Defense-In-Depth (w/ case studies)

**🔍 Case Study 1: Stuxnet (2010)**

**Overview of the Attack**

* Stuxnet was a nation-state malware that targeted Iran's nuclear centrifuges.
* It was designed to infect Windows machines via USB drives.
* Once inside, it propagated through local networks and identified Siemens Step7 systems controlling centrifuges.
* It altered PLC logic while feeding false feedback to operators.

**ICS Components Targeted**

* Engineering workstations (Windows PCs)
* Siemens S7-300 PLCs
* Step7 programming environment
* WinCC HMI systems

**Attack Vectors and Bypass**

* USB introduction: bypassed air-gapped network
* Zero-day Windows vulnerabilities used for escalation
* Stolen driver certificates gave malware legitimacy
* PLC logic manipulated with stealthy feedback to hide change

**Defense Layers That Failed**

* **Physical Layer**: USB allowed, no port lockdown
* **Host Layer**: No antivirus or patching for zero-days
* **Application Layer**: No logic validation or change monitoring
* **Procedural Layer**: No policy preventing USB device usage

**How DiD Could Have Helped**

* USB port control + removable media policies
* Application whitelisting to block unauthorized Step7 code
* Behavior-based IDS detecting unusual PLC logic behavior
* Strong change management and logic verification on PLCs
* Log integrity monitoring for unauthorized access attempts

**❓ Stuxnet Q&A**

**Q1:** *How did the attacker gain access to the ICS network?*  
**A1:** Through infected USB drives used by insiders or contractors.

**Q2:** *Which DiD control could’ve blocked the initial infection vector?*  
**A2:** USB port disabling, Data Loss Prevention (DLP), or endpoint control policies.

**Q3:** *What detection layer was missing when PLC logic was changed?*  
**A3:** Logic integrity monitoring and automatic configuration auditing.

**Q4:** *What human procedure could have caught the manipulation sooner?*  
**A4:** Regular manual or automated validation of control logic by OT engineers.

**🔍 Case Study 2: Ukraine Power Grid Attack (2015)**

**Overview of the Attack**

* BlackEnergy malware was delivered via phishing to IT systems in Ukrainian energy companies.
* Attackers gained remote access and pivoted to ICS systems.
* They issued SCADA commands to open breakers and caused a blackout for 230,000+ residents.

**ICS Components Targeted**

* SCADA operator workstations
* Remote terminal units (RTUs)
* Network segmentation between IT/OT was weak

**Attack Vectors and Bypass**

* Phishing emails to IT staff with malicious macros
* Lateral movement to ICS through shared credentials and poor segmentation
* Destruction of firmware in field devices to delay recovery

**Defense Layers That Failed**

* **Network Layer**: Flat network, insufficient firewall rules
* **Procedural Layer**: Weak user awareness and phishing defense
* **Host Layer**: No endpoint protection or detection of macro payloads
* **Data Layer**: No integrity checking of firmware images

**How DiD Could Have Helped**

* Strong IT/OT segmentation with firewalls and DMZ
* User training and email filtering to catch phishing
* Jump servers and isolated admin networks for SCADA access
* Firmware integrity checking and secure boot for RTUs
* Behavior-based alerts for unusual breaker operations

**❓ Ukraine Q&A**

**Q1:** *What was the attack’s initial access vector?*  
**A1:** A spear-phishing campaign with Microsoft Office macro payloads.

**Q2:** *Which DiD control could have prevented lateral movement?*  
**A2:** Network segmentation and use of separate authentication domains.

**Q3:** *How did attackers evade detection once inside?*  
**A3:** They used legitimate tools and admin credentials to avoid alarms.

**Q4:** *What procedural defense would’ve helped?*  
**A4:** Incident response drills and firmware backup protocols.

**🔍 Case Study 3: Triton/Trisis (2017)**

**Overview of the Attack**

* Targeted a petrochemical plant’s Safety Instrumented Systems (SIS) using Schneider Triconex controllers.
* Attackers used custom malware to reprogram SIS logic.
* Intended to disable fail-safes, potentially enabling a catastrophic physical event.

**ICS Components Targeted**

* SIS controllers (Triconex)
* Engineering workstations connected to SIS
* Remote access infrastructure (likely compromised externally)

**Attack Vectors and Bypass**

* Exploited remote access and likely poor credential hygiene
* Reprogrammed SIS logic using a fake engineering tool
* Caused system shutdown when attackers made a coding error

**Defense Layers That Failed**

* **Host Layer**: Insecure or compromised engineering workstation
* **Application Layer**: No logic change verification
* **Network Layer**: No isolation between process control and SIS
* **Procedural Layer**: No access logging or change auditing

**How DiD Could Have Helped**

* Network segmentation between SIS and other OT networks
* Strict user access policies and physical disconnection of SIS programming ports
* Application-layer logic integrity monitoring
* Alerting on abnormal engineering workstation behavior
* MFA and hardened remote access protocols

**❓ Triton Q&A**

**Q1:** *What made this attack especially dangerous?*  
**A1:** It targeted the SIS layer, potentially disabling life-saving emergency shutdowns.

**Q2:** *Which DiD technique could have prevented SIS access?*  
**A2:** Network segmentation and enforced physical-only access to SIS.

**Q3:** *What would have detected the attack earlier?*  
**A3:** Logic change audits, code signing for SIS programs, and behavioral baselines.

**Q4:** *Was this a random attack or targeted?*  
**A4:** Highly targeted; malware was customized for Triconex systems.

**🔍 Case Study 4: Oldsmar Water Facility (2021)**

**Overview of the Attack**

* Attacker used remote desktop software (likely TeamViewer) to access the SCADA system.
* Increased sodium hydroxide levels in drinking water to dangerous levels.
* Operator noticed change in real-time and reversed it.

**ICS Components Targeted**

* SCADA operator workstation
* Remote access software
* Human-machine interface (HMI) controlling water chemistry

**Attack Vectors and Bypass**

* No MFA or access control on remote software
* Password reuse likely across multiple systems
* Lack of alerts on process changes

**Defense Layers That Failed**

* **Application Layer**: Unrestricted HMI controls
* **Host Layer**: Poor endpoint security and credential reuse
* **Network Layer**: No VPN or remote access control
* **Procedural Layer**: No alerting on abnormal chemical setpoints

**How DiD Could Have Helped**

* Implement MFA and logging for remote tools
* Disable remote access by default, or use a secure jump server
* Set thresholds for chemical dosing with alerts to supervisors
* Role-based access control (RBAC) for SCADA operators
* Continuous monitoring and alerting on parameter deviations

**❓ Oldsmar Q&A**

**Q1:** *What was the most critical point of failure?*  
**A1:** Insecure remote access with no MFA or session monitoring.

**Q2:** *What DiD controls would reduce this risk?*  
**A2:** RBAC, network monitoring, SCADA lockouts, and alarm thresholds.

**Q3:** *How was the attack caught?*  
**A3:** A live operator witnessed the change and manually corrected it.

**Q4:** *What procedural fix could help prevent a repeat?*  
**A4:** Disable unnecessary remote tools and require just-in-time access with MFA.

**✅ Summary: DiD Controls Across All Cases**

| **DiD Layer** | **Stuxnet** | **Ukraine** | **Triton** | **Oldsmar** |
| --- | --- | --- | --- | --- |
| Physical | USB blocking | - | Port lockdown | - |
| Network | VLANs, firewalls | Segmentation | SIS isolation | VPN, RDP control |
| Host | Patching, antivirus | Endpoint detection | Engineering hardening | RBAC, access logs |
| Application | Logic validation | Command alerts | SIS logic auditing | SCADA input limits |
| Procedural | USB policies | Phishing training | Access audit logs | Operator SOPs |