Cyber Security in Power Systems

Matus Korman

Industrial information and control systems, KTH

<matusk@ics.kth.se>
www.ics.kth.se

Contents

- Why security is important
- IT security – introduction
- Standards and guidelines
- Cyber security in power systems
    - General IT vs. ICS
    - Vulnerability and threat landscape
    - How to secure ICS environments

Why security is important

- Imagine a power infrastructure with highly insecure IT.
- What kinds of incidents can happen?

Power outage

Damaged equipment

Blackout

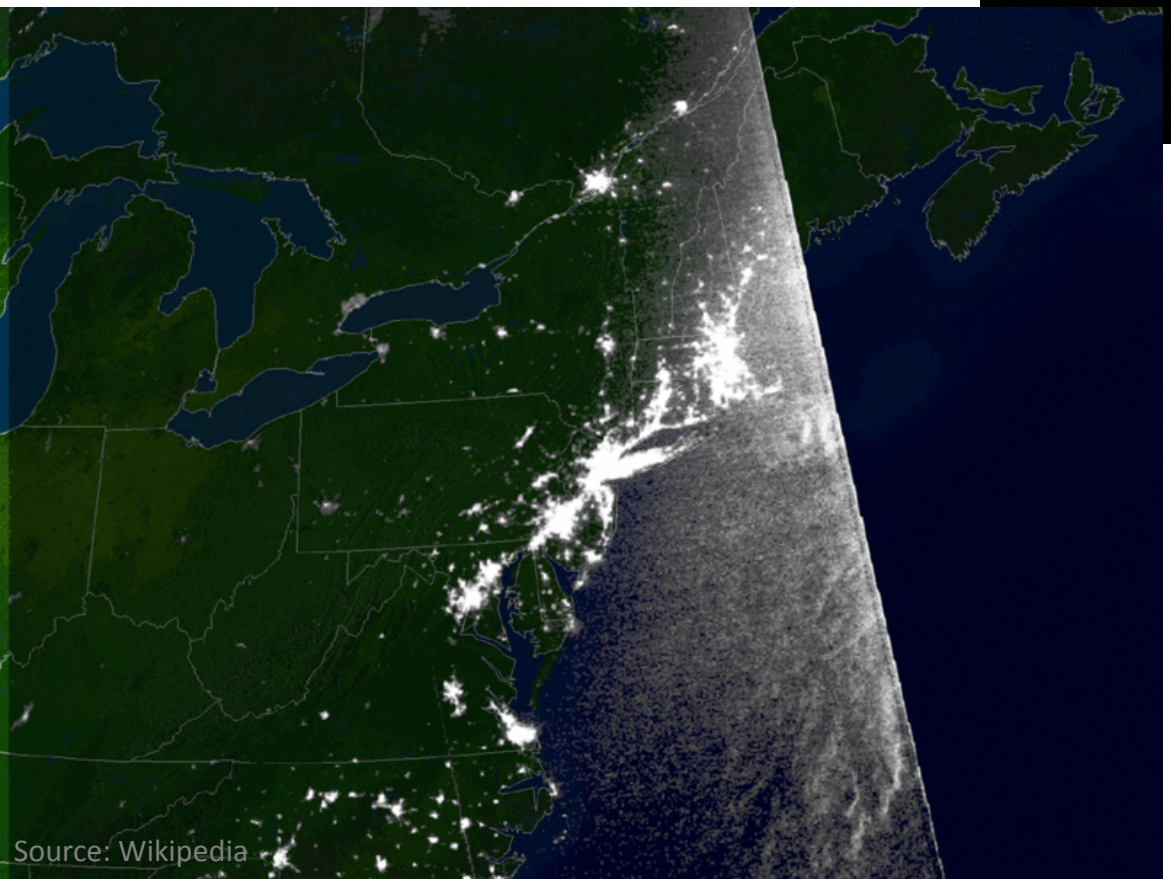
Misuse of IT resources for illegal purposes

Espionage resulting in strategic vulnerability

etc.

Example of possible consequences:

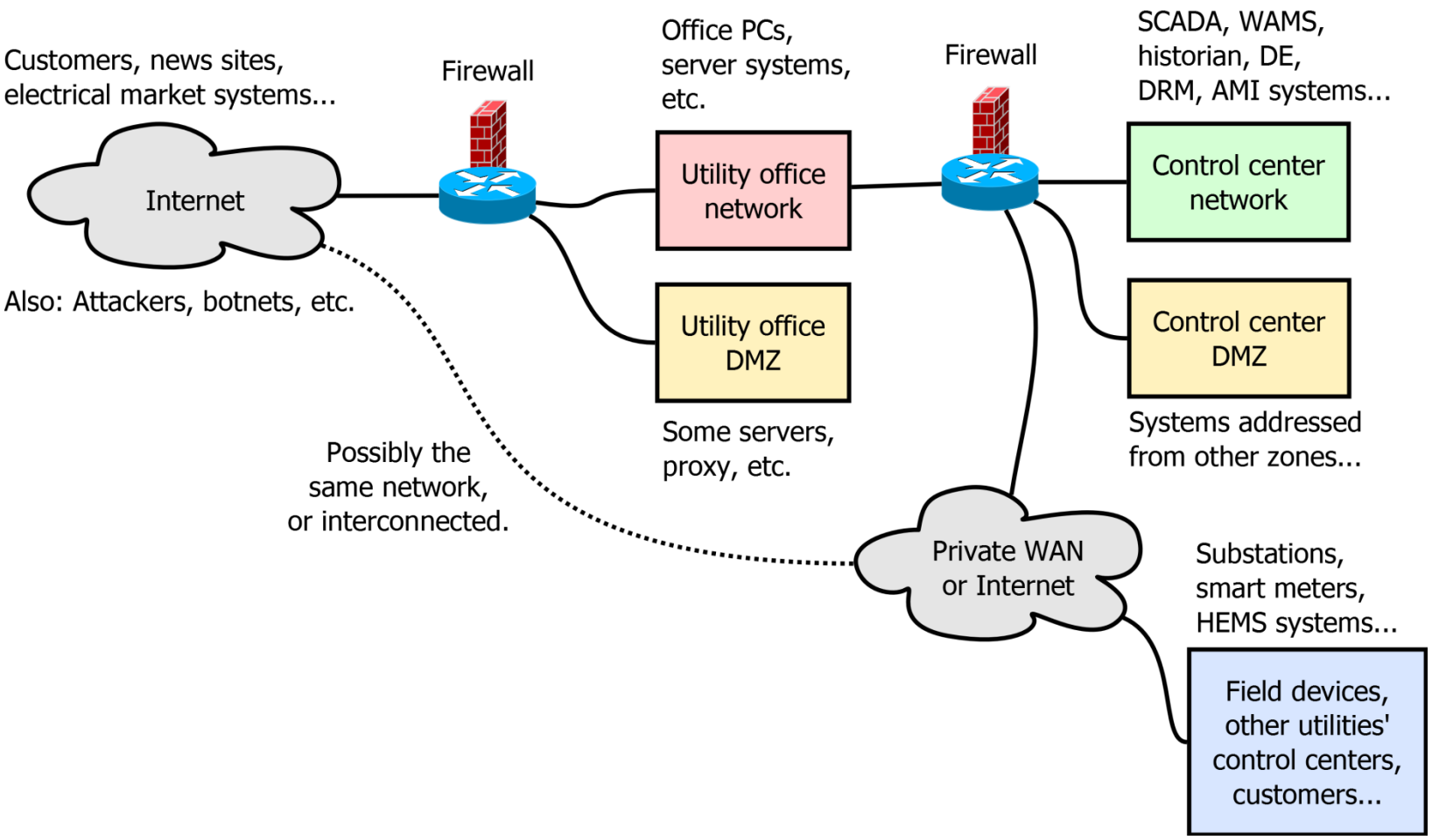
US Northeast blackout (2003)



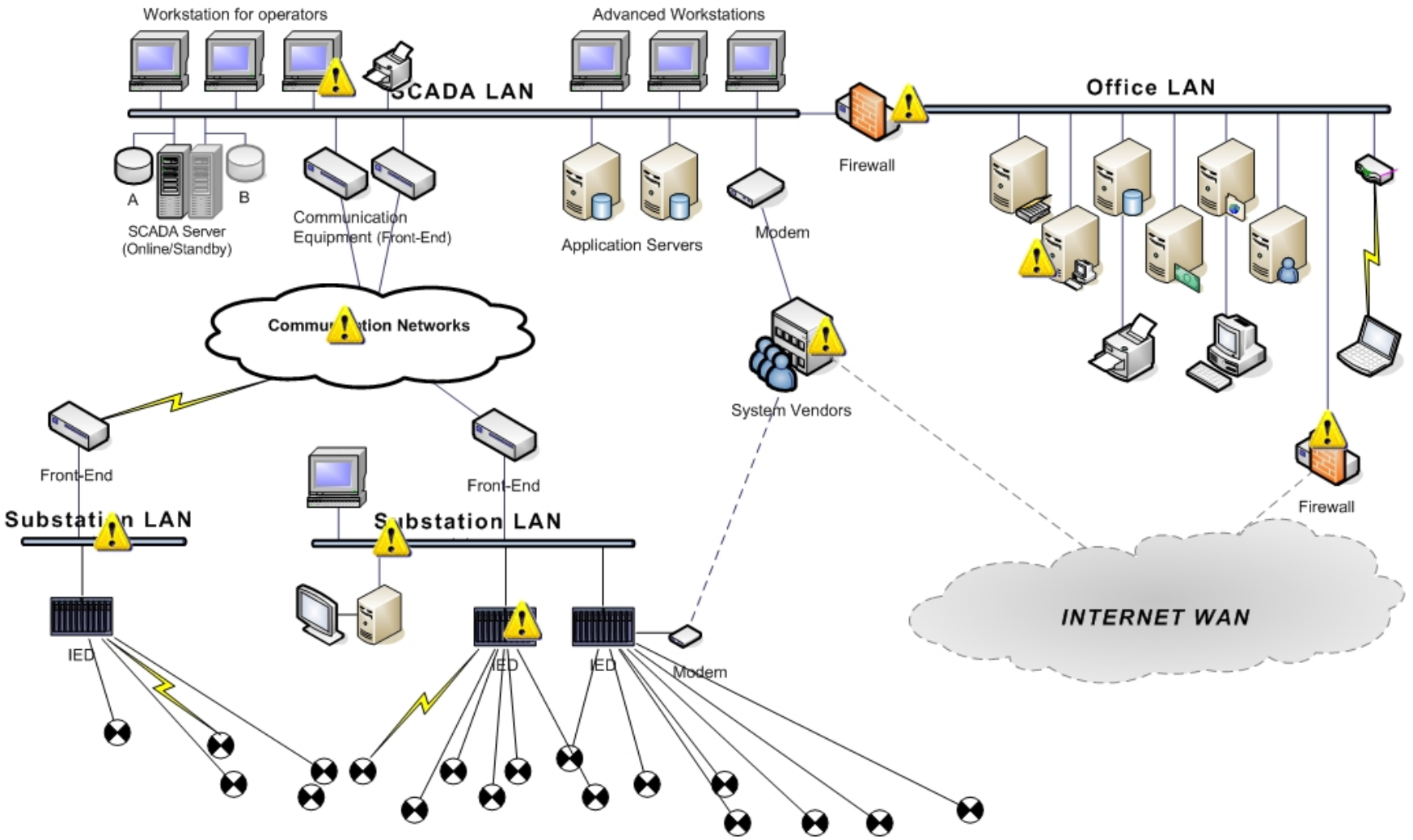
Source: Wikipedia

- **45M** people in 8 US states
- **10M** people in Canada
- Healthcare facilities experienced **$100M** lost revenue
- **6 hospitals bankrupt** one year after

A simple ICS architecture (overview)



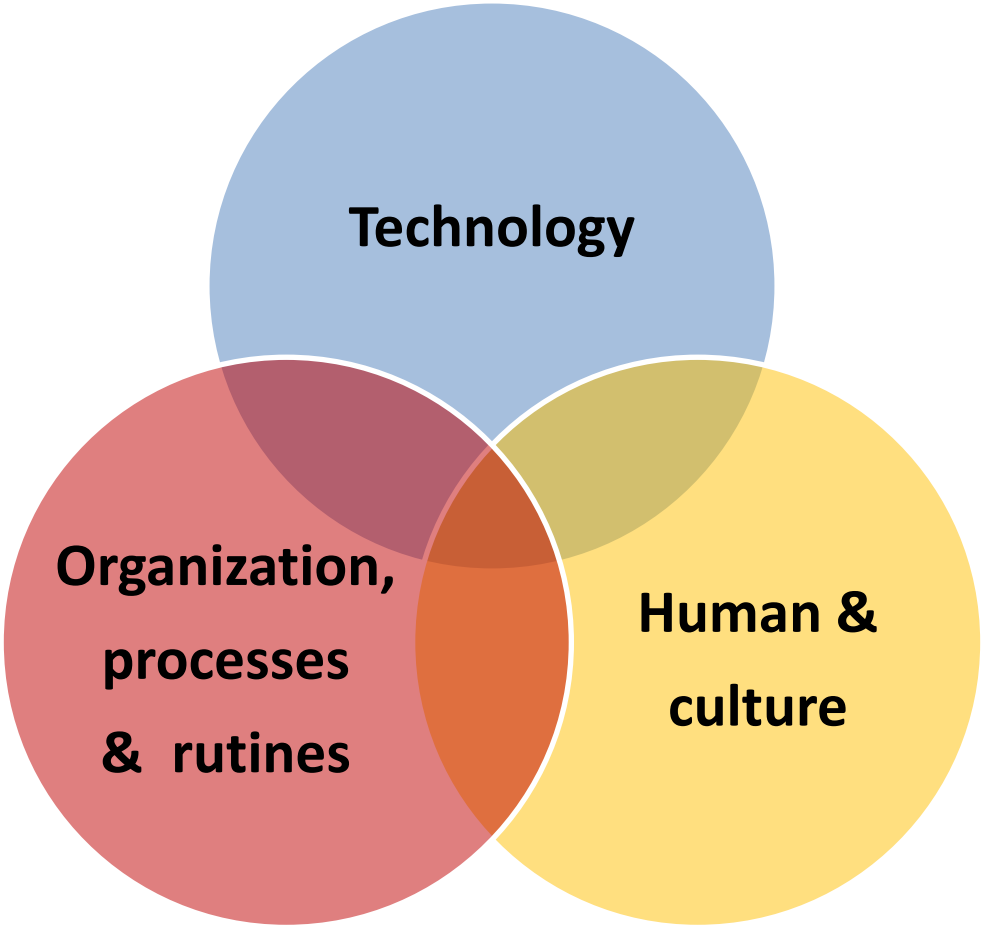
A simple ICS infrastructure (detail)



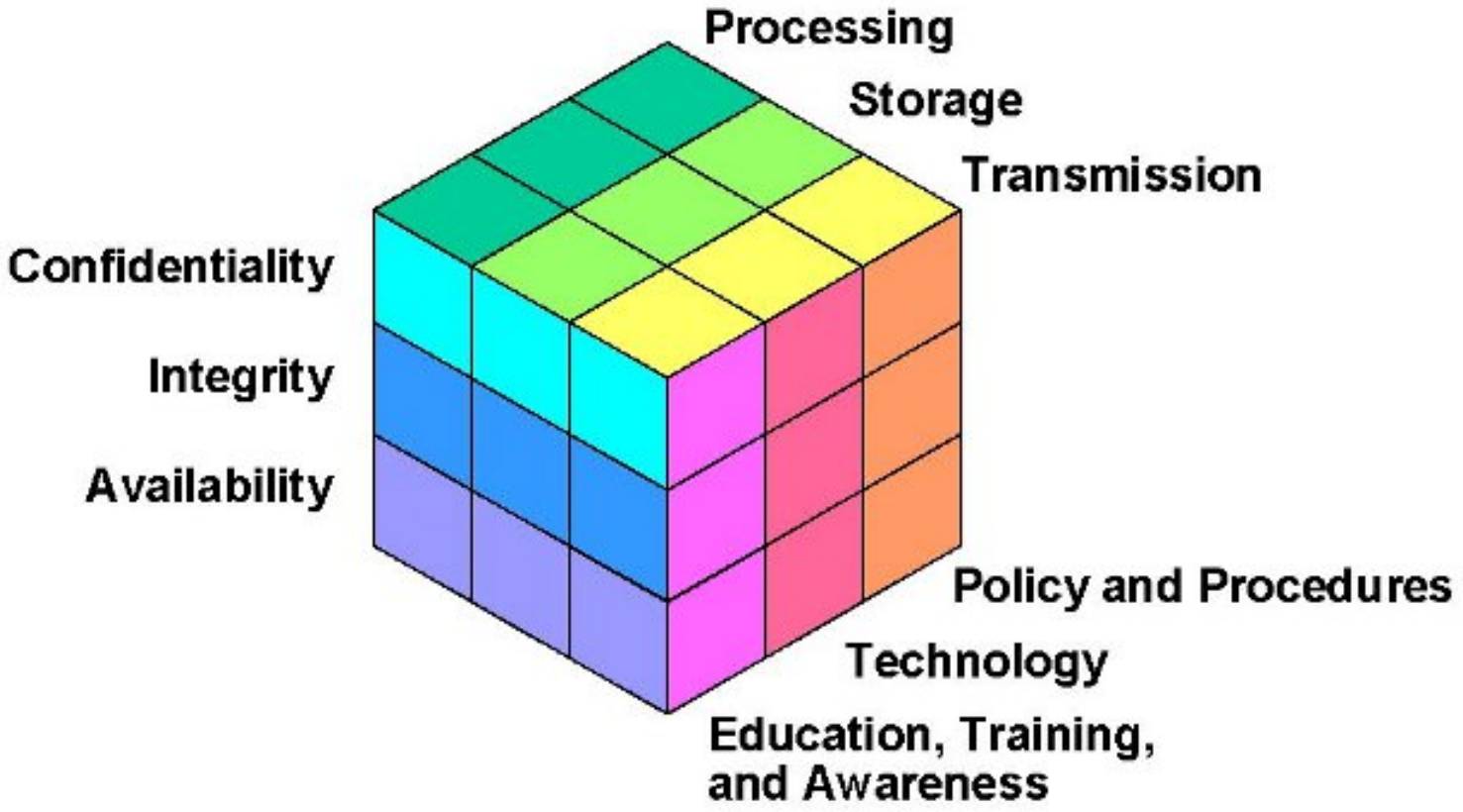
IT-security: Introduction

CIA triad:

Technology alone is not enough



McCumber cube:



Security requirements…

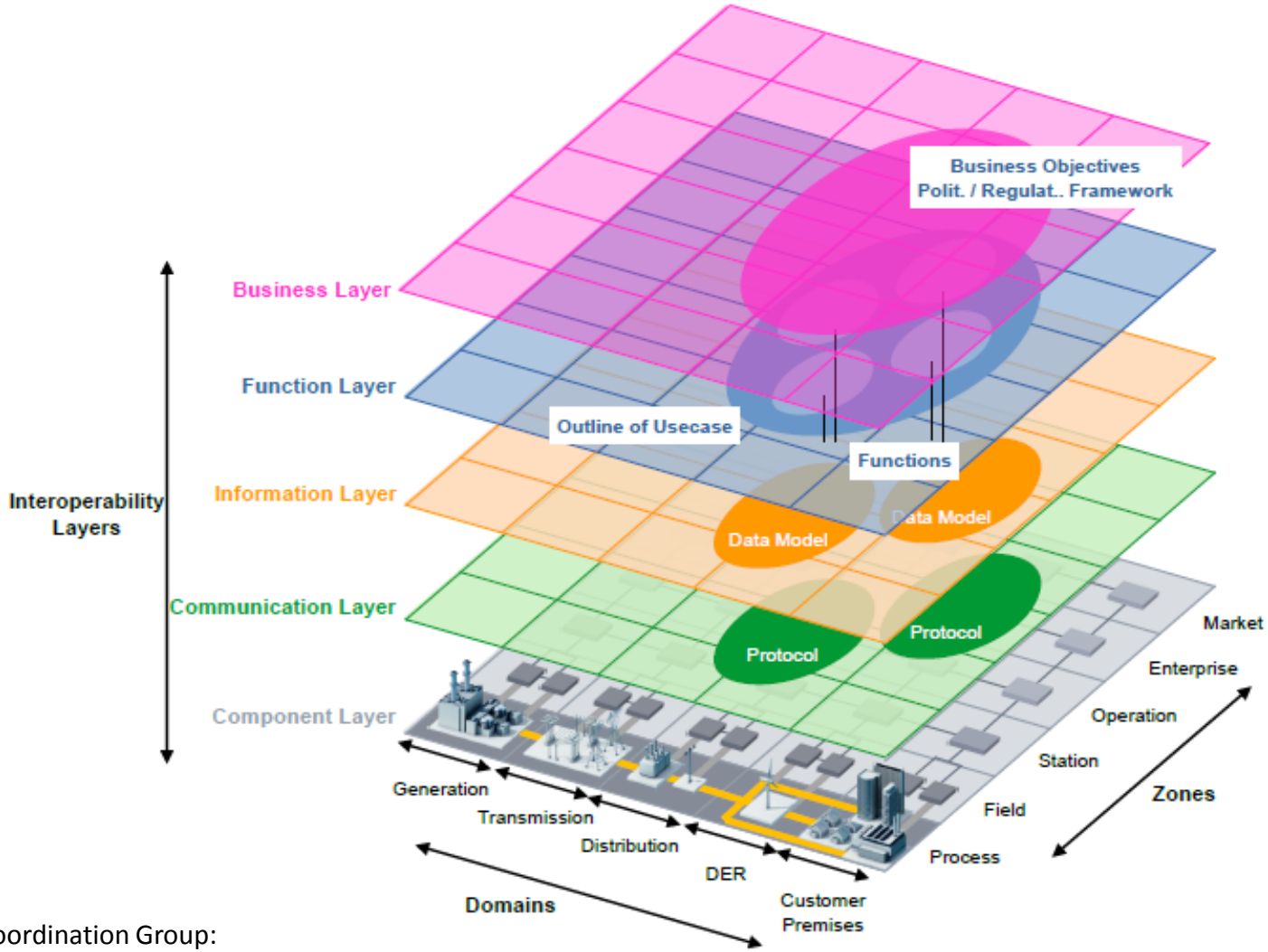
- Why do we need them?

…designing new systems

…changing existing systems

…evaluating IT security in them

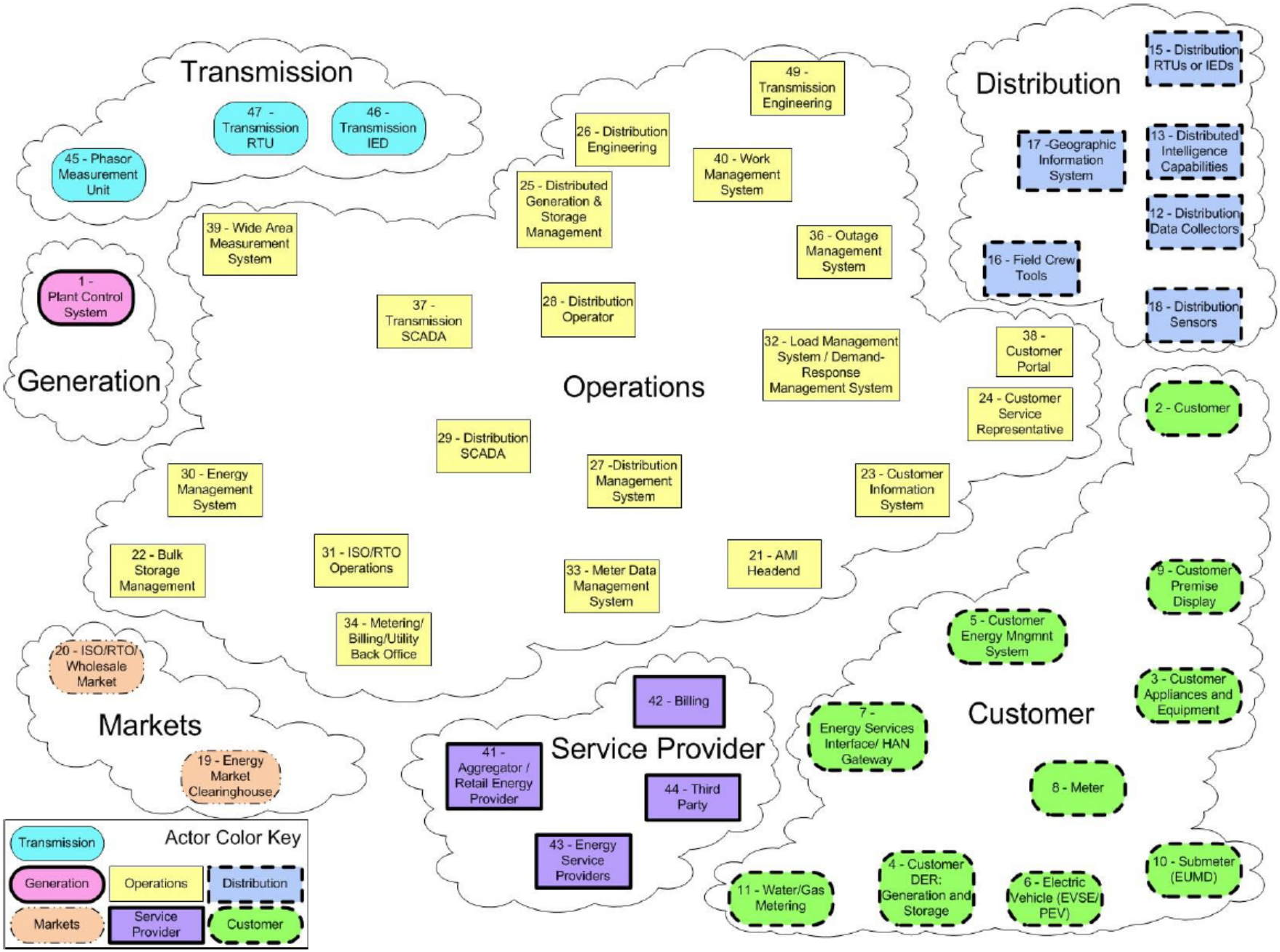
Reference model: security in power systems…



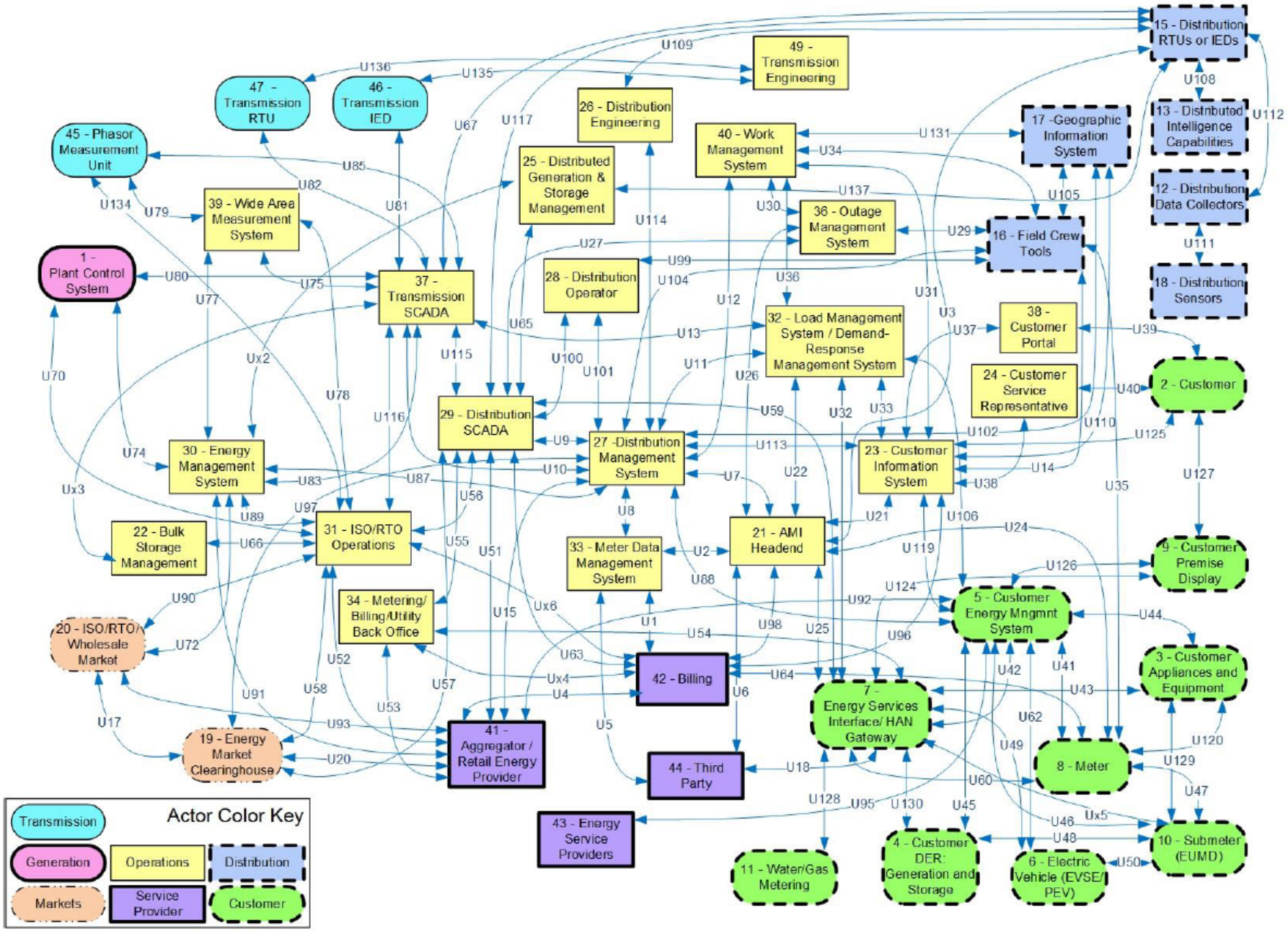
Smart Grid Architecture Model.

CEN-CENELEC-ETSI Smart Grid Coordination Group:
Smart Grid Information Security

NISTIR 7628 rev. 1: reference model – entities, actors



NISTIR 7628 rev. 1 – entities and data flows



Standards/guidelines

- General IT:
  - **ISO/IEC 27000-series** (27001, 27002)
- Control systems in general:
  - **NIST SP 800-82** (rev. 1):
    Guide to Industrial Control Systems (ICS) Security
- Power systems and other critical infrastructure:
  - **NISTIR 7628 (rev. 1)**: Guidelines for Smart Grid Cyber Security
  - **IEC 62351** (developed by WG15, IEC TC57)
  - **NIST Framework for Improving Critical Infrastructure Cybersecurity**
  - **NERC CIP** (Critical Infrastructure Protection)

General IT-security vs. ICS

- ICS are in many ways different from common IT:
  - **Performance requirements** (realtime/time-critical, guaranteed throughput, guaranteed response times…)
  - **High availability** required
  - **Difficult to update & upgrade** systems (compatibility, etc.)
  - **More fragile**(less redundancy, sharper requirements)
  - **Longer lifetime** (=> older systems – legacy)
  - **Confidentiality** is much less required
- However:
  - It is not correct to generalize to all non-ICS systems...

Certain IT-systems are critical, too

- **Banking** sector
  - IT-infrastructure that handles thousands of sensitive transactions per second.
- **Telecommunications** sector
  - IT-infrastructure that enables us to surt on Internet, talk on phone etc; as well as pay for these services.

- *ICS security is [perceived to be] lower than security in these sectors.*
  Question: *Why? What are the differences between them?*

Why ICS are vulnerable…

- Why?
- Systems are mainly designed and tested to:
  - Work in **ideal conditions** (*functionality*)
  - Work even under different **expected variation** in the process environment (*field-robustness – ”rugged”*)
  - To some extent work under **cyber-noise** (higher network load, patching of systems, etc.)
- However often not with the goal to resist **sophisticated attacks** from intelligent and capable threat agents (*cyber-security*).

Why ICS are vulnerable… cont'd

- Convergence between ICS and general IT
  - More and **more general IT-technology** is used in ICS (IP-networks, Windows, Linux, … all well-known, well-compromised)
  - **Connections** between the process network and other networks is often too liberal – lacking security protection
  - A **variety of data flows** that flow between networks, various protocols
  - Sometimes unprotected control system components (IEDs, RTUs, etc.) – for flexible access for technical personnel, consultants/contractors…
- Physically and logically distributed environments
- Not so many known major incidents so far… maybe has lead to less investments in ICS security?
  (compare with banking sector, telecommunications sector…)

Different levels of vulnerabilities

- Organization, people and operations:
  - Security policy and security culture in the organization;
  - How people carry out different technical operations;
  - What devices one can use in the ICS-network (own computers on which one surfs in private, USB-sticks etc.);
- Network design:

  - ICS-network (process network) + its connection to the office network (and other networks);
  - Application services running on machines in the ICS-network;
  - Configuration of IT-security protection in the network (firewalls, IDS/IPS, configuration of operating systems)…
- System/component design:
  - SCADA-software, HMI and workstation operating systems (Windows, Linux), other systems;
  - PLC, RTU, IED, switches, routers…

Common vulnerabilities in ICS

- **Documentation and processes**:
  - Lack of formal documentation
  - Lacking change management process
  - Lacking security policy and security culture (awareness, attitudes etc.)

- **Access control**:
  - Lacking access control (which user/role has access where, when, how, etc.)
  - Vulnerable handling of authentication data (e.g., passwords)
  - Over-privileged access accounts, old accounts, etc.

- **Network design**:
  - Vulnerable network design
    (e.g., unnecessarily broad exposure of systems and data traffic in a network)
  - Insufficient separation of different networks
    (process network, office networrk, DMZ, Internet…)

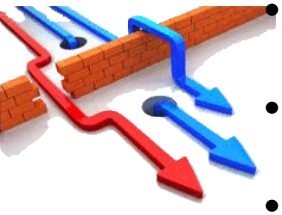
- **Security countermeasures**
  (firewalls, IDS/IPS, configuration, security operations):
  - Weak network protection (firewall restrictions such as what ports, what IP ranges, what intensity of communication, etc.)
  - Lacking security reviews and accountability
  - Vulnerable configuration of system such as unnecessary services and software installed and even running

Threat picture

- Dedicated human attackers
  - **Hackers** (e.g., youngsters with a lot of free time…);
  - **Hacktivists** (with political agenda), also terrorists;
  - **Criminals** (e.g., out after profit or misuse of computers and their network connections for illegal activity);
  - **Professional hackers** (e.g., for industrial espionage);
  - Foreign **nation states** (espionage and cyber-warfare)
- Other attackers (machines, humans)
  - **Botnets** and their operators
  - **Malware** in general that spreads through e-mail, web browsing etc. (showing a web-site is sometimes enough to become infected!)
  - **Own personnel** or company **partners** – so called *insider threat* (harm done deliberately or unconsciously; e.g., human mistakes)
- **Failures of equipment**, bad overview of own systems, lack of awareness, failure of power supply, natural phenomena, etc…

How an attack can take place…

- A network can be penetrated e.g.:

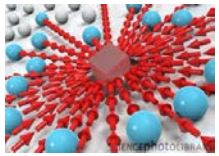


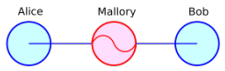
  - **Directly**: An attacker manages to get into a network from outside (e.g., by obtaining an own IP-address in there, ARP-spoofing some other machine, …)
  - **Indirectly**: An attacker exploits that personnel surfs on Internet, reads e-mail, etc… in order to infect the personnel's machine(s), and then attack further and deeper
  - **Social engineering**: An attacker tricks personnel to do something compromising (e.g., give away a username, password etc.) – through pretending to be a legitimate person, commonly in an urgent situation (e.g., a technician who quickly needs some non-standard help to prevent a major failure/incident from happening…)

- A software can be infected through (*a single* data flow can be enough) e.g.:

  - **Known vulnerabilities** (on outdated systems) – statistically frequent and often unnecessary vulnerability. Whoever can get exploits and shoot them at a system.
  - **Zero-day vulnerabilities** (**0-days**, yet publicly unknown) – majority is not captured even by advanced, expensive, collaborative security solutions (NGIPS). Luckily, 0-days are very expensive to buy usable exploits for (e.g., black market) and very demanding to identify and develop on own for a generic software.

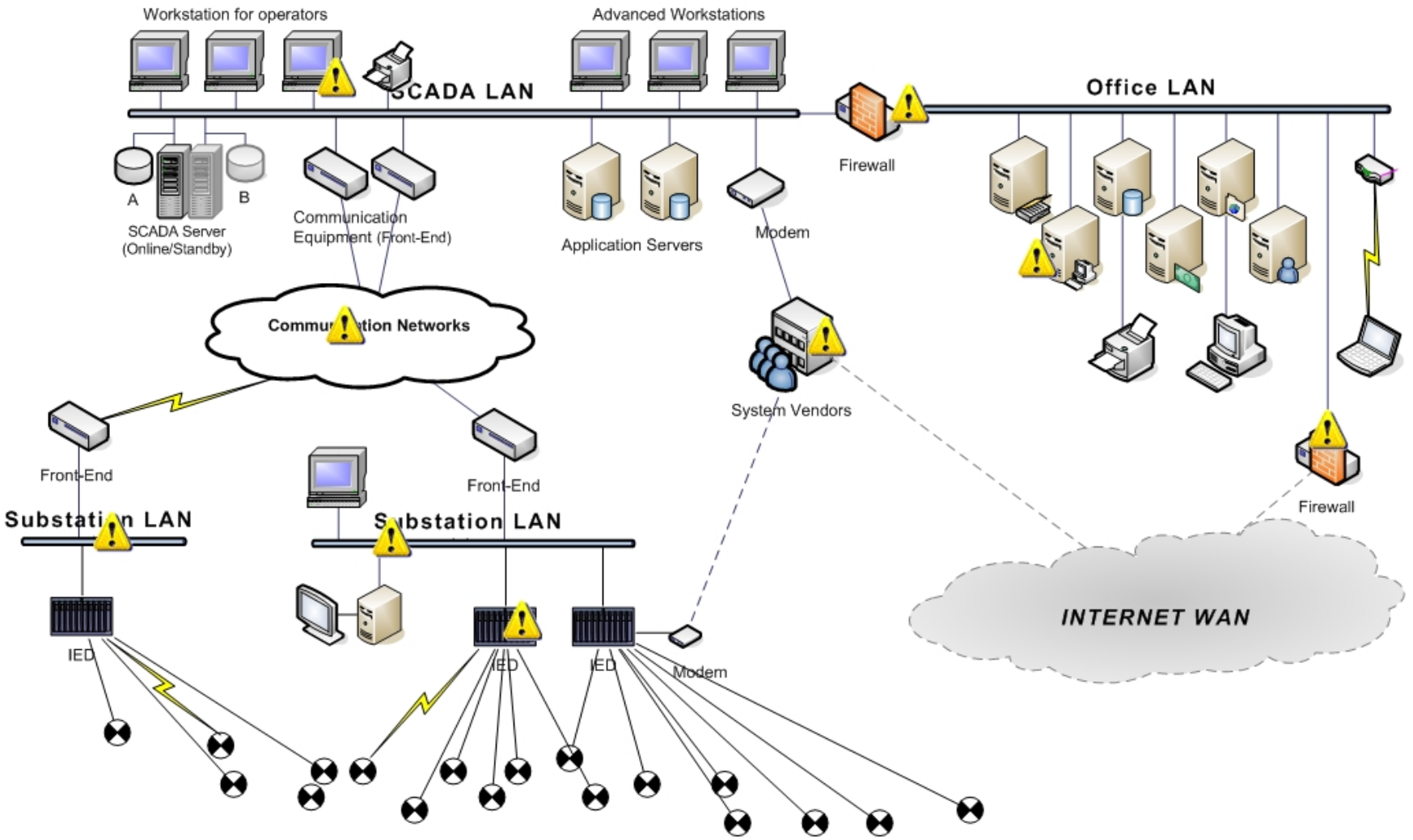
- There are different types of attacks, e.g.:





  - **DoS** (Denial of Service), **DDoS** (distributed DoS) – sabotage that blocks, saturates, locks in or takes down systems/functions so that they no longer are available (temporarily or permanently)
  - **MITM** (Man-In-The-Middle) – hidden manipulation of data communication…
  - **Intrusion** – leads to illegitimate control over a system or a part of it, which then can lead to modifications/sabotage, mapping/espionage, etc…

A simple ICS infrastructure…



How to secure ICS-environments

- Identify and **eliminate greatest security holes**
- **Robustify** systems and IT-environment as such
- **Scan for vulnerabilities, penetration test** (e.g., ICS test beds, even on-site if you dare)
  - … to **measure** security
- Establish a **systematic work** with IT security
  - **Risk analyses** and risk treatment
  - **Reviews** and **log analyses**
  - Analysis of in- and outgoing **network traffic** (e.g. Netflow)
  - **Updates** + other security maintenance
  - **Education and training** of personnel

Different analyzes and measurements…

- **We must measure** to effectively control – even information security.
- Many measurements are qualitative and build on expert evaluation or guessing
- Quantitative ones are being developed and established; e.g.:
  - SAL – Security Assurance Level (a bit like Common Criteria in IT-world)
  - OEE – Overall Equipment Effectiveness
    := (availability * performance * quality) – all quantitatively defined
  - OEE-Security (?)
    := deviation from compliance with guidelines/requirements/goal
    - How many accounts are over-privileged/outdated/unidentified?
    - How much logs can be found and analyzed w.r.t. requirements?
    - How effective are security countermeasures w.r.t. penetration tests?
    - How many undocumented network access points? Et cetera…
- **Quantitative measurements & methods are needed.**
  They are believed to be the future.

Measurements and other is inspired by Macaulay & Singer (2012): Cybersecurity for Industrial Control Systems

Example countermeasure (1)

- If a computer from ICS-network tries to connect to the Internet…

  (…is such communication needed at all?)

  – Where and how does it connect (IP addr., port, etc.)?
    - **Blacklisting**– if it connects to suspicious sites/IPs, an alarm goes to the security personnel;
    - **Whitelisting**– if it connects to other than allowed sites/Ips (e.g., for downloading updates), an alarm goes.

Example countermeasure (2)

- Data traffic in ICS follows quite predictable patterns
  - at least when it comes to what protocols are used and what machines or IP intervals are being addressed…
- It should be possible to specify and apply firewall rules that safely keep operations, but cut/limit unnecessary flexibility (= possibility for activity needed for a hacker to succeed with an intrusion).
  - This even applies to the computer-level (host-level), not only network-level.
- It should be possible to detect a lot of suspicious (anomalous) traffic patterns in the data traffic (that can point to an attack or an actual intrusion).

Example countermeasure (3)

- Special systems that are placed out in the ICS-network and run or pretend to run different services, but in fact are there only to issue an alarm if someone tries to connect to them or use the services on them
  - Because these services are not used in legitimate operation, which an external attacker mapping or scanning the network has a low chance of knowing!
  - They work like a mine field – if you don't know where the mines are, you are quite likely to get into trouble… get your attack detected and likely contained before you manage to do what you wanted (as the attacker)
- Such systems are even called *honeypots/honeynets*. They trick the attacker into revealing herself, getting identified and losing time by attacking fake systems or whole networks.

Robustification

- Robustness...
  - ICS-systems are designed for robustness in process control, to secure that a bit more than expected variation in process variables won't lead to invalid outputs or states
  - In the development process, vendors simulate different variation in process variables (e.g., temperature, humidity, EM-fields…) and test systems under different conditions
  - Do they do it sufficiently even for cyber-variation (including directed attacks)? (likely answer: NO)
  - If not, the result is doubtful cyber-robustness (including cyber-security)

This topic is inspired by Langner (2012): Robust Control System Networks

Challenges with robustness (1)

Control logic with unconstrained input/output-mapping

	Valid output	Invalid output
Valid input	Can be robust, can be fragile (difficult to say)	Wrong
Invalid input	Robust	Fragile

I.e., system behavior for valid inputs say nothing about robustness. Robustness is tested and measured through **negative testing** (testing with different invalid inputs).

Challenges with robustness (2)

Sensitivity to cyber-noise

Cyber-noise means cyber-environmental factors that temporarily or permanently can affect systems and their function, for example:

- Increased network load
- Ongoing network scanning
  (invalid packets, many packets)
- System updates
  (newer system libraries,
  modified system configuration…)

Problems and challenges in control infrastructure

1. **Multiple control paths**
   (shared authority to control or configure)

2. **Undetected broken control paths**
   (one assumes to have control when one doesn't have it)

3. **Undetected modification** of data communication (messages/packets) – *broken integrity*
   - E.g., MITM attacks (Man-In-The-Middle)

4. ***"Antistructure-attractors"***
   - Generic services as SMB/CIFS file servers, NFS, DCOM, FTP…
     - Commonly heavily outdated – systems are updated seldom in ICS-environments
     - Too flexible, so they "attract" lacking design patterns, lacking software processes and of course **malware** within an attack
       - » They are very thankful attack targets with a broad spectrum of publicly known vulnerabilities. Therefore, they are simple to compromiose (penetrate).
     - **Tip**: If you need to use such generic services, use their secured variants (SSH, SFTP, etc.) – that are tunneled over an authenticated and encrypted channel.

Abstract principles for robustification

- **Blockout** principle
  - Blocking invalid input
  - Limiting transfer function interval (”input → output”)
  - Blocking invalid output
- **Avoid-mess** principle
  - Try to maintain orderliness – as much as possible
- **Consistency** principle
  - Use similar systems and configurations in similar environments – when able
- AERA principle (**Adequate Execution Resources Allocation**)
  - E.g., to have sufficient physical redundancy, performance margins, enough personnel, responsibility coverage etc.

Robustification strategies

- **Reduction strategy** (by imposing structure)
  - Remove unnecessary applications, services, functions
  - Remove generic software services and interfaces
  - Use application-specific interfaces with least possible functionality
  - Reduce shared folders/drives etc.
  - Remove hidden untrusted systems (e.g., laptops that are being used in different places)
  - Limit user access and user interaction
  - Minimize variation in procedures
  - Minimize network exposure
  - Minimize variation in equipment types, product versions and configuration alternatives
- **Surplus strategy** (by enforcing and reinforcing structure)
  - Resilient code and architecture (defensively developed systems/architectures)
  - Mechanisms for control and detection of code execution and modification of configuration (e.g., antivirus, IPS, special OS hardening solutions, …)
  - Mechanisms for control of integrity (metainfo. such as digital signatures, checksums, etc.)
  - Context based (state-based) constraints of control authority (role- and state-based access control)
  - Protection mechanisms and process monitoring
  - Redundancy of different kind
- **Change management** (by modifying structure)
  - Examples of general attributes (that apply to technology, organization and human/culture) to take into account: adaptability, modifiability, maturity och quality of processes, among other change management itself

Compromises with robustness

Surplus strategies →			
(high) ↑ **Function-ality** ↓ (low)	Rich functionality, high flexibility, fragile (*typical COTS system*)	Rich functionality, high flexibility, robust (*unrealistic*)	**Reductions-strategies** ↓
	Necessary functionality, no/low flexibility fragile (*typical ad-hoc developed system*)	Necessary functionality, no/low flexibility robust (*real-world robust system*)	
(lower) ←	**Variation tolerance**	→ (higher)	

It says that robustness costs in different ways.
It is unrealistic to achieve both rich functionality and high robustness.

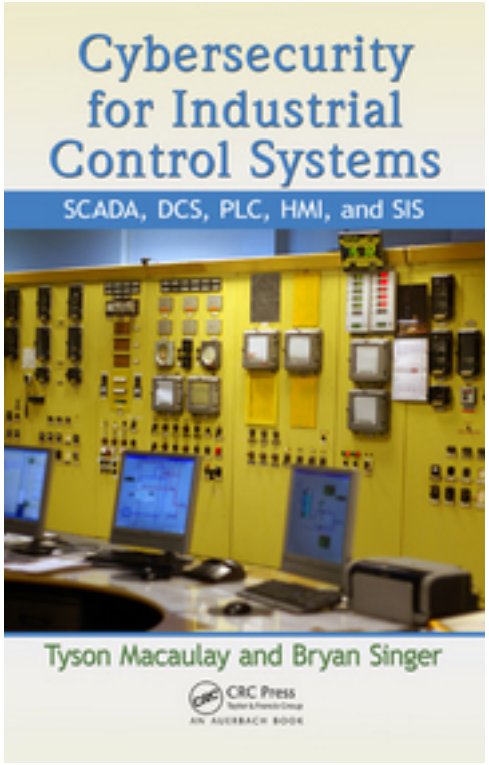
New technology – soon on the way in…

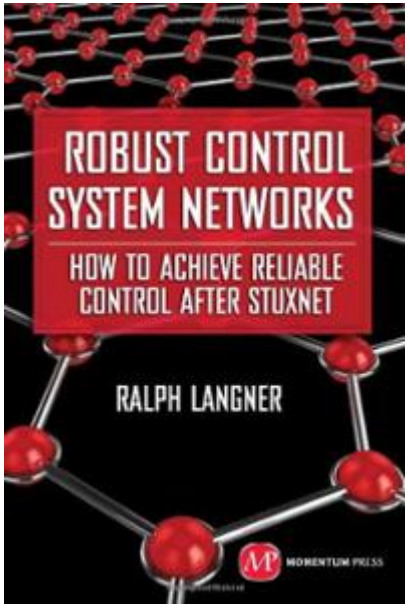
… bringing new challenges to cyber-security:

- **Internet of Things** (IoT)
  - Many small devices (e.g., sensors) soon connect to the net and ”talk” IP
- **IPv6**
  - Address space of IPv4 is saturated; new public IP-addresses must be IPv6
  - Transition to IPv6-support is a demanding, long process (several years in ICS)
- **Cellular networks** (wireless) in ICS
  - Devices are going to use wireless networks of different kind
- **Low-power Wireless Personal Area Networks** (LoWPAN)
  - Extremely energy-efficient devices with the capability of IP-based wireless communication

Books with highly applied, practical focus: Recommended reading in case of further interest

Tyson Macaulay & Bryan Singer (2012):
Cybersecurity for Industrial Control Systems





Ralph Langner (2012):
Robust Control System Networks

Yet another book:

Eric D. Knapp & Raj Samani (2013):
Applied Cyber Security and the Smart Grid

Thanks for your attention.