COMPTIA SECURITY+ SY0-701 · WEEK 7

Security Implications of Architecture Models



Objective 3.1 · 90-minute lesson

Presenter: Trapper Brown

What you will be able to do in 90 minutes

1. Assign security duties in IaaS, PaaS, and SaaS using a responsibility matrix.
2. Explain the security implications of hybrid cloud and third-party vendors.
3. Compare IaC, serverless, microservices, containers, and virtualization for security.
4. Distinguish air-gapped, logically segmented, and software-defined networks.
5. Describe the risks of IoT, ICS and SCADA, RTOS, and embedded systems.
6. Weigh an architecture against availability, cost, patching, power, and other factors.
7. Calculate yearly downtime from an availability percentage.

AGENDA

3 min	Architecture decides the security work
12 min	Cloud and the responsibility matrix
7 min	Hybrid considerations and third parties
12 min	IaC, serverless, microservices, containers
8 min	Network infrastructure choices
6 min	Centralized and decentralized control
10 min	Exercise 1: pick the architecture
10 min	IoT, ICS and SCADA, RTOS, and embedded
8 min	Considerations for choosing a model
5 min	High availability by the numbers
8 min	Exercise 2: a hospital's specialized systems
1 min	Wrap-up

The responsibility matrix moves with the model

Layer	IaaS	PaaS	SaaS
Data, users, and access	Customer	Customer	Customer
Application	Customer	Customer	Provider (customer configures)
OS, runtime, middleware	Customer	Provider	Provider
Virtualization, hardware	Provider	Provider	Provider
Physical data center	Provider	Provider	Provider

Hybrid setups and vendors add seams

Hybrid considerations

- Two sets of tools, logs, and identities to align
- Data moving between sites must be encrypted
- One policy applied on premises and in the cloud
- Network links and latency between environments

Third-party vendors

- Their security becomes part of yours
- Contracts set duties: SLA, breach notice, audits
- Ask for assurance such as a SOC 2 Type II report
- Plan for the vendor failing or leaving

Code, functions, services, and containers

Infrastructure as code

Servers and networks defined in versioned files.

Terraform, CloudFormation

Serverless

The provider runs your functions; no servers to patch.

AWS Lambda, Azure Functions

Microservices

Small services talk over APIs; each scales alone.

Separate billing service

Containers

Apps packaged in isolation, sharing the host kernel.

Docker, Kubernetes

Virtualization

Each VM runs its own OS on a shared hypervisor.

Hyper-V, VMware ESXi

Separate networks physically or logically

Approach	What it is	Tradeoff
Air-gapped	No network path to other networks at all	Strongest, but updates move by hand
Physical isolation	Separate switches and cabling per zone	Costly, but no shared gear to misconfigure
Logical segmentation	VLANs, subnets, and ACLs on shared gear	Cheap and flexible; one bad rule opens it
Software-defined (SDN)	Control plane split from the data plane	Central, automated control; guard the controller

Centralized and decentralized control

Centralized

- One place for policy, logs, and identity
- Easier to audit and keep consistent
- A single point of failure and a rich target

Decentralized

- Control spread across sites or teams
- No single point of failure
- Harder to keep policy and patching consistent

Pick the architecture that fits



8

MINUTES · PAIRS

For each need, choose the architecture model from 3.1 that fits best and name one security concern it brings.

1. A pharmacy app whose traffic spikes only on flu shot days
2. A water utility control network that must never reach the internet
3. A school that wants every lab server built identically from a file
4. A bank that wants to update each payment feature on its own
5. A clinic that wants email without running any mail servers
6. A startup that wants apps to run the same on laptops and cloud
7. A university that wants one rule pushed to 200 switches at once

Specialized systems are hard to patch



IoT

Connected devices with weak defaults and few updates.

Smart thermostats, IP cameras



ICS and SCADA

Control physical processes; safety and uptime come first.

Water treatment, power grid



RTOS

An OS that must respond within strict time limits.

Infusion pumps, vehicle brakes



Embedded systems

A computer built into a device for one job, often unpatchable.

Badge readers, medical carts

Every architecture is a set of tradeoffs

Consideration	Question to ask
Availability and resilience	Does it stay up, and recover when parts fail?
Cost and risk transference	What does it cost, and who carries the loss?
Responsiveness, scalability	Is it fast enough, and can it grow with demand?
Deployment and recovery	How easily can we build it, and rebuild it?
Patch availability	Will patches come, and can we install them?
Power and compute	Does the device have power and CPU for security?

Each extra nine cuts downtime tenfold

8.76 h

Yearly downtime at 99.9%

52.6 min

Yearly downtime at 99.99%

5.26 min

Yearly downtime at 99.999%

A year has 8,760 hours. Allowed downtime = 8,760 hours x (1 minus the availability).

Secure a hospital's specialized systems



7

MINUTES · SMALL GROUPS

For each system, name its type, the consideration that makes it hard to secure, and one compensating control.

1. Infusion pumps on Wi-Fi whose vendor ships updates once a year
2. HVAC controllers that anyone on the staff network can reach
3. Security cameras still using the factory admin password
4. A pharmacy robot on an RTOS that cannot run an EDR agent
5. Battery-powered door sensors too weak to run strong encryption
6. A SCADA system that runs the hospital's backup generators

REVIEW

Five questions: write your answers first

1. In a SaaS service, which security responsibilities always stay with the customer?
2. Why do containers provide weaker isolation than virtual machines?
3. How does an air-gapped network differ from logical segmentation?
4. What does inability to patch mean, and how should you respond to it?
5. About how much downtime per year does 99.9% availability allow?

TAKEAWAYS

Four things to take with you

01

Data and access stay yours

In IaaS, PaaS, and SaaS the customer owns data, accounts, and settings.

02

New models move the surface

IaC, serverless, microservices, and containers each shift the risks.

03

Guard what you cannot patch

Isolate IoT, ICS, RTOS, and embedded devices and watch their traffic.

04

Every model is a tradeoff

Weigh availability, cost, recovery, patching, power, and compute.

NEXT Lab 7: Case study on shared responsibility in a cloud migration