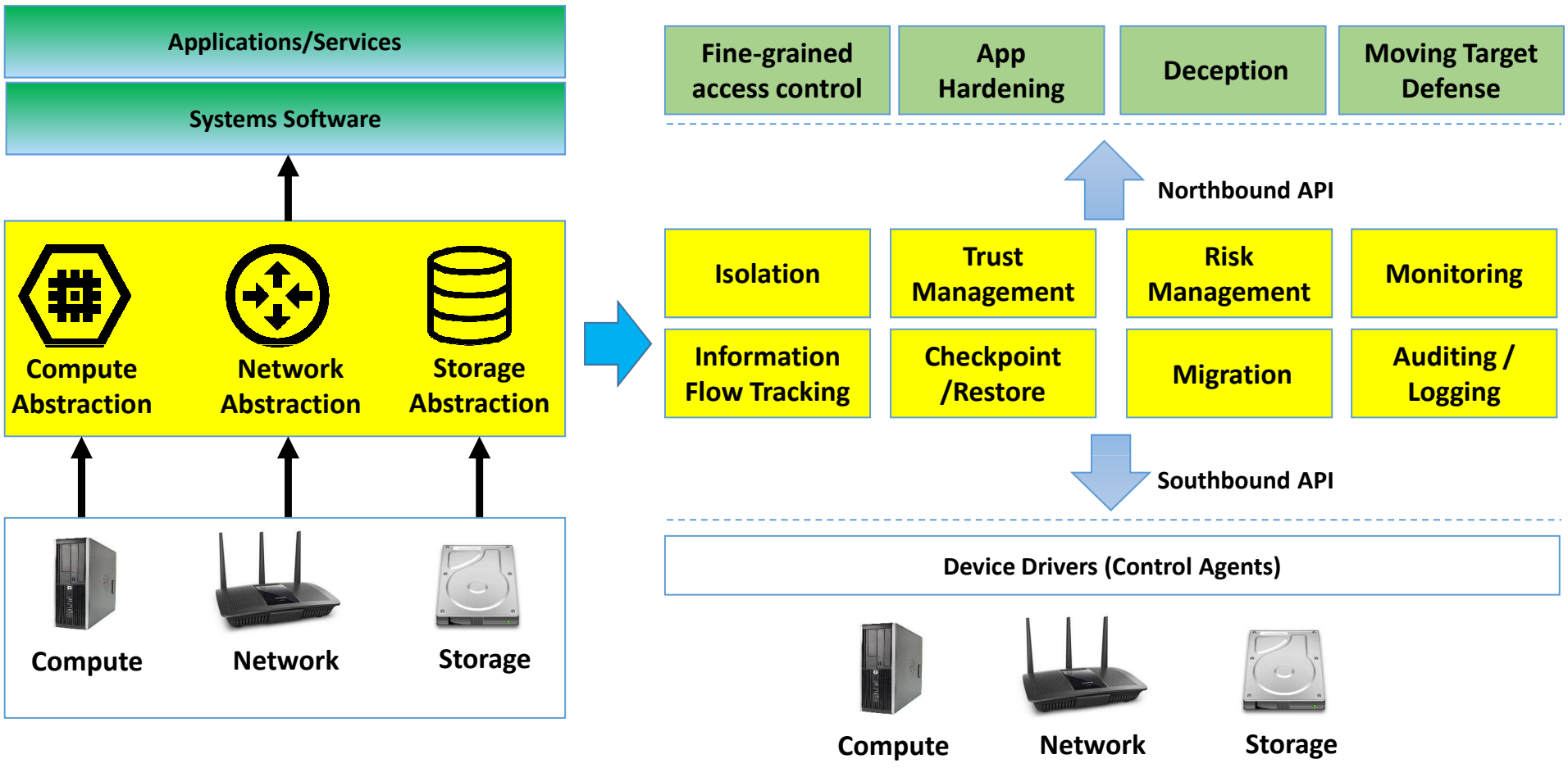
1. **Isolation**
2. Encapsulation
3. **Interposition**
4. Migration
5. Replication
6. Multiplexing
7. Portability
8. Scalability
9. Sandboxing
10. Elasticity
11. ...

Outline

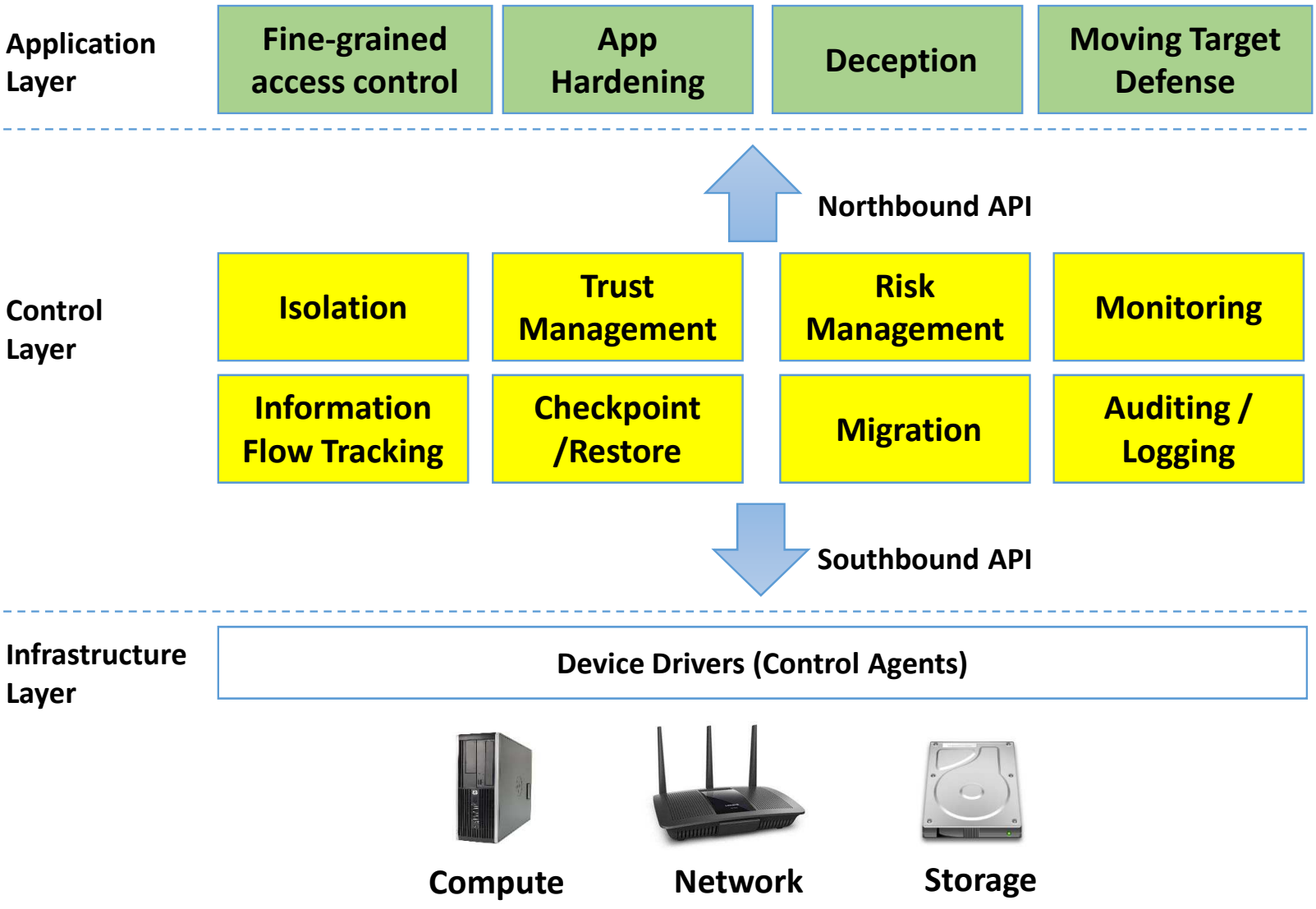
S²OS:
Introducing **S**DI Defined **S**ecurity **OS**



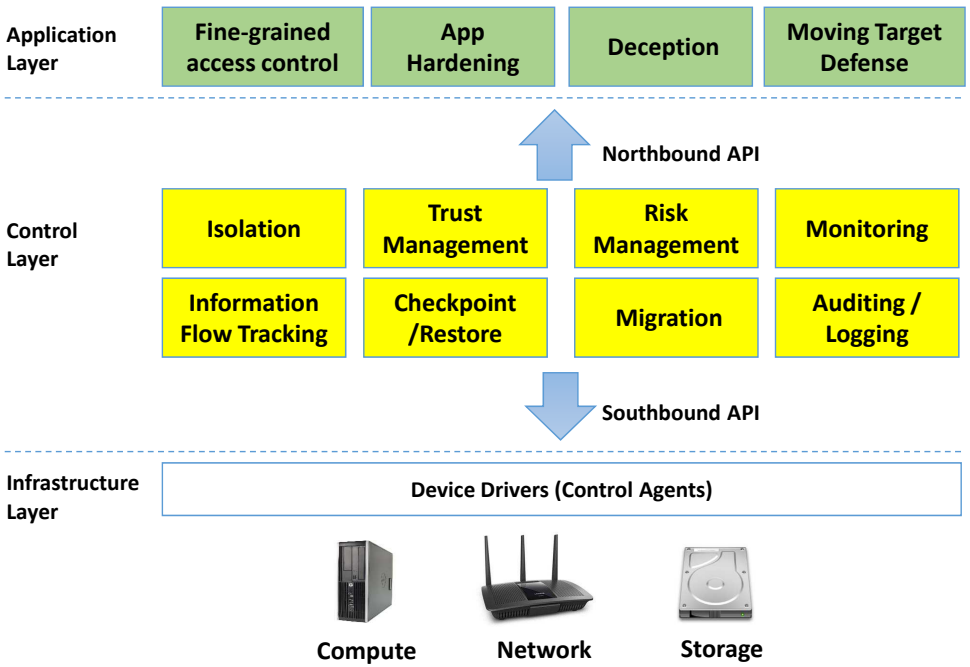
S²OS:
Introducing **S**DI Defined **S**ecurity **OS**



S²OS:
Introducing **S**DI Defined **S**ecurity **OS**



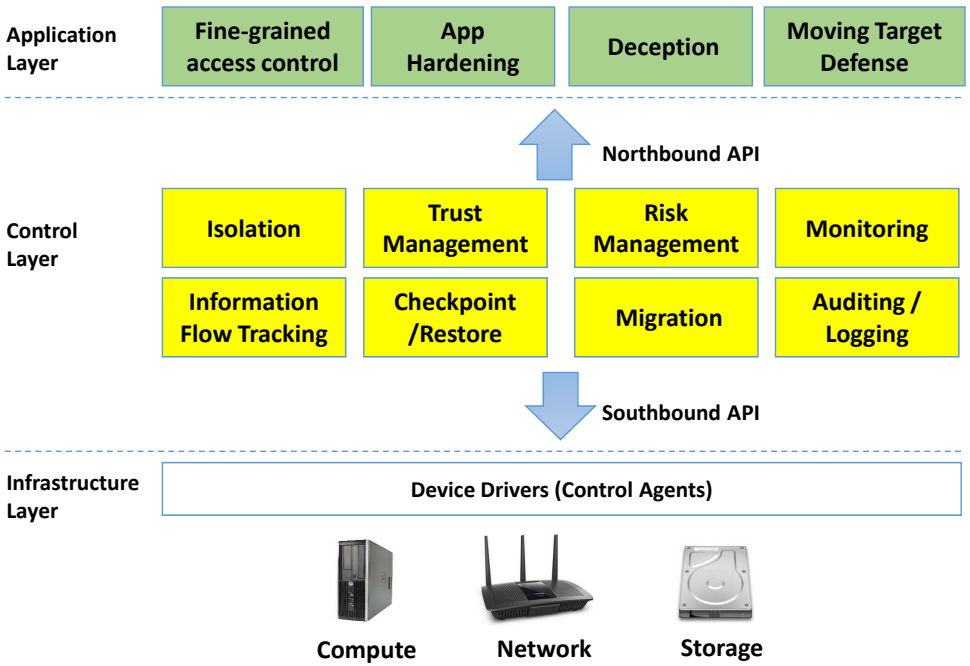
S²OS:
Key objectives



1. Abstracts **security** capabilities and **primitives** at both host OS and network levels
2. Offers an easy-to-use and **programmable security** model for monitoring and dynamically securing applications

S²OS:

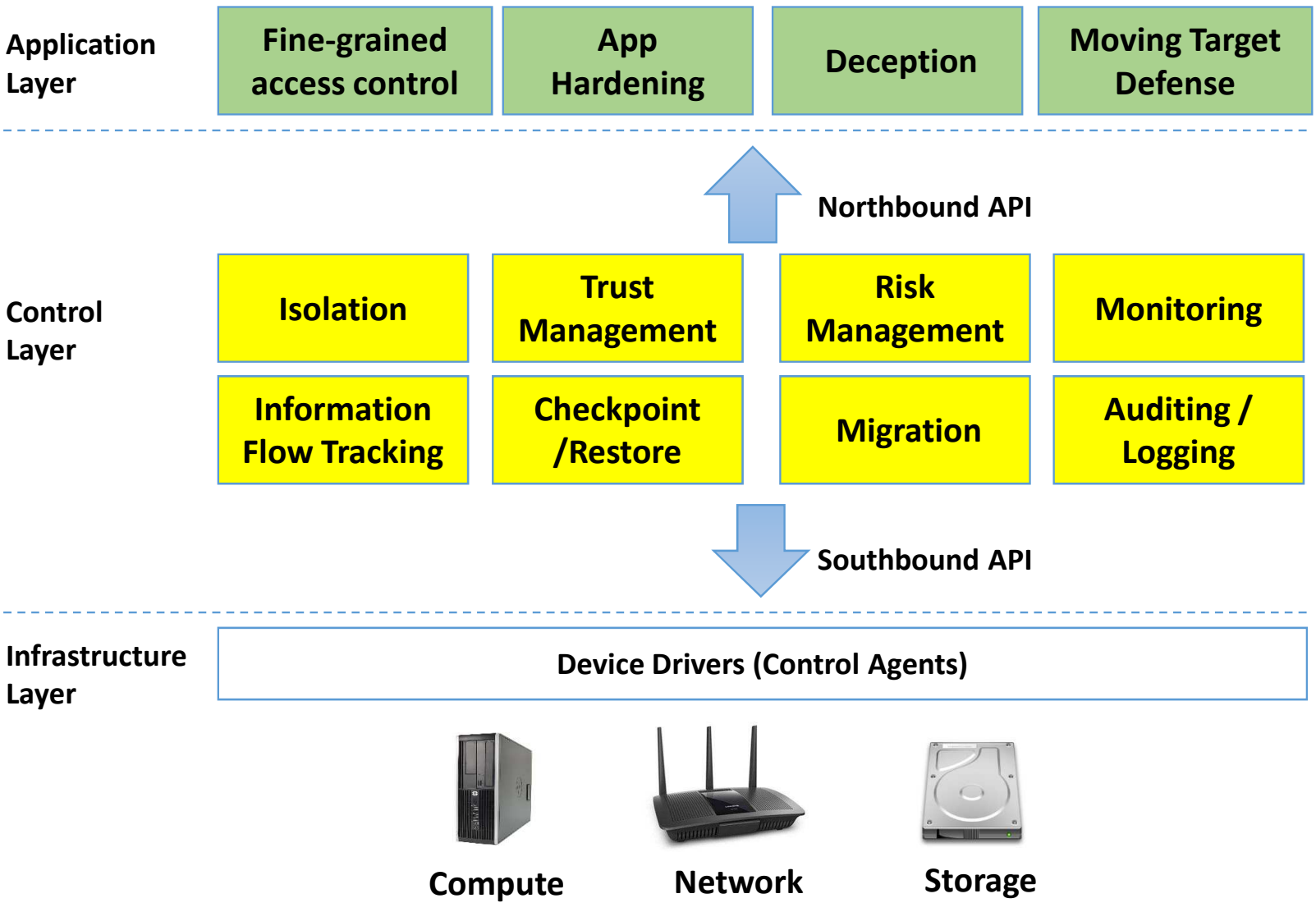
It could unlock a range of unprecedented security opportunities



1. Fine-grained, dynamic security **programmability** at entire infrastructure scale
2. **Information flow tracking** across an entire data center
3. Easily translating global security goals into local OSes and network **policies** securing applications

Outline

S²OS:


Application Layer
Fine-grained access control
App Hardening
Deception
Moving Target Defense
Northbound API
Control Layer
Isolation
Trust Management
Risk Management
Monitoring
Information Flow Tracking
Checkpoint /Restore
Migration
Auditing / Logging
Southbound API
Infrastructure Layer
Device Drivers (Control Agents)
Compute
Network
Storage

Hardware Layer
I. Host Building Blocks

1. Providing process virtualization (e.g., using Open Container Interface)
2. Providing configurable process-level virtualization (in support of multiple library OS interfaces)

Infrastructure Layer

Device Drivers (Control Agents)

Compute

Network

Storage

Hardware Layer

II. Networking Building Block

1. Extending OpenFlow-enabled Switches with Custom Functions
2. Extending OVS w/ App and Context Awareness on Hosts and Mobile Devices

Infrastructure Layer

Device Drivers (Control Agents)

Compute

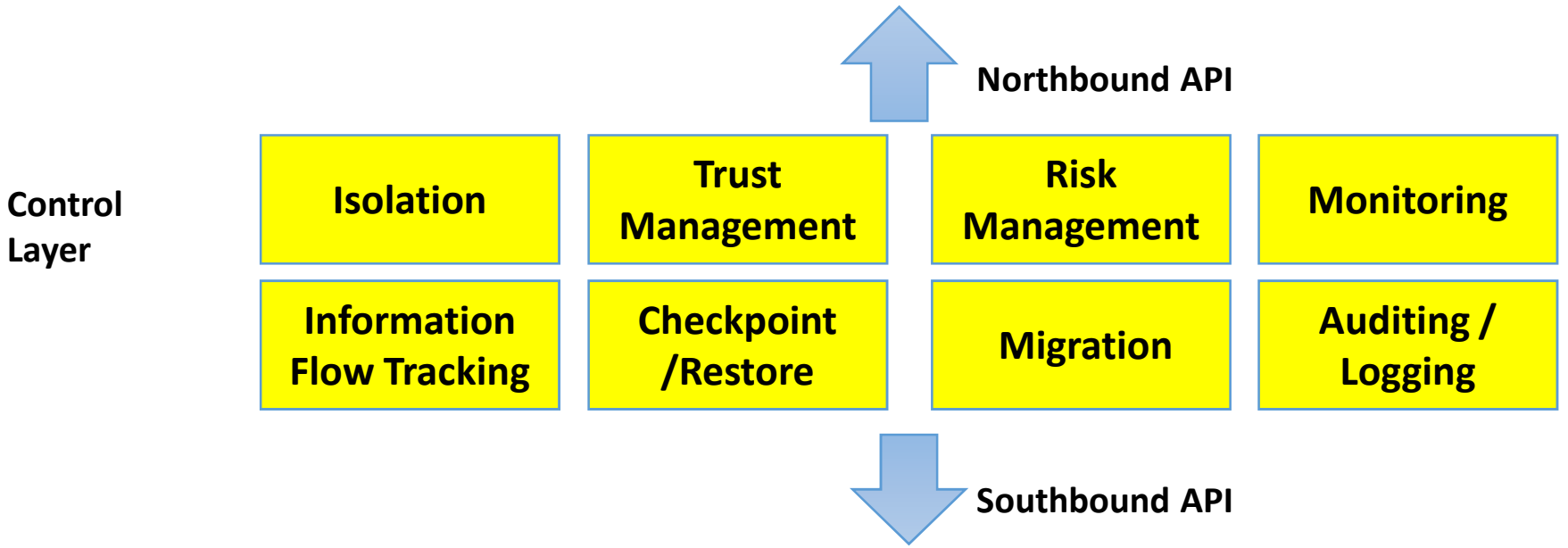
Network

Storage

Control Layer
Security APIs

- Strong Isolation
- Least Privilege
- Complete Mediation
- Holistic Programmability
- Complete Visibility

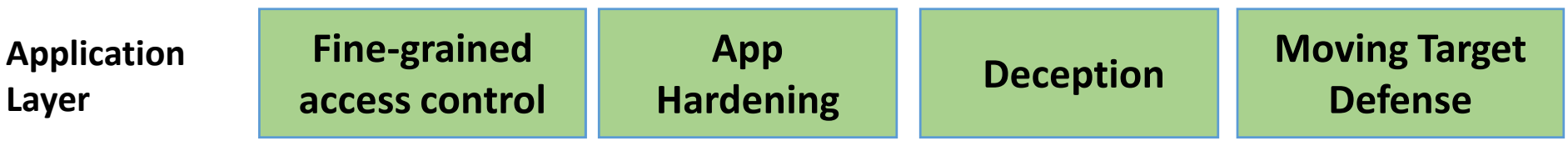
- High Flexibility
- Full Automation
- Trustworthiness
- Maneuvering
- Diversity



Application Layer

Developing Security Applications w/ the Abstractions (APIs)

- Application Development Script Language
- Example Security Applications
  1. Fine-grained Access Control
  2. Application Hardening
  3. Deception
  4. Moving Target Defense



Research Challenges

Control layer vs. infrastructure layer/plane separation

1. **Computing/OS**
   - Control Plane: Reference monitors
   - Data Plane: I/O paths (IPC, net, storage)
2. **Networking**
   - Similar to SDN
3. **Storage**
   - Control Plane: VFS, Ref. Monitor, Storage Servers
   - Data Plane: I/O path through hypervisor, over SDN channels to device

Research Challenges

Challenges stem from virtualizations

1. **Trust**
   - The root of trust
   - Discovering of usable policies and abstractions for non-binary trust
2. **Introspection**
   - The semantic-gap
3. **Practical information flow tracking**
   - Enforce useful policies without choking on “label creep”

Outline

Outcome and Evaluation

1. Outcome
   - The security OS, S^2OS, and a set of components/modules
   - A set of example security apps on top of S^2OS
2. Testbed
   - CloudLab
   - Campus SDN/Internet 2/GENI platform
3. Metrics
   - Performance Overhead
   - Effectiveness
   - Scalability
   - Usability

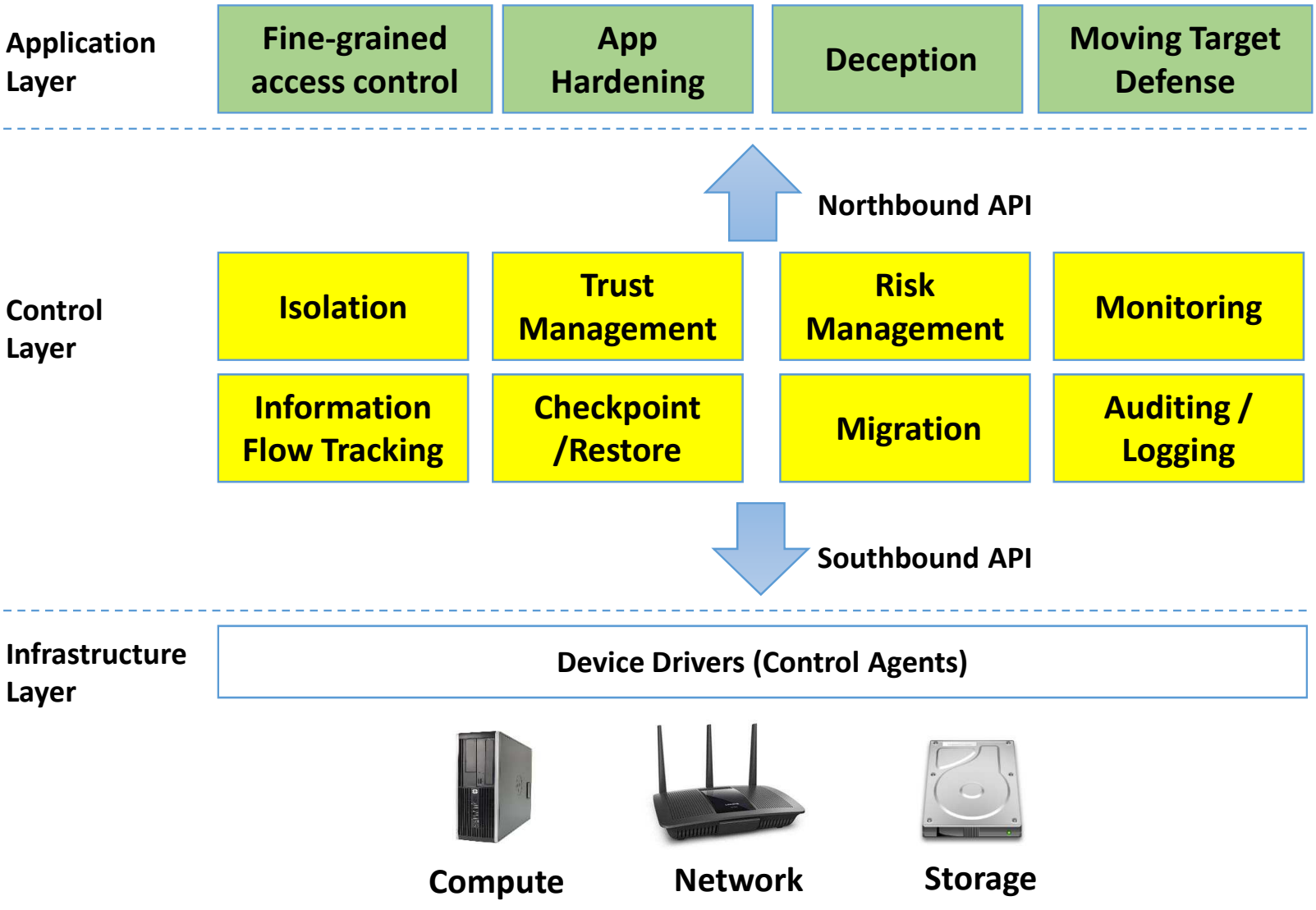
Outcome and Evaluation

1. Outcome
   - The security OS, S^2OS, and a set of components/modules
   - A set of example security apps on top of S^2OS
2. Testbed
   - CloudLab
   - Campus SDN/Internet 2/GENI platform
3. Metrics
   - Performance Overhead
   - Effectiveness
   - Scalability
   - Usability

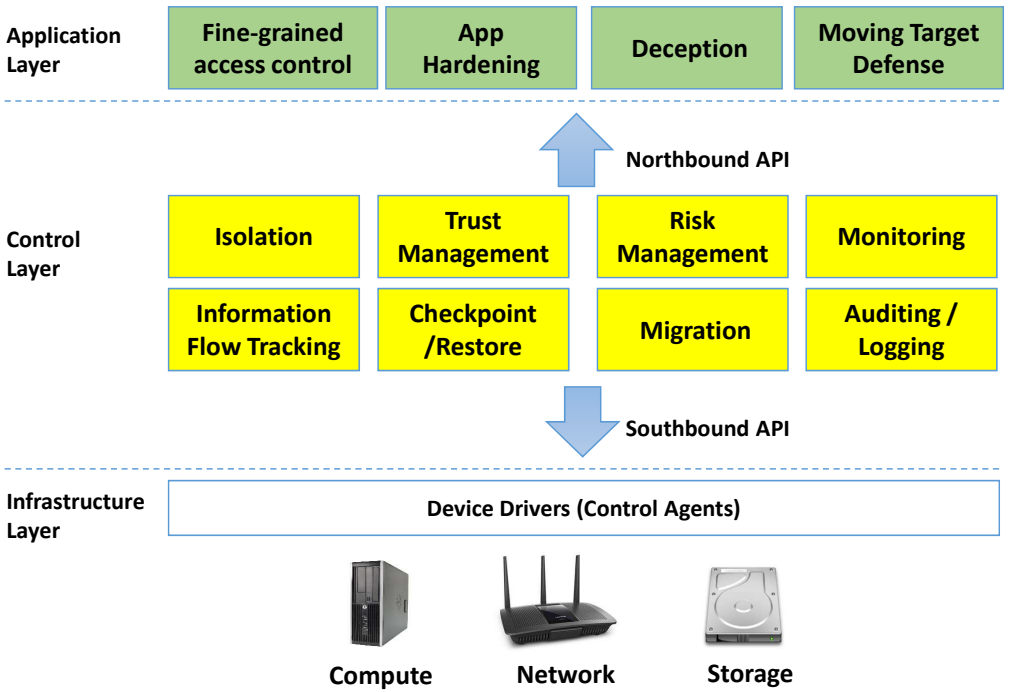
The source code related to this project will be released at
`http://success.cse.tamu.edu/S2OS/`

Outline

S²OS: **S**DI Defined **S**ecurity **OS**



Thank You



guofei@cse.tamu.edu
hongxih@clemson.edu
eric.keller@colorado.edu
zhiqiang.lin@utdallas.edu
porter@cs.unc.edu

Thanks also to the National Science Foundation (NSF) and VMware for their sponsorship of this research.