Seminar session 1: Secure System Architecture – Introductions and Assignment Review

Outline:

- Online session rules
- - Learning Outcomes
- Introductions
- Distributed Systems & Fallacies
- Seminar 1 Questions and Q&A
- Further information

Online Session Rules:

- Please keep your microphone on mute when not talking
- Use chat for general questions
- Please do not talk over others
- Everyone should contribute and opinions will be respected
- Please ask questions
- Every day is a school day

Module learning outcomes:

- The ability to contextualise the basic architecture of modern operating systems.
- An understanding of the relationship between distributed systems and operating systems.
- An understanding of the role of virtualisation in Secure Systems Architecture (SSA).
- An appreciation of current and future challenges, limitations and opportunities.
- The opportunity to reflect on and evaluate personal development.

Team Assignments:

- Set up teams of three to five people to collaborate on the programming assessment (2 teams with five team member).
- Personal introduction.

Recap:

Previous Modules:

SSD – looked at the cause of vulnerabilities, and how the mitigate at source level

NISM – looked at how to detect vulnerabilities

IRM – looked at how to detect vulnerabilities

SSA – takes a system view, looks at modelling threats and vulnerabilities, and how to mitigate; most systems today are distributed systems

What is Secure Systems Architecture?:

Definitions:

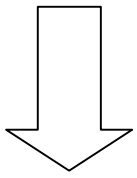
- Distributed Systems: Tightly Coupled vs. Loosely Coupled
- " Security Architecture and Design describes fundamental logical hardware, operating system, and software security components and how to use those components to design, architect, and evaluate secure computer systems." (Conrad et al., 2012)
- "If we can develop a separation kernel for a secure system such that it provides and execution environment similar to that of a distributed system, then we can simplify system design, development, verification and validation." (Alves-Foss, 1998)
- Based on JM Rusby "Proof of Separation", also Andrew Tanenbaum
- SDLC – Systems vs. Software not the same thing

Systems Design Life Cycle (from INCOSE):

- Combination of Agile approach and the waterfall life cycle
- Larger scope of resources of components and hardware
- Also a scope of regional regulations like GDPR

Monolithic vs. Micro: Operating System Kernels:

- Monolithic
- Traditional
- Hybrid Kernels
- "Wild West e.g. Exo-Kernels
- Micro Kernel



Mono vs. Micro:

Characteristic	Monolithic	Microkernel
Code Size	Large	Small
Communication	Shared space/ variables	Messaging
Complexity	High	Low
Memory	Single Address Space	Multiple Regions
Execution Mode	Kernel	Kernel & User
Security	Privileged Execution	Mixed
Risks	Drivers, 3rd party code	Messages, performance

Operating Systems – by Kernel (Examples):

Original Unix (9k LOC whole OS) ~ 16kB RAM → monolithic

- No monolithic System used today anymore
- Most of the OS today are hybrid
- E.g. MS NT/ Windows, Linux (5M LOC+) 10 -15 MB RAM, Mac OSX
- It enables to ad drivers to the OS without re compile the kernel system
- Examples of micro kernels: QNX, HURD, Minix (5k LOC)

Applications:

- Applications ahre similar characteristics with OS kernels:
- → Monolithic apps use shared memory, tend to be bigger, more complex and harder to maintain
- → Micro-services use message passing protocols, tend to be smaller and simpler (per micro-service)
- Need more careful design

Fallacies of Distributed Systems:

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.

6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

- (Various, 1991 – 94, SUN Microsystems).

Assessment:

You are required to evaluate a provided scenario which models the activities associated with networked devices in a smart home. A smart home, in general terms, involves multiple IoT devices that are able to communicate with a centralised controller, and/or with each other You are required to model the threats to and vulnerabilities of the system using Attack Trees.

Using Codio, Python, and/or containers, simulate the provided scenario. One container should be responsible for generating data to emulate messages being sent from a connected IoT device – select two smart devices within your imaginary smart home. Configure data producing streams which are representative of the devices that you have selected. Pay attention to the data generation rate, and the message transmission size, so that it is representative of activities that take place in real life. Messages should be transmitted to the central controller, connected as a second container, which will consume the data and take appropriate action in response. Additionally, you should provide the ability for a resident within the home to enforce actions on the IoT device. Demonstrate how you would address some of the threats and vulnerabilities, and discuss how you decided which threats to mitigate.

In the design of your system, you are required to consider the priorities of wireless communication protocols, including energy efficiency and security of the device itself, in relation to the transportation of messages from connected devices. Furthermore, when messages arrive at the central controller, capability should be integrated to support the concurrent execution of threads by implementing a Producer-Consumer scenario, and demonstrating that attention is given to thread priority, which is dependent on the Quality-of-Service requirements of the data.

Assessment – Part 1: Report:

Create an Attack-Defence Tree (AD-Tree) that models the security vulnerabilities of both a ‘client’ smart device (I.e. one that controls household equipment such as lights or heating) and the controller hub (that gathers the data and co-ordinates the activities). The tree should display typical vulnerabilities and you should select a suitable domain to allow quantitative evaluation of security vulnerabilities.

Checklist:

• Compile a list of potential vulnerabilities from academic resources (remember to cite all sources)

• Create an AD Tree using the Luxembourg (or alternative) software for both the client

device and the controller/ co-ordinator to display the vulnerabilities

• Select a suitable domain to assign threat values to each element of the tree (justify your selection of a domain)

• Based on your model, suggest suitable mitigation(s) to ameliorate the vulnerabilities

All decisions should be supported by related academic literature.

Further Information:

- Assessment – group assessment teams of 3-5 (probably 5)
- Design/Report (week 3)
- Working & Tested Application (week 6)
- Individual Reflection (week 6)
- Group catch-up this Friday

Next Week:

- Research vulnerabilities & create draft AD tree
- We'll also do an AD tool walkthrough
- Review other seminar questions