



Collins Aerospace
An RTX Business

RTXTGE
Technology & Global
Engineering



Cyber Class 303 (C303)

Embedded Systems Security

Interactive Participant Guide

Days 01-02

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

All rights reserved. No part of this file may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the written permission of the Publisher, except where permitted by law.

Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

For information, address Technology and Global Engineering (T&GE).

Special thanks to everyone who helped create this course.



Collins Aerospace

An **RTX** Business

RTXTGE
Technology & Global
Engineering



Published by Technology and Global
Engineering (T&GE)
Raytheon Technologies Corporation
870 Winter Street
Waltham, Massachusetts 02451-1449

Instructions: How to use your downloadable, interactive (iPDF) participant guide – understanding the symbols

Advantages	<ul style="list-style-type: none">Your notes are stored electronically with this file.Your notes are searchable.The entire file with your annotations is printable.
Symbols	<p style="text-align: center;">Description</p>
 1	<p>The page contains embedded animations/videos/other media. The number indicates the <i>number of clicks</i> required to completely display the animation and/or embedded media on the page.</p> <ul style="list-style-type: none">Select this box to activate the embedded media content in a separate window. Note: if marked with an asterisk *, the animation appears ONLY in the instructor's displayed version.Close/move this separate window to return to your guide.
 1*	
 3	<p>The page contains hyperlinks. The number indicates the number of embedded hyperlinks.</p> <ul style="list-style-type: none">Select the link(s) ON THE PAGE to go to the linked reference to open a separate window.Close/move this separate window to return to your guide.Note about video hyperlinks: <i>single-click</i> to launch the video in the size of the on-screen window; <i>double-click</i> to launch the video in full-screen mode.
	
	<p>Navigation within your downloadable participant guide</p> <ul style="list-style-type: none">Right-click inside the toolbar near top of screen and select Show All Page Navigation Tools.Select the appropriate arrow to advance directly to first or last page, to previous page, to next page.Enter a page number in the field to go immediately to that slide.

Instructions: Videos in the interactive (iPDF) participant guide

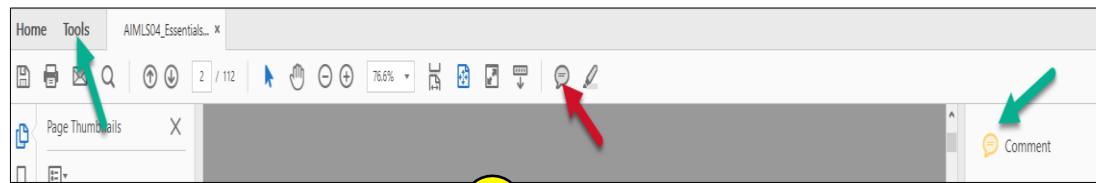
Video	<p>Marked with the word “Video” and the slide number in the title</p> <ul style="list-style-type: none">Example: Video 53: shaped charge
Location	<p>Videos may be embedded or linked to an outside source (e.g., YouTube).</p> <ul style="list-style-type: none">Embedded videos significantly enlarge the file size, making the file difficult to email or share without a server (Rshare, OneNote, OneDrive)Linked videos require that the user be online in order for the video to function properly.
The video file type (.MP4, .MPEG, .AVI,etc)	<p>How a video launches and plays in an iPDF depends upon the file type.</p> <ul style="list-style-type: none">To launch: Tap the image OR tap the forward arrow. The forward arrow appears in a white window that obscures the static image of the video clip. Initially, the video plays in the size of the on-screen window.To open in full-screen mode: Double-tap the launched video.To stop play: Tap the video it plays.To exit full-screen mode: Press the ESCAPE key on the keyboard.To re-launch a previously-played video:<ul style="list-style-type: none">Right-select the stopped video image.Select Disable Contents to reset the video so that the forward arrow is visible.
Warnings:	<p>Because of security issues, PDFs support fewer and fewer file types; for example, Flash files are now prohibited. Unfortunately, many clips in this course were created in non-supported file types, and converting them to a more secure file type often results in the loss of functionality. Consequently, you may or may not have the navigation controls (timing slider bar for moving to an exact time in the video) that you see in the instructor’s presentation.</p>



Instructions: How to use your downloadable, interactive (iPDF) participant guide – taking and preserving your class notes

Creating and saving your notes

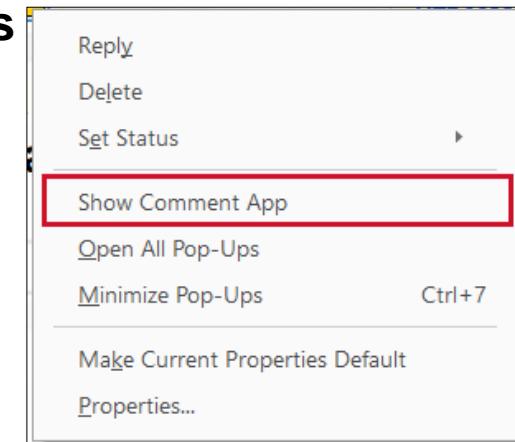
- Open the file in Adobe Acrobat Reader (Note: If Reader is not installed on your laptop, please download it from the Raytheon App store. These instructions do NOT apply to Adobe Acrobat!)
- Select Comment. There are three methods of opening the commenting function; the one shown with the red arrow is the quickest if it is available.



- Select the conversation icon to change the cursor to the conversation bubble.
- Drag the bubble to the location on the slide where you wish to make a comment.
- Select to position the comment.
- Enter your comment in the square window that opens.
- After completing your comment, DON'T FORGET TO SELECT POST to save it as annotation to your participant guide file. **Note:** When you save the file (you may want to use a new name), your comments/notes are saved as part of the file.

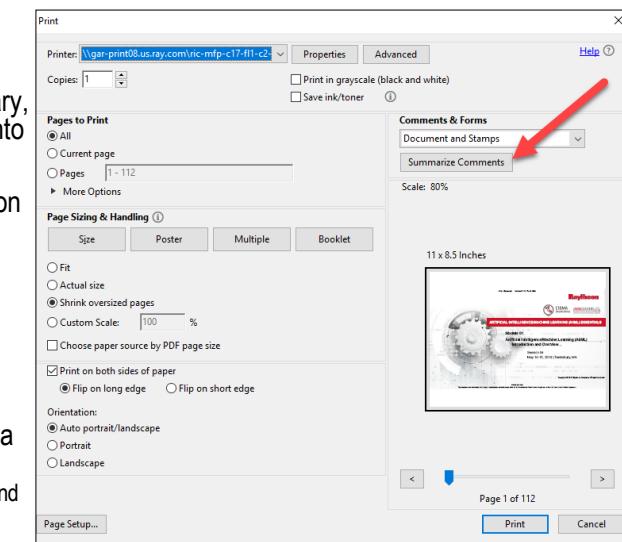
Displaying/printing your notes

- Locate any comment icon and right-click to open this submenu.
- Show Comment App opens a window in right margin that displays all your notes in the file in a searchable list. This is a VERY USEFUL feature! You can use this list as a *quick index* to document's pages
- Open All Pop-Ups opens all the note boxes so they are visible as you display the slide. Minimize Pop-Ups closes the note boxes, leaving only the comment icon visible.



Printing the file with your electronic notes

- Option 1:
 - Before printing, ensure that all comment boxes are situated ON the slide. If necessary, drag those that are outside the frame back onto the slide.
 - Select Print. This prints the comment boxes on the slides.
- Option 2:
 - In the Comments and Forms field of the Print window, select Documents in pulldown menu
 - Select Summarize Comments.
 - Select Print. This prints all the comments on a separate page after the slide. This is IDENTICAL to how PowerPoint prints slides and comments.



**Table of
Contents:
Days 01 - 02**

Each module name in the schedule is a hyperlink to that module's content in this guide.

Schedule:
All times shown here in local time.

**Cyber Course TGE CYBER EMBED SEC
Embedded Systems Security - India**
Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Bangalore, India

		6.42 hours	Monday, January 29, 2024
Day 01 Intro and Product Cybersecurity	9:00 AM - 9:30 AM	00 Introductions and Expectations Randall Brooks	
	9:30 AM - 10:00 AM	01 Product Cybersecurity General Overview Collins	
	10:00 AM - 10:45 AM	02 Aviation Product Cybersecurity Challenges Collins	
	10:45 AM - 11:00 AM	Break	
	11:00 AM - 11:25 AM	03 Collins Product Cybersecurity Collins	
	11:25 AM - 12:10 PM	Lunch	
	12:10 PM - 1:40 PM	04 Collins Product Cybersecurity SSDLC Collins	
	1:40 PM - 1:55 PM	Break	
	1:55 PM - 2:55 PM	05 Cybersecurity Certification and Standards Considerations Collins	
	2:55 PM - 3:25 PM	06 DO-326A Airworthiness Certification Process Collins	
Day 01 SSDLC and Airworthiness Security	3:25 PM - 3:40 PM	Break	
	3:40 PM - 4:25 PM	07 Aviation Threat Example Collins	
	4:25 PM - 4:55 PM	Daily wrap-up/Q&A	

		7.5 hours	Tuesday, January 30, 2024
Day 02 Crypto and Architecture Principles	9:00 AM - 10:30 AM	08 Cryptography Pt 1 Patrick S.	
	10:30 AM - 10:45 AM	Break	
	10:45 AM - 11:00 AM	08 Cryptography Pt 2 Patrick S.	
	11:00 AM - 11:45 AM	09 Architecture Pt 1 Randall Brooks	
	11:00 AM - 11:45 AM	Lunch	
	11:45 AM - 1:00 PM	09 Architecture Pt 2 Randall Brooks	
	1:00 PM - 1:15 PM	Break	
	1:15 PM - 1:45 PM	10 Secure Boot Randall Brooks	
	1:45 PM - 2:30 PM	11 Real-time Operating Sys (RTOS) Randall Brooks	
	2:30 PM - 2:45 PM	Break	
Day 02 Architecture, Secure Boot, and Secure Coding	2:45 PM - 3:15 PM	12 Software Assurance/Application Security (Short Course) Randall Brooks	
	3:15 PM - 4:45 PM	13 Secure Coding: Source Analysis and Bug Patterns Japheth Light	
	4:45 PM - 5:15 PM	Daily wrap-up/Q&A	

Reminders

- You must be connected to the internet for the hyperlinks in this document to work.
- In some cases (e.g., videos, Rshare, RTXConnect), the connection must be through Raytheon corporate servers.
- You are expected to use this document to accompany the instructor's presentation DURING the virtual class on ZfG. It should be open at all times *on your RTX-issued device ONLY*. **Note:** Be sure that you handle this interactive PDF according to the applicable security, intellectual property, and EX/IM compliance policies of Raytheon Technologies. (See the publication details on page 2 for additional details.)
- Direct any additional questions, feedback, or concerns to cyberlearning@rtx.com

Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

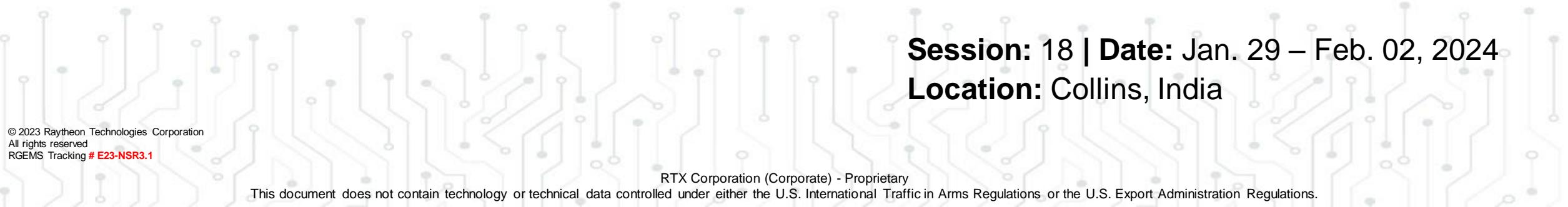
Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





Collins Aerospace
An RTX Business



TGE CYBERSEC

Embedded Systems Security

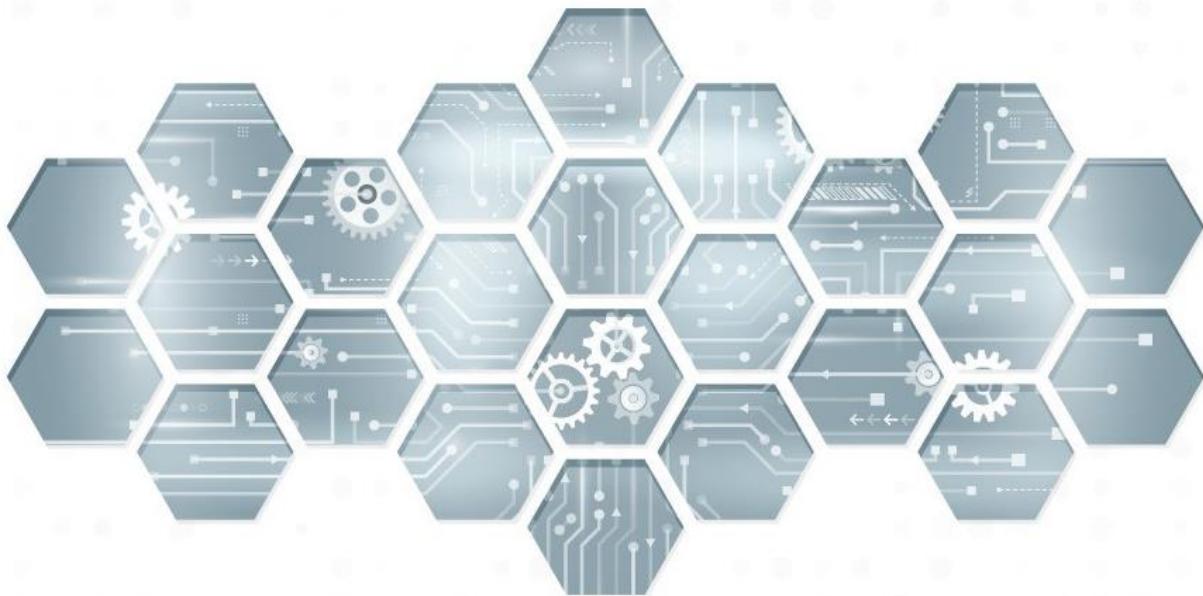
Module 00

Course Introduction and Expectations

Instructor: Randall Brooks

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India



Course Introduction and Expectations

Introduction Questions?

S.P.A.C.E.R.



- Safety & Security**
 - Fire Exits
 - Tripping/slipping hazards
 - Breaks & Restroom locations
 - Water/coffee locations
 - Security / ITAR
 - Workplace Safety Rules
- Purpose**
 - Cyber Training!
- Agenda**



- Code of Conduct**
 - Computer Screens down
 - Cell phones silent
 - One person speaking at a time
 - Keep our workplace clean
- Expectations**
 - Capture Team's Expectations
- Roles & Responsibilities**
 - Facilitators
 - Team Participants



Course schedule

Days 1-2

All times shown here are for the local time zone.

	6.42 hours	Monday, January 29, 2024
Day 01 Intro and Product Cybersecurity	9:00 AM - 9:30 AM	00 Introductions and Expectations Randall Brooks
	9:30 AM - 10:00 AM	01 Product Cybersecurity General Overview Collins
	10:00 AM - 10:45 AM	02 Aviation Product Cybersecurity Challenges Collins
	10:45 AM - 11:00 AM	Break
	11:00 AM - 11:25 AM	03 Collins Product Cybersecurity Collins
	11:25 AM - 12:10 PM	Lunch
	12:10 PM - 1:40 PM	04 Collins Product Cybersecurity SSDLC Collins
	1:40 PM - 1:55 PM	Break
Day 01 SSDLC and Airworthiness Security	1:55 PM - 2:55 PM	05 Cybersecurity Certification and Standards Considerations Collins
	2:55 PM - 3:25 PM	06 DO-326A Airworthiness Certification Process Collins
	3:25 PM - 3:40 PM	Break
	3:40 PM - 4:25 PM	07 Aviation Threat Example Collins
	4:25 PM - 4:55 PM	Daily wrap-up/Q&A

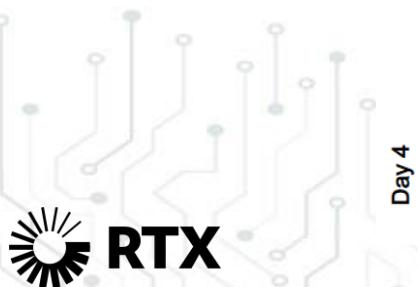
Cyber Course TGE CYBER EMB SEC
Embedded Systems Security
Session: 18 | Date: January 29 – February 2, 2024
Location: India

	7.5 hours	Tuesday, January 30, 2024
Day 02 Crypto and Architecture Principles	9:00 AM - 10:30 AM	08 Cryptography Pt 1 Patrick S.
	10:30 AM - 10:45 AM	Break
	10:45 AM - 11:00 AM	08 Cryptography Pt 2 Patrick S.
	11:00 AM - 11:45 AM	09 Architecture Pt 1 Randall Brooks
	11:00 AM - 11:45 AM	Lunch
	11:45 AM - 1:00 PM	09 Architecture Pt 2 Randall Brooks
	1:00 PM - 1:15 PM	Break
	1:15 PM - 1:45 PM	10 Secure Boot Randall Brooks
Day 02 Architecture, Secure Boot, and Secure Coding	1:45 PM - 2:30 PM	11 Real-time Operating Sys (RTOS) Randall Brooks
	2:30 PM - 2:45 PM	Break
	2:45 PM - 3:15 PM	12 Software Assurance/Application Security (Short Course) Randall Brooks
	3:15 PM - 4:45 PM	13 Secure Coding: Source Analysis and Bug Patterns Japheth Light
	4:45 PM - 5:15 PM	Daily wrap-up/Q&A



Course schedule Days 3-5

All times shown here are for the local time zone.



	7 hours	Wednesday, January 31, 2024
Day 03 Secure Coding and Labs	9:00 AM - 10:00 AM	14 SAST/DAST (includes Coverity Overview) Randall Brooks/Japheth Light
	10:00 AM - 10:30 AM	14a Lab: SAST/DAST Lab (including Fuzzing) 150 min
	10:30 AM - 10:45 AM	Break
	10:45 AM - 12:15 PM	14a Lab: SAST/DAST Lab (including Fuzzing) 150 min
Day 03 ARM and Reverse Engineering	12:15 PM - 1:00 PM	Lunch
	1:00 PM - 2:30 PM	15 ARM Assembly Japheth Light
	2:30 PM - 2:45 PM	Break
	2:45 PM - 4:15 PM	16 Software Reverse Engineering: Ghidra and Debuggers Japheth Light
	4:15 PM - 4:30 PM	Break
	4:30 PM - 5:00 PM	16a Lab: ARM Reverse Engineering Pt 1 75 min
	5:00 PM - 5:30 PM	Daily wrap-up/Q&A
	7.25 hours	Thursday, February 1, 2024
Day 04 Reverse Engineering	9:00 AM - 9:45 AM	16a Lab: ARM Reverse Engineering Pt 1 75 min
	9:45 AM - 10:30 AM	16a Lab: ARM Reverse Engineering Pt 2 120 min
	10:30 AM - 10:45 AM	Break
	10:45 AM - 12:00 PM	16a Lab: ARM Reverse Engineering Pt 2 120 min
Day 4 RE, Software Protection, and Pentesting	12:00 PM - 12:45 PM	Lunch
	12:45 PM - 1:45 PM	16a Lab: Reverse Engineering Walkthrough Japheth Light
	1:45 PM - 2:00 PM	Break
	2:00 PM - 2:30 PM	17 Software Protection Japheth Light
	2:30 PM - 3:30 PM	16a Lab: ARM Reverse Engineering Pt 3 60 min
	3:30 PM - 3:45 PM	Break
	3:45 PM - 4:45 PM	18 Pentesting (including Scapy Protocol Fuzzing) Patrick S.
	4:45 PM - 5:15 PM	18a Lab: Pentesting 60 min
	5:15 PM - 5:45 PM	Daily wrap-up/Q&A

Cyber Course TGE CYBER EMBED SEC

Embedded Systems Security

Session: 18 | Date: January 29 – February 2, 2024
Location: India

	6.75 hours	Friday, February 2, 2024
Day 05 Hardware Hacking	9:00 AM - 9:30 AM	18a Lab: Pentesting 60 min
	9:30 AM - 10:15 AM	19 JTAG and Hardware Hacking Patrick S.
	10:15 AM - 10:30 AM	Break
	10:30 AM - 11:15 AM	19a Lab: Hardware Hacking 135 min
Day 05 Capstone	11:15 AM - 12:00 PM	Lunch
	12:00 PM - 1:30 PM	19a Lab: Hardware Hacking 135 min
	1:30 PM - 1:45 PM	Break
	1:45 PM - 3:45 PM	20 Lab: Capstone 120 min
	3:45 PM - 4:30 PM	20 Lab: Capstone Out-brief
	4:30 PM - 5:00 PM	Final wrap-up/Q&A

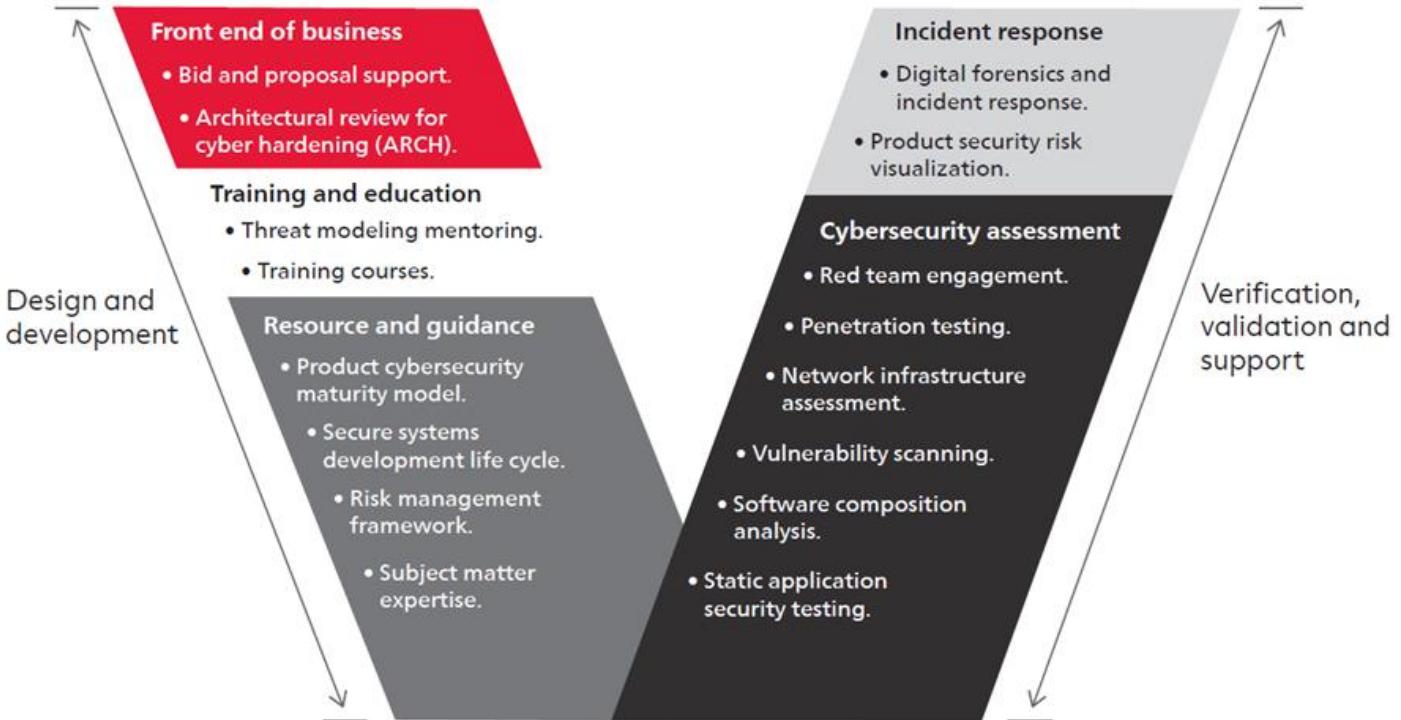
CODE Center Mission

CODE Center Goals:

- Provide classified and unclassified cyber services to internal and external customers
- Test resiliency of mission-critical systems against cyber attacks from sophisticated adversaries
- Offer continually evolving technology, tools and tactics against increasing cyber threats
- Provide engineering environment to integrate cyber technologies from RTX, vendors and partners
- Evaluate cyber technologies
- Develop cyber security workforce



The CODE Center supports all stages of a program life cycle.



Subject Matter Experts (SME) can support programs and assess product cybersecurity on demand.

Deliver efficient and cost-effective cyber services across the RTX enterprise



CODE Center Organization and Teams

The CODE Center Director is the Chief Product Cyber Security Officer.



Security Architecture Team (PSARE)

- The PSARE team serves as a product security hub for all RTX products.
- We provide security guidance and training, architecture evaluation, threat modeling, security strategy and tools, Secure Systems Development Lifecycle (SSDLC) practices, and the Product Cybersecurity Maturity Model (PCMM).

Product Security Incident Response Team (P-SIRT)

- The P-SIRT team leads activities related to Significant Product Cybersecurity Incidents (SPCI).
- Activities include detection, analysis, response, and possible remediation of product cybersecurity incidents.

Project Zero (P-Zero)

- P-Zero performs internal vulnerability assessments on hardware and software to research and patch serious security vulnerabilities, to improve our internal understanding of how weaponized exploit attacks work and to drive long term security engineering improvements to our product and service offerings

Program Services

- The Program Services team serves as a team of Cyber Security Evaluators to assist RTX programs with Cyber needs
- We provide Vulnerability assessments, network assessments, penetration tests or subject matter expertise to assist programs with identifying cyber vulnerabilities. The team's efforts allow programs to work towards their Authorization to Operate (ATO) accreditations.

Product Security Intelligent Infrastructure Team (PSII)

- The PSII team applies automated cybersecurity techniques to DevOps in order to achieve DevSecOps and continuous ATO.

Operations (OPS)

- OPS maintains CODE Center facilities, IT assets and processes.
- OPS acquires and maintains hardware and software assets needed to accomplish the CC mission.

The CC provides value added services based on RTX program needs.



CODE Center Services

The CODE Center Services and Capabilities catalog provides detailed service descriptions

Proposal Support

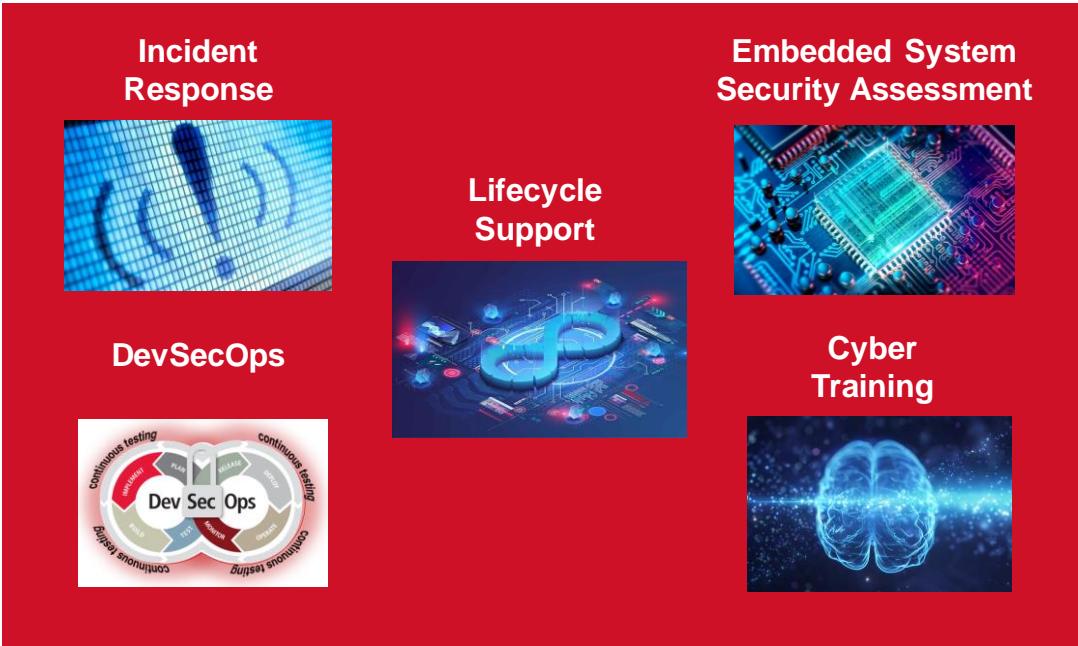
- RFI/RFP response
- Oral presentations
- Demonstrations

Infrastructure

- RF assessment facilities
- Commercial and proprietary tools library
- DevSecOps
- Cyber range

Security Architecture

- Architectural Review for Cyber Hardening (ARCH)
- Factory cyber assessments
- Threat modeling
- Security tool integration
- Product cyber maturity model assessments
- Secure systems development lifecycle



Product Security Incident Response

- Corporate level incident response for fielded products
- Digital forensics
- Validate reported issues and assist in risk assessment, mitigation, and customer communications

code.center@rtx.com

Program Services

- RMF support
- Cyber/information assurance subject matter expert program support
- Software assurance testing
- Penetration testing
- Vulnerability assessments
- STIGLER support

Project Zero

- Full scope cyber vulnerability assessment of embedded systems
- Hardware, software, networks and system level analysis

Training

- Training for cybersecurity
- Certification bootcamps
- Cyber learning center
- Penetration testing
- RMF training and lessons learned



Housekeeping

- Please set your cell phones to silent.
 - No recording allowed (audio or video)
 - If you need to take a call, please do so outside the classroom.
 - There are breaks scheduled mid-morning and mid-afternoon and for lunch, which would be optimal times to check email, text, and make calls.
 - This course is intended to be interactive and hands on.
-
- **Please ask questions and contribute to the conversation!**



Material is Raytheon Technologies Proprietary and export-controlled as marked.



Cybersecurity



C-I-A Triad



Class Discussion — cybersecurity background



Q1: Do you have a background in Cybersecurity?

- a) Yes
- b) No

Q2: Do you have a background as a Developer? (e.g., C/C++, Java, Python, scripting, etc.)

- a) Yes
- b) No



Whiteboard class discussion: Based on the schedule, which topics most interest you?



Instructions

- Review the course schedules on the previous slides.
- Choose 1-3 topics that interest you most.
- Write your choices in the flip chart provided.



Initial Questions?

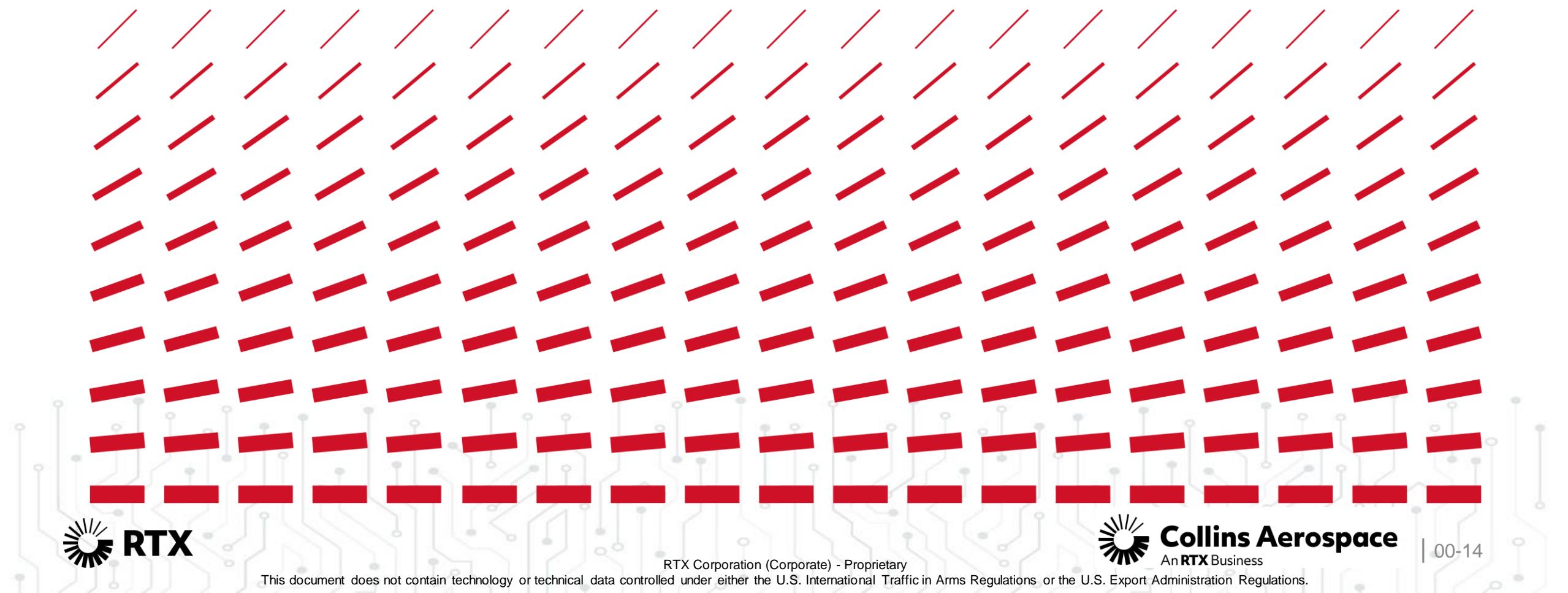


Thank you.



Remember:

- You have 30 minutes to complete the 25-question assessment.
- If you complete the assessment early, your instructor has additional guidance on when the class resumes.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





Collins Aerospace
An RTX Business



TGE CYBERSECURITY

Embedded Systems Security

Module 01

Product Cybersecurity Training

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

Agenda

COLLINS PRODUCT CYBERSECURITY TRAINING



- **Module 1: Cybersecurity General Overview**
- Module 2: Aviation Product Cybersecurity Challenges
- Module 3: Collins Product Cybersecurity: Policy Flow down
- Module 4: Collins Secure System Development Life Cycle
- Module 5: Cybersecurity Certification and Standards Considerations
- Module 6: DO-326A Airworthiness Certification Process
- Module 7: Aviation Threat Example



Module 1: Product Cybersecurity General Overview



- What is Product Cybersecurity
- Why we care
 - Customer & Regulatory Landscape
 - Threat Risks & Threat Actors
- Security properties: C-I-A Triad
- Vocabulary and definition in aviation



Product Cybersecurity

WHAT IS IT?



Product Cybersecurity is defined as the mitigation of risks with respect to confidentiality, integrity or availability of Collins products or services to standards agreed to by the customer and meet any internal minimum-security requirements.



Product cybersecurity



Growth in regulatory and customer requirements due to increased cyber attacks in industry

Regulatory



EXECUTIVE ORDER
ON IMPROVING THE NATION'S
CYBERSECURITY



Civil:

- FAA/Customer: Code Signing
- Maintain confidentiality of airborne software
- Vulnerability Disclosure Program

DoD / NIST:

- Secure Software Development Framework (SSDF)
- SW Modernization & Zero Trust
- Cyber reporting requirements

Customers



BOEING



BOMBARDIER

Critical customer requirements

- Digital signature
- Hardware based security features
- Static Code Analysis
- Supply chain risk management
- Cyber resiliency requirements

Vulnerability scanning & reporting

Industry & DoD

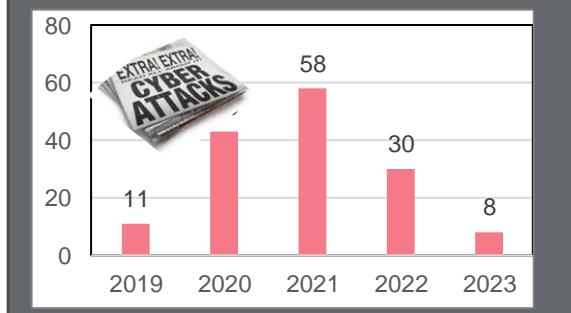


AVIATION ISAC



- Aerospace industry a target for cybercriminals / data theft;
- Industry standard solutions for operators
- Expanding academic research focus
- DoD cyber supply chain research

Incident Response



Increase in Issues, Incidents & Cost

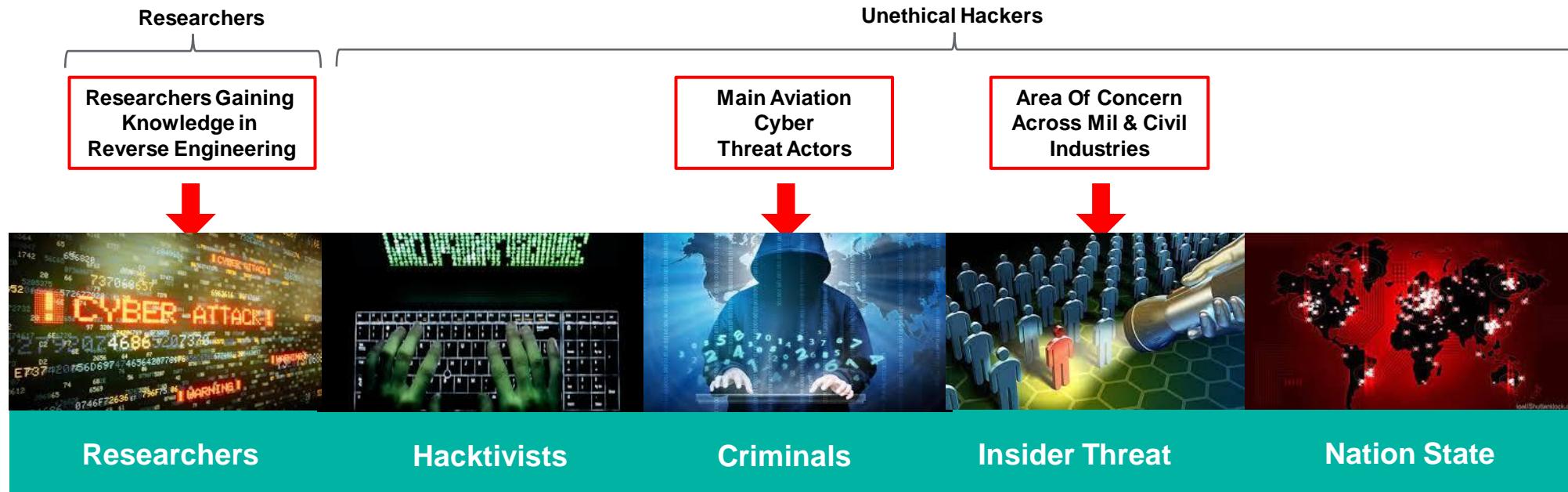
- Security researcher publishes paper on reverse engineering of Collins software (04/22)
- Increase in COTS vulnerabilities – manual analysis
- Increase in insider cyber threat

Continue to drive use of proactive cyber practices to ensure our products remain safe & secure



Threat risks & Threat Actors

Evolving threat landscape: people and groups of concern are increasing



- Expose Vulnerabilities**
- IOActive AFD-3700 research
 - Garmin GPS / Mode-S Spoofing
 - Collins Cabin Systems
 - Boeing 787 CIS / MS

- Disruption**
- Russian Military Records & TV Signals
 - iPhones during Hong Kong protests
 - Nestle sensitive data

- Ransomware / IP Theft**
- SITA (2022)
 - Swissport (2022)
 - E.M.I.T (2021)
 - Embraer (2020)
 - ST. Eng. Aero. (2020)

- Data Theft & Lack of Trust**
- DHS (2022)
 - City of Dallas (2021)
 - GE (2020)
 - Microsoft (2020)

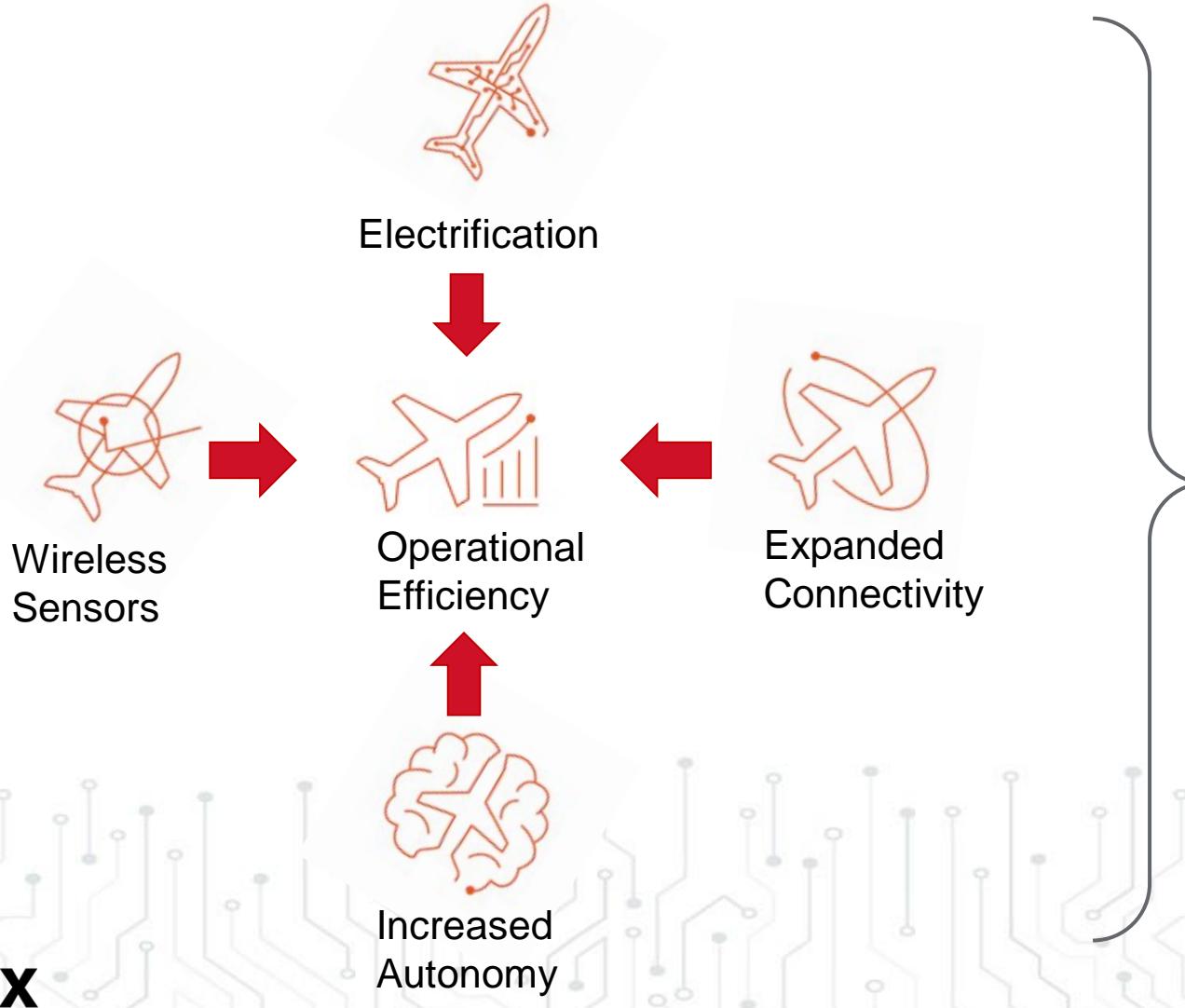
- Advanced Persistent Threat**
- Russia & Ukraine (2022)
 - North Korea vs. Fintech/media (2022)
 - Russia & SolarWinds (2021)

Cyber threats and attacks facing aviation industry continue to grow



Product Cybersecurity

WHY DO WE CARE?



Security Properties

C-I-A TRIAD

Confidentiality

- A function, service, or data is only accessible by intended systems or parties.
- Often achieved with encryption or access control features

Integrity

- A function, service, or data can only be modified consistently with a prescribed process.
- E.g., Dataload, ATC message, Airspeed from a sensor

Availability

- A function, service, or data is available when it is needed.
- Not terribly different than safety, security often deals with denial-of-service issues (DoS).



Vocabulary and definition in aviation



Asset:

The logical and physical resources of the aircraft which contribute to the airworthiness of the aircraft, including functions, systems, items, data, interfaces, processes and information

Threat:

Any circumstance or event with the potential to harm an information system through unauthorized access, destruction, disclosure, modification of data, and/or denial of service. Threats arise from human actions and natural events. [CNSSI 4009, adapted]

Vulnerability:

A flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy.

Source: EUROCAE AERONAUTICAL INFORMATION SYSTEM SECURITY GLOSSARY ER-013



Vocabulary and definition



Risk (Security)

Exposure to the possibility of harm. The risk of an event is a function of the severity of the adverse event and the level of threat of that event.

$$\text{Asset} + \text{Threat} + \text{Vulnerability} = \text{Risk}$$

Threat Condition

A **condition having an effect on the airplane and/or its occupants**, either direct or consequential, which is **caused** or contributed to by one or more acts of **intentional unauthorized electronic interaction, involving cyber threats**, considering flight phase and relevant adverse operational or environmental conditions.

Failure Condition

A **condition having an effect on the airplane and/or its occupants**, either direct or consequential, which is **caused** or contributed to by **one or more failures or errors**, considering flight phase and relevant adverse operational or environmental conditions, or external events

Source: EUROCAE AERONAUTICAL INFORMATION SYSTEM SECURITY GLOSSARY ER-013



Vocabulary and definition



Security Countermeasure

A feature or **function used to mitigate or control a threat condition**, including its operation, management, and maintenance. A security countermeasure may be implemented by systems, people, or procedures, on-board or off-board, or some combination thereof.

Airworthiness Security

The **protection of the airworthiness of an aircraft and its occupants from the information security threat**: harm due to human action (intentional or unintentional) using access, use, disclosure, disruption, modification, or destruction of data and/or data interfaces. This also includes the consequences of malware and forged data and of access of other systems to aircraft systems.

Source: EUROCAE/RTCA Term and definition: 120-21 PMC-2151 (rtca.org)

Source: EUROCAE AERONAUTICAL INFORMATION SYSTEM SECURITY GLOSSARY ER-013



QUIZ 1 on C-I-A TRIAD



How would you rate the following security attributes below?

Rate based on order of importance(1 to 3, N/A; 1 being most important)

	Confidentiality	Integrity	Availability
Safety Critical Function			
Air Traffic Control Communication			
Passenger Data			
Military Mission Data			



QUIZ 1 Results



How would you rate the following security attributes below?

Rate based on order of importance(1 to 3, N/A; 1 being most important)

	Confidentiality	Integrity	Availability
Safety Critical Function	NA	2	1
Air Traffic Control Communication	NA	1	1
Passenger Data	1	2	3
Military Mission Data	1	2	3

Note: Order of importance of these attributes can vary with each specific implementation.



Questions?



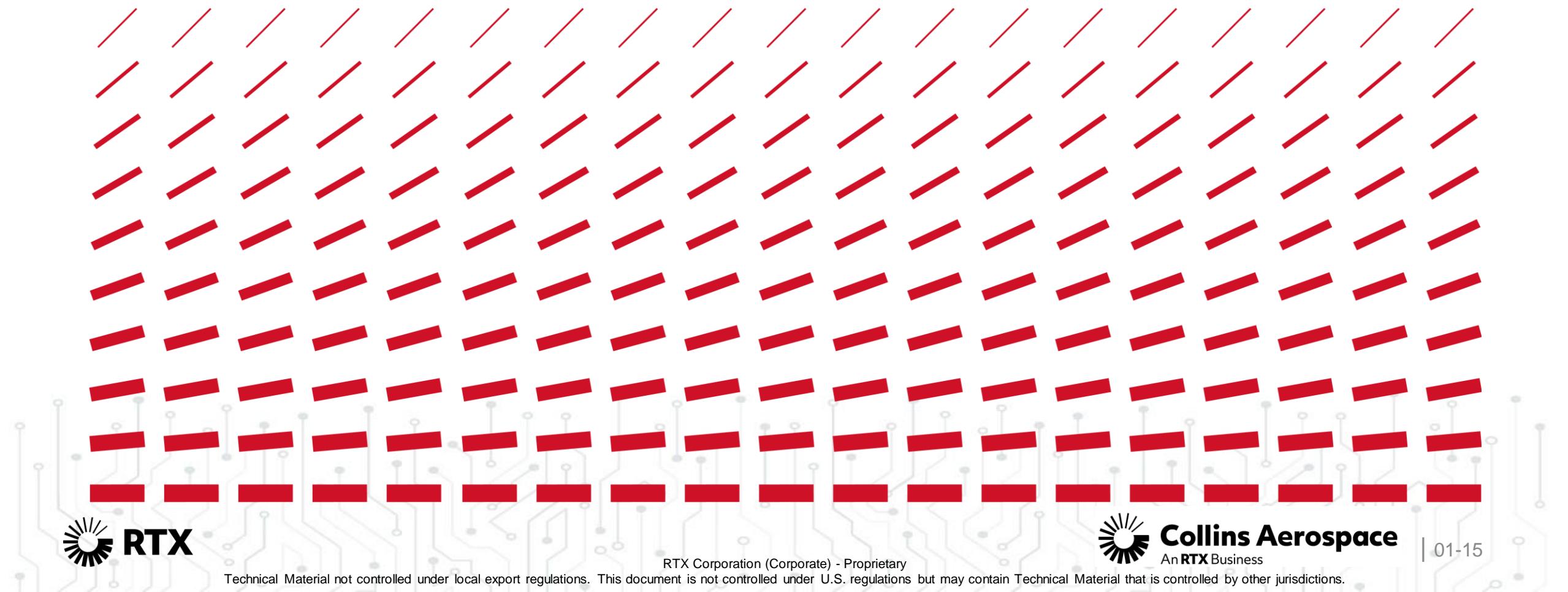
RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.



| 01-14

Thank you.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





RTX TGE
Technology & Global
Engineering



TGE CYBERSECURITY

Embedded Systems Security

Module 02

Aviation Product Cybersecurity Challenges

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India



Collins Aerospace
An RTX Business

Agenda

COLLINS PRODUCT CYBERSECURITY TRAINING



- Module 1: Cybersecurity General Overview
- **Module 2: Aviation Product Cybersecurity Challenges**
- Module 3: Collins Product Cybersecurity: Policy Flow down
- Module 4: Collins Secure System Development Life Cycle
- Module 5: Cybersecurity Certification and Standards Considerations
- Module 6: DO-326A Airworthiness Certification Process
- Module 7: Aviation Threat Example



Agenda

COLLINS PRODUCT CYBERSECURITY TRAINING



- Module 1: Cybersecurity General Overview
- **Module 2: Aviation Product Cybersecurity Challenges**
- Module 3: Collins Product Cybersecurity: Policy Flow down
- Module 4: Collins Secure System Development Life Cycle
- Module 5: Cybersecurity Certification and Standards Considerations
- Module 6: DO-326A Airworthiness Certification Process
- Module 7: Aviation Threat Example



Module 2: Aviation PRODUCT Cybersecurity Challenges



- Aviation Cybersecurity Challenges
- Aviation Threat Evolution
- Threat Landscape in Aviation
- Threat Evolution in Aviation
- EUROCONTROL Aviation Cyber Events Map
- Aviation Cybersecurity in the Press
- Collins Cybersecurity Issues



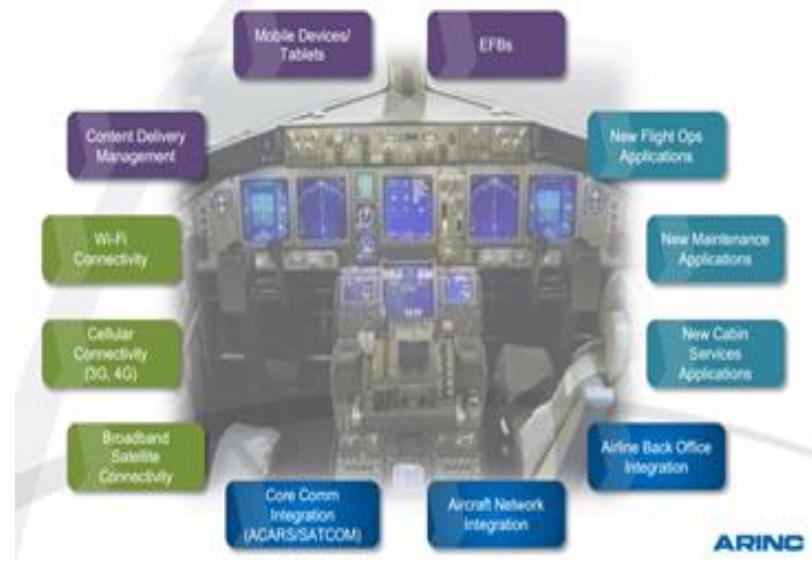
Aviation Cybersecurity Challenges

DRIVEN BY THE DIGITAL TRANSFORMATION



Aerospace is moving to digitalization and hyper-connectivity

- Multiple communication means (VHF, SATCOM, LTE, Wi-Fi,...)
- More IT technologies (TCP/IP) being used
- Cost effective solution lead 3rd Party Component (TPC) usage
- IT related vulnerabilities are now applicable to the A/C
- Increased ***opportunities for attacks***
- Successful attacks can have ***safety and/or operational impact***



Source: "Product Security: manufacturer point of view" (Journee INSA 2015, Airbus)



Aviation Threat Evolution



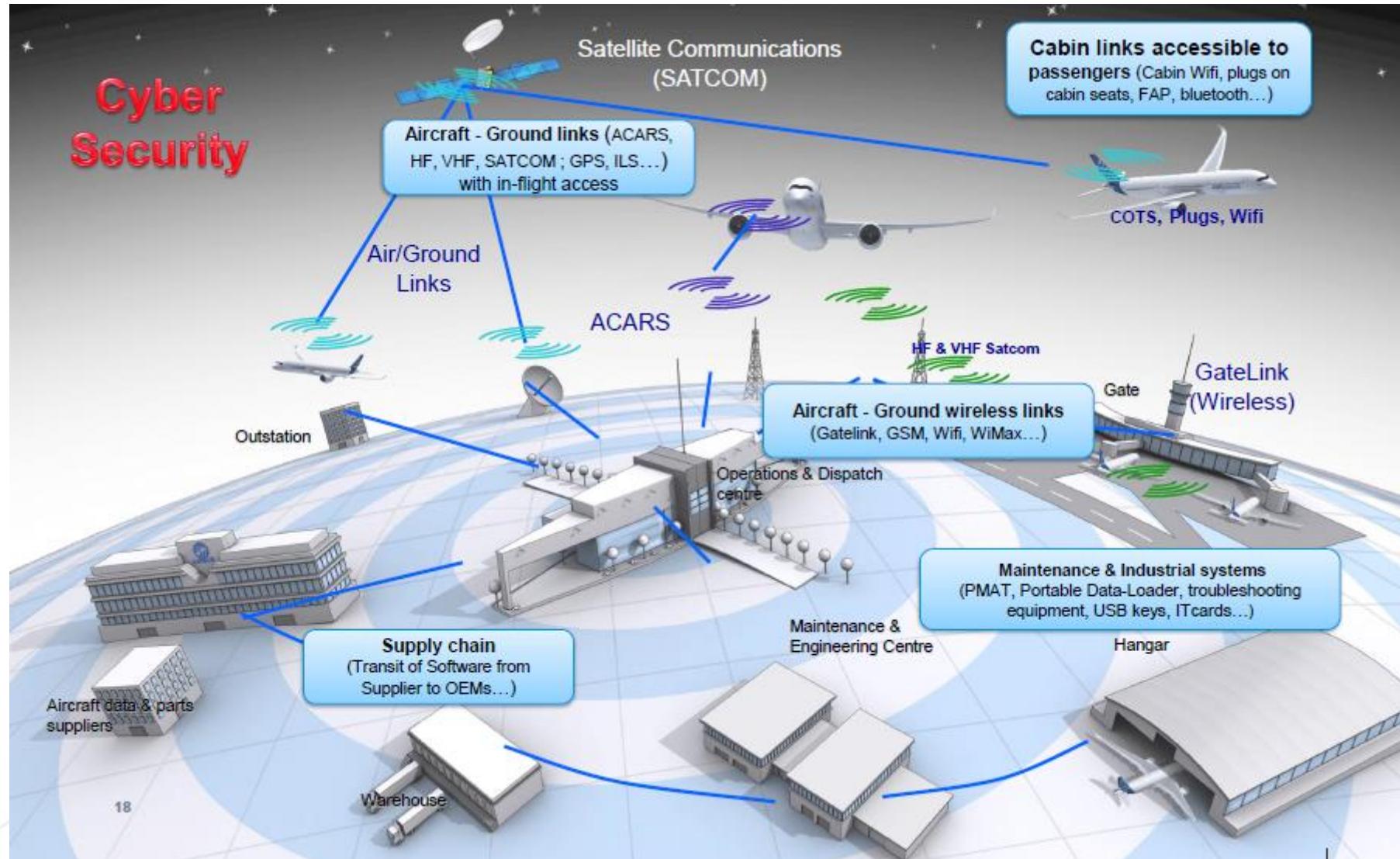
Failure Conditions

Physical Security Threats

Cybersecurity Threat Conditions



Threat Landscape in Aviation



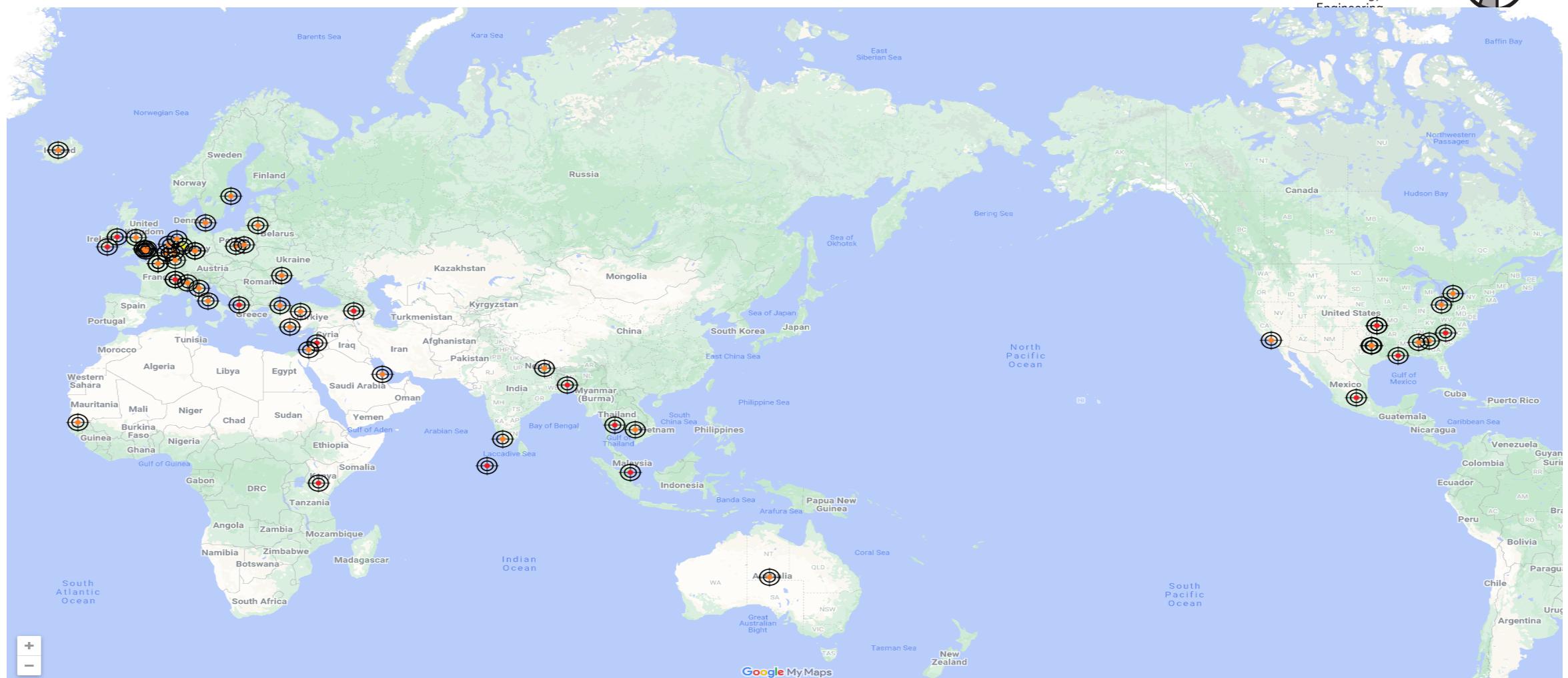
Source: "Threatscape" of Airbus Group products (Journée INSA 2015, Airbus)

Threat Evolution in Aviation



- Physical threats
- Cybersecurity threats

Eurocontrol Aviation Cyber Events Map



<https://www.google.com/maps/d/embed?mid=1ptVlma0CZqoPiN-zsomzbRVQDRS7BXGk&ll=37.03962057979058%2C147.86449642519779&z=3>

<https://www.eurocontrol.int/cybersecurity>



Aviation Cybersecurity in the Press



2012: -DEFCON 2012 Brad Haynes, ADS-B Fantom Flight demo

2013: -Hugo Téso, "Hack-in-a Box" - ACARS, FMS, CEH like methodology

2014: -Ruben Santamarta, A wake-up call SATCOM Security

-David Stupples, SAI Conference 2014- Is it possible to "cyberjack" modern airplanes

2015: -DEFCON 2015 Dr. Phil Polstra and Captain Polly – Cyber hijacking Airplanes: Truth or Fiction

-Chris Roberts IFE C2 & FBI Investigation

2016: -Ruben Santamarta, IOActive "In Flight Hacking System" - Panasonic IFE

2017: -Dr. Robert Hickey, DHS Led Team Demonstrates That Commercial Aircraft Can Be Remotely Hacked

-Anthony Lam José Fernandez Richard Frank, Cyberterrorists Bringing Down Airplanes: Will it Happen Soon?

-Hugo Teso, High Level Conference on Cybersecurity in Civil Aviation

-Dave/Karit, DEFCON25 GPS time spoofing



Aviation Cybersecurity in the Press



2018: -Ruben Santamarta, Black Hat USA 2018 "Last Call for SATCOM Security"

2019: -Ruben Santamarta, BlackHat USA "Arm IDA and Cross Check: Reversing the Boeing 787's Core Network"

-Rapid7, Investigating CAN Bus Network Integrity in Avionics Systems

2020: -Oxford TCAS/GPWS/ILS Spoofing

-FOMAX - UK security researcher presents at DefCon's Aerospace Village on perceived vulnerabilities found in our router application interface.

-CMU-900 presentation at DefCon Aerospace Village

2021: -USENIX Conference Presentation on rehosting CMU-900 in an emulator for security vulnerability discovery

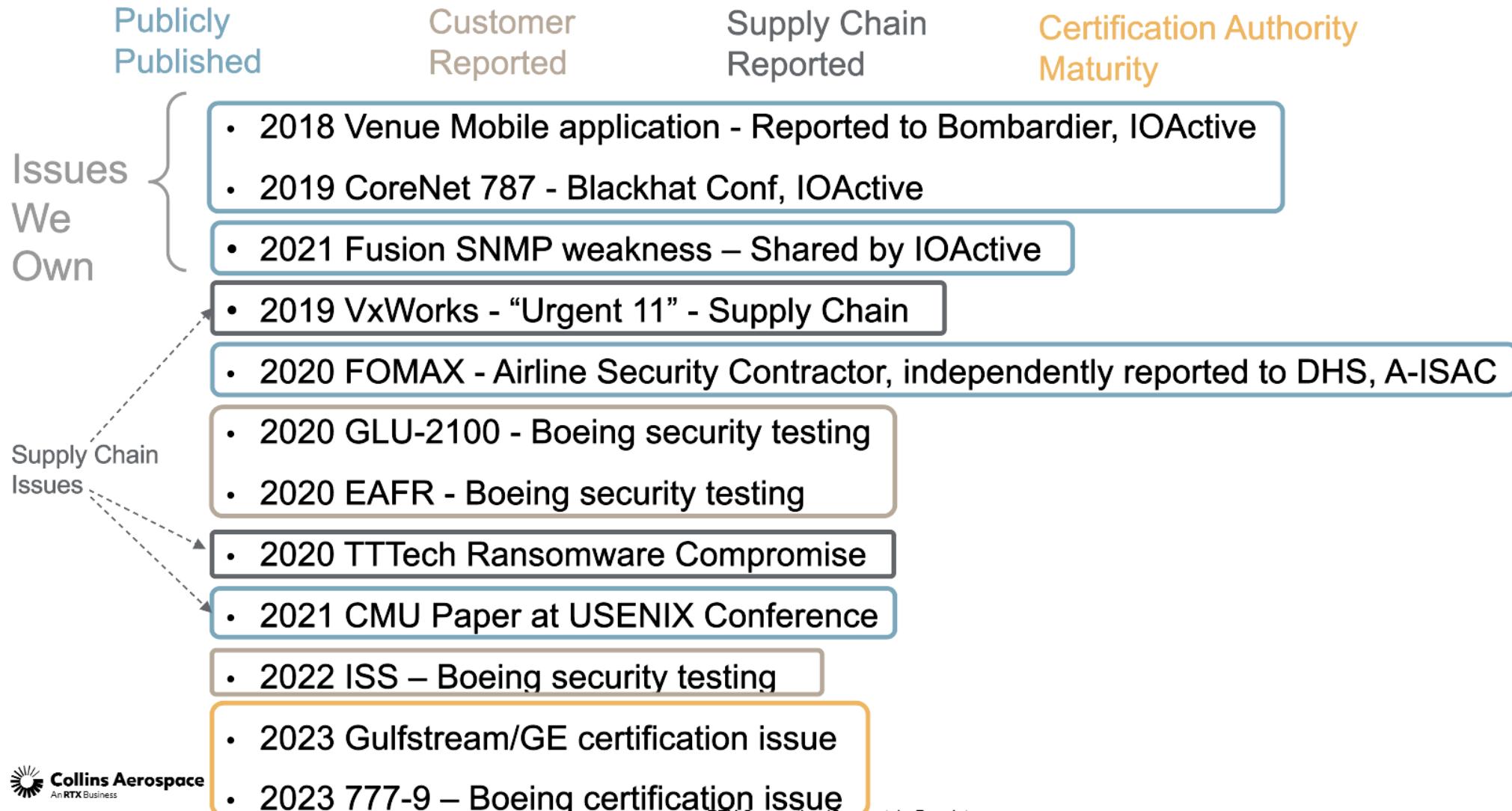
2022: -Ruben Santamarta, IOActive, publication of "Reverse Engineering of DAL-A Certified Avionics: Collins' Pro



Line Fusion—AFD-3700™

Collins Cybersecurity Issues

5 Years of Externally Reported Product Cybersecurity Issues



 Collins Aerospace
An RTX Business



 Collins Aerospace
An RTX Business

Questions?

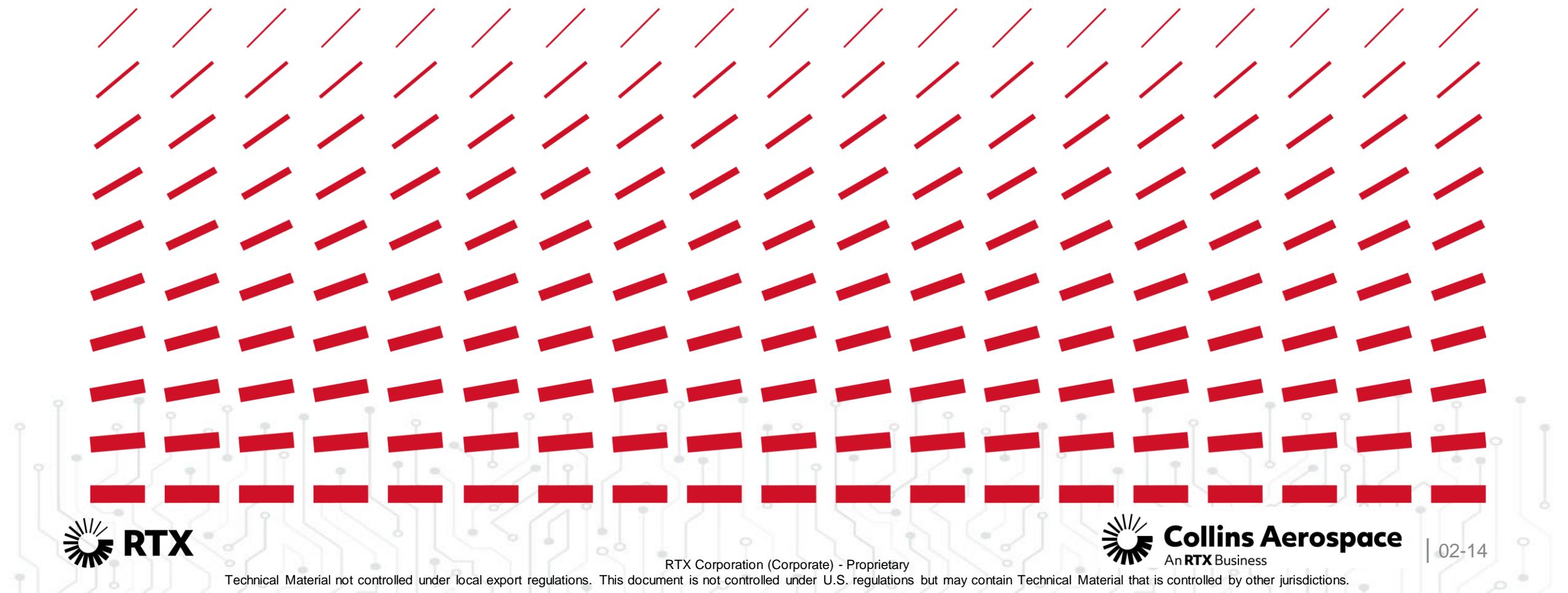


Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.



| 02-13

Thank you.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





RTX TGE
Technology & Global
Engineering



TGE CYBERSECURITY

Embedded Systems Security

Module 03

Collins Product Cybersecurity

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India



Collins Aerospace

An **RTX** Business

Agenda

COLLINS PRODUCT CYBERSECURITY TRAINING



- Module 1: Cybersecurity General Overview
- Module 2: Aviation Product Cybersecurity Challenges
- **Module 3: Collins Product Cybersecurity: Policy Flow down**
- Module 4: Collins Secure System Development Life Cycle
- Module 5: Cybersecurity Certification and Standards Considerations
- Module 6: DO-326A Airworthiness Certification Process
- Module 7: Aviation Threat Example



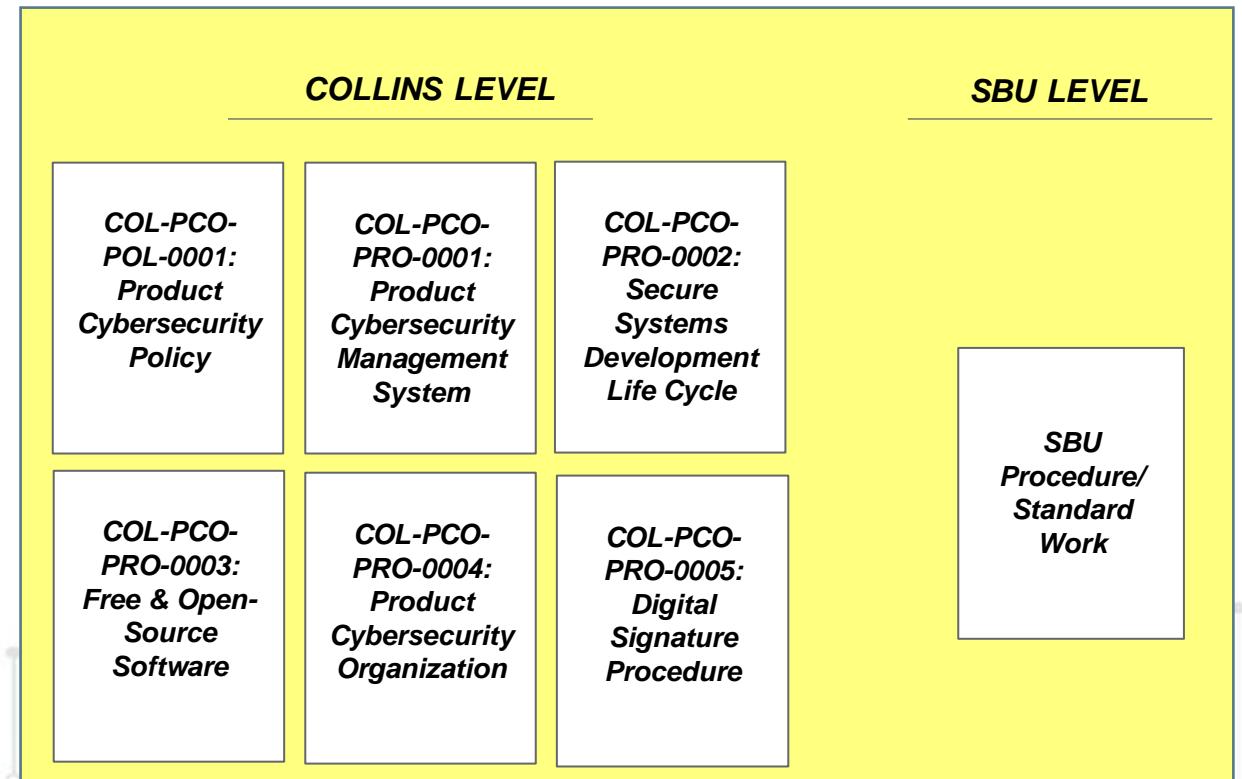
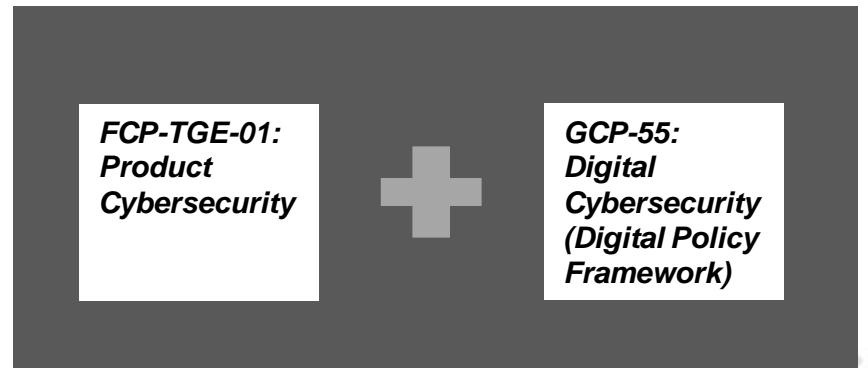
Module 3: Collins Aerospace Product Cybersecurity



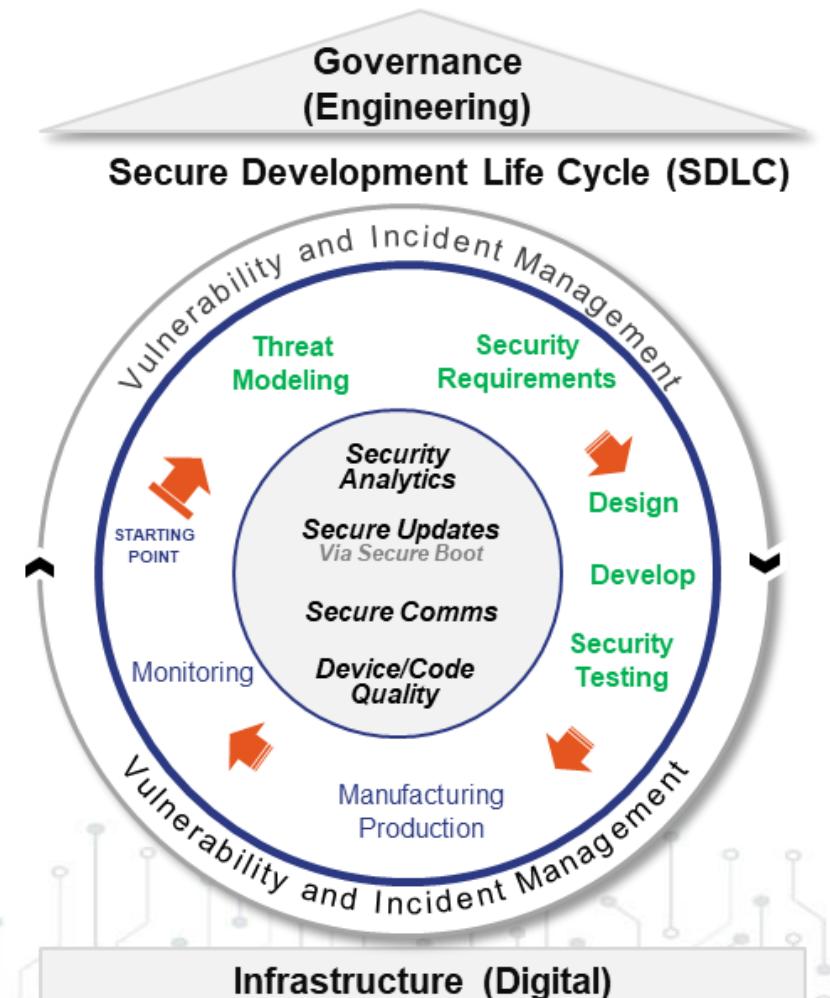
- Collins Aerospace Product Security Policy and Procedure
 - Overview
 - Approach to Product Development
 - Product Cybersecurity Framework
 - Product Cybersecurity Assurance Team (PCAT)



Product Cybersecurity Policies



Product Cybersecurity



- Policy [COL-PCO-POL-0001]
 - Establishes security framework and common expectations
- Management System [COL-PCO-PRO-0001]
 - Process for vulnerability and incident management
 - Defines criteria for security issue reporting
- Secure Development Life Cycle [COL-PCO-PRO-0002]
 - Minimum Security Related activities
 - Provides technical design considerations
 - Includes risk management of third parties
- Free and Open- Source Software [COL-PCO-PRO-0003]
 - Requires risk assessment and approval for use
- Product Cybersecurity Organization [COL-PCO-PRO-0004]
 - Defines roles & responsibilities
- Digital Signature Procedure [COL-PCO-PRO-0005]
 - Collins defined method for signing software



Approach to Product Development



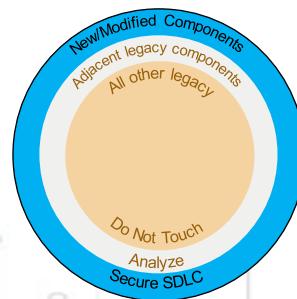
New Products

- Governed by secure development process (SSDLC)
- Security Risk Assessment (SRA) is required
- SRA drives technical requirements and validations
- Includes supply chain security obligations



Modified Products

- Same process as “New” for modified portions
- Code reuse allowable based on threat model assessment
- Vulnerabilities found in legacy code are dispositioned based on criticality



Legacy Products

- Assess product if a vulnerability is discovered or disclosed externally
- Assess product if internal risk assessment warrants concern
- Prioritize based on potential impact and exposure



Implementing Balanced Criteria for Risk Management



Product Cybersecurity Office



Leadership & Coordination

External

- Influencing & driving Industry Activities & Standards: FAA/EASA Policies
- Cyber Assessment

Internal

- Cyber Roadmap(s)
- RTX CODE Center & Cross BU
- Cross Functional & Cross Discipline collaboration
- Support to GETC-IN

Process

Governance & Process

- Policy, Process, Procedures & Work Instructions
- Process health & assessments
- Safety Committee Rep
- Product Cyber Maturity Model
- Overall Cyber Metrics

Incident Management

- Coordinate Issue/Incident Management Response with Digital & Supply Chain
- Manage Response Metrics
- Cyber Playbook for One Collins Approach

Training, Capabilities & Support – Expanded Roles

Communication & Training

- Cyber Training (RTX & CATU)
- Provide Instructors & Mentors
- Cyber Discipline Health Assessment & Promotion
- Communication (Internal & External / Customer(s))

Security Capabilities

- Integrity: Digital Signature
- Confidentiality: Encryption
- Consistent Deployment of Secure Design Features
- Software Bill of Materials
- Pave-The-Road Capabilities

SBU Support & Deployment

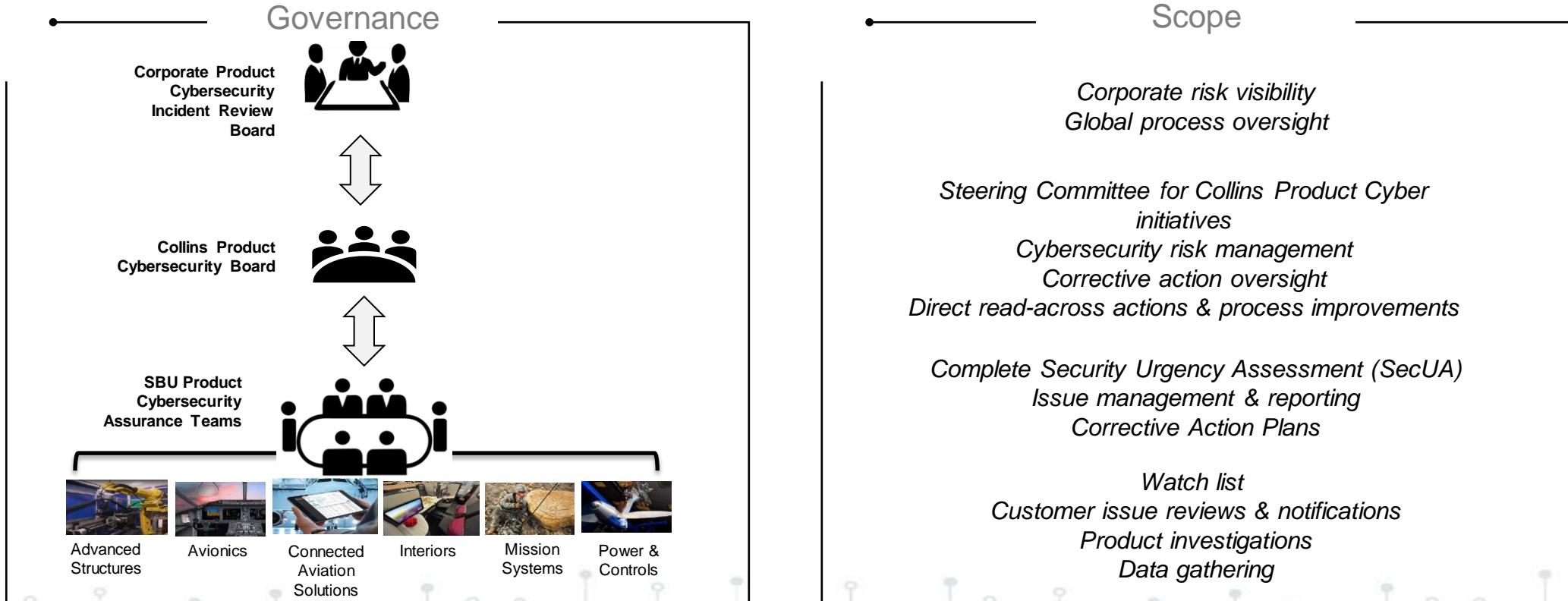
- SBU Support
- Non-Advocate Reviews
- Support to SBUs, GETC-IN & PR Capability & Tool Deployment
- DPLC / DevSecOps
- Security Tools

BU: Business Unit
 CATU: Collins Aerospace Technical University
 DPLC: Digital Product Life Cycle
 EASA: European Union Aviation Safety Agency

FAA: Federal Aviation Agency
 GETC-I/PR: Global Engineering Technology Centers, India/Puerto Rico
 SBU: Strategic Business Unit



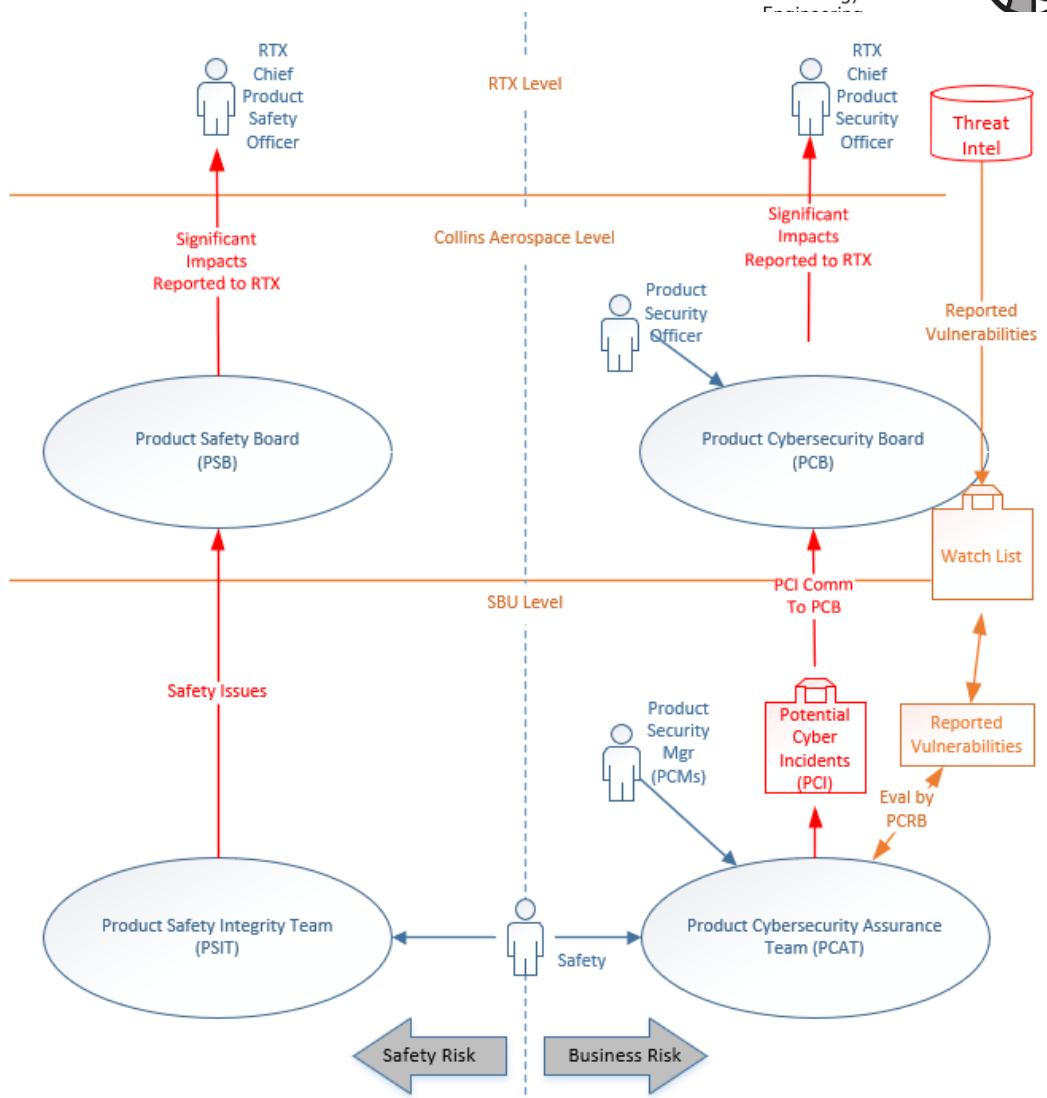
Product Cybersecurity Framework



Product Cybersecurity Assurance Team (PCAT)



- One per SBU
- Manage Security events (30d clock watchlist items)
- Determine the risk level with Security Urgency Assessment (30+30 days clock)
- Manage Potential Security Issue (PCI) and present remediation plan to PCO
- Links with other internal stakeholders
- Participate to establish the Product Cybersecurity Governance
- Promote Product Cybersecurity within the SBU



SECURITY URGENCY ASSESSMENT (SecUA)



Assessment based on:

- Vulnerability Factors – How bad is the vulnerability on its own? Similar to the Common Vulnerability Scoring System (CVSS) (<https://www.first.org/cvss/>)
- System Exploitability Factors – Is the vulnerability exploitable for this product?
- Technical Impact – How much of an impact to confidentiality, integrity or availability?
- Business Impact – What business impact could the exploitation of the vulnerability impart?

Risk: Enter the name of the product and PSEC-ID here		Likelihood		
Vulnerability Factor		System Exploitability Factor		
Instrumentation Needed	Awareness	Skill level	Opportunity	Mitigations
1 - Theoretical	3 - Vulnerability Information Available	1 - Security penetration testing skills	1 - Physical access required	2 - Existing system mitigations or protections prevent the exploit
1	3	1	1	2
Overall likelihood:		0.229	LOW	

Functional Impact				
Safety Impact	Mission Impact	Loss of Confidentiality	Loss of Integrity	Loss of Availability
5 - Major	0 -	0 -	9 - Loss of integrity of critical data/function	0 -
5	0	0	9	0
Functional impact:		9.000	HIGH	

Business Impact			
Deliverable Impact	Operational Impact	Scale of Impact	Volume
0 -	9 - Loss of end system operation	4 - Multiple on one platform	5 - Medium
0	9	4	5
Business impact:		2.222	LOW

Overall Risk Severity = Likelihood x Impact				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Low	Low	Medium
	LOW	LOW	Medium	HIGH
Likelihood				

COL-PCO-FRM-0001 - SecUA



Questions?

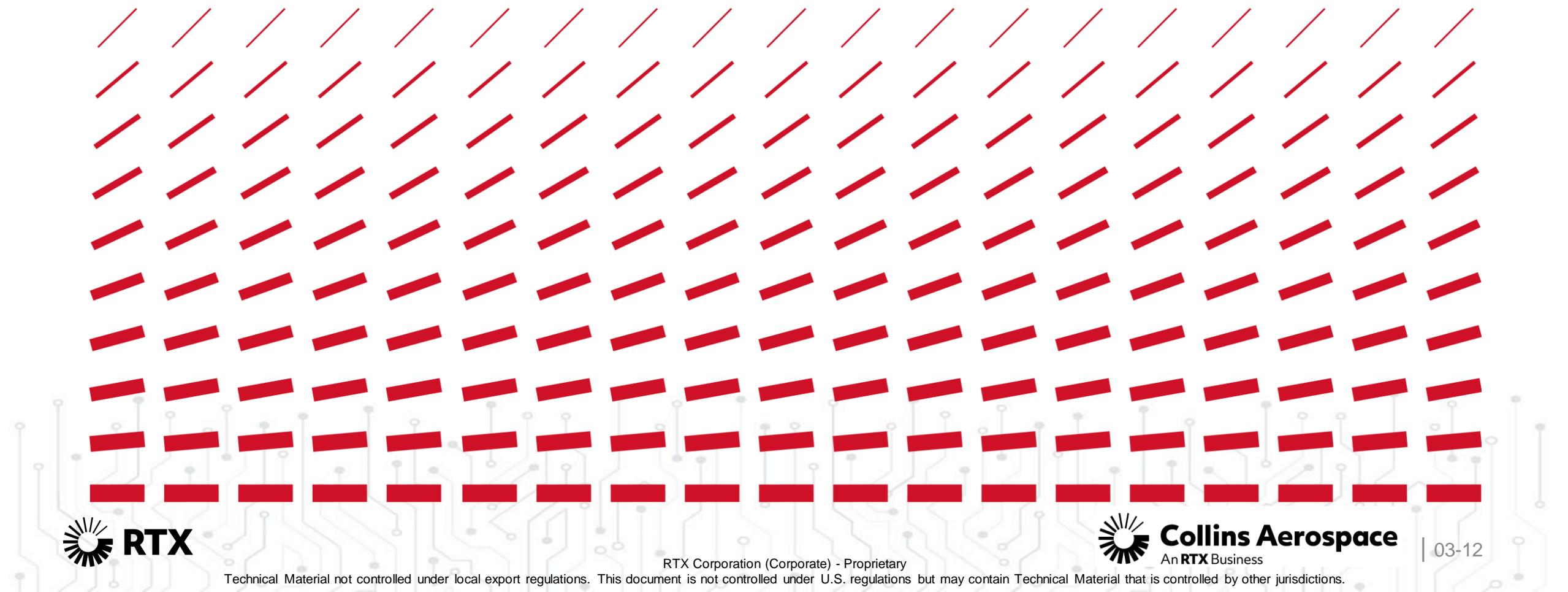


Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.



| 03-11

Thank you.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





Collins Aerospace
An RTX Business

© 2023 Raytheon Technologies Corporation
All rights reserved
RGEMS Tracking

© 2023 Collins Aerospace. | Collins Aerospace Proprietary. | This document does not include any export controlled technical data.

RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.

RTX TGE
Technology & Global
Engineering



TGE CYBERSEC

Embedded Systems Security

Module 04
Secure System Development Life Cycle
(SSDLC)

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India



Agenda

COLLINS PRODUCT CYBERSECURITY TRAINING



- Module 1: Cybersecurity General Overview
- Module 2: Aviation Product Cybersecurity Challenges
- Module 3: Collins Product Cybersecurity: Policy Flow down
- **Module 4: Collins Secure System Development Life Cycle**
- Module 5: Cybersecurity Certification and Standards Considerations
- Module 6: DO-326A Airworthiness Certification Process
- Module 7: Aviation Threat Example



Module 4: Secure System Development Life Cycle (SSDLC)



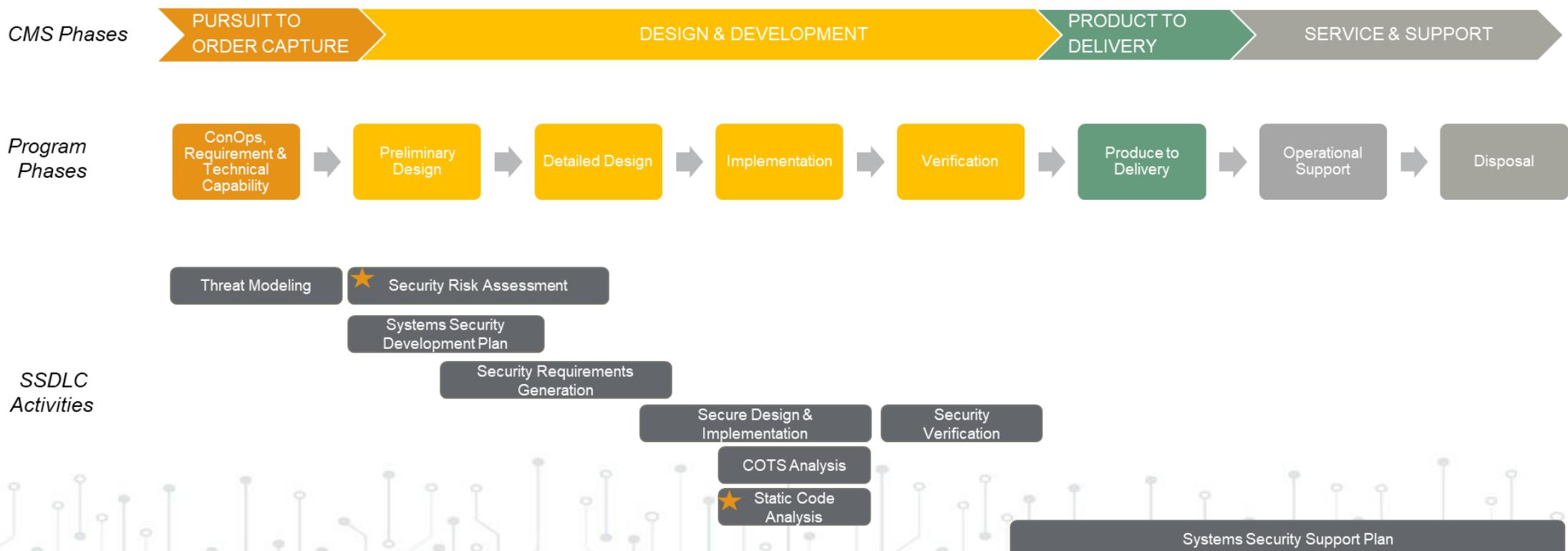
- Overview of the SSDLC
- Required SSDLC Activities
- Tailorable SSDLC Activities
- High Level Cybersecurity Assessment
- Security Development Plan
- Security Risk Assessment
- Cybersecurity Requirements
- Implementation
- 3rd Party Component Risk Management
- Cybersecurity Verification
- Operational Support & Disposal



Secure System Development Life Cycle (SSDLC)

COL-PCO-PRO-0002

Assure that SSDLC processes incorporate appropriate risk-based cybersecurity principles into the requirement, design, development and support of new or substantially modified Collins Aerospace System, Products and Unclassified Hosted Services.

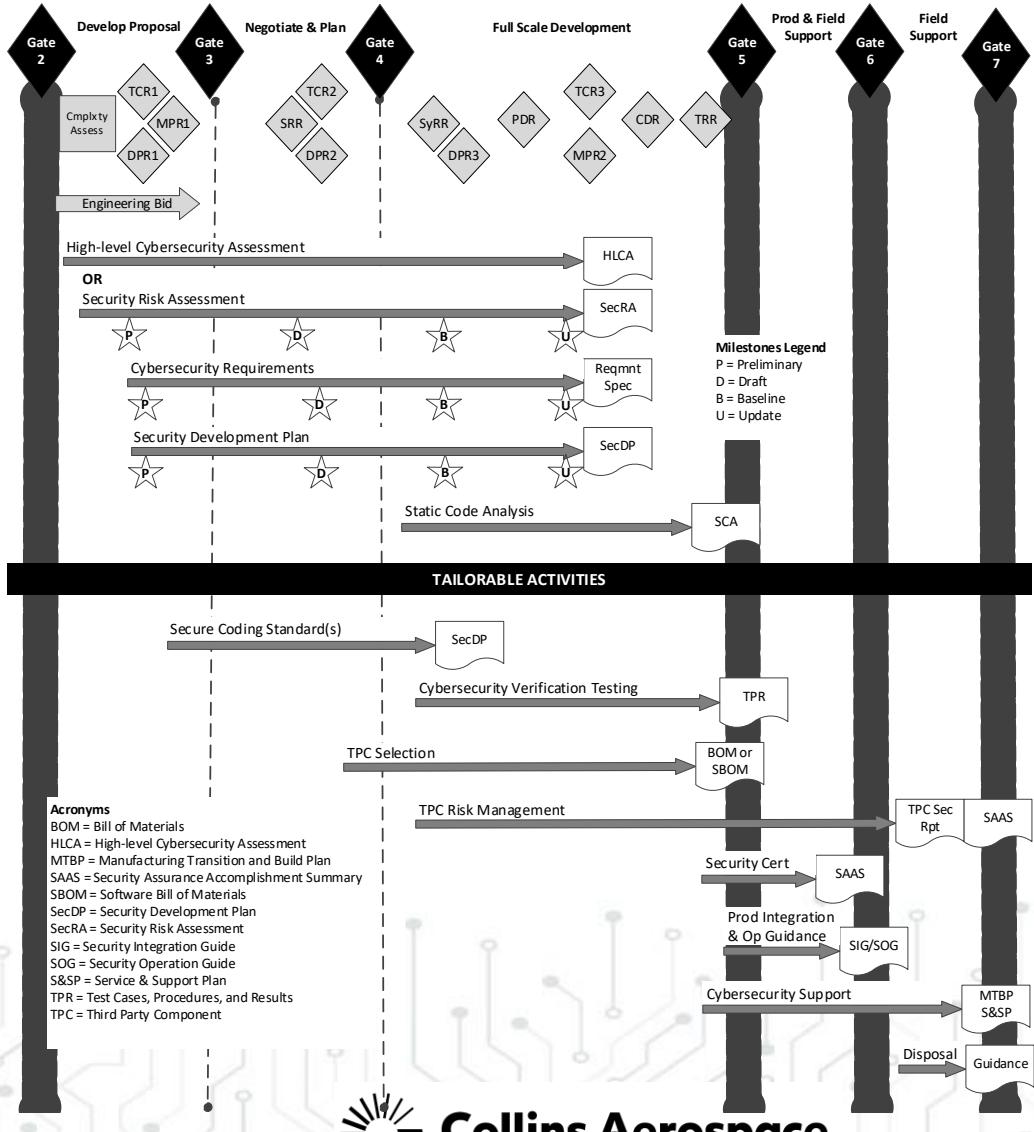


Secure System Development Life Cycle

AVI-PCO-PRO-0002 release update September 2023

- Applicable to projects that had not attained CMS Gate 4 approval as of December 2020
- Risk-based approach** to identify cybersecurity concerns
- Occur across the project life cycle
- Aligned to CMS Gates and D&D milestones
- Support compliance** to security frameworks (e.g., Risk Management Framework) and standards (e.g., DO-326A/ED-202A)
- Defined **required vs tailorable** Cybersecurity activities
- Designs address identified **cybersecurity vulnerabilities and risks**

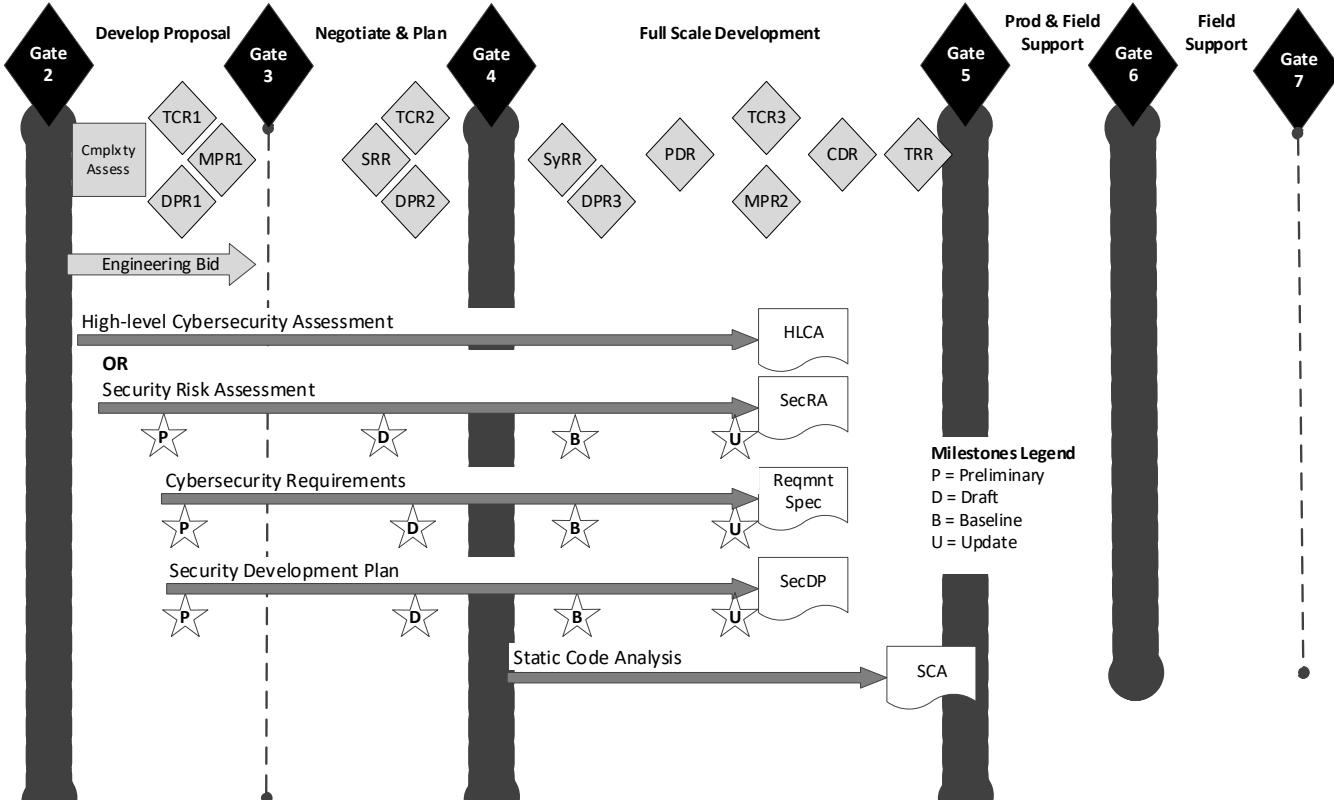
Be Proactive To Reduce Cost and Impact



Required SSDLC Activities



- Start with a **High-level Cybersecurity Assessment (HLCA)**
 - Should be completed by CMS Gate 3
- As applicable based on HLCA results:
 - Security Risk Assessment (SecRA)
 - Cybersecurity Requirements
 - Security Development Plan (SecDP)
 - Static Code Analysis (SCA)

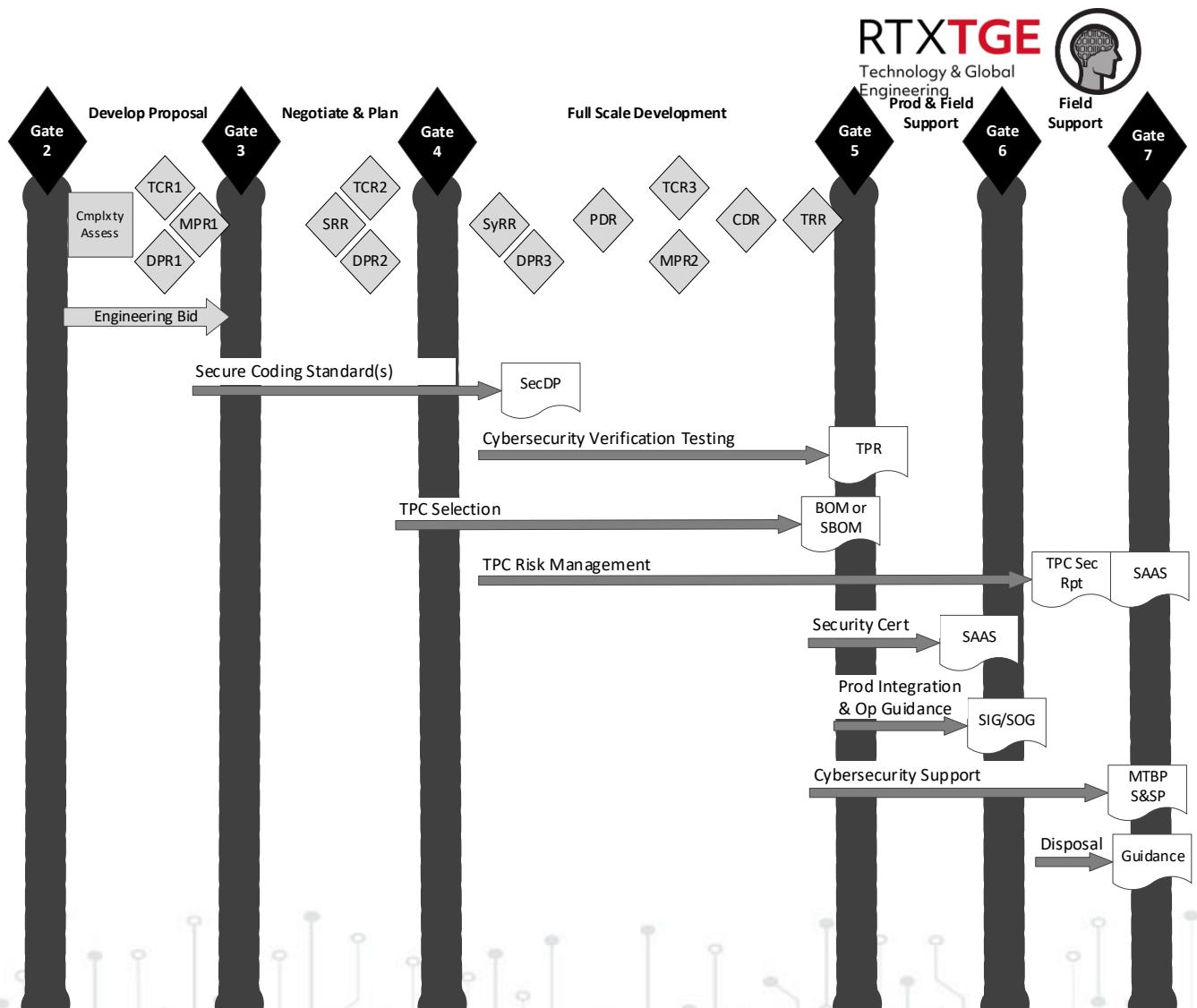


These Activities are Iterative and Mature Across the Development Life Cycle



Tailorable SSDLC activities

- Secure Coding Standard
 - Should be covered by SCA baseline
- Third Part Component Selection
- TPC Risk Management
- Product Security Verification Testing
- Security Certification
- Product Integration & Operational Guidance
- Cybersecurity Support
- Disposal



SSDLC Activities are Defined in the Security Development Plan when Applicable

High-level cybersecurity assessment



- COL-PCO-FRM-0007** : Responsible: TPM/ ENG Architect/SME; Support: Cyber SME

- Fill out the 11 questions form required **prior to gate 3**
- Send completed HLCA to **Cybersecurity SME** for review and recommendation
- High-level evaluation** to determine potential cybersecurity concerns:
 - Applicable law and regulations, security env, exposed interfaces to untrusted source and data, Third Part Component usage...
- Drive SSDLC applicability if any (activities to be included in the bid).
- Project scope changes require revisiting HLCA** to verify validity of previous answers

If the product is purely mechanical with no electronics components, then jump to question 11.				
Q #	Security Considerations	Guidance	Answer	Details Driving the Answer
1	If this a modification to an existing product, are any of the following true: - New interface(s) are being added - The operating environment is changing - New users of the product - There is a major safety change - Security reclassification is required - CIA requirements are changing	Per COL-PCO-POL-0001, the SSDLC only applies to new or substantially modified existing products. If any of these are yes, then a security risk assessment (at a minimum) must be done on the changes to the project. See the SSDLC for more detail. If this is a modification to an existing product, please answer the questions in line 5 - 15 in the scope of the modification.		
2	For the product/solution and/or any applicable data, are there any known or potential security assurances, security controls, standards, frameworks or security technical requirements that are stipulated by the customer, standard, laws, regulation, or other directive?	Examples: - FAA, EASA or DoD advisory circulars and regulations, DFARS, RMF, ISO-27000, NIST 800-53, FIPS, STIGS, ISO 15408 Common Criteria, Executive Order for Federal Systems - Airbus M1665.3 (Security Assurance Requirements for Suppliers) - RTCA DO-326A/DO-356A (Airworthiness Security Process Specification)		
3	Are there any applicable laws and/or regulations for transmitting, collecting, processing or storing data? (Consider how this will be used/installed.)	Examples: GDPR, CCPA, FISMA, U.S. state-based regulations, PII, HIPAA, PCI, export control, government classifications.		
4	Does the product/solution store sensitive data unencrypted across power cycles?	Examples: - Is PII or PCI data stored on the system? - Is there sensitive data (CUI/CDI) that needs to be protected?		
5	Will the product/solution (transported and installed) have any physical, logical, and/or functional interfaces, including removable media?	Consider: - Any physical data connection points available for use by the end user? (These could be via A429, A717, 1553, JTAG, maintenance, debug, ethernet ports or USB ports for mass storage devices or tablets, SD card, Compact Flash cards, CD/DVDs, SIM cards, or others.) - Any logical/functional interfaces available? (These could include: IP, TCP, UDP, HTTPS for user Interface; NTP for time-sync; Syslog [UDP] for logging; SSH for administrator remote access; DNS for IP / Host lookup.)		
6	Will the product/solution (transported and installed) have any wireless interfaces?	Consider: - Any physical data connection points available for use by the end user? (These could be via Wi-Fi, cellular, Bluetooth, SATCOM, HF, VHF, UHF, ADS-B, GPS/GNSS.) - Any logical/functional interfaces available? (These could include: IP, TCP, UDP, HTTPS for User Interface; NTP for time-sync; Syslog [UDP] for logging; SSH for administrator remote access; DNS for IP / Host lookup.)		
7	Will the product/solution be connected to the internet?	This includes any type of connection that may cause the product to be vulnerable (remote, cloud, internet, ground system). Consider the following conditions: - Service Support - Built-in test interfaces used in service centers and production - Customer installation		
8	Will the product/solution need to support multiple levels of trust either internally or across interfaces?	Consider: Will the product/solution store or manage unauthenticated/untrusted data? Will any data come from an untrusted domain or an open network?		
9	Will the product/solution be serviceable/reprogrammable in the field or from remote access?	Consider: Does this have the ability to reload a full or a partial version of the system directly on the operating field? This would include upgrades made at customer sites, MROs, service centers, etc. Does this have the ability to reload without physically interacting or connecting to the product?		
10	Will the product/solution include any human user interfaces?	Would a human be able to interact with the system and cause harm? - This could include any Human Machine Interface (HMI), where a user logs in or accesses the device anonymously.		
11	Will the product/solution use any third-party developed software or hardware?	Examples: - Custom third-party developed application or tool - Any Third Party Components (TPC) closed-source or open-source libraries, applications - FPGA intellectual property blocks For purely mechanical product, consider: - Product that uses additive manufacturing techniques - Product integrity cannot be verified by standard screening procedures		



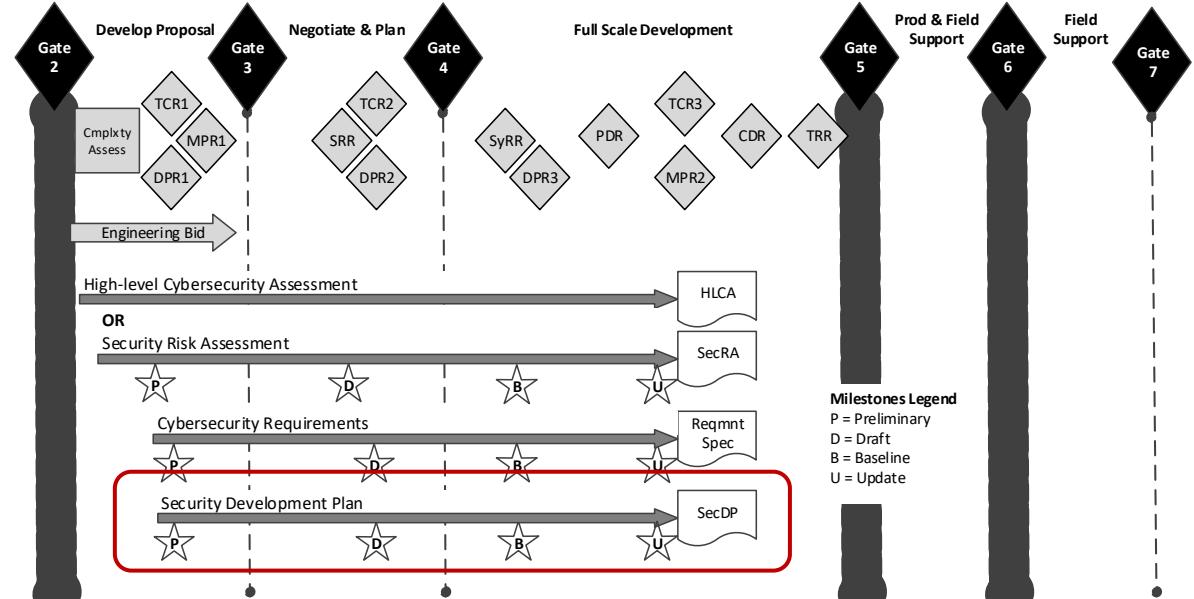
SECURITY DEVELOPMENT PLAN

Responsible: Cyber SME; Support: TPM/ Eng Architect/SME



- **Describe** the security activities:

- **What:** Identified SSDLC activities for the program,
- **Who:** Security organization roles and responsibilities,
- **Where:** Development/verification environments,
- **How:**
 - Guidelines, Process, standards, law, regulations...
 - Applicable security policies for the environment, design repositories, configuration management...
 - Plan to protect data exchanged with suppliers and customers.
- Can be a standalone document or included in another project planning document



COL-PCO-FRM-0006, Template for Systems Security Development Plan



Security Risk Assessment

- **Product Asset(s)**

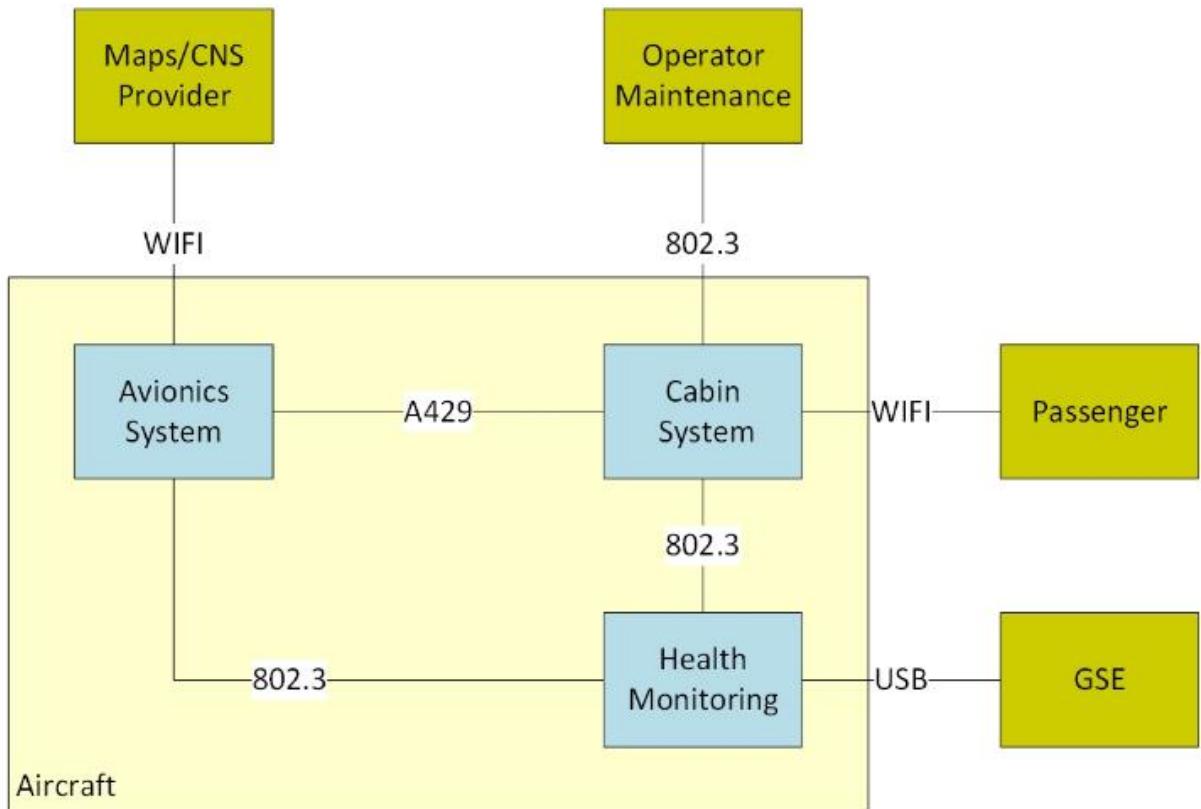
- Anything from processes, components, devices, Line Replaceable Units (LRU), features, functions (data) that pertain to the Product under development requiring cybersecurity protection

- **Security Environment**

- Identifies the operational and maintenance settings where the Product will exist, including external interfaces (e.g., systems, people, entities), and expected inputs/outputs

- **Security Perimeter**

- Establishes the boundary between the Security Environment and the Product under development

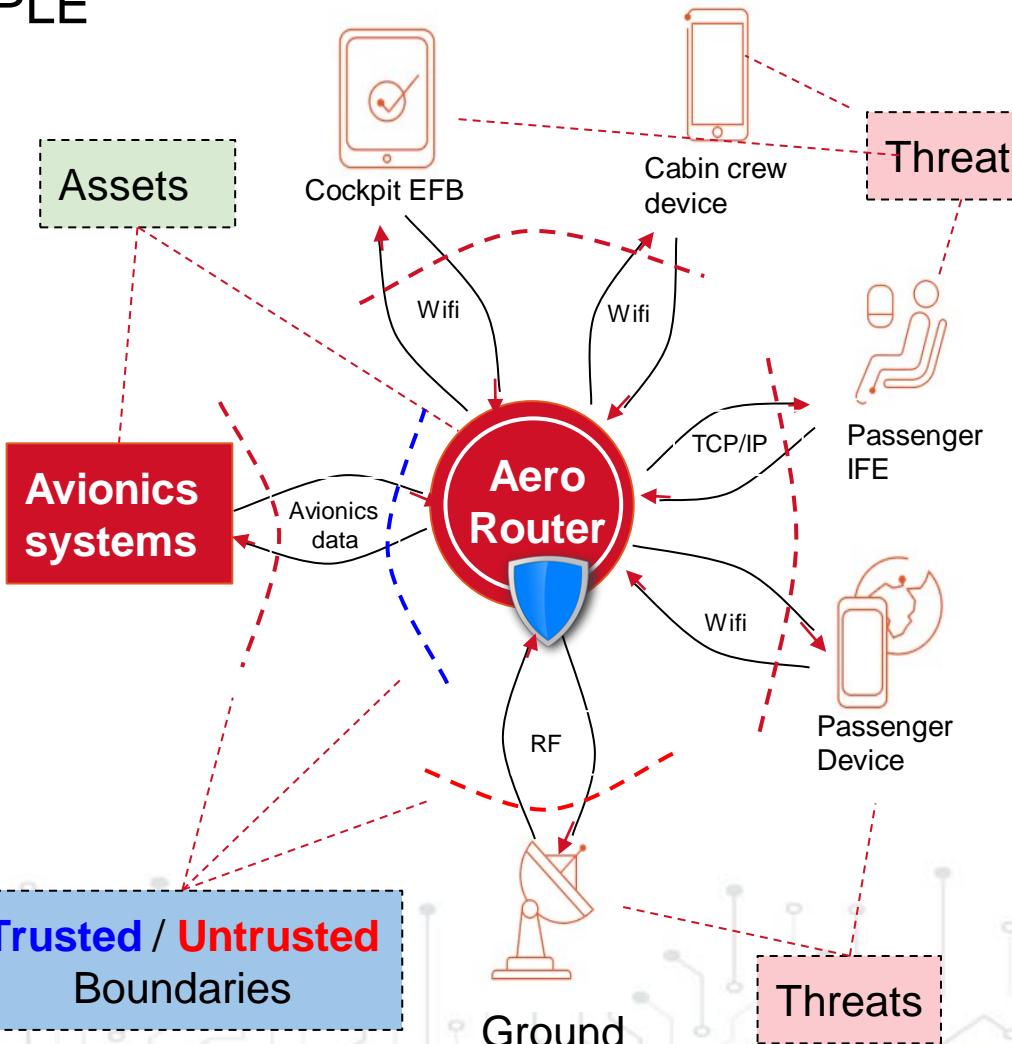


Security Risk Assessment - Preliminary



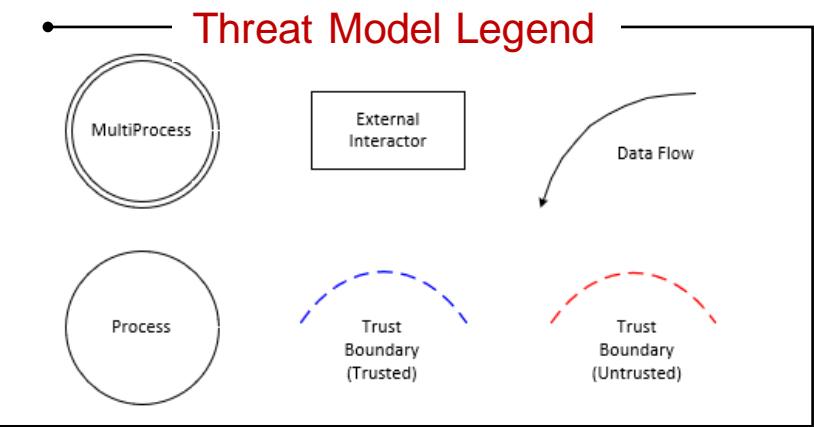
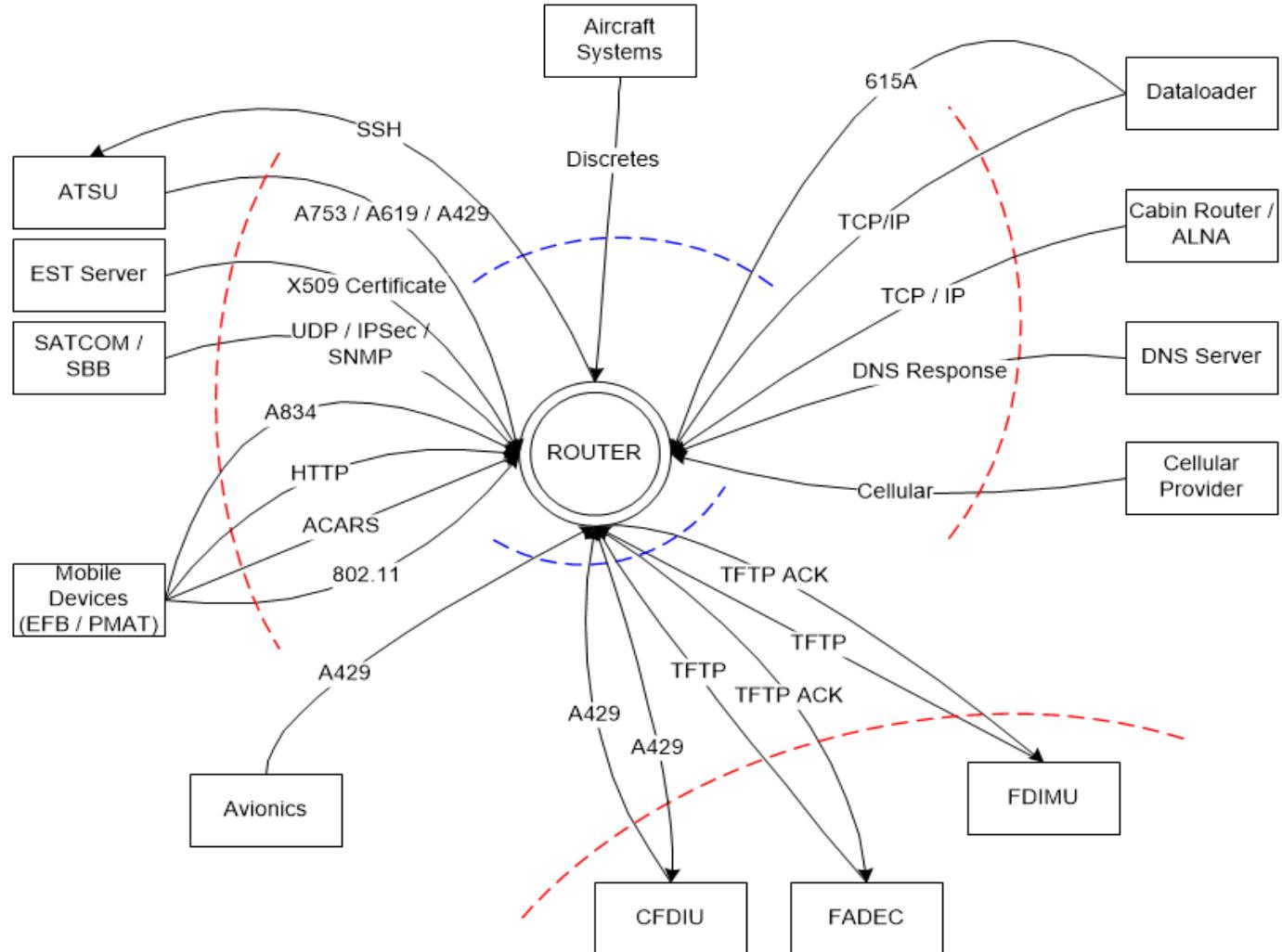
SECURITY SCOPE: THREAT MODELING EXAMPLE

- 1. Define Assets**
 - What needs protection?
- 2. Define Security Principles to Protect**
 - Confidentiality, Integrity, Availability
- 3. Define Functional Data Flows**
- 4. Define Trust Boundary**
 - Anything outside the trust boundary can be a threat source
- 5. Identify the threats, threats scenarios and assess the risks**



Security Risk Assessment

THREAT MODEL EXAMPLE



Security Risk Assessment

THREAT METHODOLOGY EXAMPLE

STRIDE Model: popular legacy Threat Modeling Method available (Microsoft)

- **Spoofing:** pretend to be something or someone you are not
- **Tampering:** manipulate/change information you are not supposed to
- **Repudiation:** ability to claim you didn't do certain actions (no matter if you did or not)
- **Information Disclosure:** leak/expose information
- **Denial of Service:** prevent the system to provide a service
- **Elevation of Privilege:** gain rights to do things you are not supposed to

Threat	Property	Threat Definition	Mitigations examples
Spoofing	Authentication	Impersonating something or someone else.	To authenticate principals: Basic/Digest/Windows Authentication PKI systems such as SSL/TLS and certificates IPSec To authenticate code or data: Digital signatures MAC & Hashes
Tampering	Integrity	Modifying data or code	ACLs Digital signatures Message Authentication Codes
Repudiation	Non-repudiation	Claiming to have not performed an action.	Strong Authentication Secure logging and auditing Digital Signatures Secure time stamps
Information Disclosure	Confidentiality	Exposing information to someone not authorized to see it	Encryption ACLs
Denial of Service	Availability	Deny or degrade service to users	ACLs Filtering Quotas High availability designs
Elevation of Privilege	Authorization	Gain capabilities without proper authorization	Group or role membership Validate Permissions Input validation



Security Risk Assessment

THREAT EXAMPLE



1. List all threats derived from crossing untrusted interfaces illuminated by threat model / data flow diagram.
2. Identify the impacted security properties (C-I-A).

Table 1-1: Router Targeted Threats

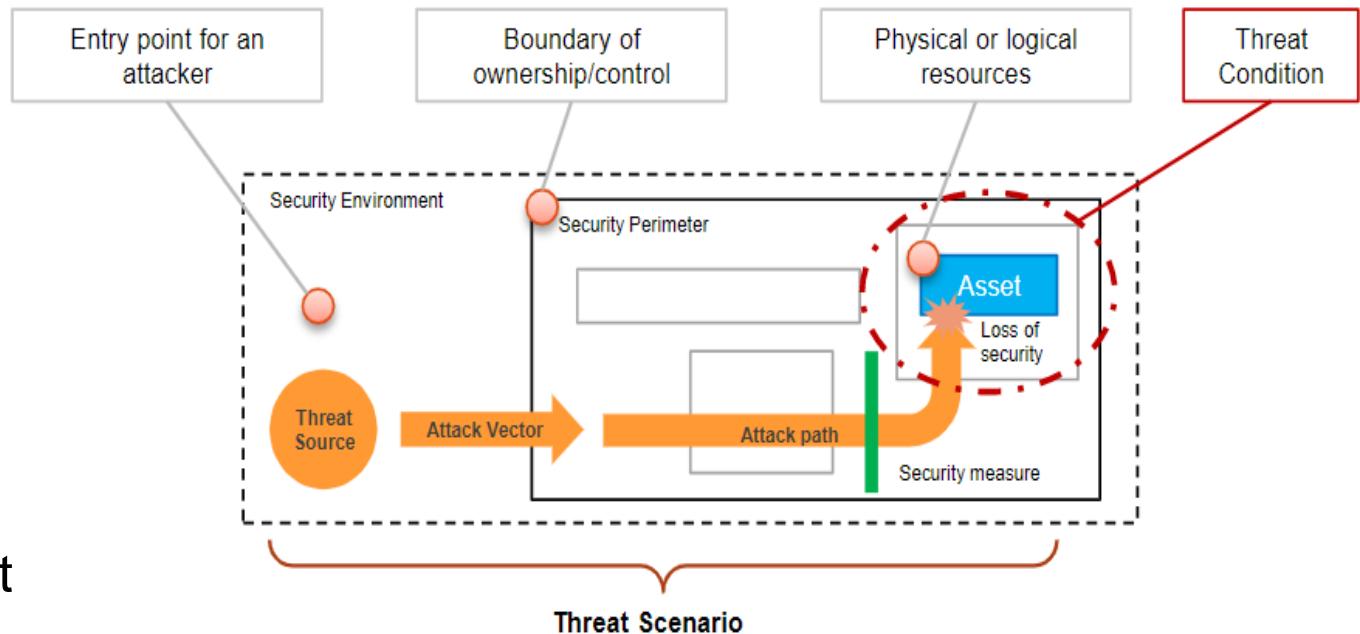
Information Disclosure	Threat	Description	Target (C,I,A)
Tampering	T1	Attack the confidentiality of the router via wireless protocols from an unauthorized device to gain access to the router.	Router (C)
	T2	Attack the integrity of the router through the <u>ACARS</u> protocol from the ACARS network with the intention of executing unexpected commands on the Router.	Router (I)



Security Risk Assessment

THREAT SCENARIO

- **Threat Source:** external system and actor utilizing an attack vector
- **Attack Path:** the interfaces exploited, the actions performed, security measures by-passed, vulnerabilities that allow the attack to succeed
- **Security Measures:** technologies or solutions meant to mitigate or prevent an attack
- **Threat Condition:** end state (safety hazard) triggered by threat scenario



Threat Scenarios Describe How to Reach a Threat Condition

Security Risk Assessment

THREAT SCENARIO AND SCORE (THREAT LEVEL RATE)

1. For each threat previously identified, define the threat scenario

2. For each threat scenario, complete the vulnerability scorecard below to determine the level of risk based on exploitation level



VULNERABILITY	Title:	Compromise the Integrity of the SSR through Malformed ACARS		Ref:	V2		
Description:	An attacker uses a weakness in the handling of ACARS messages by the SSR to execute unexpected commands on the SSR.						
Damage / Consequences:	Remotely execute commands on the SSR to view or modify data stored on the SSR.						
Exploitation scenario(s):	The attacker creates an ACARS message with proper formatting for it to be received by the ATSU on the aircraft. The message contains malicious code which takes advantage of a vulnerability in the SSR. The ATSU forwards this message to the SSR for routing to the EFB. Improper handling of the message leads to remote code execution on the SSR allowing unintended commands to be carried out.						
Mitigation plan:	The SSR implements a MAC policy through SELinux which limits the ability of applications hosted on the SSR to only access predefined objects (files, ports, etc.) on the system which are required by the function. If new unexpected commands are attempted, they will be denied by the MAC policy.						
Impact	Major						
Vulnerability Exploitation	Expertise Level: 8	Knowledge Level: 7	Equipment Level: 7	Elapsed Time Level: 7			
	Exploitation Level: 29 (Very Low)						
Vulnerability Coverage:	Mitigated						
Applicable Threats:	T2						



Security Risk Assessment

THREAT SCENARIO SCORING



Expertise Level		
Attacker Expertise	Description	Level
Layman	No particular expertise of information security / information networks.	0
Proficient	General knowledge of information security / information networks and of the security behavior of the target of the attack. High school or/and non-College Security Training (with direct work experience) or equivalency in information security.	3
Expert	High and specific knowledge of information security / information networks. High skills on the underlying algorithms, protocols, hardware, structures, security behavior. College Degree (with direct work experience) or equivalency in information security.	6
Multiple Experts	Multiple experts: The attacker is highly skilled in several fields of expertise.	8

Knowledge Level		
Knowledge of Target	Description	Level
Public Information	Known published information allows exploitation of the identified potential vulnerability on the target (e.g. available on internet).	0
Restricted Information	Restricted information allows identification of potential vulnerability on the target (e.g. knowledge that is controlled within the developer organization under a non-disclosure agreement). Examples: Development plan describing the list of COTS used.	3
Sensitive Information	Sensitive information contributes to the vulnerability exploitation on the target. The information is adequately protected against unauthorized disclosure (e.g. knowledge that is shared between discrete teams within developer organization, access to which is constrained only to members of the specified teams)	7
Critical Information	Critical information allows exploitation to directly identify potential vulnerabilities on the target. The information is adequately protected against unauthorized disclosure (e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need to know basis). Examples: Private key files.	11

Equipment Level		
Equipment	Description	Level
Standard Equipment	Equipment (hardware, software) is readily available to the attacker, either for the identification of vulnerability or for an attack. Such equipment is easy to buy or to obtain in non-specialized shops, through internet downloads: protocol analyzer or simple attack scripts.	0
Specialized Equipment	Equipment is not readily available but can be acquired without undue effort. For instance, purchase of moderate amounts of equipment (e.g. power analysis tools, use of hundreds of PCs linked across the Internet would fall into this category), or development of more extensive attack scripts or programs.	4
Custom Equipment	Equipment is not readily available in non-specialized shops or internet. For instance, hardware or software may be specially produced or developed for the attack (e.g. very sophisticated software). The equipment may be very expensive.	7
Multiple Custom Equipment	Several custom equipment types are necessary. This could include the development of customer hardware or software.	9

Elapsed Time Level		
Elapsed Time	Description	Level
Very Easy	Very easy means that the Elapsed time is inferior to one day.	0
Easy	Easy means that the Elapsed time is superior to one day and inferior to one week.	1
Moderate	Moderate means that the Elapsed time is superior to one week and inferior to one month.	4
Difficult	Difficult means that the Elapsed time is superior to one month.	7



Security Risk Assessment

THREAT SCENARIO AND SCORE (THREAT LEVEL RATE)

- For each threat previously identified, define the threat scenario
- For each threat scenario, complete the vulnerability scorecard below to determine the level of risk based on exploitation level

VULNERABILITY	Title:	Compromise the Integrity of the SSR through Malformed ACARS	Ref:	V2
Description:	An attacker uses a weakness in the handling of ACARS messages by the SSR to execute unexpected commands on the SSR.			
Damage / Consequences:	Remotely execute commands on the SSR to view or modify data stored on the SSR.			
Exploitation scenario(s):	The attacker creates an ACARS message with proper formatting for it to be received by the ATSU on the aircraft. The message contains malicious code which takes advantage of a vulnerability in the SSR. The ATSU forwards this message to the SSR for routing to the EFB. Improper handling of the message leads to remote code execution on the SSR allowing unintended commands to be carried out.			
Mitigation plan:	The SSR implements a MAC policy through SELinux which limits the ability of applications hosted on the SSR to only access predefined objects (files, ports, etc.) on the system which are required by the function. If new unexpected commands are attempted, they will be denied by the MAC policy.			
Impact	Major			
Vulnerability Exploitation	Expertise Level: 8	Knowledge Level: 7	Equipment Level: 7	Elapsed Time Level: 7
	Exploitation Level: 29 (Very Low)			
Vulnerability Coverage:	Mitigated			
Applicable Threats:	T2			

Exploitation Level		
Sum of Values	Level of Threat	Description
29-35	Very Low	Very unlikely that the vulnerability may be exploited
22-28	Low	Unlikely that the vulnerability may be exploited
15-21	Moderate	Likely that the vulnerability may be exploited
8-14	High	Highly likely that the vulnerability may be exploited in the future.
0-7	Very High	No doubt that the vulnerability may be exploited in the short term

Severity of the Threat Condition Effect					
Level of Threat	No Safety Effect	Minor	Major	Hazardous	Catastrophic
Very High	Acceptable	Acceptable	Unacceptable	Unacceptable	Unacceptable
High	Acceptable	Acceptable	Unacceptable	Unacceptable	Unacceptable
Moderate	Acceptable	Acceptable	Acceptable	Unacceptable	Unacceptable
Low	Acceptable	Acceptable	Acceptable	Acceptable	Unacceptable
Very Low	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable

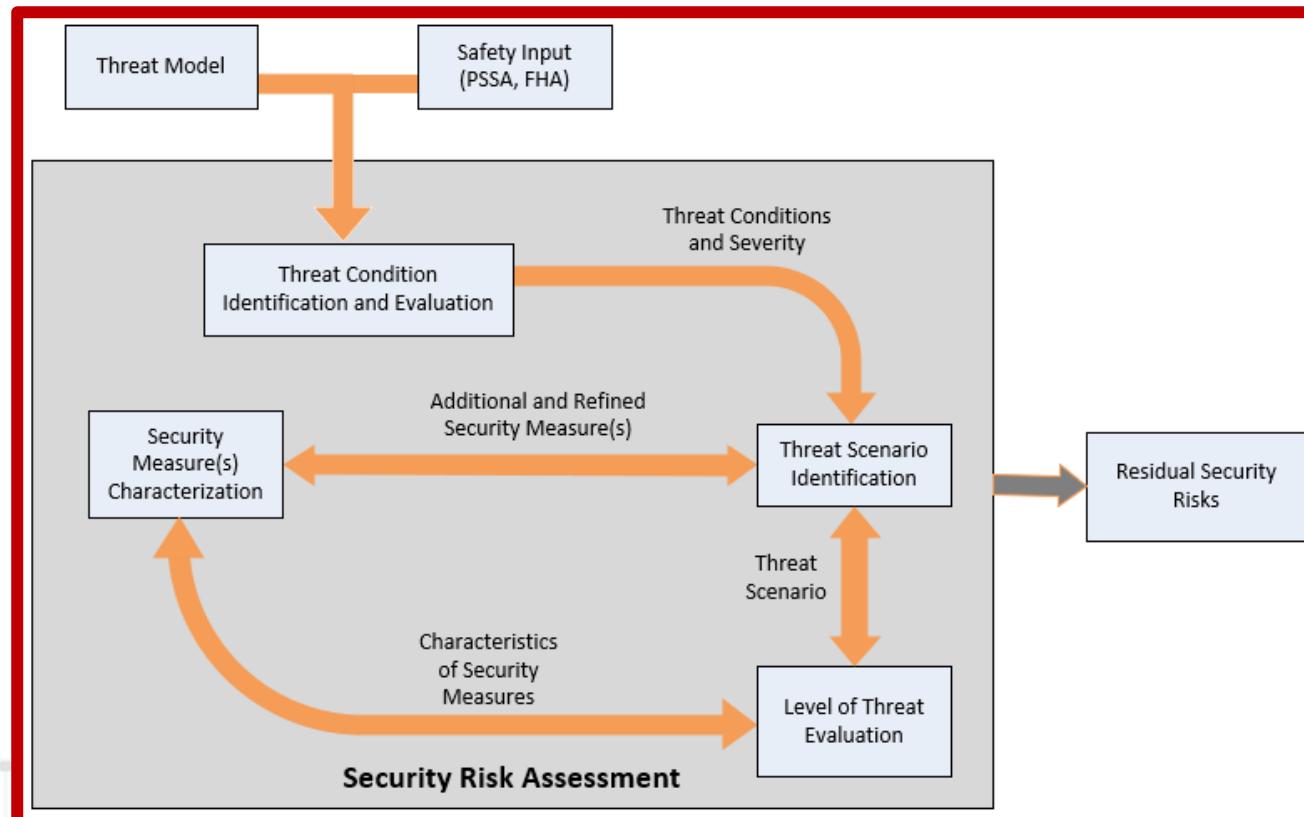


Security Risk Assessment

Responsible: Cyber SME; Support: System / Software Architect/SME



- Determine Threat Conditions by correlating **Threats to Failure Conditions** from Safety FHA/SSA
- Identify **Threat Scenarios** (what steps need to occur to reach the Threat Condition)
- Identify existing **Security Measures** to reduce unacceptable risks (existing architecture and requirements)
- Define/design new **Security Measures** to reduce unacceptable risks (updated architecture and new requirements)
- Process is complete when **Threats** are mitigated to an acceptable level

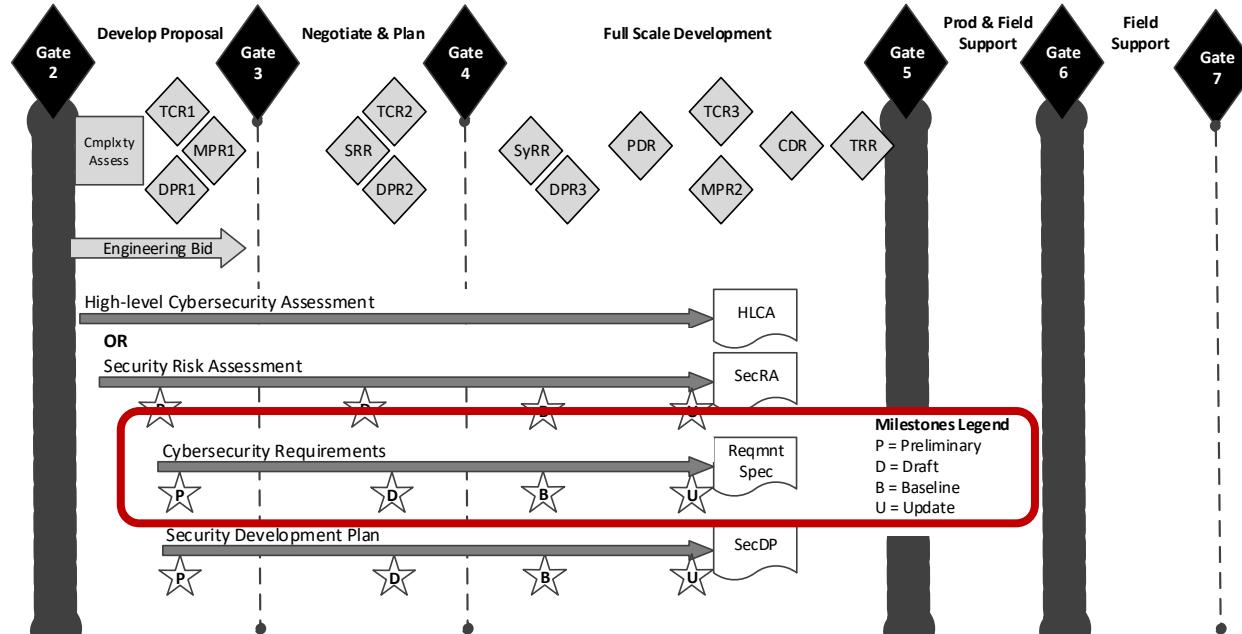


Security Requirements

Responsible and/or support: Cyber SME/ System and Software Architect/SME



- Security requirements come from many locations:**
 - Directly **from the Customer**
 - Indirectly from standards, laws, regulations.
 - Derived **from the Security Risk Assessment**
 - Mitigating unacceptable risks
- Flow into the normal requirements process**
 - Identified (tagged) as security requirements
- Drive the security architecture and design in way that security countermeasures cannot be tampered or bypassed



Security Requirements Are Not Always Explicit



Security Contractual Obligations



Security requirements related to Product are not only on the product itself **but also on the different environments in which the product is handled** during development and in-service !

AIRBUS Airbus Company Directive	AIRBUS Airbus Company	AIRBUS Customer Services Security Assurance Technical Report	AIRBUS Method	
Specific Requirements for Security	Requirements	Customer Service Technical Report	Security Assurance Supplier Support	CAGE Code 81205
PURPOSE/SCOPE: This directive is a complementary document for Suppliers* (A1015.0), to which it is minimum security requirements for areas not limited to managed IT service tasking, IT outsourcing or cloud provider support services for OT/IoT) relevant to Airbus Procurement shall apply the Air Directive, without any deviation, to all Airbus entities, sites, locations including controlling interests. This Directive is relevant for the contracted work & information/connectivity or applicability.	PURPOSE/SCOPE: This Directive defines the purpose of the security of Airbus business and facilities that are located either on or off AirBus premises. It covers the security of Customer Service products and services. It also specifies the requirements for the security of Customer Service products and services. It also specifies the requirements for the security of Customer Service products and services.	REFERENCE: ACN APPLICABILITY ATA APPLICABILITY CUSTOMER CONFIDENTIALITY DOCUMENT LEVEL	PURPOSE: The main aim of this document is to define the requirements for the supplier development process (SDP) (including software) requiring Security Assurance Level (SAL) concept and the associated levels of assurance assigned to each level, in order to develop the required assurance levels.	System Cybersecurity Compliance Methodology and Requirements
Document Owner: Name: KNUEPPEL, Dietrich Function: Directive Owner	Document Owner: Name: KNUEPPEL, Dietrich Function: Directive Owner	NAME AUTHOR(S): LOCATELLI, APPROVAL: DESMARTIS Marie AUTHORIZATION: WOHLERS, V.	KEYWORDS RELATED DOCUMENTS NAME Document Owner: Name: PIEBOIS Laurent Function: Aircraft Security Governance - EID	DOCUMENT NUMBER: D6-87601 RELEASE/REVISION: New RELEASE DATE: September 14, 2023 For Document Release Use
<small>© Airbus SAS 2021. All rights reserved. Confidential and proprietary document. To be used only by authorized personnel and not distributed outside of the company.</small>	<small>© Airbus SAS 2021. All rights reserved. Confidential and proprietary document. To be used only by authorized personnel and not distributed outside of the company.</small>	<small>© Airbus SAS 2021. All rights reserved. Confidential and proprietary document. To be used only by authorized personnel and not distributed outside of the company.</small>	<small>Copyright © 2023 Boeing. All rights reserved. BOEING PROPRIETARY - ECCN: 7E994</small>	<small>BCA Cybersecurity Certification (66-CB-EC20)</small>



Example of Security Measures

SECURITY MEASURES CAN BE TECHNICAL OR OPERATIONAL



Isolation/Separation

- Secure Gateway/Diode
- Secure Ethernet Filter (FPGA)
- SELinux MAC policy
- Secure hypervisor (containerization)
- VLANs

Platform Hardening

- Compiler/Linker Flags for Hosted Applications
 - ASLR and stack protection for COTS libraries
- Isolation of Kernel and User Space Memory
- Memory Execution Protection
- Disabling Unused Kernel Modules
- Streamline OS Footprint
- TCP/IP Network Stack Hardening

Operational Guidance

- Password Management
- Certificate Management

Filtering / Policing

- Network Firewall
- Application Proxy
- Intrusion Prevention System

Authentication/Authorization

- Digital Certificates
- Access Control List (ACL)
- Multi-Factor Authentication

Integrity Checker

- Secure Boot / Trusted Boot
- Security Integrity Checker
- Digitally Signing Load sets

Secure Communication

- Virtual Private Network (IPSEC, TLS, DTLS)
- HTTPS

Need to Consider Performance/Operational Impact when Implementing Security Measures



Implementation

SECURE CODE GUIDELINES AND SECURE DEVELOPMENT

Responsible: Engineer Software or Hardware Architect/SME; Support: Cyber SME

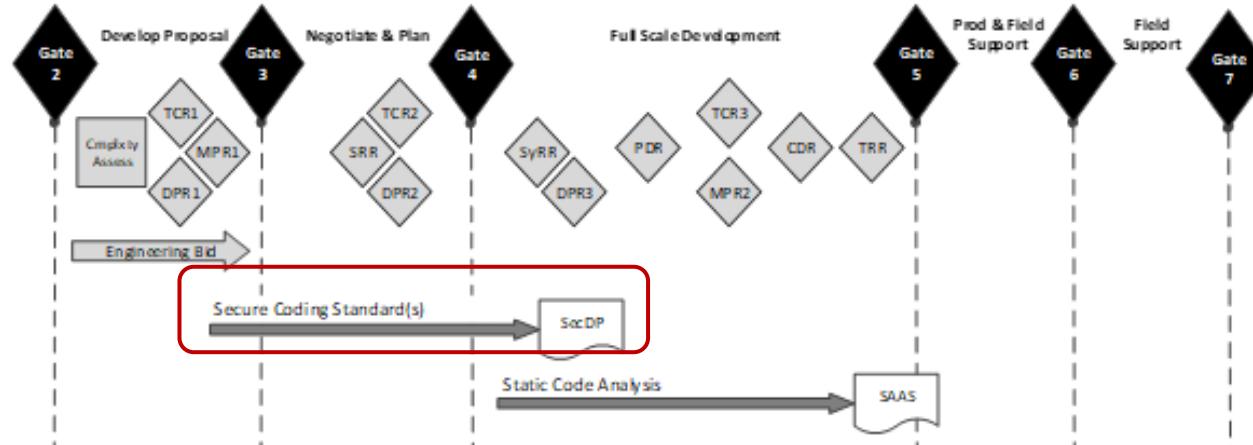


- **Manage Secure Coding guidance***

- Define the Secure coding rules for the program, should be covered by SCA baseline
- Train software engineer

- **Apply secure development principles:**

- Input validation: especially from untrusted data sources
- Buffer handling / Memory management
- Do not use weak functions
- Principle of least-privilege
- Error / Exception management is paramount
- Keep it simple design



Helpful material:

- Secure Coding Guidance Shared Practices on [the CELL](#)
- Secure Coding C & CPP Standards : [CELL](#)

Manage Secure Development and Code Analysis Earlier in the Program



Implementation

SECURE CODE GUIDELINES AND SECURE DEVELOPMENT



CWE: Common Weakness Enumeration

CWE - 2023 CWE Top 25 Most Dangerous Software

2023 CWE Top 25



Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2022
1	CWE-787	Out-of-bounds Write	63.72	70	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.54	4	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	34.27	6	0
4	CWE-416	Use After Free	16.71	44	+3
5	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	15.65	23	+1
6	CWE-20	Improper Input Validation	15.50	35	-2

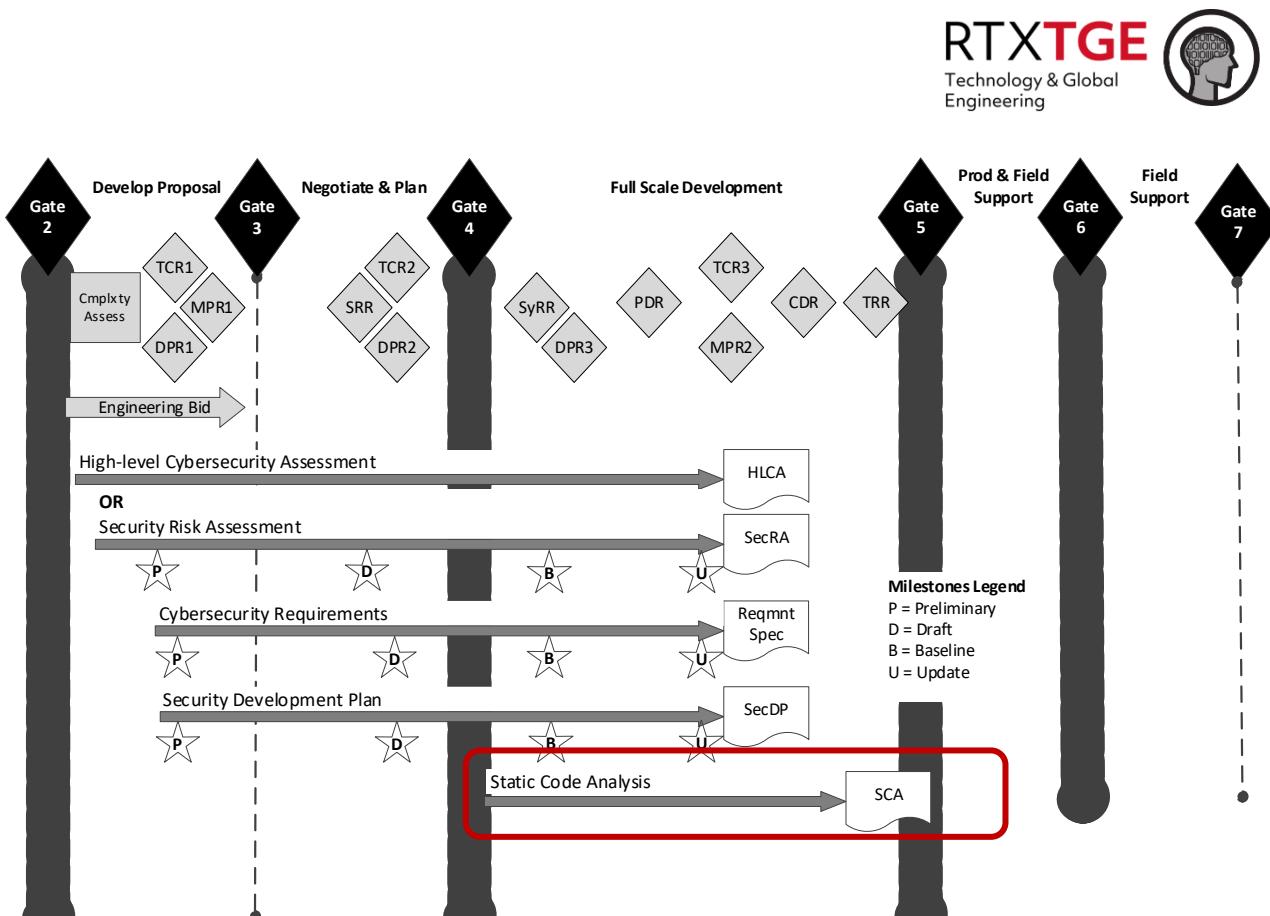
CW-119 Buffer overflow example



Implementation

STATIC CODE ANALYSIS (SCA)

- **SCA** enables developers to find and fix software defects by scanning the structure and composition of software against known vulnerabilities.
- **SCA tool shall be utilized** for the review of Cybersecurity aspects of **Collins-developed Products** unless they are unavailable.
- **When SCA tool is not available** for use due to technology restrictions or legacy program plans, then **code peer reviews based on Secure Coding standards shall be performed**.
- The Cybersecurity SME, shall evaluate all findings leave as is for risk evaluation.
- SCA is a part of Static Application Security Testing (SAST)



Fixing Defects Early in Development Reduces Overall Cost



Implementation

STATIC CODE ANALYSIS & MANUAL CODE REVIEWS



Static Code Analysis allows to:

- Continuously inspect code in the DevSecOps pipeline
- Detect coding standard violations
- Identify defects which could result in vulnerabilities
- Identify code quality issues
- Assist to fix unacceptable defects
- Track and justify outstanding defect
- Manage defect reporting

Manual Code Reviews

- Supplements automatic code analysis
- Ensure intent of security measure is met
- Check the implementation of security related security measure is correct

Goal: Zero Code Defects Introduced During Implementation



Implementation

STATIC CODE ANALYSIS

- Coverity SCA tool is currently one of our corporate tool
- Can analyzes C, C++, C#, Java, JavaScript, Objective-C, PHP, and Python source code to detect potential quality defects and a variety of security vulnerabilities.
- Provides extensive coverage for MISRA-C and CERT C, coding standards
- Coverity comes with a suite of security checkers that find many coding defects that can lead to security defects:
 - improper stack or heap access,
 - integer overflows, buffer overflows,
 - race conditions,
 - UNIX- and Windows-specific bugs,
 - insecure coding practices,
 - improper management of user-controllable strings.
- These defects can lead to memory corruption, privilege escalation, unauthorized reading of memory or files, process or system crashes, and denial of service.



Coverity Demo

Helpful material:

- SCA Endorsed Practices on [the CELL](#)
- Coverity SCA [getting started](#)
- [Coverity as a Service](#) managed by DPLC
- CATU Training: [SCA intro](#), [Coverity intro](#)



Implementation

STATIC CODE ANALYSIS

- Coverity will **find a lot of defects** (quality + security) but **not all of them!**
- Main sources of concern found in some programs during development:
 - Failure of deny-by-default logic
 - Buffer management / Pointer dereference / memory leaks
 - Format strings
 - Denial of Service conditions : infinite loops and Regular Expression evaluation
 - Log injection: unsanitized data reaching log content
 - File path injection / Path traversal
 - Command injection: unsanitized data reaching system()/popen()
 - XML injection / SQL injection
 - Bad ACL implementations, hardcoded password...



Fixing Bugs During Development Phase Is Cheaper Than During Operational Phase

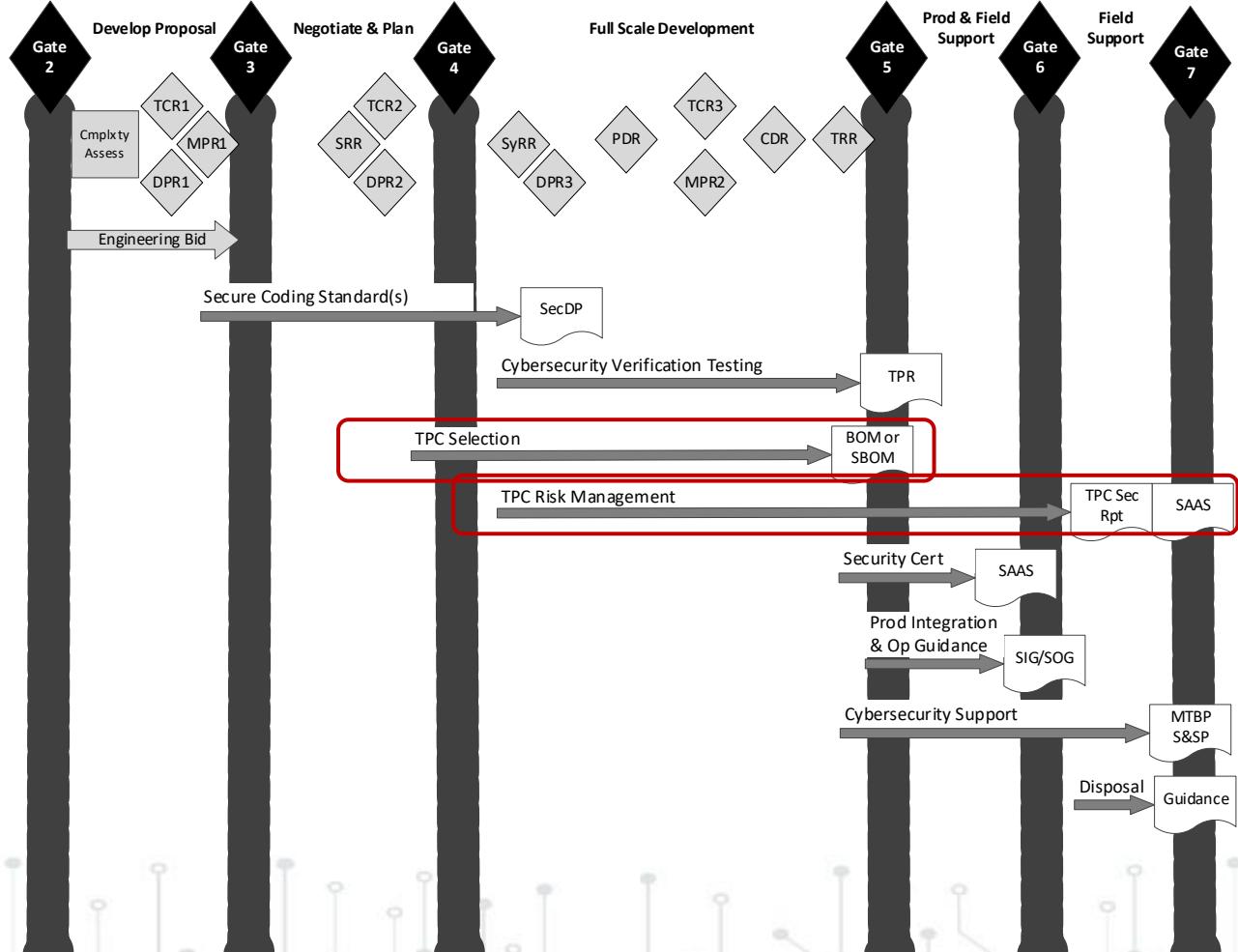


Third Party Component Risk MGT

Responsible: Eng Software SME (Selection); Cyber SME (vuln assessment)



- Complete TPC selection (COTS/FOSS)**
 - Assess the risk to use this TPC (TPC & supplier pedigree, security support, known vulnerabilities...)
 - Define the SBOM
- TPC Vulnerability Assessments**
 - Identify TPC vulnerabilities and the potential product impact
 - Fix unacceptable vulnerabilities
 - [COL-PCO-GUI-0002: Third Party Component Management Guidance](#)
 - [Management of Free and Open-Source Software Activities](#)



TPC Security Risk Management Needs To Performed To Limit Program Risk



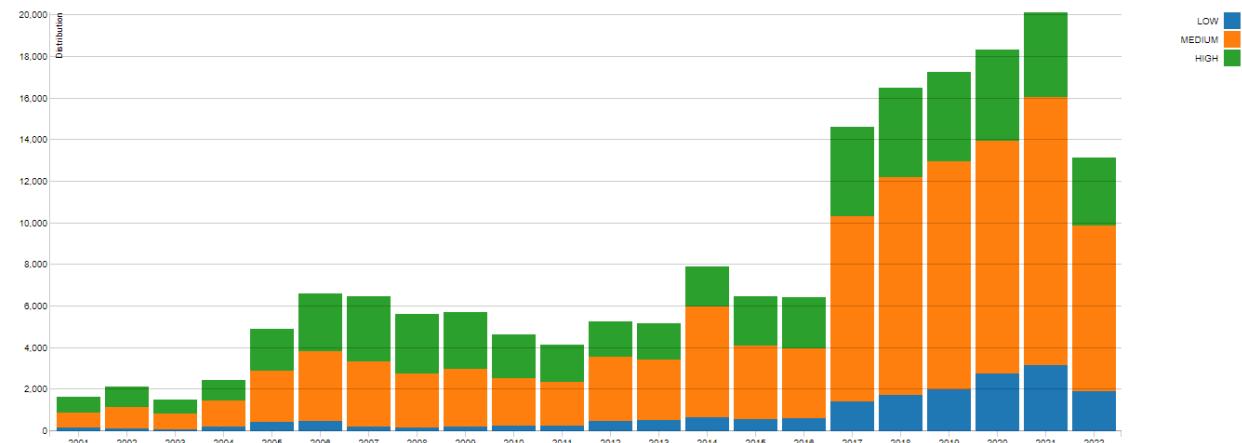
Third Party Component Risk MGT

TPVULNERABILITY MANAGEMENT CONTEXT

- The 2023 Synopsis OSSRA report's [[link](#)]:
 - 97% of Aerospace, Aviation, Auto... scanned Code based contain Open-Source Software (OSS)
 - 60% of scanned Code based contain vulnerabilities
- Significant increase in **CVEs** published since 2016*
- Embedded equipment lifecycle is different than IT
- Supply chain complexity makes TPC vulnerability management a challenge
- Time to deploy a security update is longer than w/ IT
 - We can't push an update out overnight



Aerospace, Aviation, Auto,
Transportation, Logistics



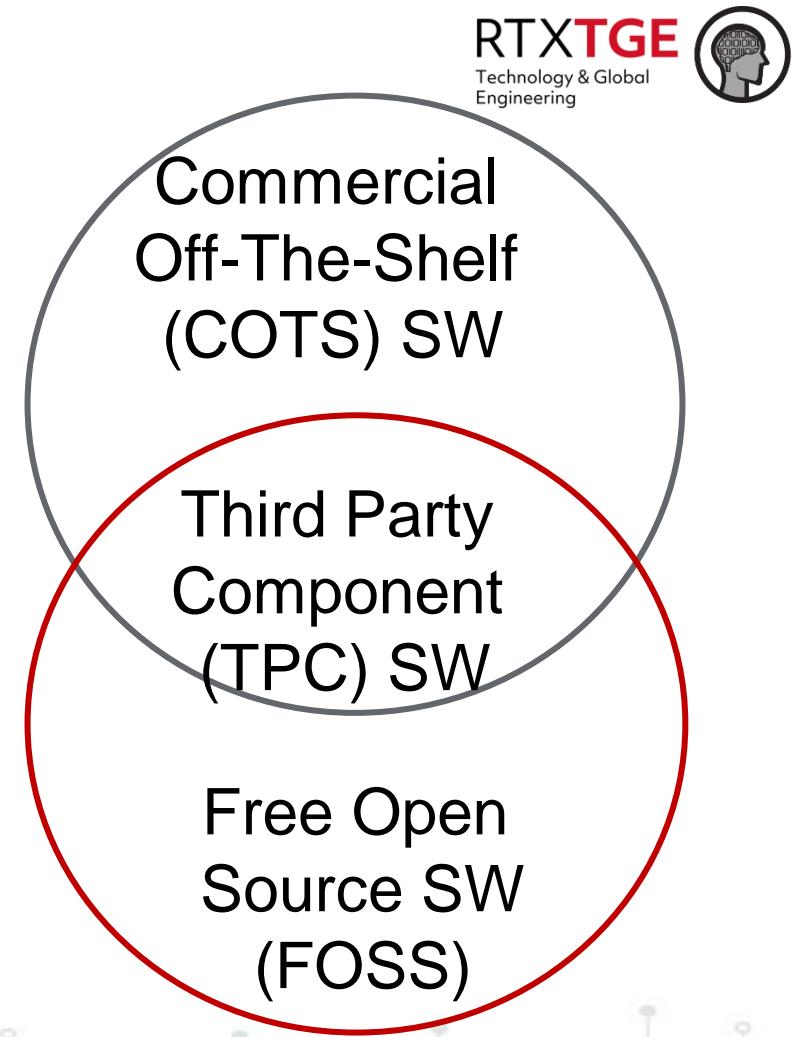
Source: <https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time>



Third Party Component Risk MGT

TPC RISKS TO CONSIDER

- **Incompatibility** between the equipment and the TPC lifecycle
 - Security vulnerability bulletin, TPC maintenance, security update availability...
- Unknown **TPC composition** (unknown Bill Of Materials)
- Lack of supplier **security assurance** process
- Supplier or OSS **community failure**
- Software **supply chain attacks** [[CISA](#)]:
 - Hijacking updates
 - Undermining code signing
 - Compromising supplier development environment
 - Compromising open-source code



TPC Cybersecurity Risk Shall be Anticipated at the Early Stage of the Product Life Cycle



Third Party Component Risk MGT

THIRD PARTY COMPONENT (TPC) SELECTION



Evaluation Criteria	
TCP Supplier Evaluation	TPC Evaluation
<ul style="list-style-type: none"> - Vendor location / community members - Vendor financial health - Vendor / community reputation - OSS Community project activity and robustness (number of active members and commits) - Security Culture, governance, assurance - Security vulnerability and patch management strategy <p>...</p>	<ul style="list-style-type: none"> - Meet assurance/technical requirements - Dependencies and composition known - Lifecycle fit with the equipment lifecycle - Total of known fixed/unfixed vulnerabilities (CVE details [4], NVD database [5]) - Patch reactivity (average time to get a fix) - Existing OSS source code defects: https://scan.coverity.com/ <p>...</p>

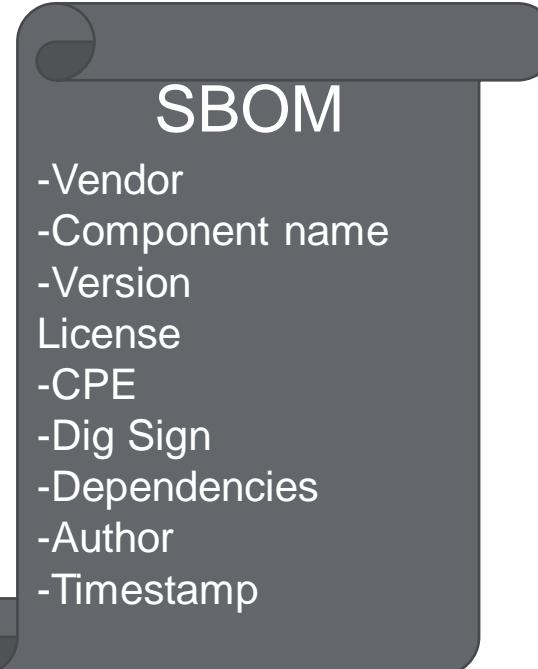
[COL-PCO-GUI-0002 -Third Party Component Management Guidance](#)



Third Party Component VULNERABILITY MGT

SOFTWARE BILL OF MATERIAL (SBOM)

- Knowing the System/software composition is key to manage cyber risks
- Build the inventory/ SBOM and manage it in configuration
- Useful information are:
 - Vendors, Community web site, origine
 - Component Name
 - Version
 - License
 - Unique identifier like the CPE (Mitre ID: Common Product Enumeration)
 - Digital signature or hash
 - Dependencies, Relationship (contains)
 - SBOM author name and a timestamp (creation/last modification)
- Standard format like SPDX, CycloneDX exist to facilitate SBOM management

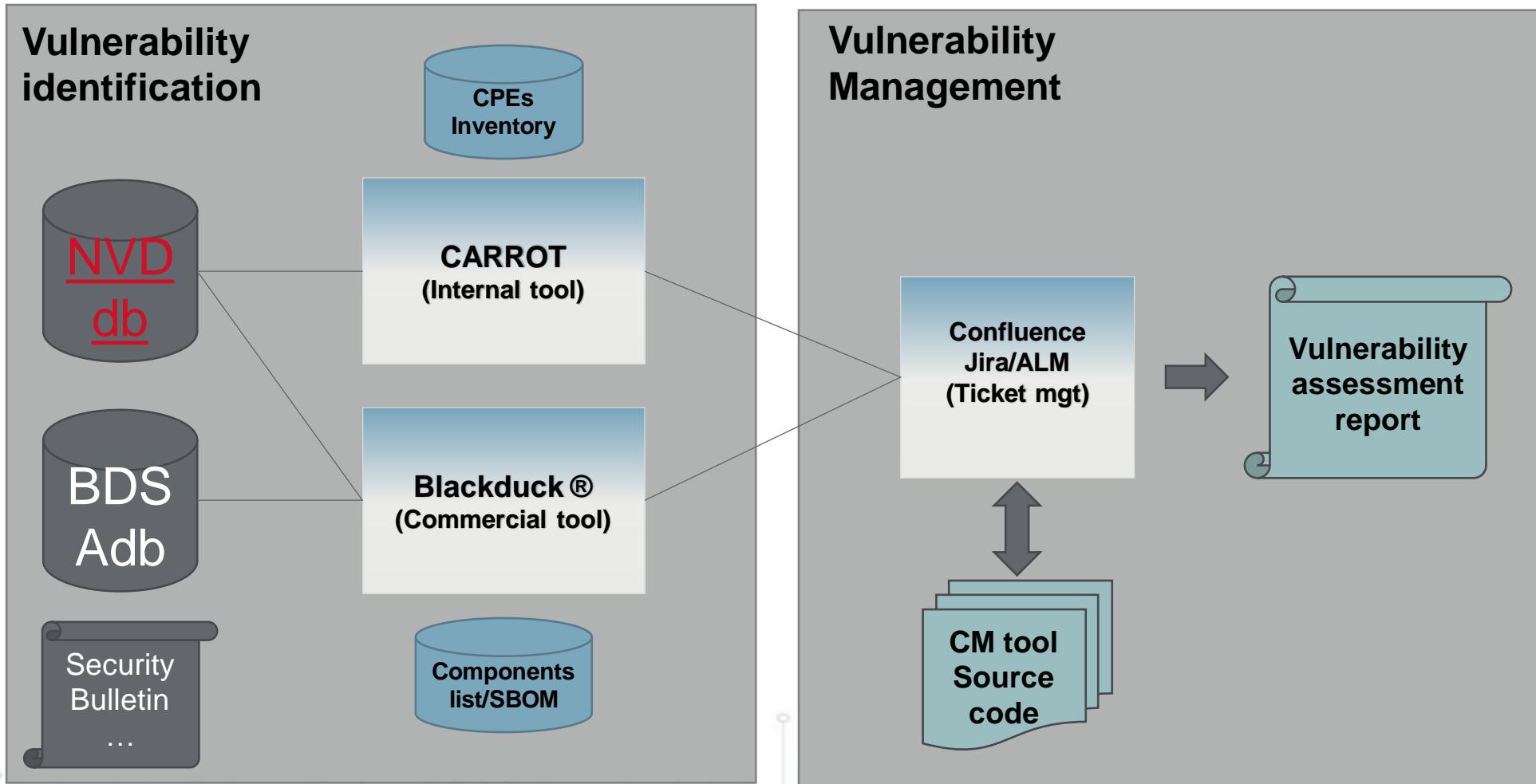


TCP inventory is key in the vulnerability management



TPC Vulnerability Management Tools

COLLINS AVIONICS TOOL CHAIN



Third Party Component Risk MGT

THIRD PARTY COMPONENT (TPC) VULNERABILITY MANAGEMENT

Steps

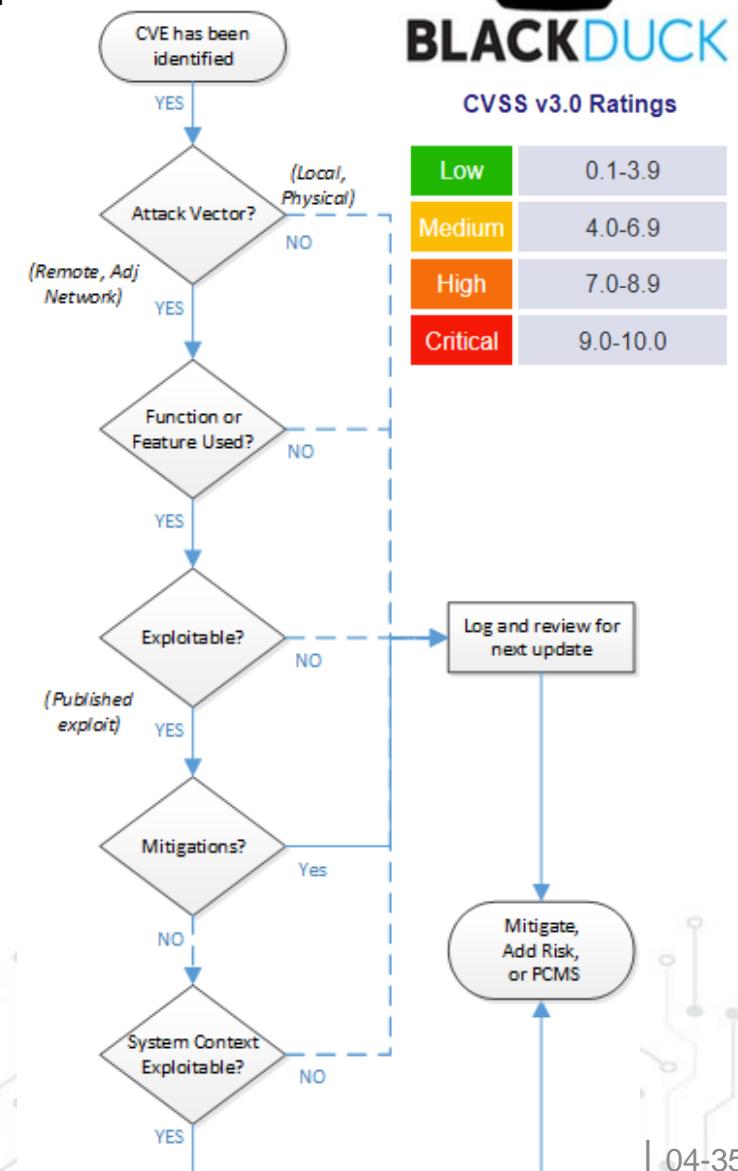
1. Identify public known vulnerabilities (ex: CVE*s) based on TPC SW package versions used in the product
2. Disposition CVEs based on flow chart
 - The threat model help determining the vulnerability exposure
 - When considered applicable the impact analysis require a high expertise level on the targeted system
3. Determine mitigation of concerning CVEs via patching or component version upgrades

References:

- [CVE Details](#)
- [NVD Database](#)

CVE = Common Vulnerability and Exposure

TPC Vulnerability Demo



Security Verification

Responsible: Cyber SME; Support: System & Software Architect/SME



Functional Security Testing

- Verify the functional aspects of the security measures

Robustness / Application Fuzzing

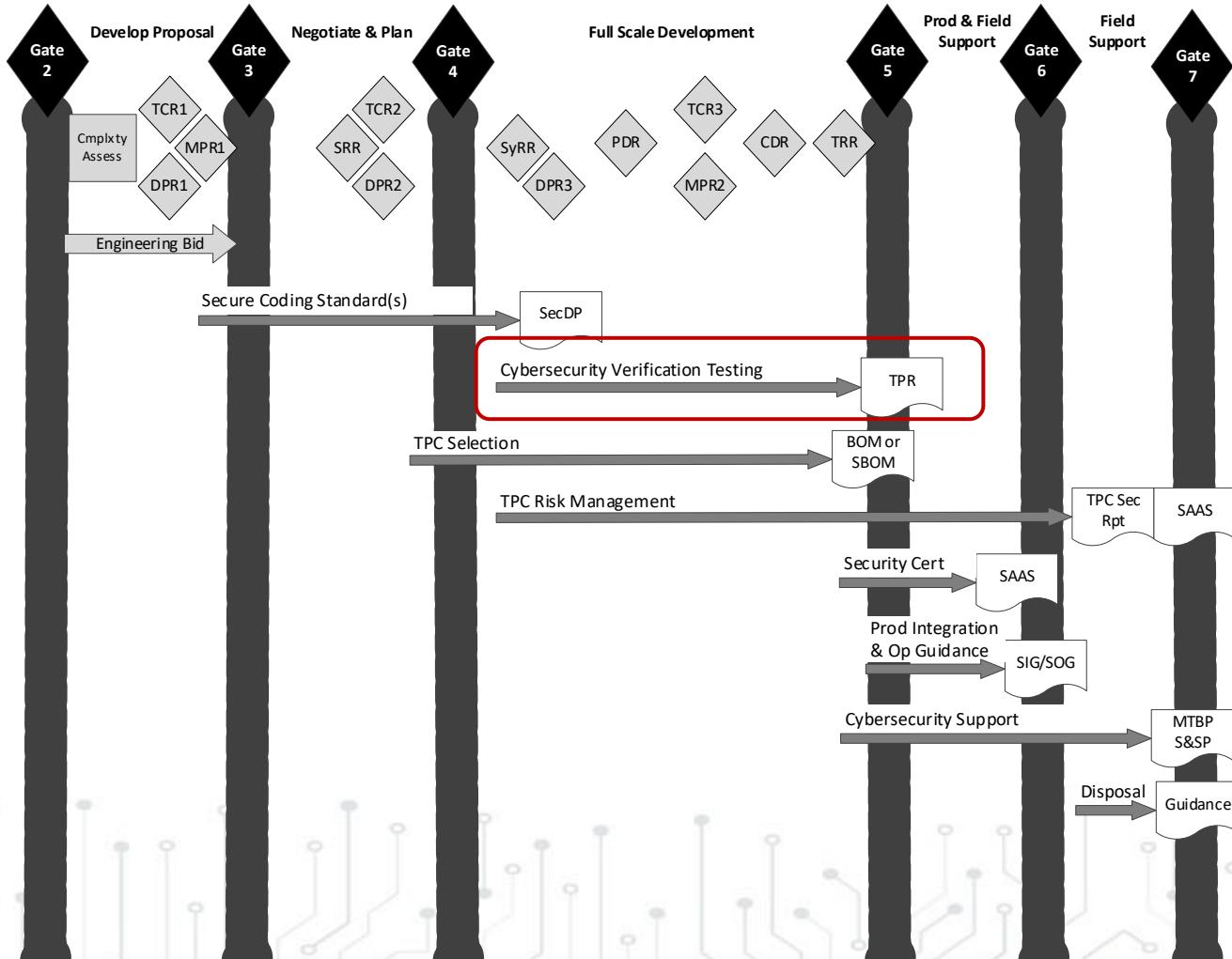
- Send data into exposed functions with incorrect data types, rate, length... to trigger abnormal behavior
- Look for crashes, instability, adverse system impacts

Penetration Testing following SecRA

- Attempt to compromise a system the way an attacker would

Finalize Security Risk Assessment

- Update the SecRA risk level based on Security verification results



Security Verification

ROBUSTNESS / APPLICATION FUZZING

Benefits:

- Generates large volumes of tests
- Small development time and effort
- Sends invalid and near invalid data
- Discover potential zero-day vulnerabilities
- Automated generation of mutated messages for testing software libraries by learning the input format

Issues Discovered:

- Services Crashing
- Format String Errors
 - Length Errors
 - Invalid Characters
 - Too Much Data
 - Out of Order Protocol Operations



Fuzzing Frameworks:

- BooFuzz
- Fuddly
- libfuzzer
- American Fuzzy Lop
- Scapy
- AHS (Collins Aero Hacking Suite)

Denial of Service (DoS):

- Pushing too much data in a single packet
- Pushing data too quickly
- Filling logs w/ useless data, masking attacks



Security Verification

INTRUSION AND PENETRATION TESTING

- Free-form testing that evolves based on the information discovered
- Generally, from external interfaces (take the hacker posture)
- Used to validate SecRA assumptions (reproduce threat scenarios)
- General steps:
 1. **Planning and Preparation:** sets scope and goals of the test.
 2. **Information Gathering:** Probing to determine what services are open and available.
 3. **Vulnerability Detection:** Analysis to determine if any vulnerabilities exist that can be used to gain unauthorized access to system resources.
 4. **Exploitation:** Exploiting discovered vulnerabilities on the target to verify the targets exposure to risk.
 5. **Post Exploitation:** Determine persistence in the system can be achieved.
 6. **Reporting:** Report of all findings discovered during the penetration test.



Operational Support

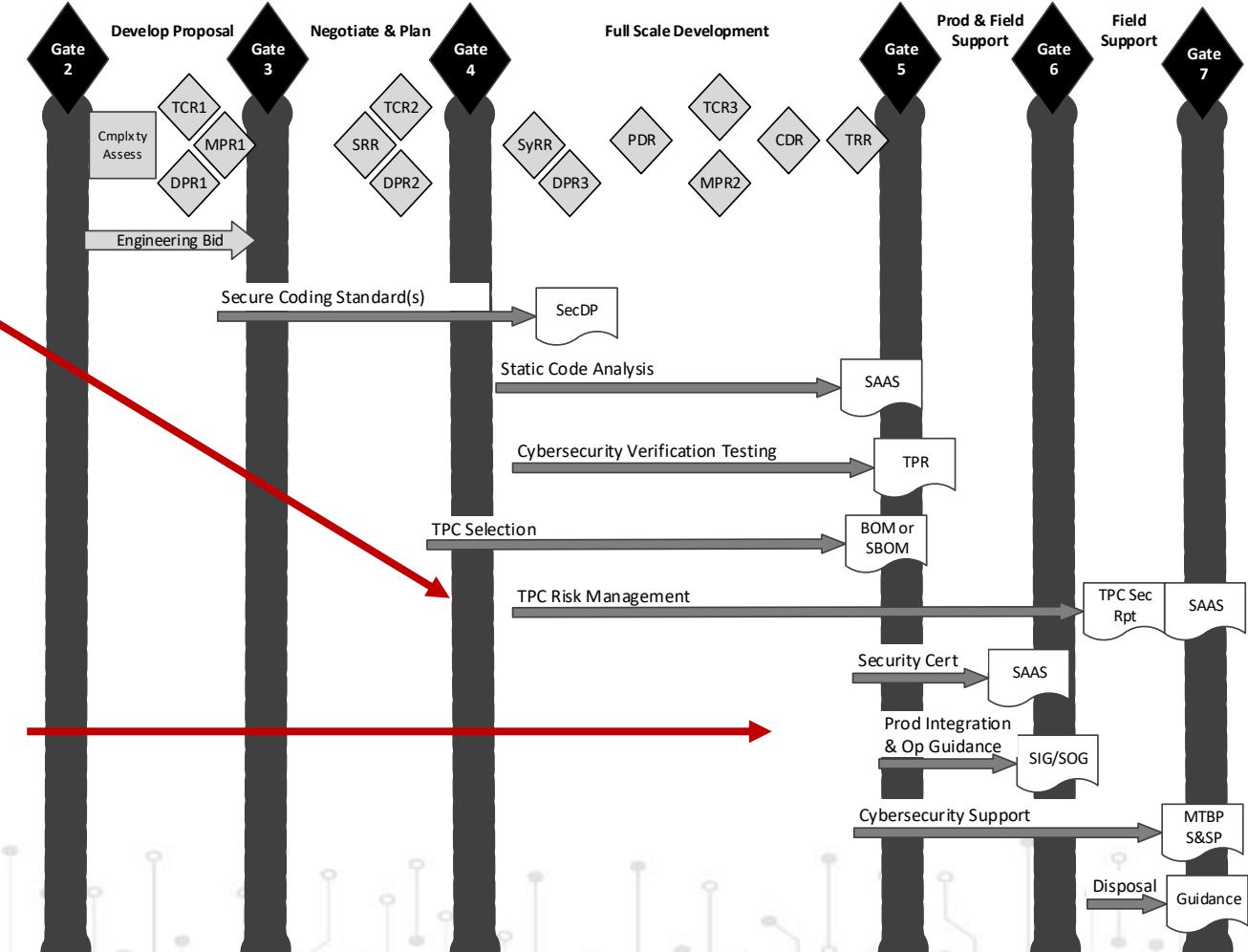
Responsible: Cyber SME; Support: Eng Architect/SME



- Continued Risk / Vulnerability Assessment**

- Assess new vulnerabilities
- Product Security scope change
- New attackers' capabilities
- Technology obsolescence (ex: crypto algorithm...)
- ...

- Develop the **Security integration** and **operational guidance** to make sure the product is correctly deployed and operated
 - Identify **Security actions** and **controls** required during the maintenance phase



Maintain the security effectiveness during the overall life cycle



Operational Support

CONTINUOUS RISK VULNERABILITY ASSESSMENT



- Define the process specific to the system required to maintain the Security effectiveness (SecRA update frequency, CVE analysis...)
- Ongoing Vulnerability Assessment
 - New impactful vulnerabilities?
 - How do they affect our products?
- Security scope and environment change
- Survey on research group disclosure
- New technologies/tools available for the attackers (ex: Quantum computer...)
- Survey on technology obsolescence
 - Crypto algorithm or key size obsolescence
 - How do they affect our products



Operational Support

CONTINUOUS RISK VULNERABILITY ASSESSMENT



- Define the process specific to the system required to maintain the Security effectiveness (SecRA update frequency, CVE analysis...)
- Ongoing Vulnerability Assessment
 - New impactful vulnerabilities?
 - How do they affect our products?
- Security scope and environment change
- Survey on research group disclosure
- New technologies/tools available for the attackers (ex: Quantum computer...)
- Survey on technology obsolescence
 - Crypto algorithm or key size obsolescence
 - How do they affect our products



Operational Support

INTEGRATION AND OPERATIONAL USER GUIDE



How the system needs to be integrated/deployed to prevent security issue

What does the end user have to do to maintain security effectiveness?

Are there security logs?

- What do they do with them
- What do the logs mean?

How do they configure device identity, certificates, passwords & keys?

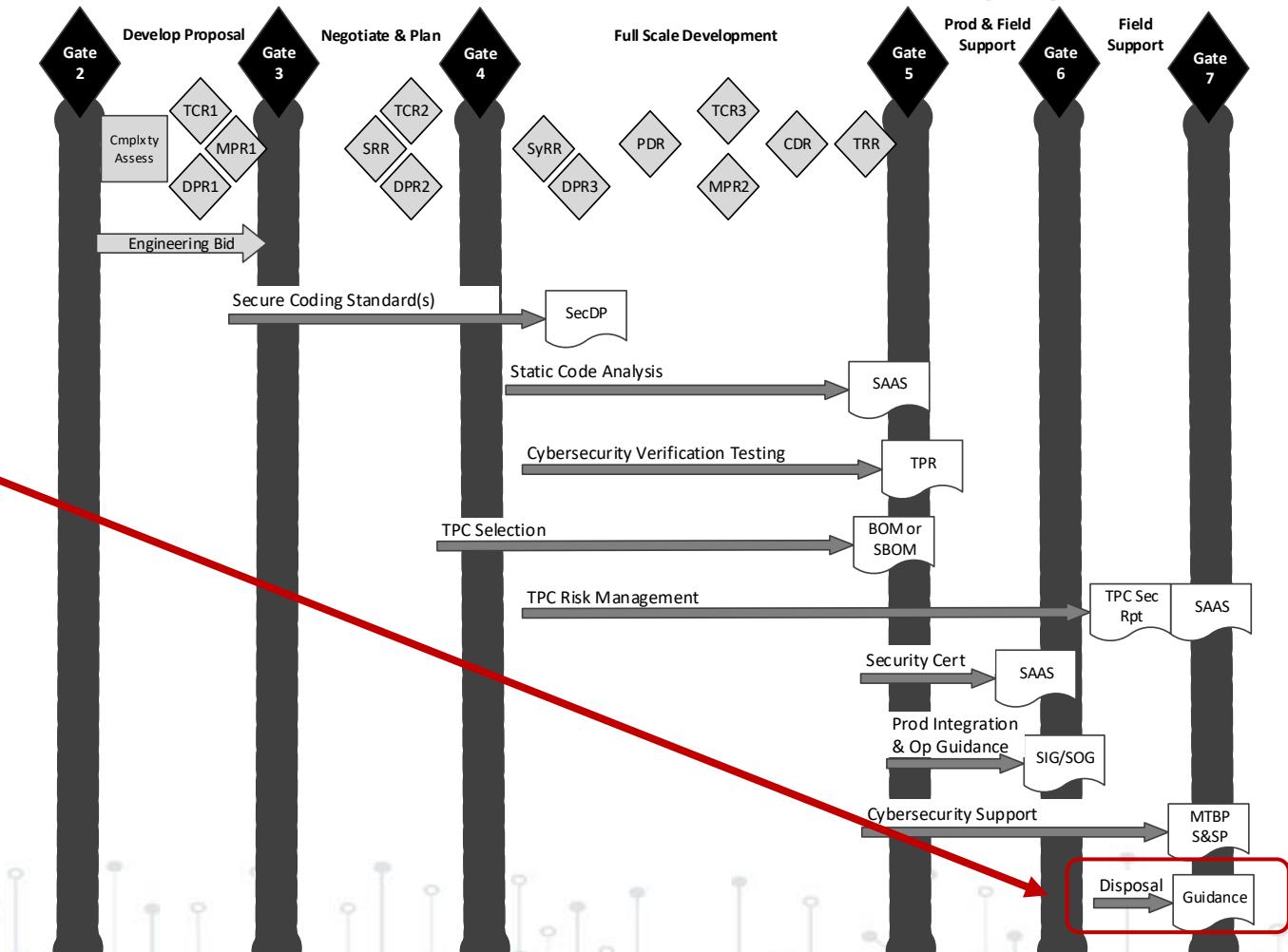


Disposal

Responsible: Eng Architect/SME; Support: Cyber SME



- Identifying if there is any **specific actions required for the Product disposal phase**
 - Ex: Hard Disk destruction, certificates revocation...



Is There Sensitive Data Residing On Product We Want To Protect?



Summary of Engineering ROLE



- Raise to the PCM any potential Product Cybersecurity incident and risk
- Support the PCMS with watchlist items and incident investigation
- Identify and engage with a Cybersecurity SME for the project
- Complete the preliminary High-level Cybersecurity Assessment prior Gate 3
- If HLCA drive SSDLC applicability:
 - Support the Cybersecurity SME on the SecRA across development life cycle
 - Support the Cybersecurity SME on Security Development Plan definition
 - Support the cybersecurity requirements definition
 - Manage the secure implementation (Security coding rules guidance and Static Code Analysis)
 - Support the Cybersecurity SME on TPCs cybersecurity risks management (if applicable)
 - Develop and maintain the SBOM (Software Bill Of Material)
 - Manage Security functional tests
 - Support the Cybersecurity SME on Security testing (e.g., verification, penetration, fuzzing)
 - Support continuous risk assessment (TPC, vulnerability mgt, security events...)

Product Cybersecurity Requires Teamwork



Product Cybersecurity Contacts



- SBU Product Cybersecurity Managers
 - Specific focal for SSDLC deployment for a product portfolio (SBU)
 - Transitions the SSDLC (What) to the business (How)
 - First point of contact for questions on “How do we do it for project XYZ?”
- Find & use your SBU focal to guide your projects: [SBU Product Cybersecurity Managers](#)
- Collins Product Cybersecurity Office: GCollinsProductSecurityOffice@collins.com



Resources



- Product Cybersecurity Office (PCO) Portal: [Collins PCO Site](#)
- High Level Cybersecurity Questionnaire (HLCA): [COL-PCO-FRM-0007](#)
- Additional Resources:
 - TPC vulnerability management and Blackduck tools [Link](#)
 - CELL Static Code Analysis Endorsed Practices : [Link](#)
 - CELL Secure Coding Guidance Shared Practices: [Link](#)
 - [CATU - Avionics PCMS and SSDLC Introduction](#)
 - Collins Product Cybersecurity Office (PCO) Documentation: [Link](#)



Questions?



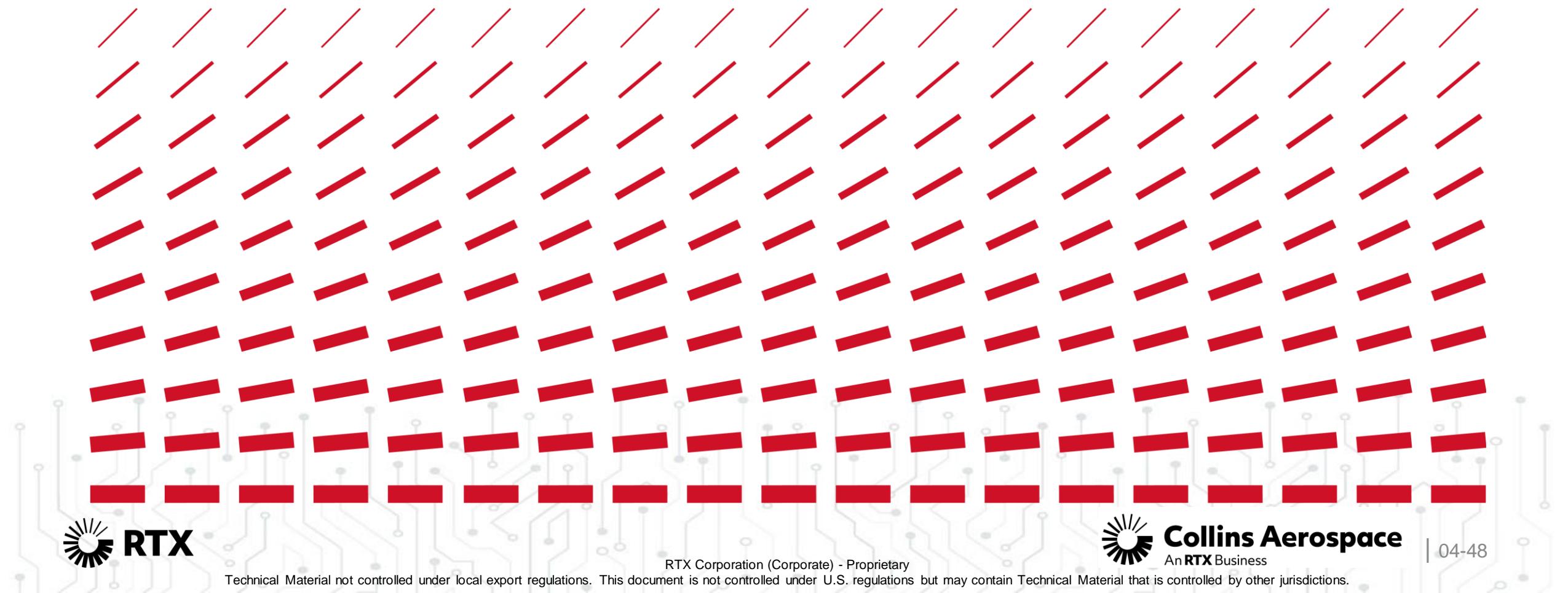
Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.

RTX Corporation (Corporate) - Proprietary



| 04-47

Thank you.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





Collins Aerospace
An RTX Business

© 2023 Raytheon Technologies Corporation
All rights reserved
RGEMS Tracking

© 2023 Collins Aerospace. | Collins Aerospace Proprietary. | This document does not include any export controlled technical data.

RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.

RTX TGE
Technology & Global
Engineering



TGE CYBERSECURITY

Embedded Systems Security

Module 05
Cybersecurity Certification and Standards
Considerations

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

Agenda



COLLINS PRODUCT CYBERSECURITY TRAINING

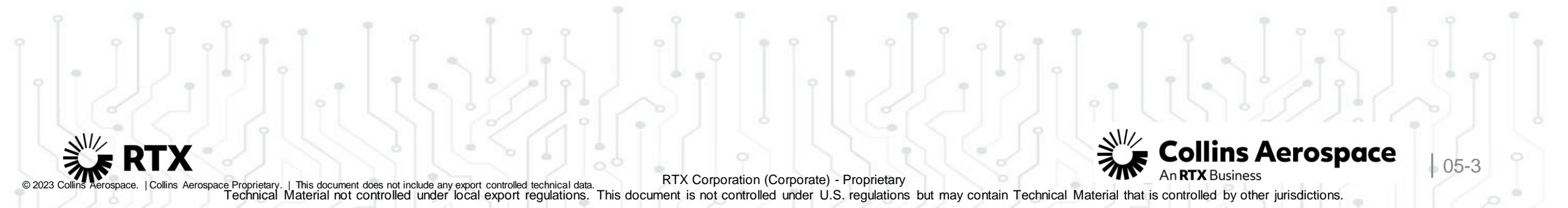
- Module 1: Cybersecurity General Overview
- Module 2: Aviation Product Cybersecurity Challenges
- Module 3: Collins Product Cybersecurity: Policy Flow down
- Module 4: Collins Secure System Development Life Cycle
- **Module 5: Cybersecurity Certification and Standards Considerations**
- Module 6: DO-326A Airworthiness Certification Process
- Module 7: Aviation Threat Example



Module 5: Cybersecurity Certification and Standards Considerations



- Safety vs Aviation Cybersecurity
- Safety Certification and standard
- Cybersecurity Certification, Standard and guidance
- Cybersecurity process overview



Aircraft Certifications Types



Type certification (TC): Approval of the design of the aircraft and all component parts (including propellers, engines, control stations, etc.). It signifies the design is in compliance with applicable airworthiness, noise, fuel venting, and exhaust emissions standards.

Supplemental type certificate (STC): A type certificate (TC) issued when an applicant has received FAA approval to modify an aeronautical product from its original design. The STC, which incorporates by reference the related TC, approves not only the modification but also how that modification affects the original design.

Technical Standard Order (TSO): Minimum performance standard for specified materials, parts, and appliances used on civil aircraft.

Parts Manufacturer Approval (PMA): Combined design and production approval for modification and replacement articles. It allows a manufacturer to produce and sell these articles for installation on type certificated products.

Source: www.faa.gov



Aviation Regulatory Landscape for Civil Certification



Product	FAA	EASA
Gen Av (small A/C)	Part 23	CS-23
Air Transport (Large A/C)	Part 25	CS-25
Light Rotary	Part 27	CS-27
Big Rotary	Part 29	CS-29
Engines	Part 33	CS-E
Propellers	CS-P	CS-P
European TSOs		CS-ETSO
Aux Power		CS-APU

Aircraft certification rules are based on latest amendment of the Certification Specification



Cybersecurity Certification Applicability



EASA

- Applies to all new Type Certificates
- Applies to all existing products with a significant change to their TC
- EASA Part 21 Appendix A (.21.A.91) gives examples of Major Changes.
 - Change that may introduce the potential for unauthorized electronic access to product systems should be considered 'major' if there is a need to mitigate the risks for an identified unsafe conditions.
 - Examples: new digital communication means, new service (data flow) accessible from an untrusted source, etc.

Jan 2021

Legacy Aircrafts

Original certification
+ special conditions for
cybersecurity issue

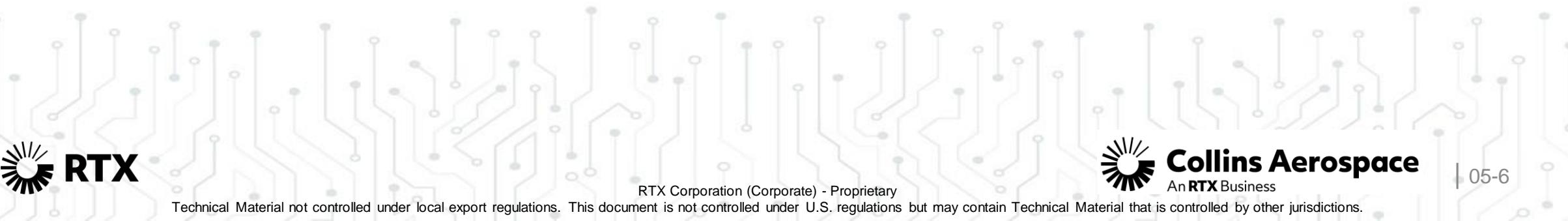
No change, voluntary compliance

Significant change

New rules apply
to changes only

New Aircrafts

New rules apply



Cybersecurity Certification Applicability – cont.



FAA

Issued the OpSpec D301 / AC 119-1A for operator (Post TC)

- Apply to e-enabled/connected aircraft with Cybersecurity special conditions.
- Describes an acceptable means to obtain authorization to operate an A/C (Similar to DO-355A)
- Operator to define an Aircraft Network Security Program
 - Ensure that data security protection is sufficient to prevent access by unauthorized devices or personnel external to the aircraft.
 - Ensure that security threats specific to the operator's fleets, routes, and maintenance practices are identified and assessed, and that risk mitigation strategies are implemented...
 - Prevent inadvertent or malicious changes to the aircraft network, systems, and software, including those possibly caused by maintenance activity...

Note: An operator's ANSP should not include independent aircraft testing that tampers with the certified system. This could result in nonconformance to type design and render the aircraft unairworthy.



Aviation Safety & Cybersecurity



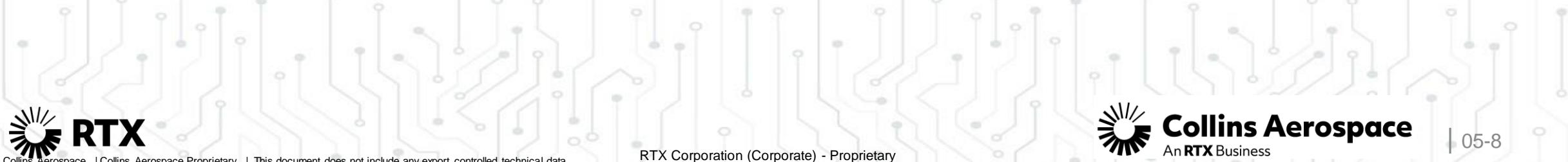
Aviation Safety

Within the context of aviation, safety is the state in which risks associated with aviation activities, related to, or in direct support of the operation of aircraft, are reduced and controlled to an acceptable level. (Source: ICAO)

The state in which risk is acceptable. (Source: ARP4754A)

Aviation Cybersecurity for Safety (Airworthiness Security)

The protection of the airworthiness of an aircraft from intentional unauthorized electronic interaction: harm due to human action (intentional or unintentional) using access, use, disclosure, disruption, modification, or destruction of data and/or data interfaces. This also includes the consequences of malware and forged data and of access of other systems to aircraft systems. (Source: DO-326A / ED-202A)



Quiz 2: Safety vs Cybersecurity

Which of the following properties are valid for the Safety domain, Cybersecurity domain or both?



	Safety	Cybersecurity	Both
Based on Risk management			
Quantify the risk			
Estimate the risk			
Manage intentional unauthorized acts			
Experience in service is an advantage (mature technology),			
Static operational environment			
Manage sys failure and human error			
Fail-Safe approach			
Fail Secure approach			



Quiz 2: Safety vs Cybersecurity

Which of the following properties are valid for the Safety domain, Cybersecurity domain or both?



	Safety	Cybersecurity	Both
Based on Risk management			x
Quantify the risk	x		
Estimate the risk		x	
Manage intentional unauthorized acts		x	
Experience in service is an advantage (mature technology),	x		
Static operational environment	x		
Manage sys failure and human error			x
Fail-Safe approach	x		
Fail Secure approach		x	

Safety vs Cybersecurity



Safety	Cybersecurity
Based on Risk management	Based on Risk management
Quantify the risk	Estimate the risk with a likelihood
Safety objective is the reliability (Integrity, Availability)	Can serve safety objectives I-A (Cybersecurity for safety) but also confidentiality C.
Manage hardware/software failure and human error	Manage intentional acts in addition
Static environment	Dynamic environment (Threat environment)
Experience in service is an advantage (mature technology), avoid change	Mature technology can come obsolete from a Cyber perspective (new vulnerability, need for patch regularly)
Fail-Safe approach	Fail Secure approach



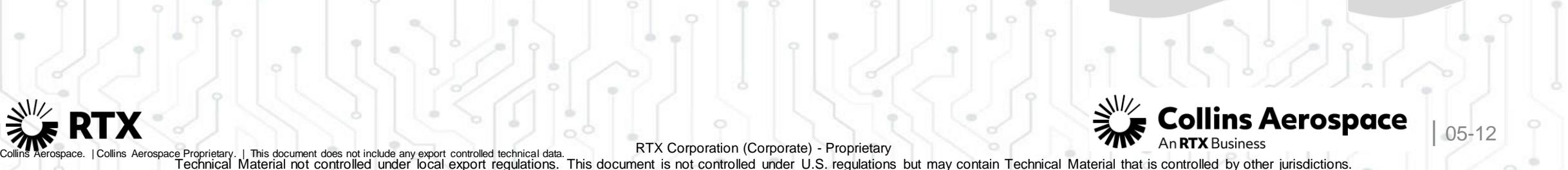
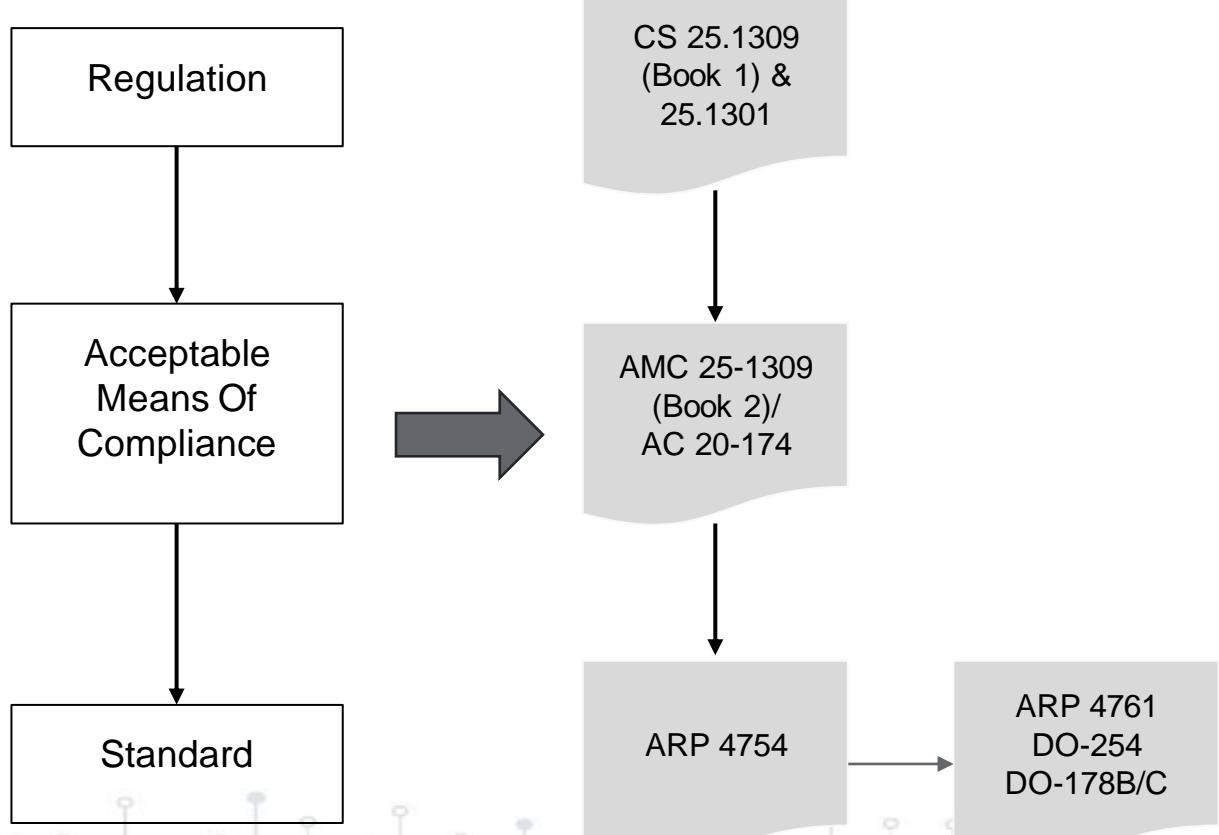
How does it work for Aircraft Safety



(a) The aeroplane equipment and systems must be designed and installed so that:

(1) Those required for type certification or by operating rules, or whose improper functioning would reduce safety, perform as intended under the aeroplane operating and environmental conditions.

(2) Other equipment and systems are not a source of danger in themselves and do not adversely affect the proper functioning of those covered by sub-paragraph (a)(1) of this paragraph.



Design Assurance LEVELS (DAL)



DAL	Failure Condition	Description
A	Catastrophic	Failure may cause deaths, usually with loss of the approach
B	Hazardous	Failure has a large negative impact on safety or performance or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload or causes serious or fatal injuries among the passengers.
C	Major	Failure significantly reduces the safety margin or significantly increases crew workload. May result in passenger discomfort (or even minor injuries).
D	Minor	Failure slightly reduces the safety margin or slightly increases crew workload. Examples might include causing passenger inconvenience or a routine flight plan change.
E	No Safety Effect	Failure has no impact on safety, aircraft operation, or crew workload



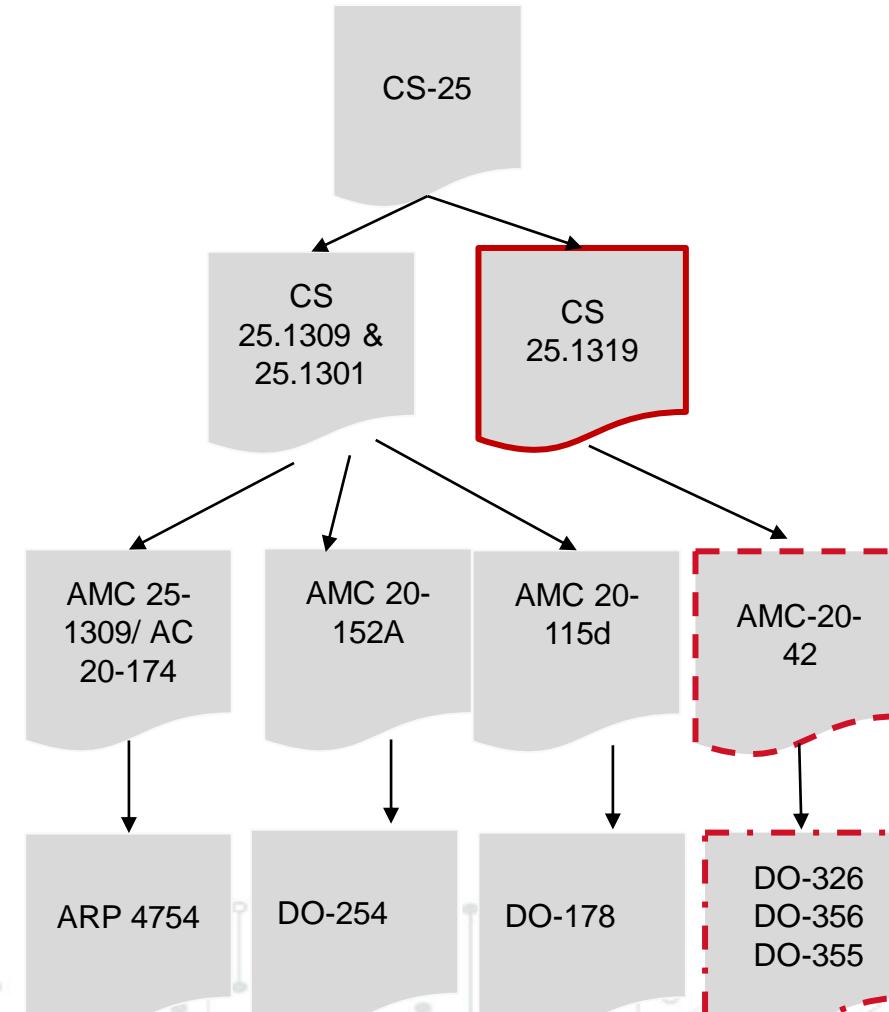
How does it work for Cybersecurity?



CS 25.1319

"....Equipment, systems and network information protection
(a) Aeroplane equipment, systems and networks, considered separately and in relation to other systems, must be protected from intentional unauthorised electronic interactions (IUEIs) that may result in adverse effects on the safety of the aeroplane. Protection must be ensured by showing that the security risks have been identified, assessed and mitigated as necessary.
(b) When required by paragraph (a), the applicant must make procedures and Instructions for Continued Airworthiness (ICA) available that ensure that the security protections of the aeroplane's equipment, systems and networks are maintained

Applicable to all new TC since January 2021 and STC, ETSO considered major change.



How does it work for Cybersecurity?



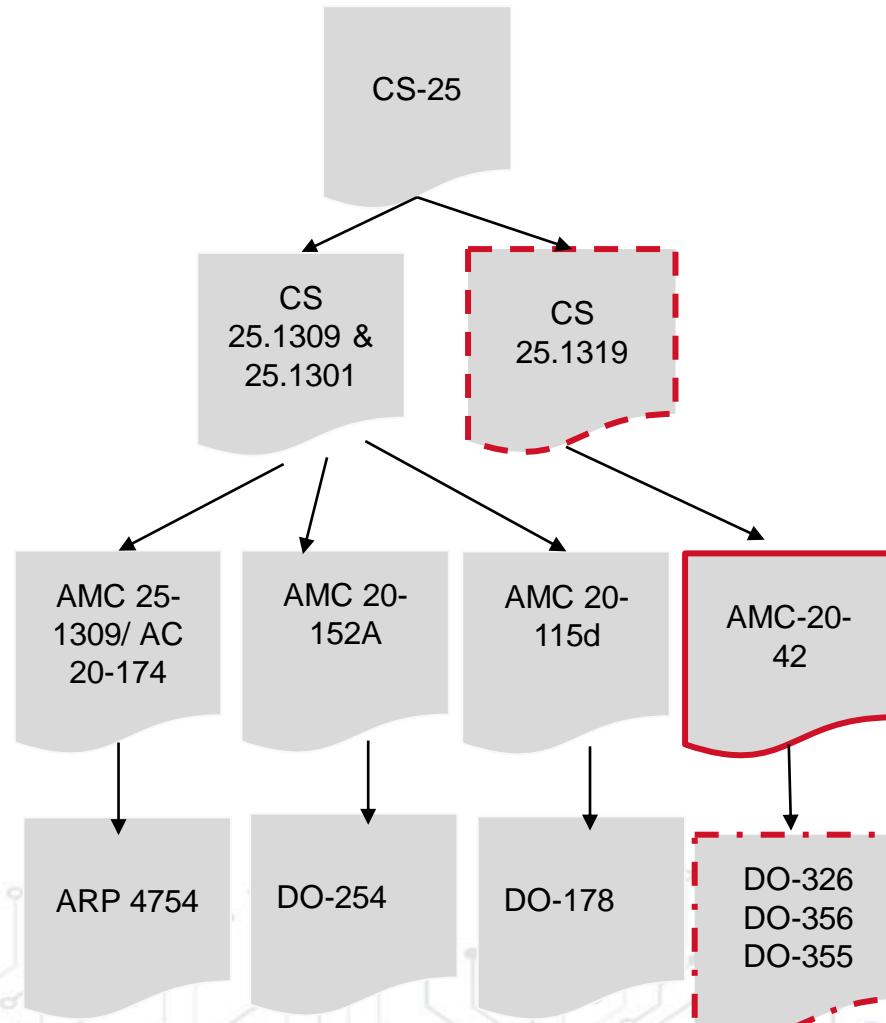
AMC 20-42 Airworthiness information security risk assessment:

(a) This AMC describes an acceptable means, but not the only means, to show compliance with the applicable rules for the certification of products and parts.....

(b) This AMC recognizes as an acceptable means of compliance the following (EUROCAE) and (RTCA) documents:

- **EUROCAE ED-202A / RTCA DO-326A**, Airworthiness Security Process Specification,
- **EUROCAE ED-203A / RTCA DO-356A**, Airworthiness Security Methods and Considerations,
- **EUROCAE ED-204A / RTCA DO-355A**, Information Security Guidance for Continuing Airworthiness,

https://www.easa.europa.eu/sites/default/files/dfu/amc-20_amendment_18.pdf



Security Standards and Guidance



DO-326A / ED-202A - Airworthiness Security Process Specification

What needs to be done

DO-356A / ED-203A - *Airworthiness Security Methods and Considerations*

How you do it

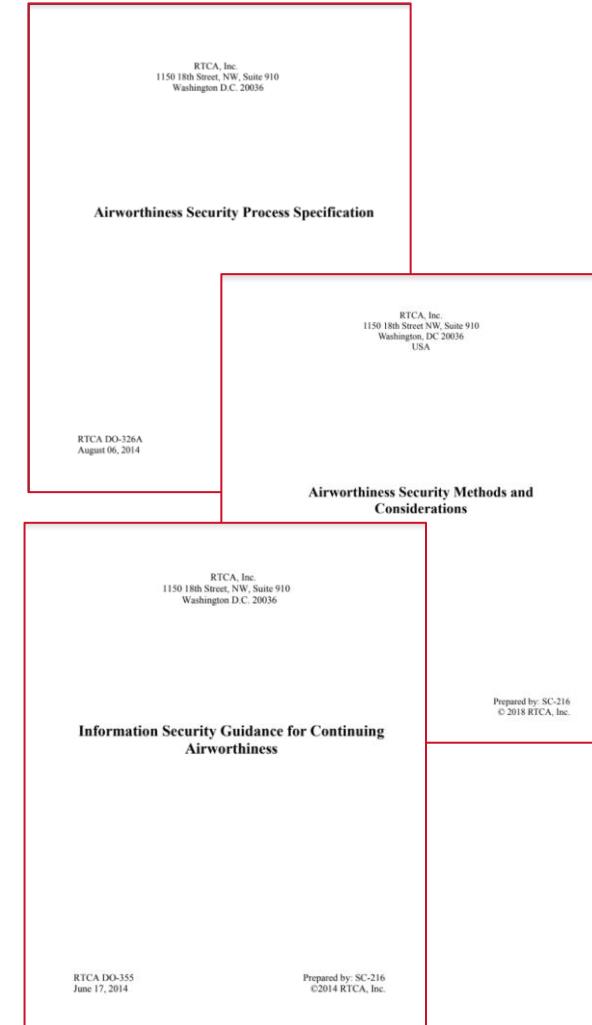
DO-355A / ED-204A - Information Security Guidance for Continuing Airworthiness

How do you maintain it (Post TC)

DO-393 / ED-205 - *Process Standard for Security Certification and Declaration of ATM ANS Ground Systems*

DO-392 / ED-206 - Guidance on Security Event Management

How to detect and manage incident (Post TC)



Questions?



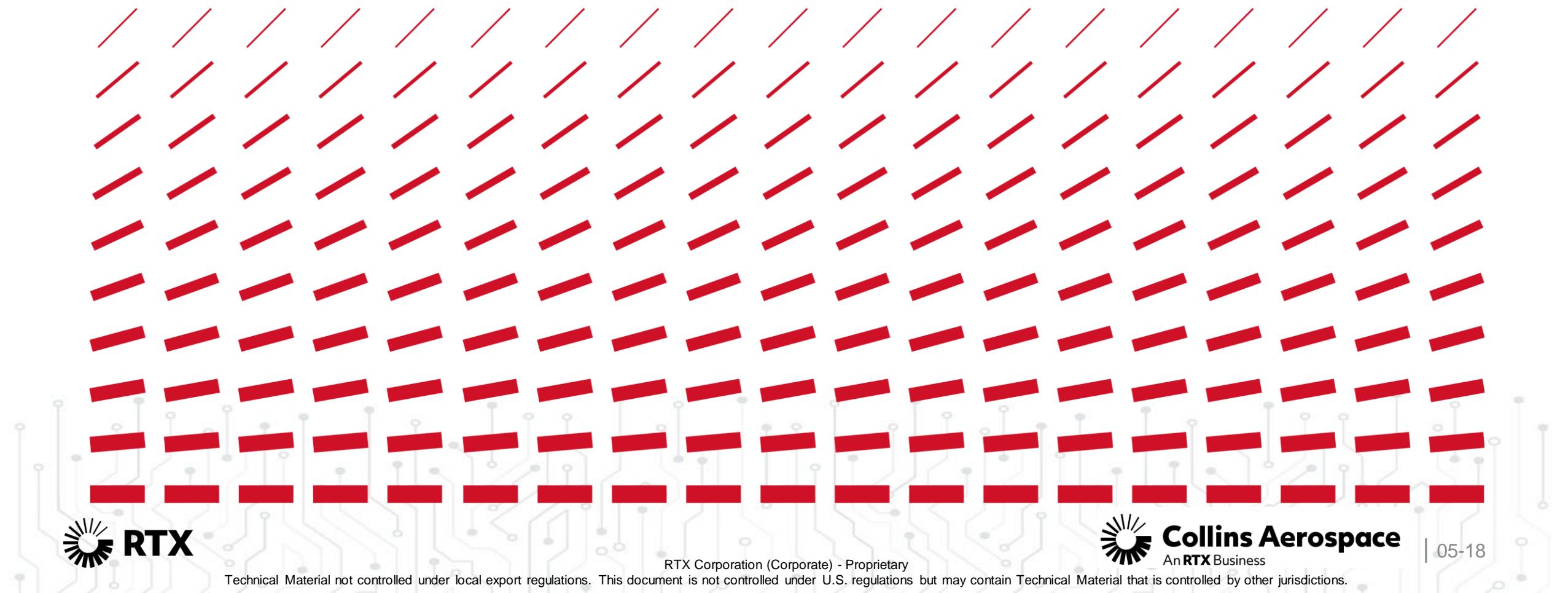
RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.



0517

Thank you.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





Collins Aerospace
An RTX Business

RTXTGE
Technology & Global
Engineering



TGE CYBERSEC

Embedded Systems Security

Module 06

DO-326A Airworthiness Certification Process

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

Agenda



COLLINS PRODUCT CYBERSECURITY TRAINING

- Module 1: Cybersecurity General Overview
- Module 2: Aviation Product Cybersecurity Challenges
- Module 3: Collins Product Cybersecurity: Policy Flow down
- Module 4: Collins Secure System Development Life Cycle
- Module 5: Cybersecurity Certification and Standards Considerations
- **Module 6: DO-326A Airworthiness Certification Process**
- Module 7: Aviation Threat Example



Module 6: DO-326A Airworthiness Certification Process

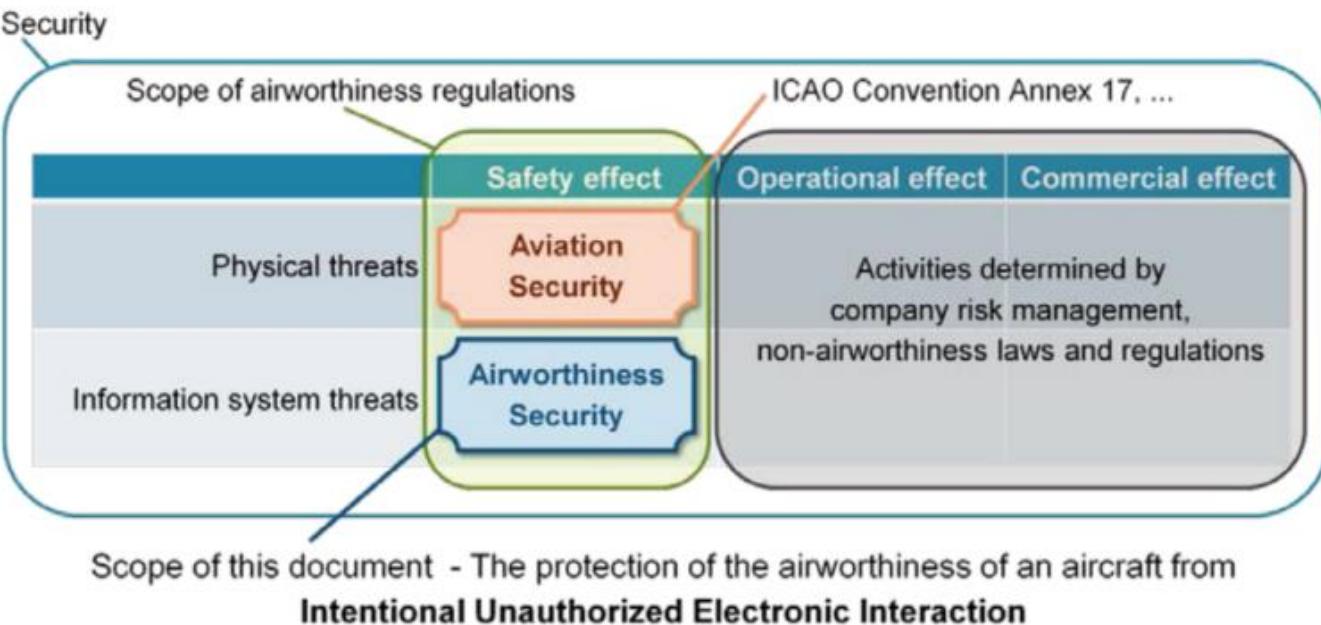


- Overview of the Activities involved in the DO-326A Certification Process



DO-326A/ED-202A Scope

INTENTIONAL UNAUTHORIZED ELECTRONIC INTERACTION (IUEI)



Intentional Unauthorized Electronic Interaction (IUEI) :

- Circumstance or event with the potential to affect the aircraft due to human action resulting from unauthorised access, use, disclosure, denial disruption, modification, or destruction of information and/or aircraft system interfaces
- This includes the consequences of malware and forged data and the effects of external systems but does not include physical attacks or electromagnetic disturbance



DO-326A/ED-202A

AIRWORTHINESS SECURITY PROCESS SPECIFICATION



- Provides the process to handle the threat of IUEI to safety.
- Intended to be used in conjunction with other applicable guidance material
 - SAE ARP 4754A/ED-79A, SAE ARP 4761/ED-135, DO-178C/ED-12C, DO-254/ED-80...
- ***Only addresses onboard equipment***
 - Does not address Physical Security, Airports / Airlines / Air Traffic Service Providers, comm., navigation and surveillance services managed by national agencies (GPS, ADS-B, ...)
 - Establish mitigations / acceptability of safety risks induced by IUEI



DO-356A/ED-203A

AIRWORTHINESS SECURITY METHODS AND CONSIDERATIONS



- Companion document of DO-326A
- Provides methods and guidelines usable within DO-326A for “showing Acceptable Means of compliance for airworthiness security during aircraft design and development life-cycle”
 - Methodology for risk assessment and Acceptable Means of Compliance
 - Security Assurance Level definition and Acceptable Means of Compliance
 - Define Security assurance relation to threat condition severity -> SAL level
 - Security Assurance Objectives (39 objectives) and related assurance activities (118) by SAL
 - Regulatory consideration : Airworthiness Risk Acceptability Matrix
 - Guidance for the Development of Security Architectures and Measures
 - Guidance for Security Event Logging and provides the Acceptable Means of Compliance



DO-326A/ED-202A

AIRWORTHINESS SECURITY PROCESS SPECIFICATION

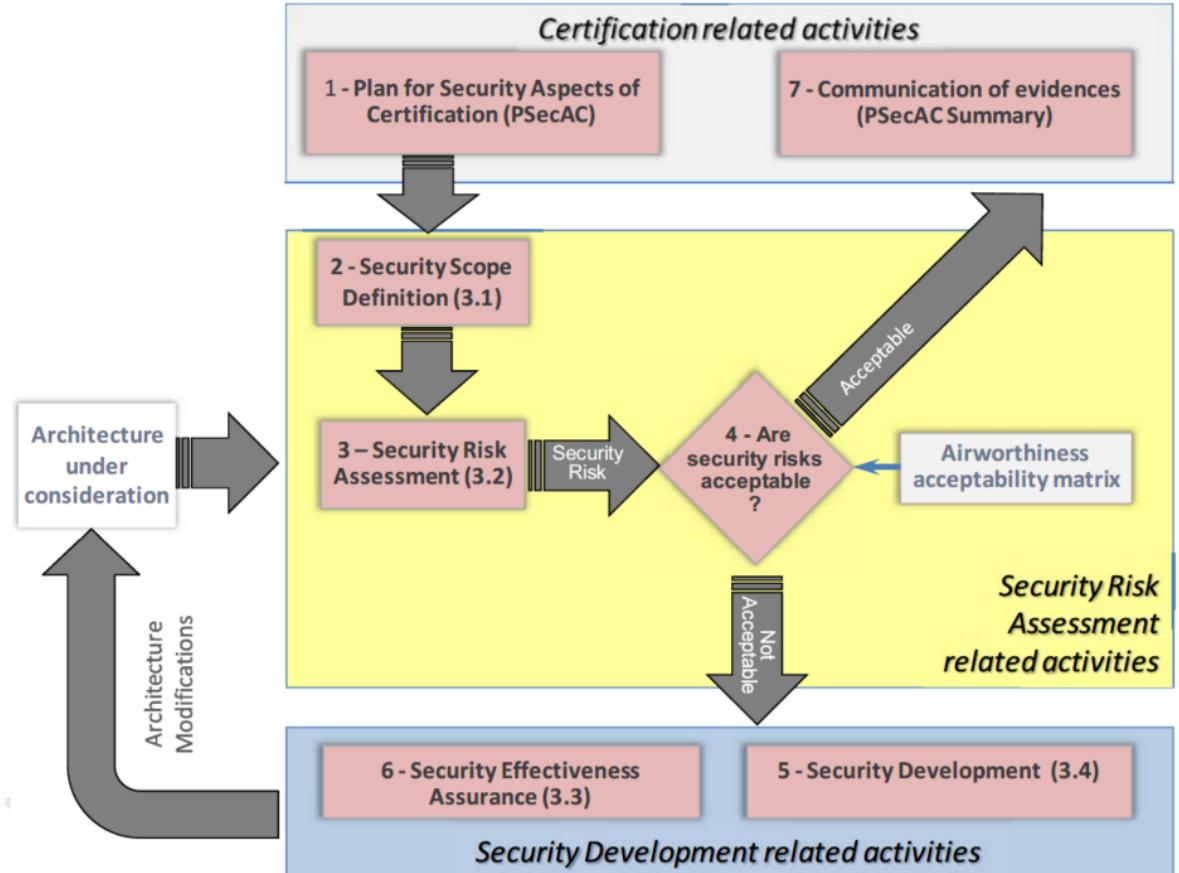


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A

1. Plan for Security Aspects of Certification (PSecAC): The plan is defined by the applicant and agreed to by the Airworthiness Authorities.
2. Aircraft/System level Security Scope Definition (ASSD, SSSD): Establish the security scope as an input for Security Risk Assessment.
3. Aircraft/System Security Risk Assessment ((P)ASRA, (P)SSRA): Identify and evaluate security risks.
4. Decision Gate 4: “Are Security Risks acceptable?”: airworthiness acceptability matrix.
5. Security Development: Risk mitigation results in the design of a security.
6. Security effectiveness assurance: Confidence that the security risks are acceptable.
7. Communication of evidences: When all risks are acceptable, the results of airworthiness security activities should be captured in the PSecAC Summary.

Step 1: Plan for Security Aspects of Certification (PsecAC)



- Defined by applicant and agreed to by the Airworthiness Authorities
- Established at the beginning of the certification process
- Identify means for demonstrating compliance with airworthiness regulations relative to security concerns
- Identify the Security Assurance Objectives and associated activities
- Identify the certification data package and deliverables
- Means and methods for showing compliance to the certification basis related to security (ex: Project organization, Risk assessment process, test strategy...)

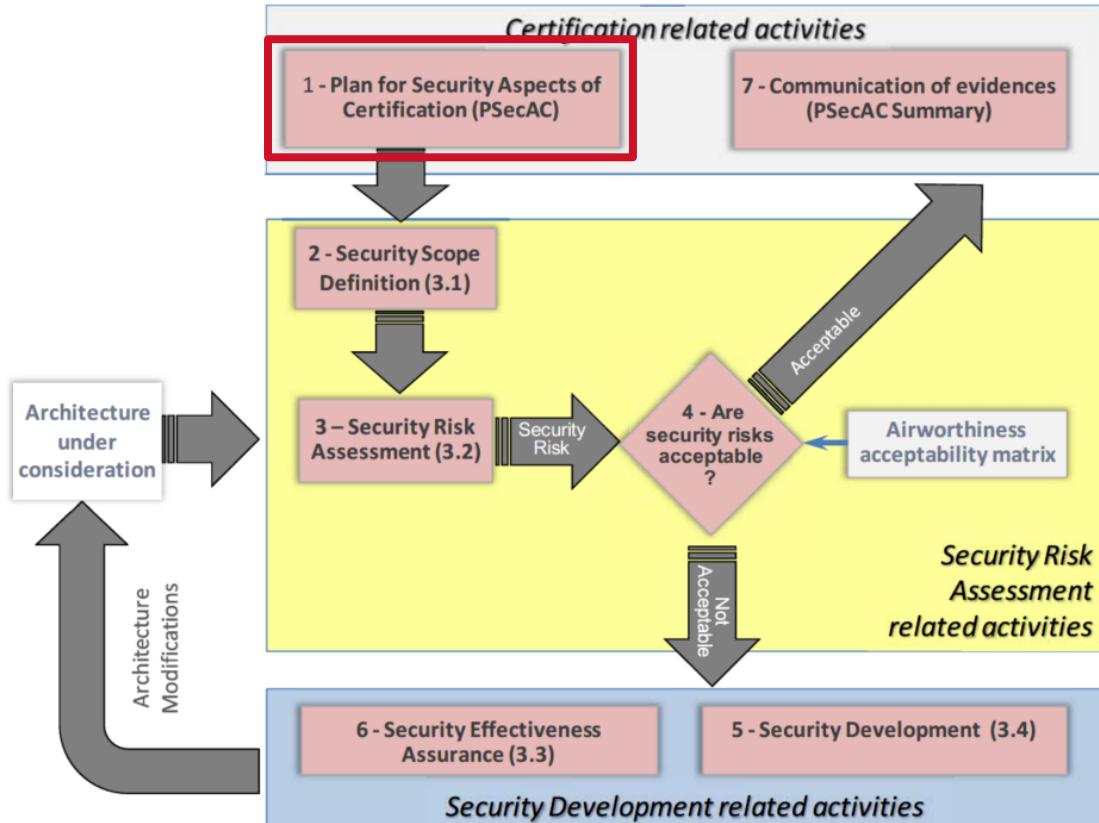


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



Step 2: Security Scope



- Assets: logical and physical resources of the aircraft that contribute to the airworthiness of the aircraft
- Identify the asset protection perimeter and interfaces with other environments
 - What should be protected against intentional unauthorized electronic interaction from a safety hazard classification viewpoint.
- Iteratively refined along system design life-cycle, layered (aircraft/system/item)

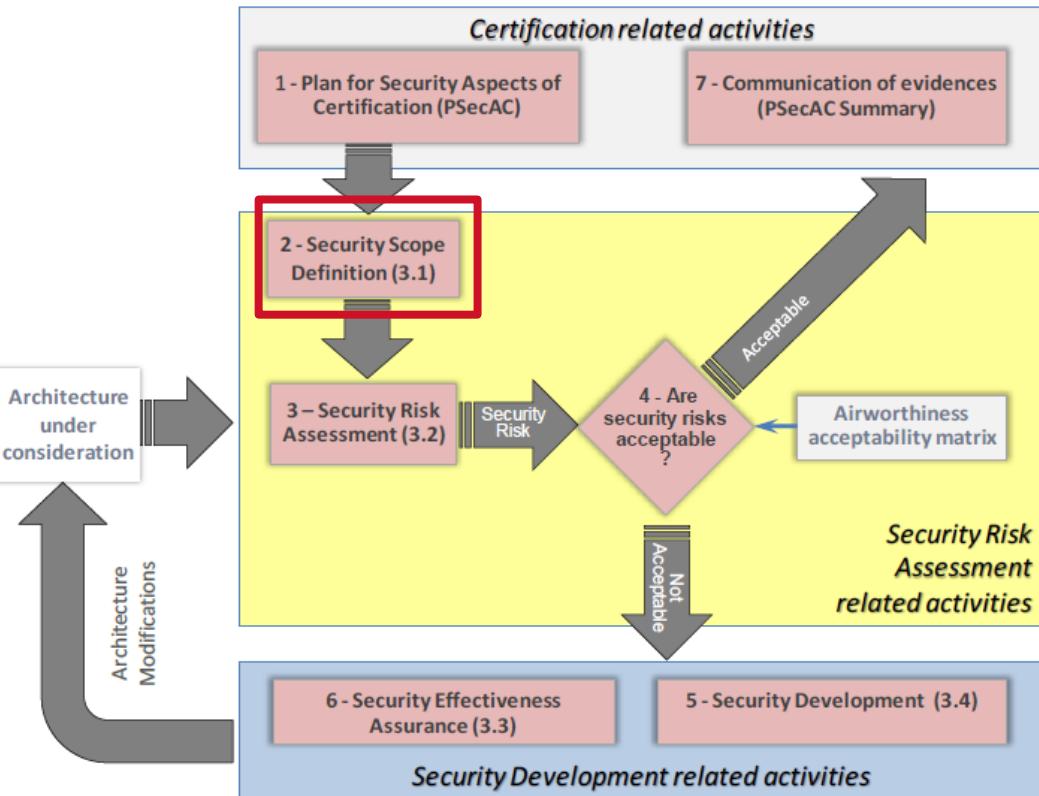


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



Step 2: Security Scope

- Assets Identification:
 - Define connections between assets (ex: inside the security perimeter)
- Definition of the security Perimeter
 - Based on a functional architecture and the aircraft/system ICD
 - Identify boundary of the target of evaluation
 - Physical/Wireless Interface, logical connections (services, protocols, information flows) with external entities
- Characterization of the Environment
 - The security environment describes the assumptions (trusted/Untrusted) and requirements about the persons, organizations, and external systems outside the security perimeter that interact with the assets under consideration.
 - Trusted interactions excluded from Assessment (Should be agreed with authority or validated)

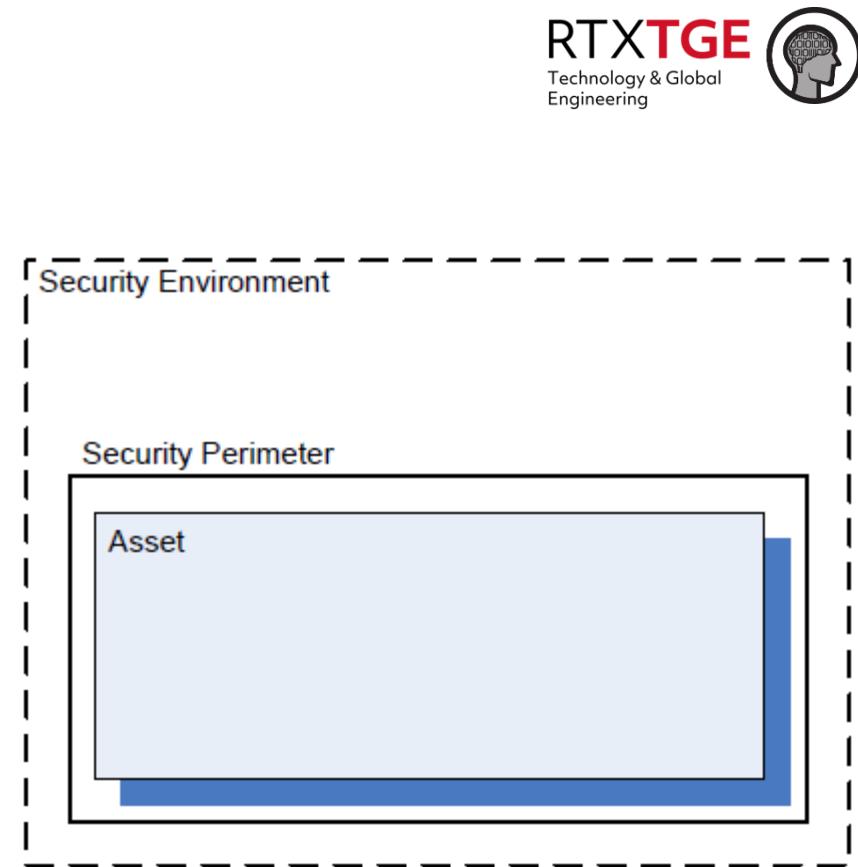
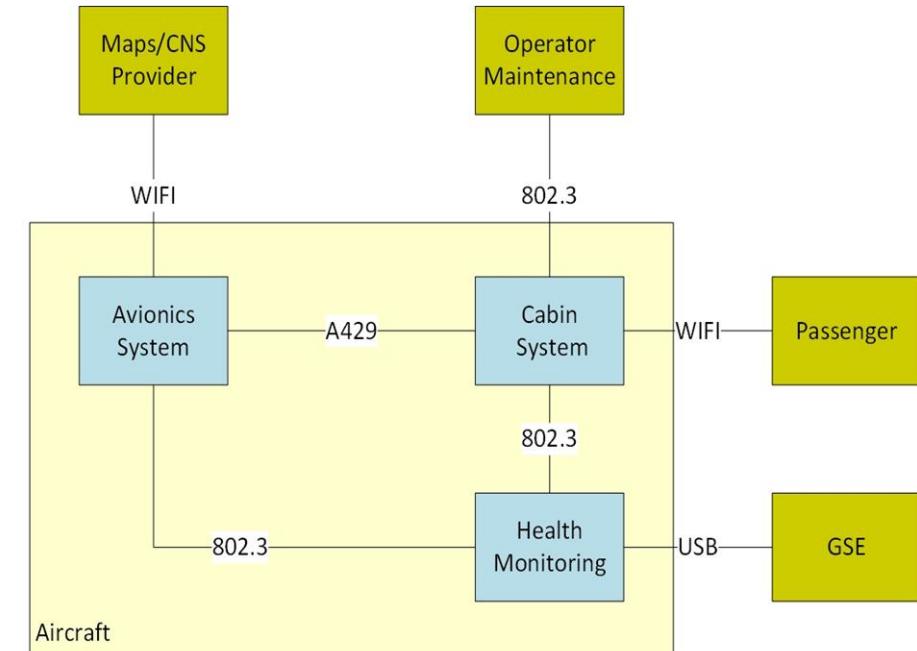
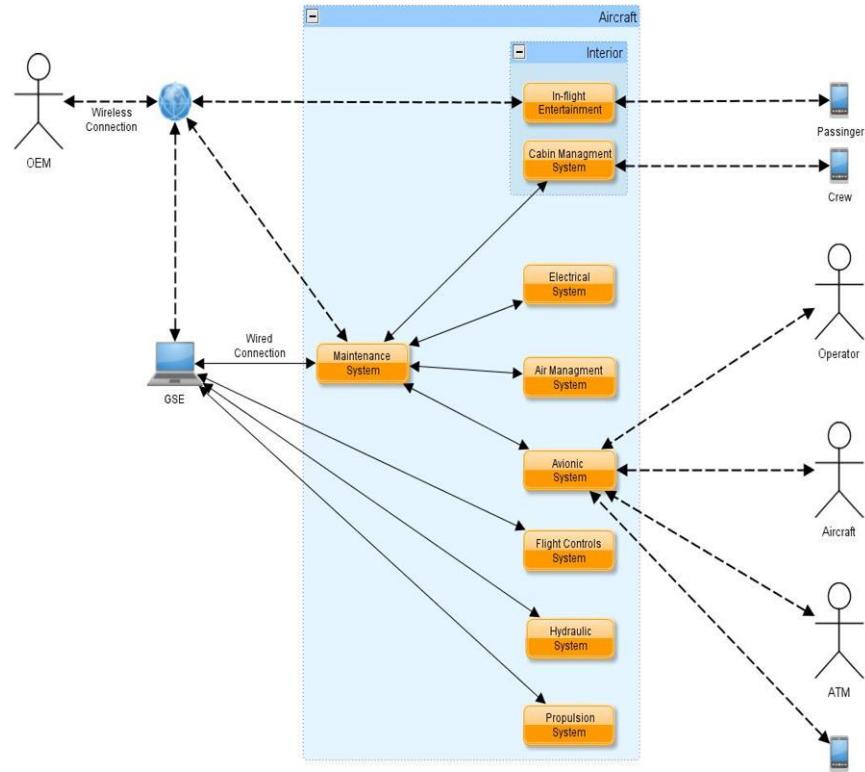


Figure 3-1 : Security Scope

Source: RTCA DO-326A



Step 2: Security Scope samples



Threat Model is a way to define security scope

Source: RTCA DO-326A



Step 3: Security Risk Assessment



- Evaluate security risk of a system from unauthorized interaction
 - Level of threat of a threat scenario and
 - Severity of the impact of the threat condition

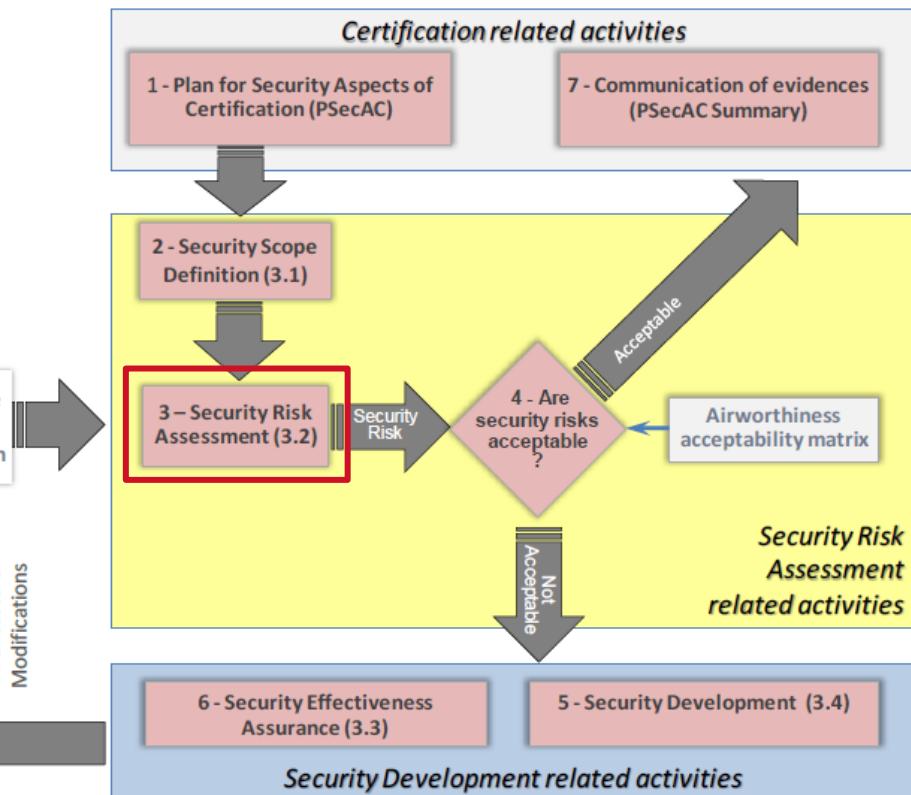


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



Step 3: Security Risk Assessment



- Performed at 2 levels
 - Aircraft
 - System
- Performed at 2 phases
 - Design Phase (PASRA/PSSRA)
 - Threat Condition identification
 - Threat Scenario identification
 - Security Measure characterization
 - Level of threat evaluation
 - Verification Phase (ASRA/SSRA)

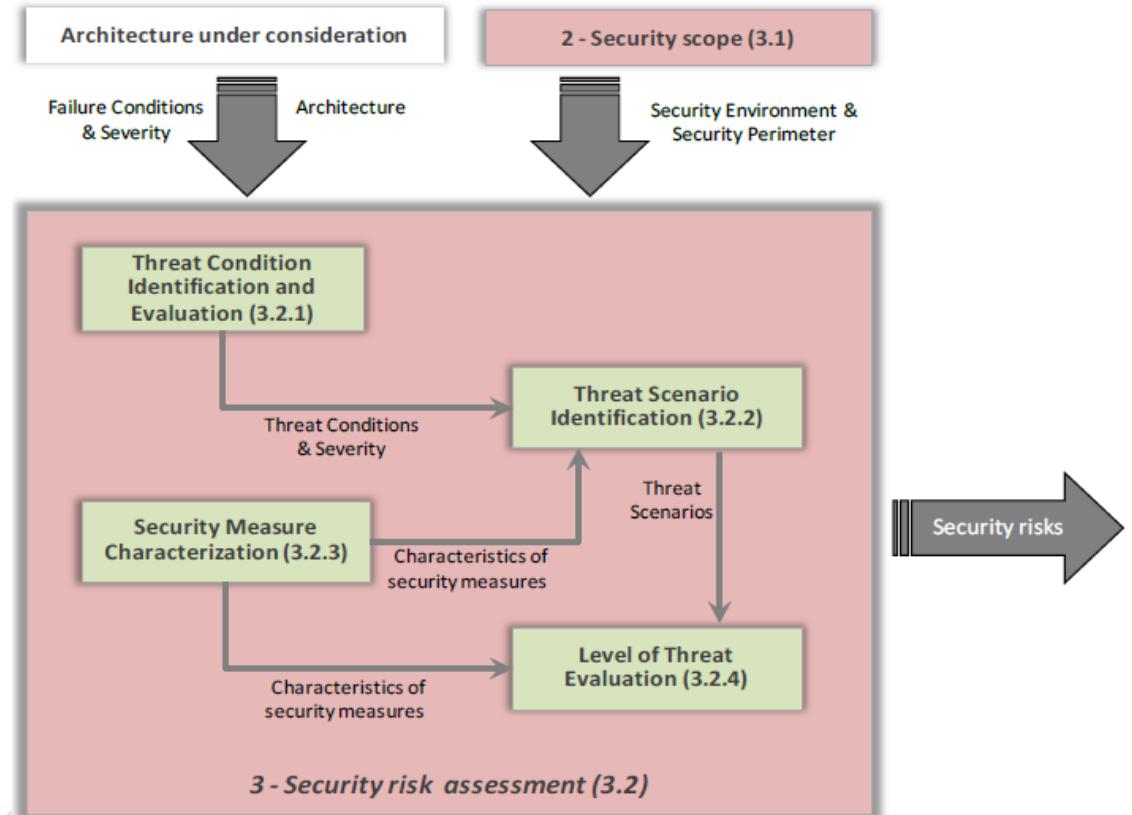


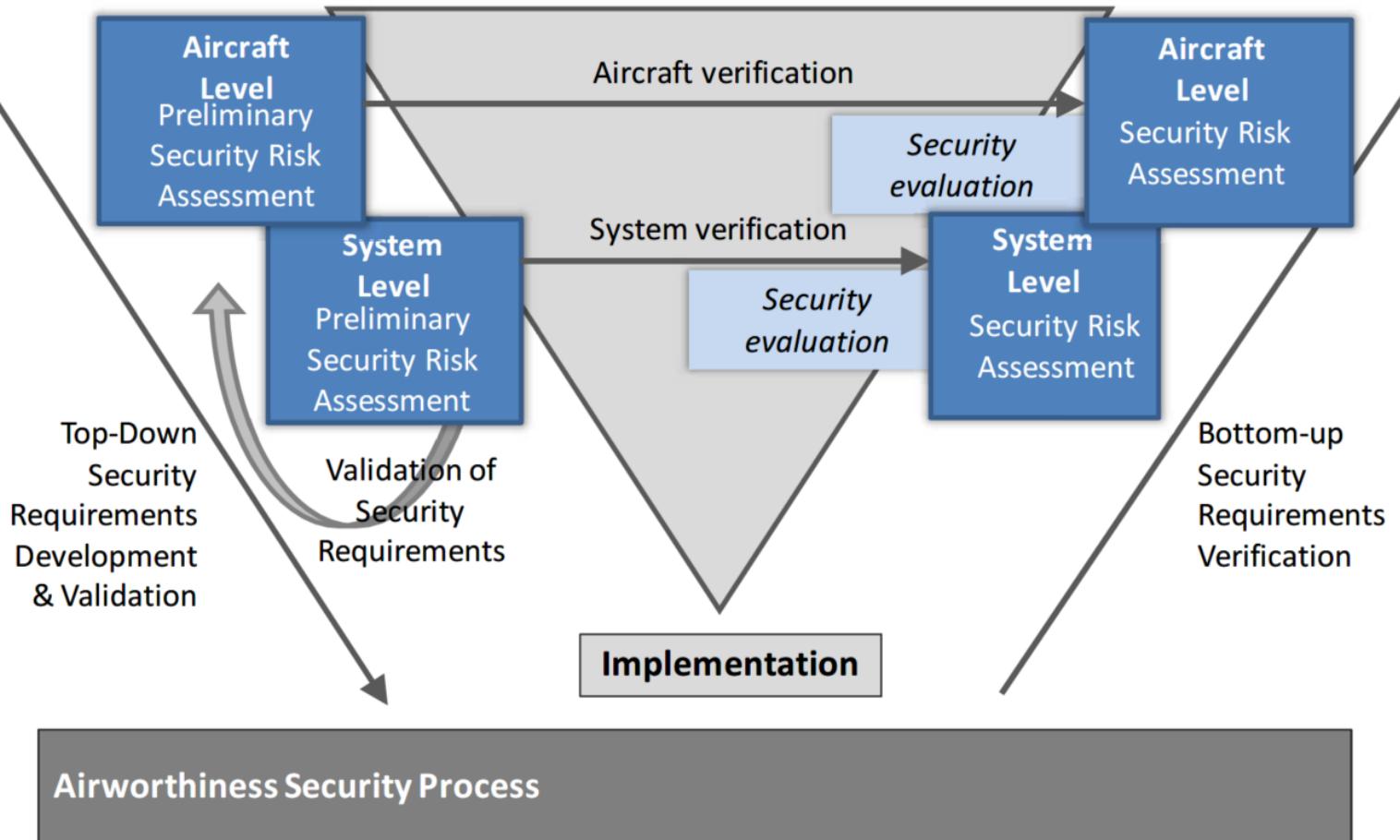
Figure 3-2 : Security Risk Assessment

Source: RTCA DO-326A



| 06-13

Step 3: Security Risk Assessment



**Figure 2-2 : Security Risk Assessment Related Activities in the development process
V-model**

Source: RTCA DO-326A



STEP 3: Security Risk Assessment

THREAT CONDITION IDENTIFICATION

- Identification of the **Threat Conditions (TC)**
 - FHA and Failure Condition (FC) are an input for Security Risk Assessment (Security for Safety)
 - TC are based on FC + TC considering total loss of asset Security attributes
 - Architecture driven
 - End safety effects that can be achieved through an attack (Threat Scenario)
 - can be triggered by one or more Threat Scenarios

Source: Diagram from EUROCAE ED-203A

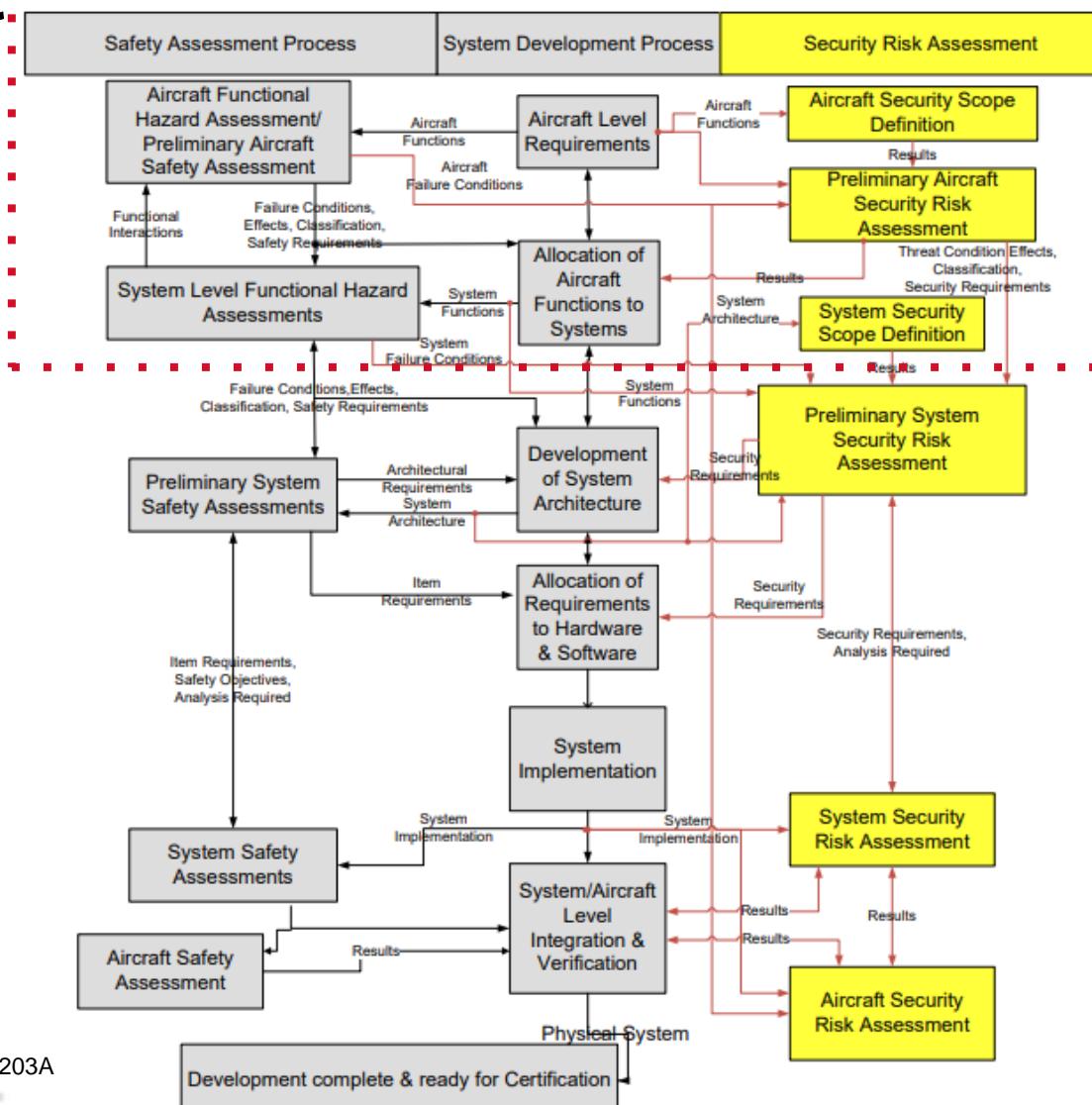


FIGURE 2-3: AWSP AS PART OF AIRCRAFT CERTIFICATION PROCESS

Step3: Security Risk Assessment

Security Assurance Level (SAL)



- Examples of Failure condition issued from FHA
 - FC.1.Loss of acclimatized environment for crew and passengers – Minor
 - FC.3.Loss of pressurization for crew and passengers – Catastrophic
 - FC.4.Loss of aircraft structural integrity – Catastrophic
 - FC.5.Loss of AMS Status Information – Major
- Examples of System asset interfaces:
 - SI.1 Physical interface to ARINC 664 switch
 - SI.2 Logical interface to Avionic system
 - SI.3 Logical interface to Bleed system
- Example of Threat conditions:
 - TC.8 is corresponding to a “Loss of Availability” of the SI.3 which bring a “Loss of pressure control” from the “Flight crew” and “Occupant” and can have a Catastrophic impact.**

TC	Asset	Attribute (CIA)	Description	FP	Effect on			Severity	Failure Cond.
					Aircraft	Flight crew	Occupant		
TC.1	AF.1	Loss of Availability	Just repeats FC.1	None	Unpleasant temperature	Unpleasant temperature		Minor	FC.1
TC.2	FC.2 to FC.7 repetition	FC.2
TC.3	SI.1	n/a	ARINC 664 interface is secure by assumptions AS.1 and AS.2	None	None	None	n/a	None	
TC.4	SI.2	n/a	ARINC 664 data flow is secure by assumptions AS.1 and AS.2	None	None	None	n/a	None	
TC.5	SI.3	Loss of Integrity	Misleading commands to bleed due to pressurization controller corruption	None	Loss of pressure control	Loss of pressure control		Catastrophic	FC.3
TC.6	SI.3	Loss of Confident.	Counterfeit LRU installation.	None	Loss of pressure control	Loss of pressure control		Catastrophic	FC.3
TC.7	SI.3	Loss of Availability	On ground the aircraft will not be dispatched.	None	Loss of pressure control	Loss of pressure control	No safety impact	FC.3	
TC.8	SI.3	Loss of Availability	On air if no messages are provided to crew.	None	Loss of pressure control	Loss of pressure control	Catastrophic	FC.3	

Asset Legend:
 AF = Aircraft Function
 SI = System Interface

Source: EUROCAE ED-203A

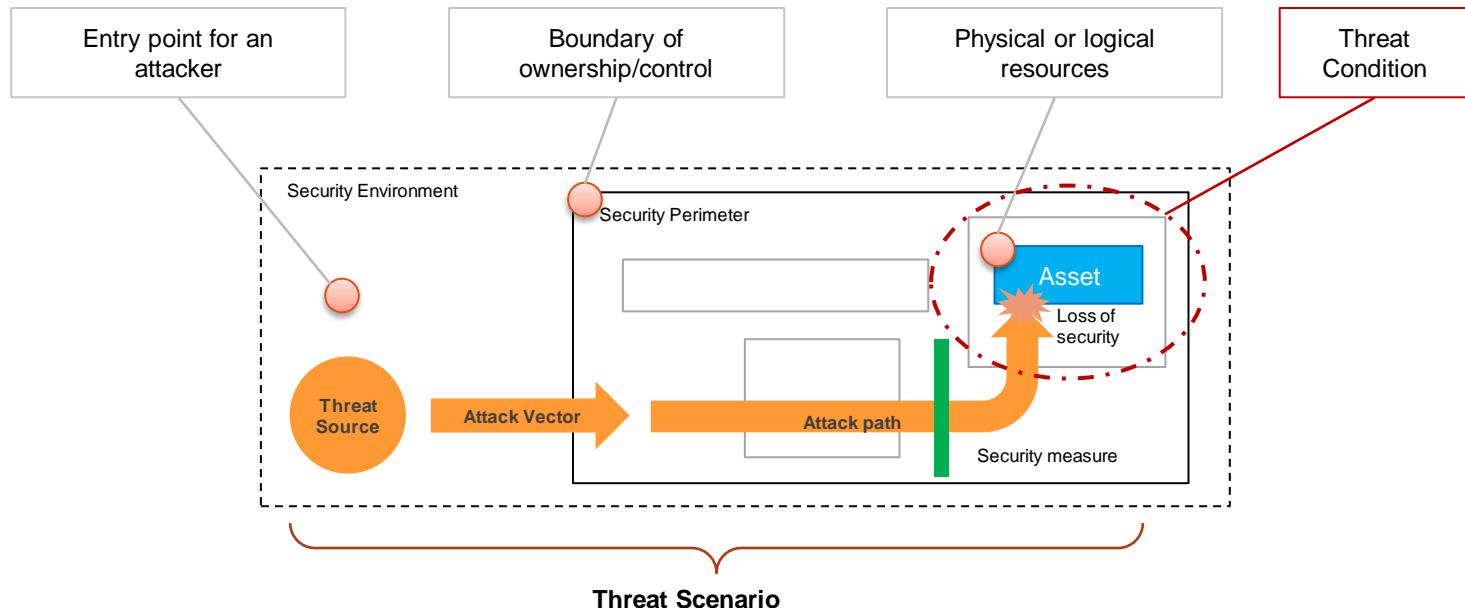


Step 3: Security Risk Assessment

Threat Scenarios



- **Source:** external systems and actors, together with attack vectors
- **Attack Path:** the interfaces exploited, the actions performed, security measures by-passed, vulnerabilities that allow the attack to succeed
- **Security Measures:** technologies or solutions meant to mitigate or prevent an attack
- **Threat Condition:** triggered because of the scenario consequence



Source: EUROCAE ED-203A



Step 3: Security Risk Assessment

Security Assurance Level (SAL)



- The Security Assurance Level (SAL) is
 - a classification for the confidence in the protection and resilience of the aircraft and aircraft systems against attacks.
 - assigned to **security measures** and **assets** that have been identified in the security scope (Function, application, component)
 - assigned early in the process, after threat conditions, severity levels and security measures have been
 - Typically identified in preliminary security risk assessments
 - Assigned to security measure itself and its dependencies (not to the whole partitions)

Security Assurance Level (SAL)	Definition
3	Strongest security assurance for security measures. All security assurance objectives defined in this document are applicable.
2	Advanced security assurance for security measures. SAL 2 is similar to SAL3 on security specific assurance objectives, but significantly less demanding on security development assurance objectives.
1	Minimum security assurance for security measures. Appropriate for additional protection or hardening/resilience.
0	No protective effect. This level is limited to the initial assessment of the protection needs (as detailed in section 2.2) and is applicable for systems and items that have no higher SAL assigned.

Source: EUROCAE ED-203A

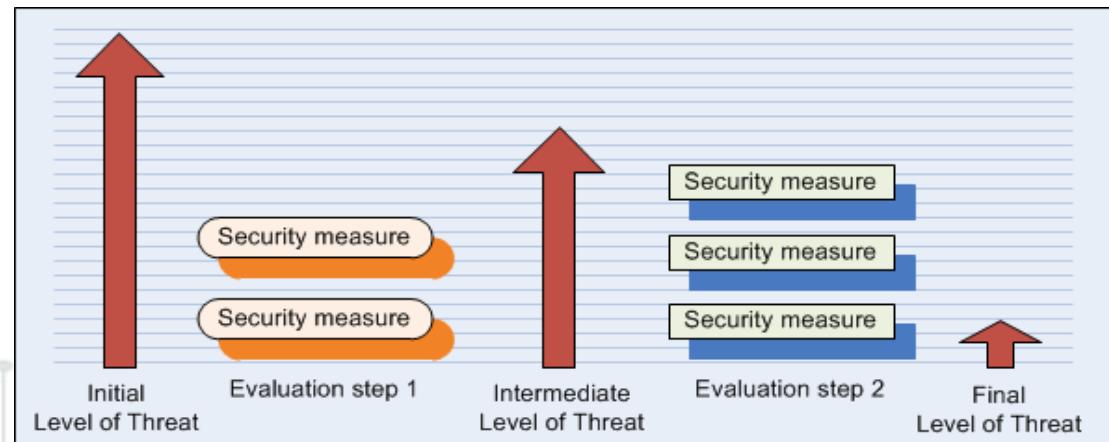
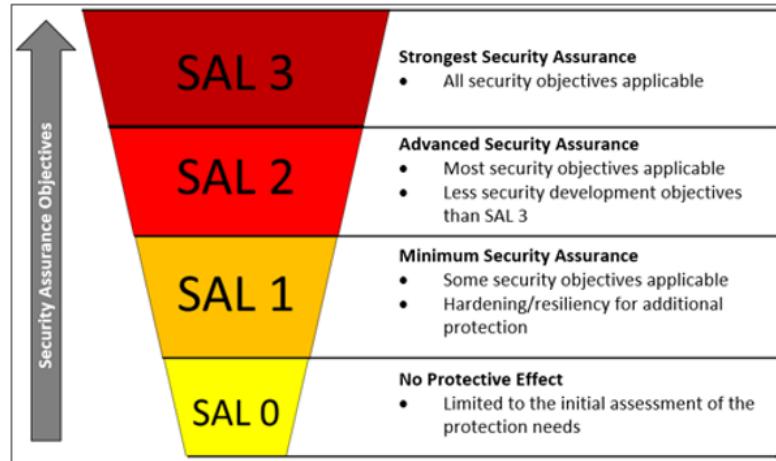


Step 3: Security Risk Assessment

SECURITY ASSURANCE LEVEL (SAL) MAPPING TO THREAT CONDITION SEVERITY



- The Security Assurance Level (SAL) Assignment
 - Security measures are designed to mitigate a cybersecurity exposure or vulnerability.
 - Security measures reduce the level of threat
 - Security measures require a level of security assurance that is specific to what they are designed to protect.



Note: Appendix A is identifying the Security Assurances Objectives for the different SAL.

Threat Condition Effect Severity	Minimum Security Assurance
Catastrophic	SAL 3 + SAL 2
Hazardous	SAL 3
Major	SAL 2
Minor	SAL 0
No Safety Effect	SAL 0

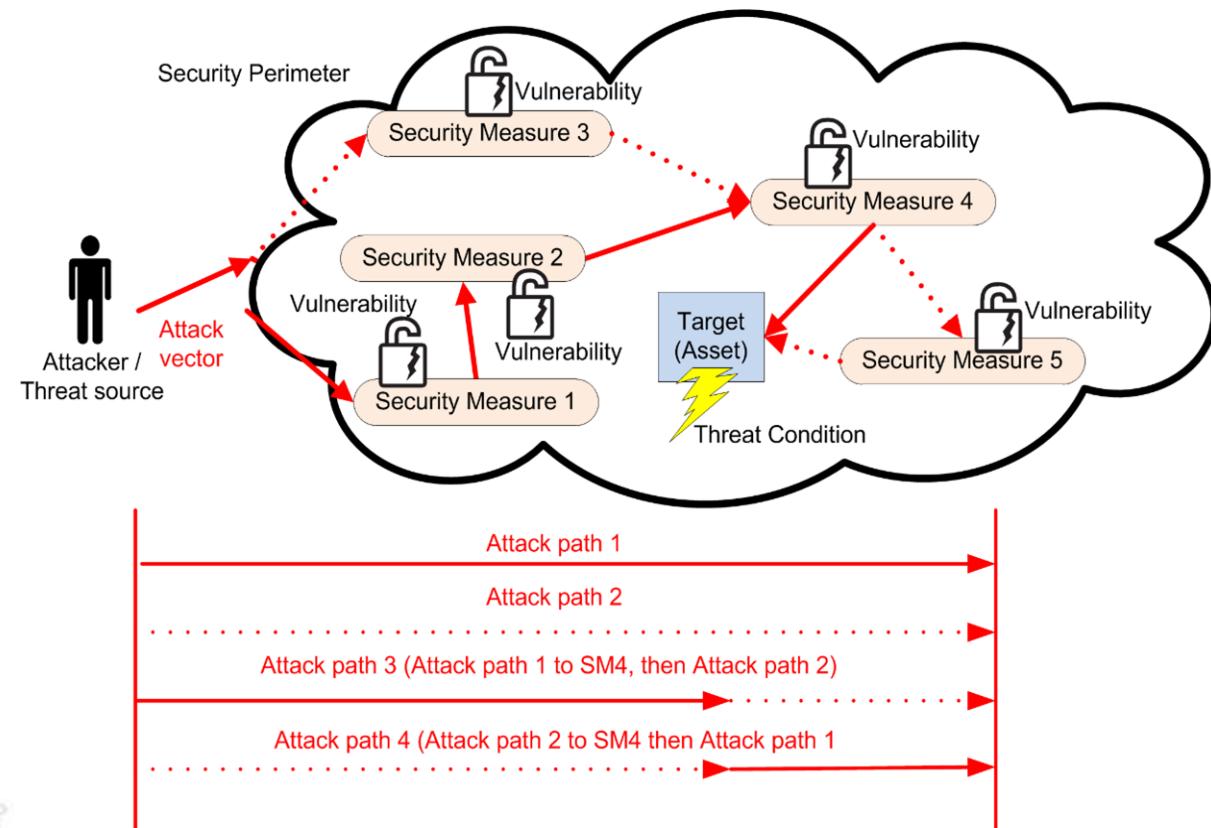
Source: EUROCAE ED-203A



Step 3: Security Risk Assessment

Security Measures

- Security measures objectives are to mitigate/reduce Security risk to an acceptable level
- Security measures can be on various form:
 - Organizational (ex: instructions to crew, maintenance...)
 - Technical (Security function, input validation, security architecture...)
- Initial security measure effectiveness shall be evaluated
- Security measure common mode analysis should be performed assessing the Independence, Diversity and Isolation to prevent bypassing, tampering, single point of failure...
- Security measure must be characterized:
 - Description, type (orga, tech, preventive, detective...), Protected assets, SAL, Threat mitigation...



Source: Diagram from EUROCAE ED-203A



Step 3: Security Risk Assessment

LEVEL OF THREAT

- Level of Threat is the general concept to describe the evaluation of a threat scenario
- Can be express by different approaches
- **Effectiveness approach** (DO-356A appendix E):
 - measures the level of threat by the degree to which the protection succeeds in stopping the attacker.
 - Calculate the strength of the security measures based on 3 principles: Preparation Means, Window of Opportunity, Execution
- **Likelihood approach** (DO-356A appendix F):
 - measures the level of threat by the degree to which the protection fails to stop the attacker.



Level of Threat scale	Effectiveness scale
Very High	None
High	Basic
Moderate	Moderate
Low	High
Extremely Low	Very High

Source: Diagram from EUROCAE ED-203A

Step 3: Security Risk Assessment

Level of Threat evaluation



- Each evaluation starts with the highest possible level of threat (None)
- Each identified security measure increases the protection against a specific attack
- The effectiveness level for a threat scenario is determined by the combined effectiveness points for all identified security measures

TABLE E-1: EFFECTIVENESS LEVEL DEFINITION

Effectiveness	Definition
None	Protection does not exist or is not effective against the specific threat
Basic	Basic protection against intentional unauthorized electronic interactions with elements of the threat scenario assessed.
Moderate	Adequate to protect against a Major safety effect by intentional unauthorized electronic interactions with elements of the threat scenario assessed
High	Adequate to protect against a Hazardous safety effect by intentional unauthorized electronic interactions with elements of the threat scenario assessed
Very High	Adequate to protect against a Catastrophic safety effect by intentional unauthorized electronic interactions with elements of the threat scenario assessed

Effectiveness of protection = Criterion computation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
None	Basic						Moderate						High						Very High											



Step 3: Security Risk Assessment

Level of Threat Evaluation

- Effectiveness of each security measure protecting against an attack in the specific threat scenario is determined by three evaluation criteria:
 - Preparation means: Knowledge of the target and required equipment (ex: on the shelf, need to be build or Bespoke above \$100K)
 - Execution means: Based on skills and equipment
 - Window of opportunity: When the attack can be done
- There is a maximum effectiveness per security measure type:
 - Technical 10,
 - Non-technical 6,
 - Simple Device 18.
- The effectiveness for a given threat scenario is calculated with a sum of security Measure effectiveness and by applying an effectiveness capping (details can be found in DO-356A Appendix E)

Execution means

Expertise Equipment	Layman	Proficient	Expert	Multiple Expert
None / Standard	0	4	6	10
Special COTS	4	4	6	10
Special	n/a	6	8	12
Bespoke	n/a	n/a	10	12

Preparation means

Knowledge Equipment	None / Public Information and no preparation time	Uncontrolled Information and no significant preparation time	Insider Knowledge or Significant preparation time
None / Standard	0	2	6
Special COTS	0	2	6
Special	n/a	4	6
Bespoke	n/a	5	6

Window of opportunity

Effect	Description
0	The attack can be carried out at any time.
1	The attack can be carried out during regular cruise flight.
2	The attack vector is available while the aircraft is on the ground.
3	Maximum effectiveness for mandatory operational procedures limiting the window of opportunity.
6	The attack vector is only available in a restricted time phase, e.g. on the ground in maintenance mode.
8	The attack can only be carried out during a very restricted time slot independent from the flight phase (e.g. during system reboot).

Source: RTCA DO-356A



Step 4: Risk Acceptability

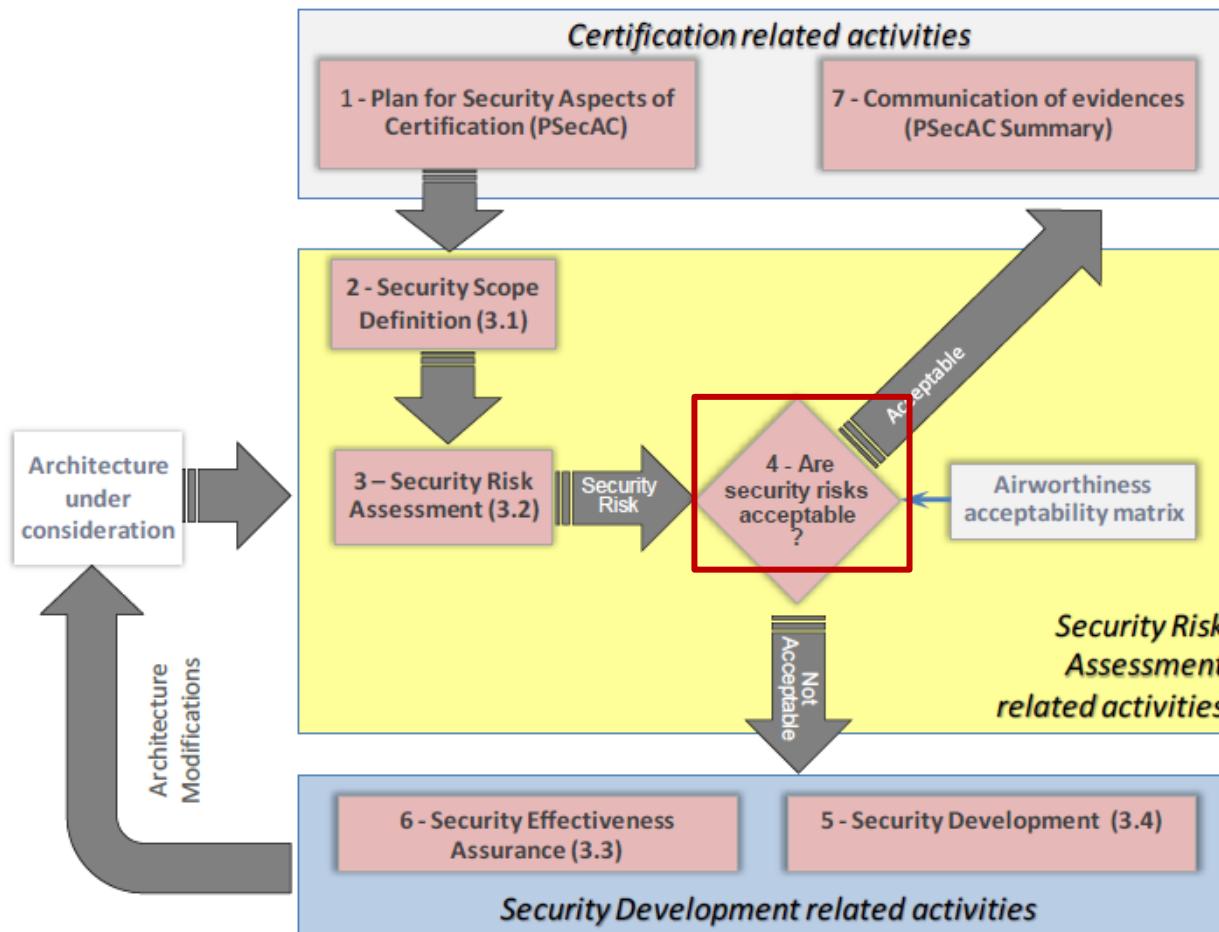


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



Step 4: Risk Acceptability



- The security effectiveness will determine the Level of Threat
- If the risk is unacceptable following the Level of Threat / Severity:
 - Need to reduce the severity of the threat condition
 - Or to reduce the level of threat by adding Security Measure

Level of Threat	Severity of the threat condition				
	No Safety Effect	Minor	Major	Hazardous	Catastrophic
Very High	Acceptable	Acceptable	Unacceptable	Unacceptable	Unacceptable
High	Acceptable	Acceptable	Unacceptable	Unacceptable	Unacceptable
Moderate	Acceptable	Acceptable	Acceptable	Unacceptable	Unacceptable
Low	Acceptable	Acceptable	Acceptable	Acceptable	Unacceptable
Extremely Low	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable*

Source: RTCA DO-356A



Step 5: Security Development



- Define the security architecture necessary to reduce unacceptable risks
- Define security measures of the security architecture
- Establish security guidance for correct integration, operation and maintenance for each aircraft system.

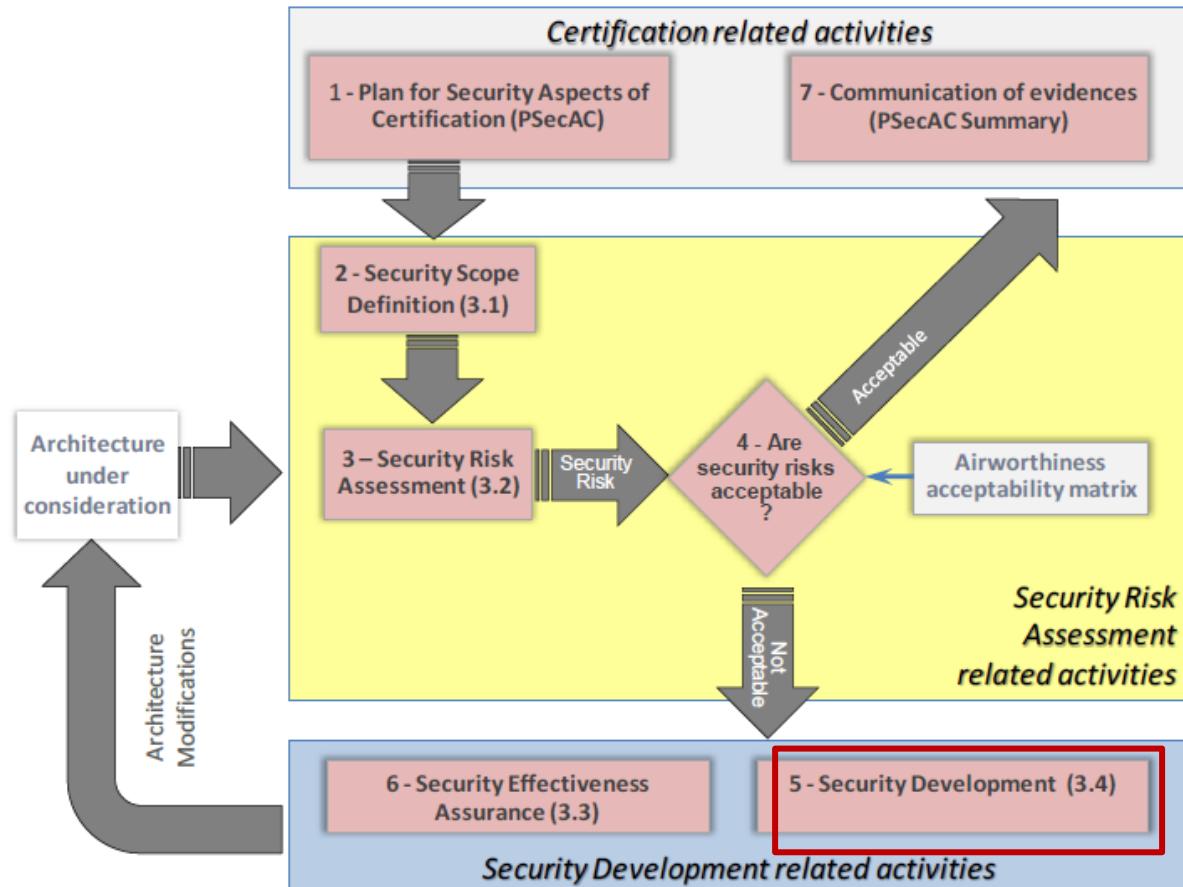


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



Step 6: Security Effectiveness Assurance



- This activity consists of determining Security Effectiveness Objectives and Requirements.
- Provide a measure of the ability of security measure / architecture to reduce the risks to an acceptable level
 - Security Assurance Objectives
 - Security Verification

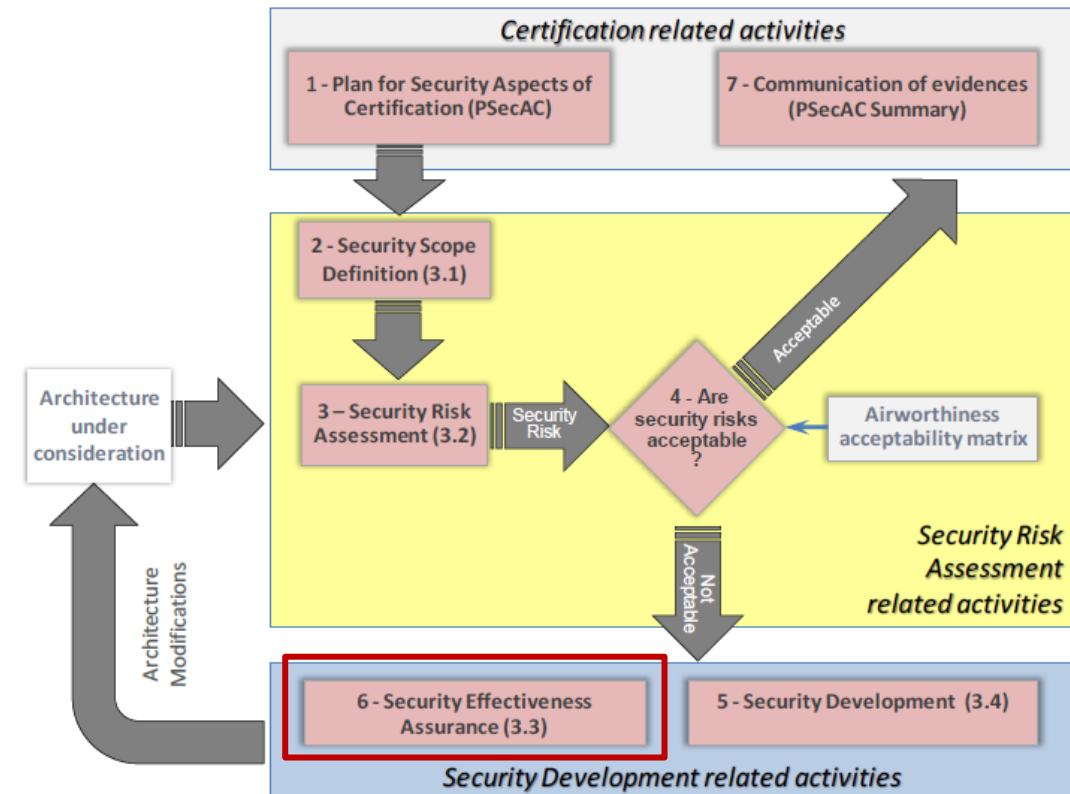


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



Step 6: Security Effectiveness Assurance

SECURITY ASSURANCE OBJECTIVES



Appendix A identify all the Security Objectives:

- To meet a SAL
 - R: Recommended,
 - R*:with independence,
 - A: As negotiated
 - N: Not required
- Scope of applicability
 - AC: Aircraft,
 - S:System,
 - I:Item
- If there are security specific
 - Yes: not covered by another assurance process
 - No: may be also satisfied by the safety process

TABLE A-1: SECURITY SPECIFIC ASSURANCE OBJECTIVES ALLOCATION TABLE

Ref.	Objective	Scope	SAL				Security specific	Document sections	
			3	2	1	0			
Security Risk Assessment Objectives									
O1.1	The security scope is established and validated.	AC, S	R	R	R	R	yes		4.1.1, B.2.1
O1.2	The Threat Condition Identification and Evaluation is complete and validated.	AC, S	R*	R	R	R	yes		4.1.1, B.2.1
O1.3	The Preliminary Aircraft/System Security Risk Assessments and Aircraft/System Security Risk Assessments are performed and consistent with related aircraft/system safety assessments.	AC, S	R*	R	A	N	yes		4.1.1, B.2.1
O1.4	Preliminary Aircraft/System Security Risk Assessment results have been processed to define aircraft/system security architecture and identify the need for security measures.	AC, S	R*	R	A	N	yes		4.1.1, B.2.1
O1.5	Aircraft/System Security Risk Assessment is consistent and complete with respect to security scope, security guidance, security requirements, security verification, security refutation and vulnerability identification.	AC, S	R*	R	A	N	yes		4.1.1, B.2.1
Vulnerability Identification Objectives									
O2.1	Vulnerabilities in security measures and assets (including COTS) are identified and evaluated for their potential impact on safety.	AC, S, I	R	R	A	N	yes		0, B.2.2, B.2.3
O2.2	Vulnerabilities are treated according to their evaluation.	AC, S, I	R	R	A	N	yes		0, B.2.2, B.2.3

Source: Table from EUROCAE ED-203A



Step 6: Security Effectiveness Assurance

SECURITY ASSURANCE OBJECTIVES



- **Security Assurance**

- Cover requirements addressing security measures
- Necessary to assure that security measures perform as intended
- Security measures are free of known and exploitable vulnerabilities, which may be introduced during development.
- The overall security architecture is consistent and security measure cannot by bypass or tamper.

- **Security Assurance Objectives**

- Security Specific Assurance

- Security Risk Assessment Objectives
- Vulnerability Identification Objectives
- Security Refutation Objectives
- Security Deployment Objectives
- Continued Security Effectiveness Objectives

- Security Development Assurance

- Requirements Objectives
- Design Objectives
- Implementation Objectives
- Security Verification Objectives
- Security Planning Objectives
- Security Configuration Management Objectives
- Security Certification Liaison Objectives
- Tool Security Objectives



Step 6: Security Effectiveness Assurance

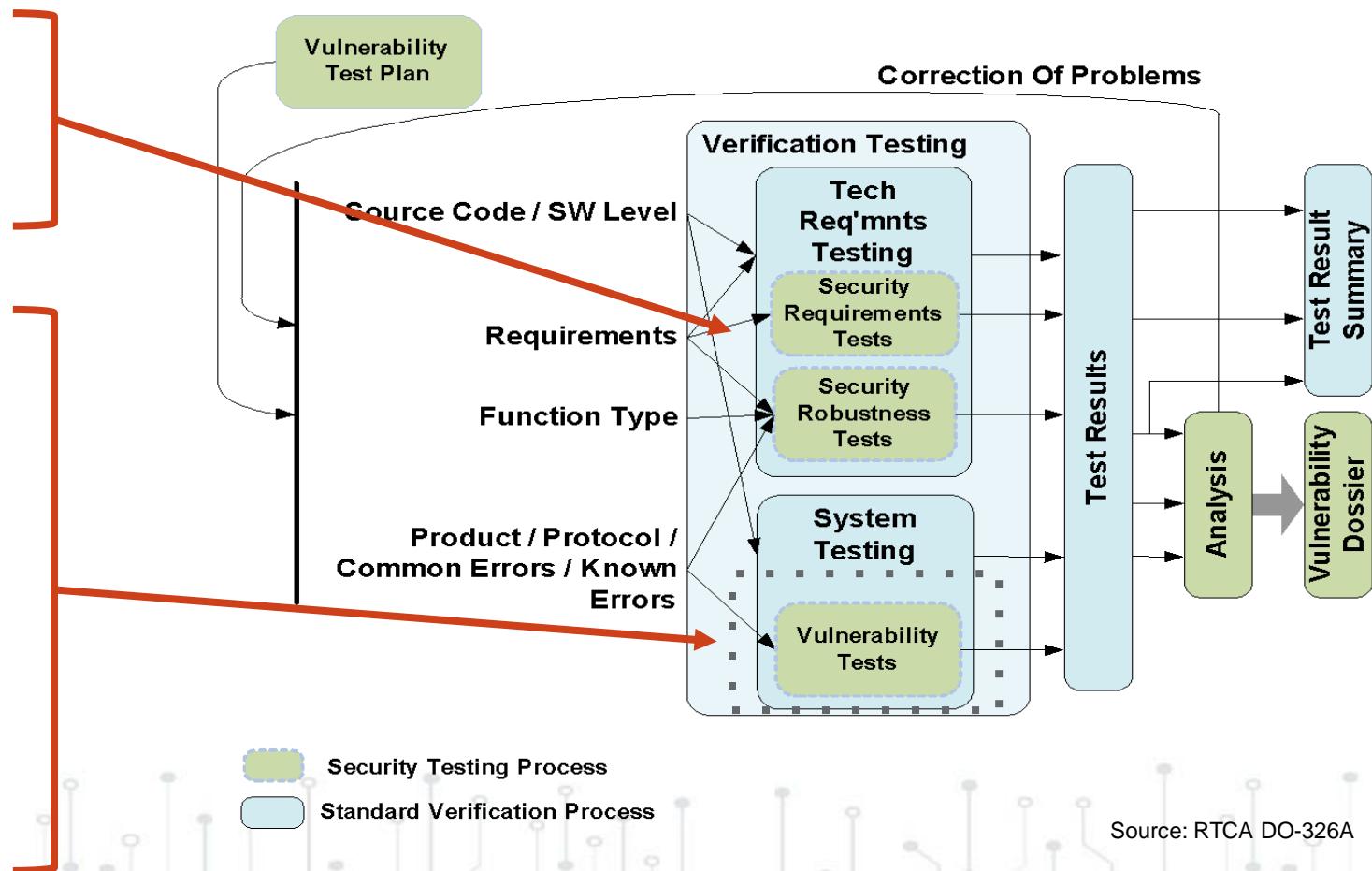
VERIFICATION TESTING



The set of requirements derived from security risk assessment tested by functional security tests for correctness and completeness

Refutation testing:

- The attacks considered in the security risk assessment cannot be captured as a precise set of input conditions
- Refutation assurance activities focus on the role of the attacker
- Refutation can be used to formally demonstrate that an unwanted behavior has been precluded to an acceptable level of confidence
- Example: Fuzzing



Source: RTCA DO-326A



Step 7: PSecAC Summary



- Summarize the certification evidence relative to the security concerns
- Provide evidence on activities covering identified objectives
- Identify and justify (acceptable) deviations to the original plan
- Summary of security analysis, verification and test results

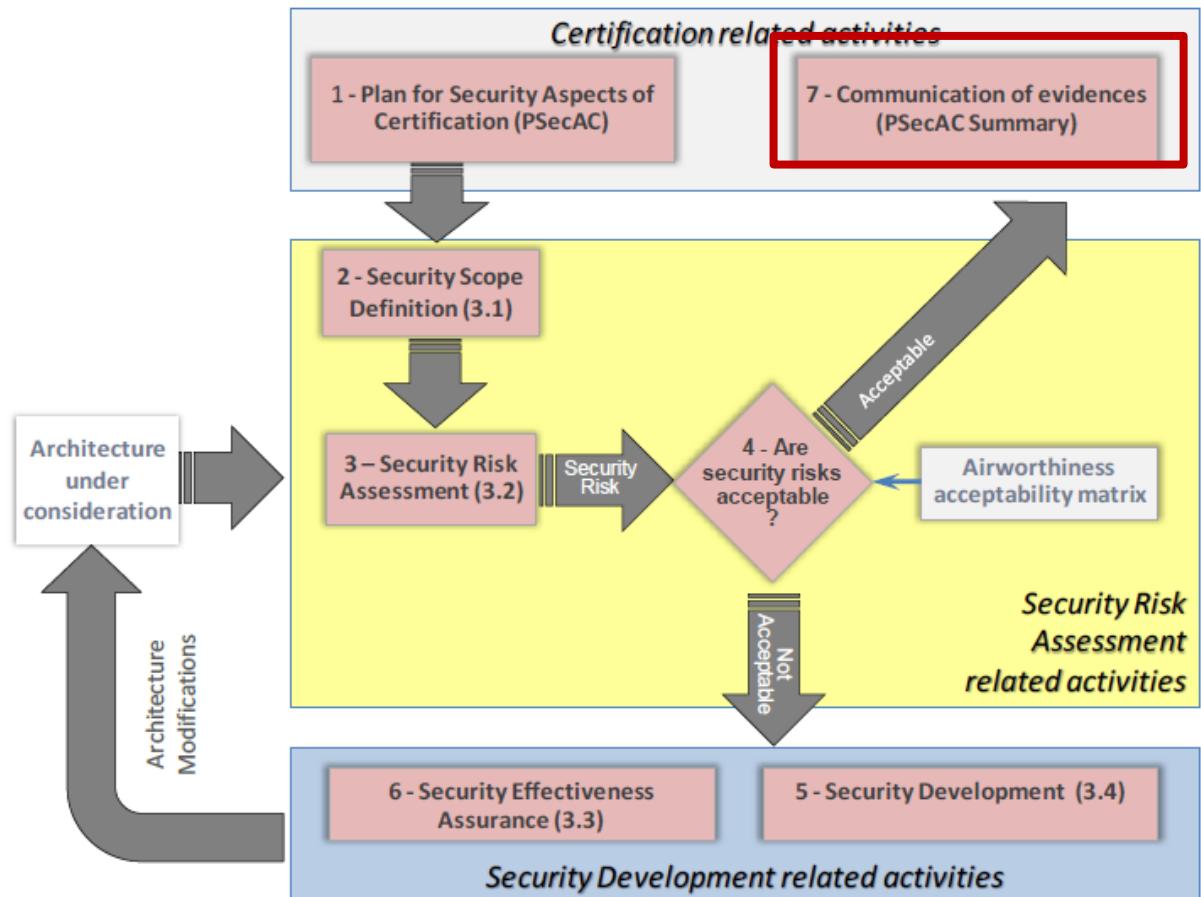


Figure 2-1 : Airworthiness Security Risk Management Framework

Source: RTCA DO-326A



06-31

DO-326A: Acceptable Compliance Demonstration Evidence Considerations



• Aircraft Level

- Plan for Security Aspects of Certification (PSecAC)
- Aircraft Security Scope Definition (ASSD)
- Preliminary Aircraft Security Risk Assessment (PASRA)
- Aircraft Security Risk Assessment (ASRA)
- PSecAC Summary
- Aircraft Security Operator Guidance (ASOG) – dependent on certification project
- Aircraft Security Verification (ASV) – dependent on certification project

• System Level

- PSecAC (tailored to the system)
- System Security Scope Definition (SSSD)
- Preliminary System Security Risk Assessment (PSSRA)
- System Security Risk Assessment (SSRA)
- PSecAC Summary (tailored to the system)
- System Security Operator Guidance (SSOG)
 - dependent on certification project
- System Security Integrator Guidance (SIG)
 - dependent on certification project
- System Security Verification (SSV) – dependent on certification project

Compliance activities depend on the SAL

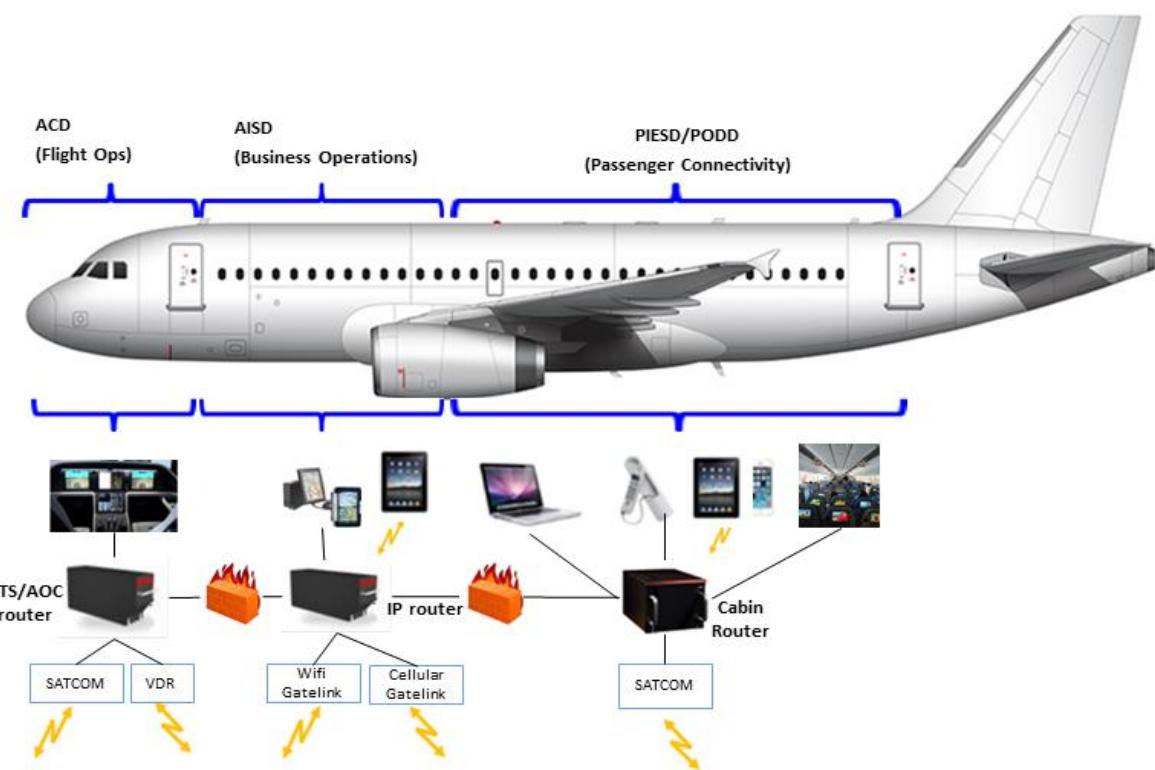
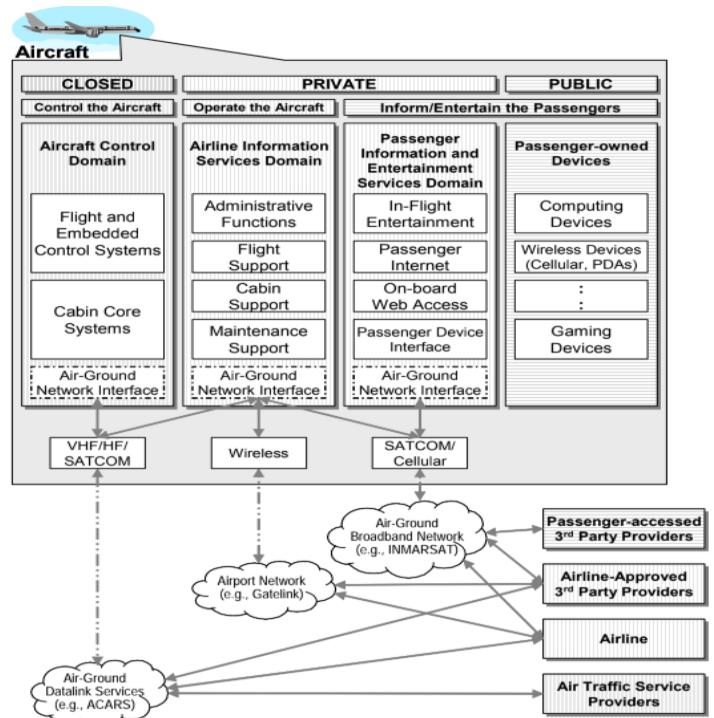


Other Noteworthy Standards



- ARINC 811: Commercial Aircraft Information Security Concepts of Operation and Process Framework
- Released in 2005, currently being updated
- Provides a common understanding of information security concepts as they relate to airborne networks.
- Provides an aircraft information security process framework relating to airline operational needs.
- Provides a general threats list

- T.ACCESS
- T.DENIAL
- T.ENTRY
-



Other Noteworthy Standards



ARINC Report 827: *Electronic Distribution of Software by Crate (EDS Crate)*

- This EDS standard is intended to promote consistent, secure distribution of EDS (A/C sw part) content to any appropriate destination.

ARINC Report 835: *Guidance for Security of Loadable Software Parts Using Digital Signatures*

- Define Airbus and Boeing loadable software digital signature methods

ARINC Report 842: *Guidance for Usage of Digital Certificates* Cybersecurity process overview

ARINC Report 852: *Guidance for Security Event Logging in an IP Environment*

- Provides the guidance for IP-based onboard networks and systems in the following aircraft domains

ARINC Project Paper 858: *Internet Protocol Suite (IPS) for Aeronautical Safety Services*

- Will define Air-Ground IPS Security measures

ATA Spec 42: *Aviation Industry Standards for Digital Information Security*

- Standardized methods to achieve an appropriate security level for the applications that rely on digital identities



Other Noteworthy Standards



ARINC Specification 664 Part 5: Network Domain Characteristics and Interconnections

- Defines aircraft domains similarly as ARINC 811

ARINC Characteristic 771: Low-Earth Orbiting Aviation Satellite Communication System

- Security objectives, architecture, security analysis, and multiple transceiver design for physical separation

ARINC Characteristic 781: Aviation Satellite Communication Systems

- Supplement 7 include Inmarsat SBB Security consideration. Define an Air-Ground VPN

ARINC Specification 822A: On-Ground Aircraft Wireless Communication

- Provides cybersecurity guidelines for the use of commercial data links connected while the aircraft is located on the ground

ARINC Specification 823: DataLink Security, Part 1, ACARS Message Security

- Provide an industry standard for ACARS Message Security (never been deployed)



Questions?



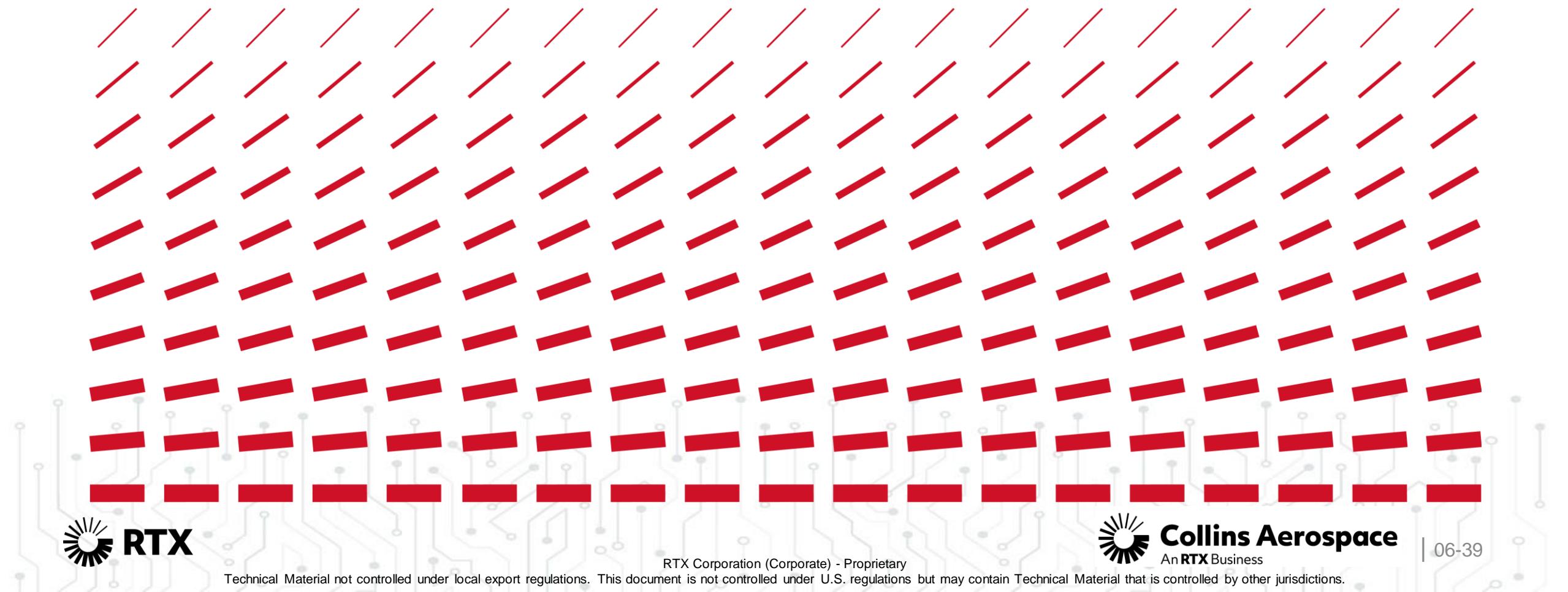
RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.



06-38

Thank you.



RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.

Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





Collins Aerospace
An RTX Business



TGE CYBEREMBSEC

Embedded Systems Security

Module 07

Aviation Threat Example

Instructor: Jason Schoenbeck

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

Agenda

COLLINS PRODUCT CYBERSECURITY TRAINING



- Module 1: Cybersecurity General Overview
- Module 2: Aviation Product Cybersecurity Challenges
- Module 3: Collins Product Cybersecurity: Policy Flow down
- Module 4: Collins Secure System Development Life Cycle
- Module 5: Cybersecurity Certification and Standards Considerations
- Module 6: DO-326A Airworthiness Certification Process
- **Module 7: Aviation Threat Example**



Section 7: Aviation Threat Example



- **Black Hat Presentation** - “Last Call for SATCOM Security”
 - Scenario description
- **Attack Tactics**
 - Reconnaissance
 - Resource Development
 - Initial Access & Execution
 - Post-Exploitation
- **Open Discussion**



Aviation Threat Example

2018 BLACK HAT SECURITY CONFERENCE



“Last Call for SATCOM Security”

- IOActive - Cybersecurity consultant firm
- Research performed by Ruben Santamarta
 - Principal consultant at IOActive
 - Focused on:
 - Penetration testing
 - Identifying system vulnerabilities
 - Researching cutting-edge technologies



YouTube Presentation: [Link](#)

Presentation Slides: [Link](#)

Presentation White Paper: [Link](#)



Attacks Tactics



MITRE ATT&CK® is a knowledge base of adversary tactics and techniques [Link](#)

1. Reconnaissance / Information Gathering:

- Passive: using network sniffer, public information...
- Active: using network scanner, vulnerability scanner, Phishing email, social engineering...

2. Resource Development:

- Involves creating/purchasing/compromising/stealing resources to enable the operation.

3. Initial Access:

- Gain initial presence within a network/target.

4. Execution:

- Run malicious code on targeted system to exploit system

5. Post-Exploitation:

- MITRE ATT&CK Tactics of: Persistence, Exfiltration, Lateral Movement, Impact, etc



Reconnaissance

ON AIRCRAFT NETWORK TRAFFIC CAPTURE

On a Norwegian Air flight, Wireshark used to capture In-Flight WiFi:

- Passengers IPs appeared as routable IPs
- Observed network scans coming from external random hosts

Continued throughout flight to passively monitor network traffic

- Mapped what devices exist on the internal network

IP assigned to passenger's are public (Routable)

3 7.281568	KontronA_26:a9:65	Broadcast	ARP	Who has 128.65.86.137? Tell 128.65.86.130
4 7.938288	KontronA_26:a9:65	Broadcast	ARP	Who has 10.178.27.43? Tell 10.142.8.217
5 8.285073	KontronA_26:a9:65	Broadcast	ARP	Who has 128.65.86.137? Tell 128.65.86.130
NetRange:		128.65.0.0 - 128.65.255.255		
CIDR:		128.65.0.0/16		
NetName:		RIPER-ERA-126-65-0-0		
inetnum:		128.65.80.0 - 128.65.95.255		
netname:		ROW44		
descr:		Hughes Network Systems GmbH		
country:		DE		

260085 1063.207963 41.235.74.58 128.65.86.156 TCP 37065 -> 23 [SYN]
Frame 260085: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: KontronA_26:a9:65 (00:10:13:26:a9:65), Dst: (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 41.235.74.58, Dst: 128.65.86.156
Transmission Control Protocol, Src Port: 37065 (37065), Dst Port: 23 (23), Seq: 0, Len: 0
Source Port: 37065 Destination Port: 23 [Stream index: 17833] [TCP Segment Len: 0] Sequence number: 0 (relative sequence number) Acknowledgment number: 0 Header Length: 24 bytes Flags: 0x002 (SYN)

Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf

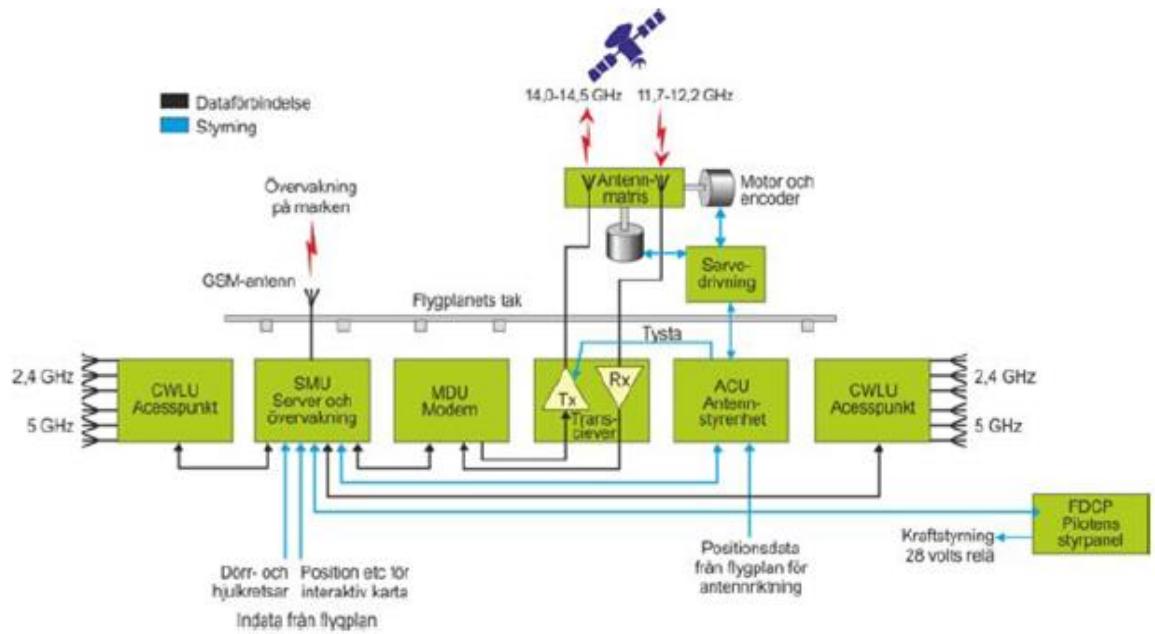


Reconnaissance

INFORMATION GATHERING: PUBLICLY ACCESSIBLE SOURCES

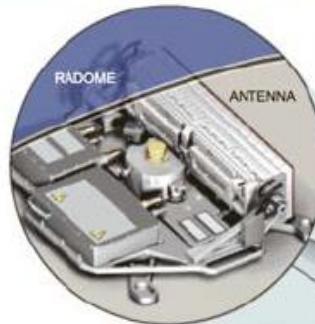


Row 44 Completes Installation Of In-Flight Entertainment Solution On 60 Of Norwegian Air Shuttle's Boeing 737-800 Aircraft



Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf

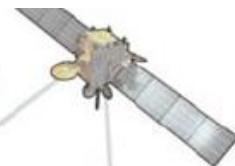
Aircraft using in-flight broadband services, like the one developed by Row 44, shown below, allow passengers to stay connected to the Internet while in the air. Here's how it works.



2. External Antenna
Mounted atop the aircraft in an aerodynamic radome, it sends and receives broadband signals, linking with an orbiting satellite.



3. Satellite
A constellation of satellites in geostationary orbit receive and transmit data between the aircraft and the ground, allowing for continuous communications.



1. Inside the Plane
Wireless LANs at each end of the plane communicate with passengers' wireless devices. Data is sent through cabin-mounted electronics boxes (inset below), then out to an external antenna.



4. Ground station
Communicates with the satellites, connecting the aircraft to the Internet.



Wireless LAN Unit
Sends and receives data from passengers' wireless devices.

To Internet

MDU: Modem Data Unit

ACU: Antenna Control Unit

HPT: High Power Transceiver

SMU: Server Management Unit

SAA: Satellite Antenna Assembly



Reconnaissance

INFORMATION GATHERING: PUBLICLY ACCESSIBLE SOURCES

Shodan (search engine for Internet-connected devices) found 3 different airlines using SATCOM Modems

[Source: Santamarta-Last-Call-For-Satcom-Security.pdf](#)

Result	IP Address	Description
1	67.143.123.121	VxWorks (VxWorks5.4.2) FTP server ready host#714300121123.dirxwvay.com Hughes Network Systems Added on 2018-03-02 08:40:23 GMT United States Details
2	67.143.120.173	VxWorks (VxWorks5.4.2) FTP server ready host#714300121123.dirxwvay.com Hughes Network Systems Added on 2018-03-02 07:42:28 GMT United States Details
3	67.143.120.22	VxWorks (VxWorks5.4.2) FTP server ready host#714300221120.dirxwvay.com Hughes Network Systems Added on 2018-03-02 09:32:29 GMT United States Details

YouTube had videos covering airline installation and flight test of SATCOM system

[Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf](#)



Resource Development

REVERSE ENGINEERING FALBACK UPDATER



FallBack Updater Procedures

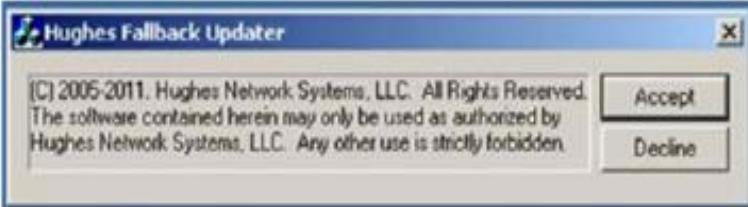
Repeat this procedure for each unit installed:

- Connect the PC and HX200/HX260 via the LAN (LAN1 connector on the HX200/HX260).
- Open the Windows Explorer and navigate to the default directory where the files were unzipped. The latest version is found on Portal and loaded to the installers PC.
- Double-click on HUGHES_Updater.

Results

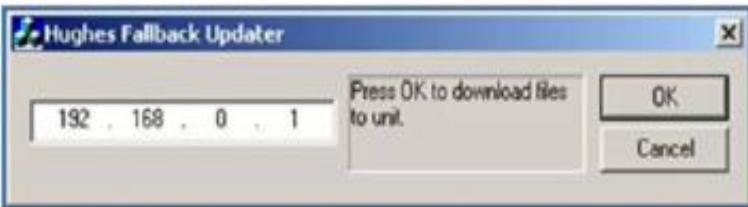
The following messages will be generated if the fallback update operation is successful.

STEP 1



Click on the Accept button to acknowledge the restricted use condition.

STEP 2



```

loc_4015DB:          ; "brighton"
mov    edi, offset abrighton
or     ecx, 0FFFFFFFh
xor    eax, eax
push   offset aPassword ; "Password: "
repne scash
not   ecx

```

Reverse engineering of the Fallback
Updater uncovers credentials

```

loc_401677:          ; "swordfish"
mov    edi, offset aswordfish
or     ecx, 0FFFFFFFh
xor    eax, eax
push   offset Str1    ; "-> "
repne scash
not   ecx

```

Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf



Initial Access & Execution

GETTING ACCESS TO THE MDU AT 30K FT



Get a command-line shell on in-flight aircraft from the internet

```
Trying 128.65.92.65...
Connected to 128.65.92.65.
Escape character is '^]'.
VxWorks login: brighton
Password:
->
```

[Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf](#)

Connecting to In-Flight aircraft from the ground and through the Internet, enables access to:

- VxWorks Shell (Telnet)
- Web User Interface (often just appeared as wrapper for VxWorks Shell)
- FTP Server (used same credentials as VxWorks Shell)
- Onboard Firmware (using FTP)
- Documentation



Post-Exploitation

2 Envisioned Attack Scenarios

- Turn the SATCOM system (Modem Data Unit (MDU), High Power Transceiver (HPT) and Antenna Control Unit(ACU) into a malicious intentional radiator by controlling the power and transmission.
- Eavesdrop and tamper with crew and passenger's communications



Additional attacks through Web UI (no login required)

- Monitor status of the unit
- Potentially initiate a Denial of Service (reboot of terminal)

S/N: 2251594
Main.bin: [6.9.0.51]
Fallback.bin: [6.9.0.20_PtD]

Advanced Configuration and Statistics

Enable Auto Refresh: Interval (sec): Submit

General Stats

Network Time: THU NOV 23 23:09:03 2017

TFC Gem Statistics

TFC Start Time	THU NOV 23 23:19:13 2017
TFC IP Address	128.65.86.65
Terminal Type	KX200M
Build Number	6.9.0.51
TFC Status	BURNING

-- Object Allocation Stats -----

Peak Upstream Connns.....	10
Curr Browser Connns.....	0
Max-Reached Upstream Connns.....	0
Peak Browser Connns(Startup).....	30
Failed Big UME Buffers.....	0
Curr Big UME Buffers.....	0
Peak Browser Connns(After Clear).....	30
Peak Big UME Buffers.....	0
Max-Reached Browser Connns.....	0
Persistent Conn Agents.....	120
Persistent Stop Threshhold Hst.....	0
Error Stats	

-- Common HPP State -----

Prefetch Coverage %.....	1
Prefetch Efficiency %.....	16
Prefetch Header Efficiency %.....	11
Prefetch total efficiency %.....	13
Prefetched Objects.....	6
Prefetched Bdr Objects.....	9
House Prefetch Obj Over See TMP 0	
Dowstream Rocket Heads Failed.....	0
LAS send Blocked %.....	0
Prefetch FAILED URLs.....	0
Prefetch Objects Purged.....	0

S/N: 2251594
Main.bin: [6.9.0.51]
Fallback.bin: [6.9.0.20_PtD]

Advanced Configuration and Statistics

Enable Auto Refresh: Interval (sec): Submit

IP Address Stats

Network Time: THU NOV 23 23:09:03 2017

INGRESS Filter IP Address based statistics

Cumulative Statistics

Vlan_id	IP	VLAN Detected TCP Pkts/Bytes	UDP Pkts/Bytes	Web Pkts/Bytes	ICMP Pkts/Bytes	IGMP Pkts/Bytes
0	10.7.0.10	3275/291136	0/0	0/0	0/0	0/0
0	192.168.0.2	324749/18278832	0/0	0/0	4090/343550	0/0
0	0.0.0.0	0/0	9/3276	0/0	0/0	0/0
0	10.2.0.5	YBS	0/0	9/2952	0/0	0/0
0	10.178.26.16	0/0	144/3306	45/1980	0/0	3/168
0	10.178.26.149	0/0	319/16702	195/8580	0/0	42/2352
0	10.178.26.88	0/0	917/56545	0/0	402/20880	20/2512

Ephemeral Statistics (Cleared every 300 seconds)

Vlan_id	IP	VLAN Detected TCP Pkts/Bytes	UDP Pkts/Bytes	Web Pkts/Bytes	ICMP Pkts/Bytes	IGMP Pkts/Bytes
0	10.7.0.10	67/2680	0/0	0/0	0/0	0/0
0	192.168.0.2	2442/146389	0/0	0/0	40/3366	0/0
0	10.178.26.149	10/898	0/0	0/0	0/0	0/0
0	10.178.26.88	91/4763	0/0	79/3640	0/0	0/0

NOTE: 169.254.xxxx.xxx and 0.0.0.0 addresses correspond to DHCP packets received by the VSAT.

Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf



Post-Exploitation

RESOURCE DEVELOPMENT & INFORMATION GATHERING



- Actual MDU firmware found accessible on FTP server
- Used decompiler to analyze the MDU firmware.
 - Firmware compiled w/ full symbol table intact.
- Multiple sources of publicly available information (HX System User Manual, Hughes Patent, FCC application for authority to operate this system) to determine firmware functionality.
- MDU firmware was verifying a digital signature at runtime, but the security scheme could be bypassed

[Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf](#)



Post-Exploitation

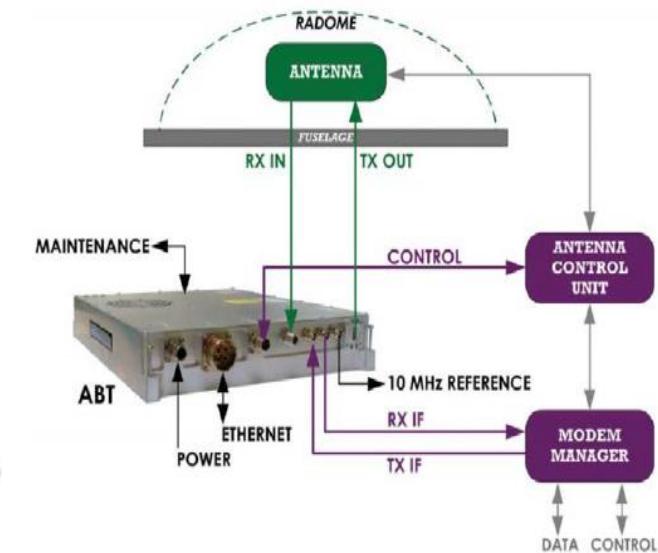
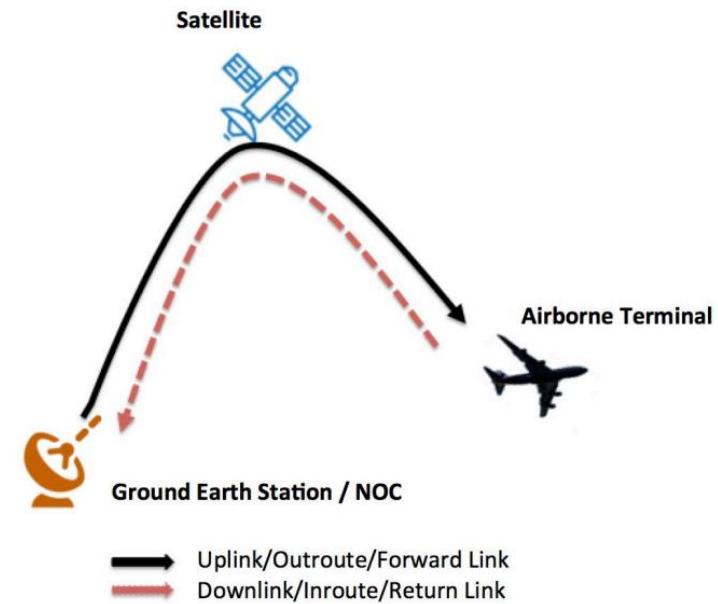
RESOURCE DEVELOPMENT & IMPACT

Turn the SATCOM System into an Intentional Radiator

SATCOM terminals often depend on a Network Operations Center (NOC) to receive instructions for proper operation

- Complex messaging protocol
- Specific checks for NOC oversight could be bypassed in the MDU firmware to make the antenna an intentional radiator

Build malicious firmware (activates the Antenna Control Unit (ACU) command & control, manages the transmission and RF power)



Source: [Santamarta-Last-Call-For-Satcom-Security-wp.pdf](#)



Post-Exploitation

TAMPER & IMPACT



Eavesdrop or tamper with cabin crew or passenger's communication

SATCOM devices often contain data accelerators to improve throughput performance by optimizing link efficiency.

MDU Firmware contained TCP/IP and web browsing acceleration.

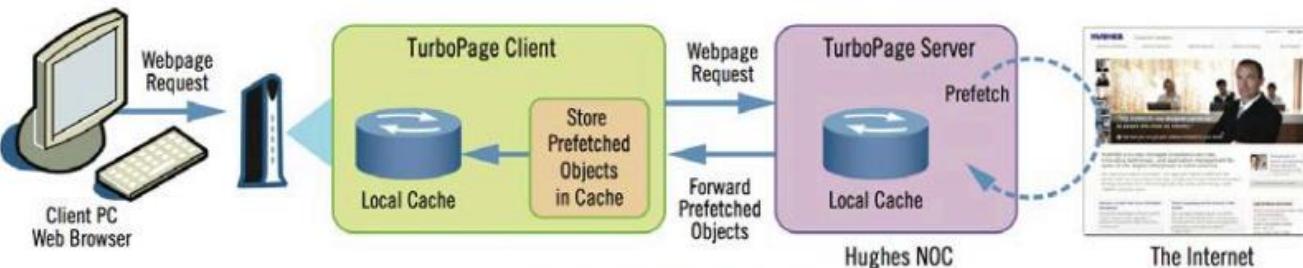


Figure 1. How TurboPage Works

Date acceleration features provide potential vector to eavesdrop or tamper with cabin crew or passenger communications.

- Logic usually involves parsing of web pages and/or TCP/IP packets

[Source: Santamarta-Last-Call-For-Satcom-Security-wp.pdf](#)



Presentation Conclusion



Industry	Security Risk	Flight Safety Risk	RF Risk	Likelihood	Attack Vector
Aviation	Yes	No*	No*	Medium	Remote

*Based on input received from the Aviation industry through the A-ISAC and our own

Industry	Threat
Aviation	<ul style="list-style-type: none">Ability to disrupt, intercept or modify non-safety communications such as In-Flight WiFi *Ability to attack crew and passenger's devicesAbility to manipulate SATCOM antenna positioning and transmissions.

(*) Typically pilot and co-pilot do not use it. In-Flight WiFi is normally used by flight attendants for PAX and PCI transactions.

(*) Configurations may vary the impact.



Open Discussion



Sources of Issues

- Design Decisions
- Software Development Practices
- Publicly Available Information
- Lack of Security Assurance



Open Discussion



Design Issues

- Lack of network segregation of command and control, maintenance/crew and cabin wireless network
- No authentication on Wireless Access point
- Passenger IP addresses Publicly routable
- Non-secure protocols directly exposed from ground (FTP, Telnet, web interface)
- Fallback Updater updates firmware w/o authentication over clear text protocol (Telnet)
- Weak data loading signature mechanism (firmware protection)
- Administrative action directly exposed to the external interface (voluntary backdoor)
- No anti-scan protection from ground

Software Development Issues

- Credential hardcoded and usable from external interface
- Software binary compiled with full symbol table intact
- Weak password – dictionary word

Publicly Available Information

- Technical data publicly available
- YouTube videos of airlines flight testing SATCOM system
- Media publications w/ technical details of airlines receiving the system

Lack of Security Assurance

- Likely no Security Risk Assessment performed
- Secure Coding Guidelines not observed
- Likely no Security verification/refutation test (Fuzzing, penetration testing)



Questions?



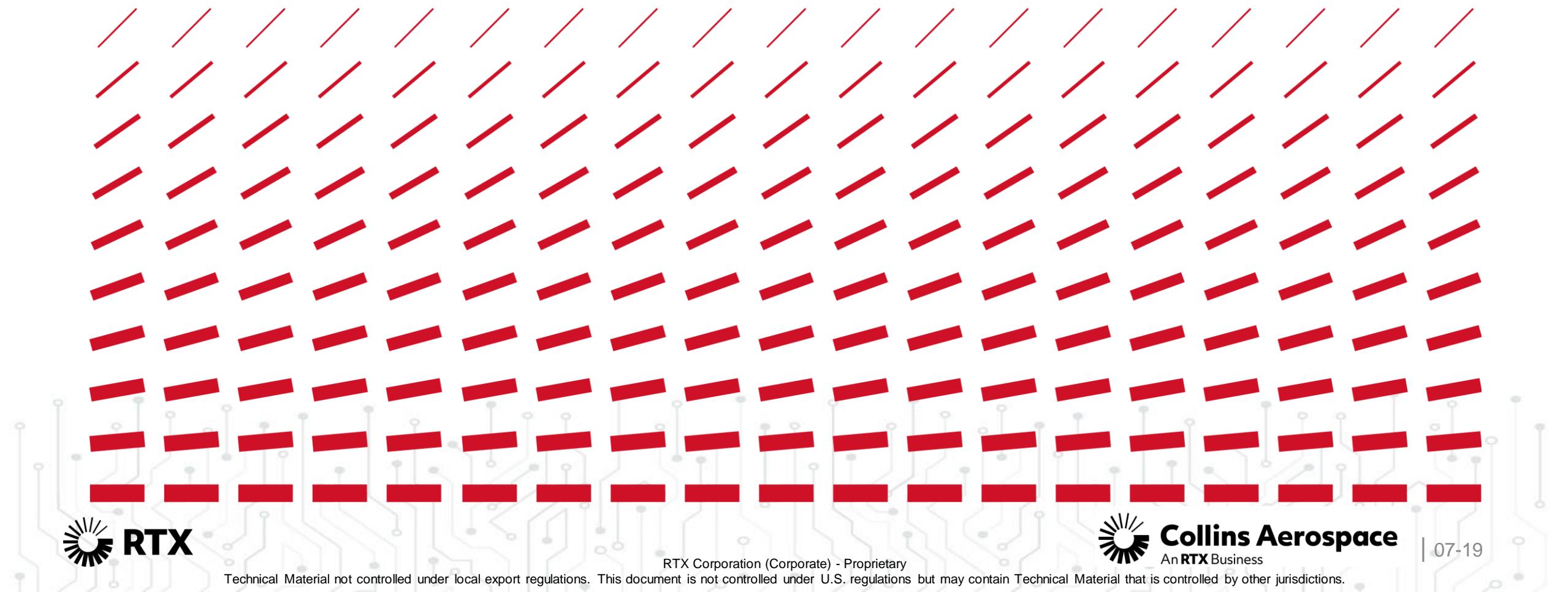
RTX Corporation (Corporate) - Proprietary

Technical Material not controlled under local export regulations. This document is not controlled under U.S. regulations but may contain Technical Material that is controlled by other jurisdictions.



07-18

Thank you.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



08-0



TGECYBEREMBSEC

Embedded Systems Security

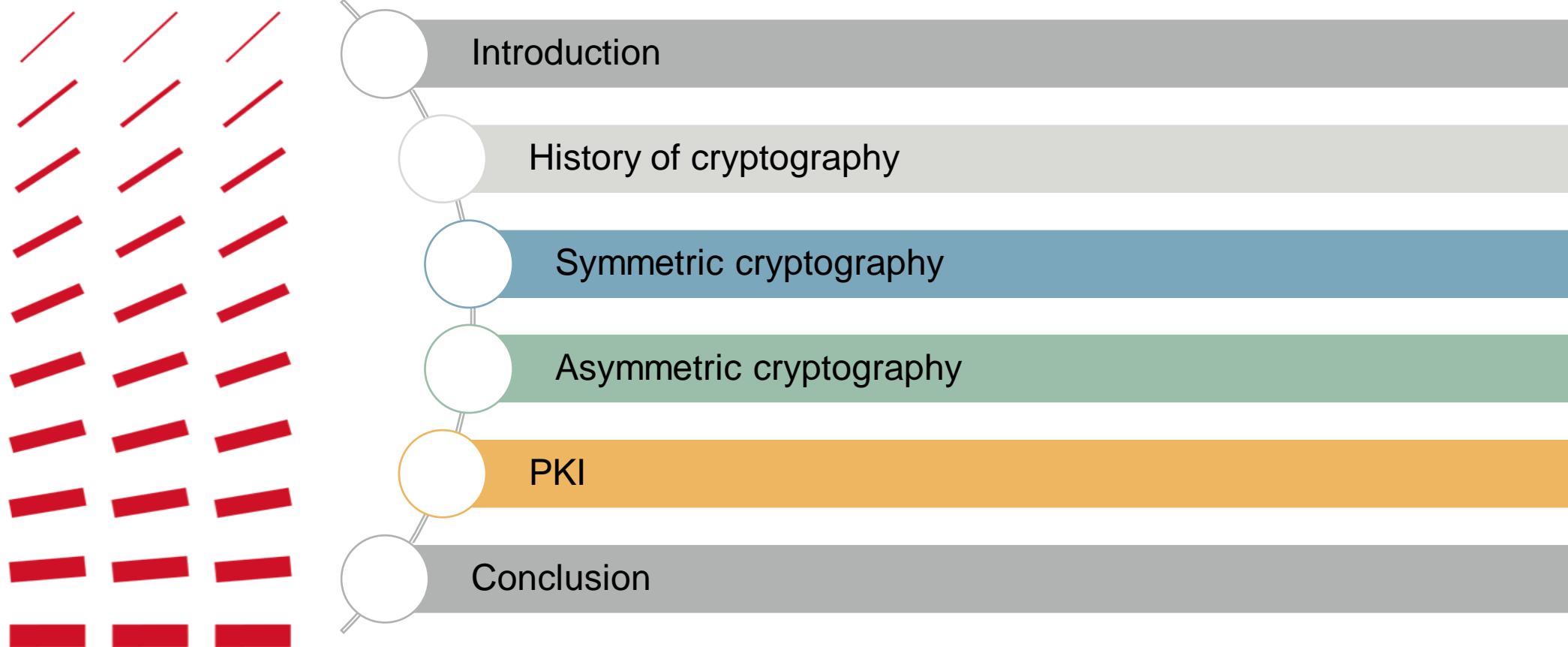
Module 08

Cryptography

Instructor: Patrick Schweickert

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

Module agenda





Cryptography

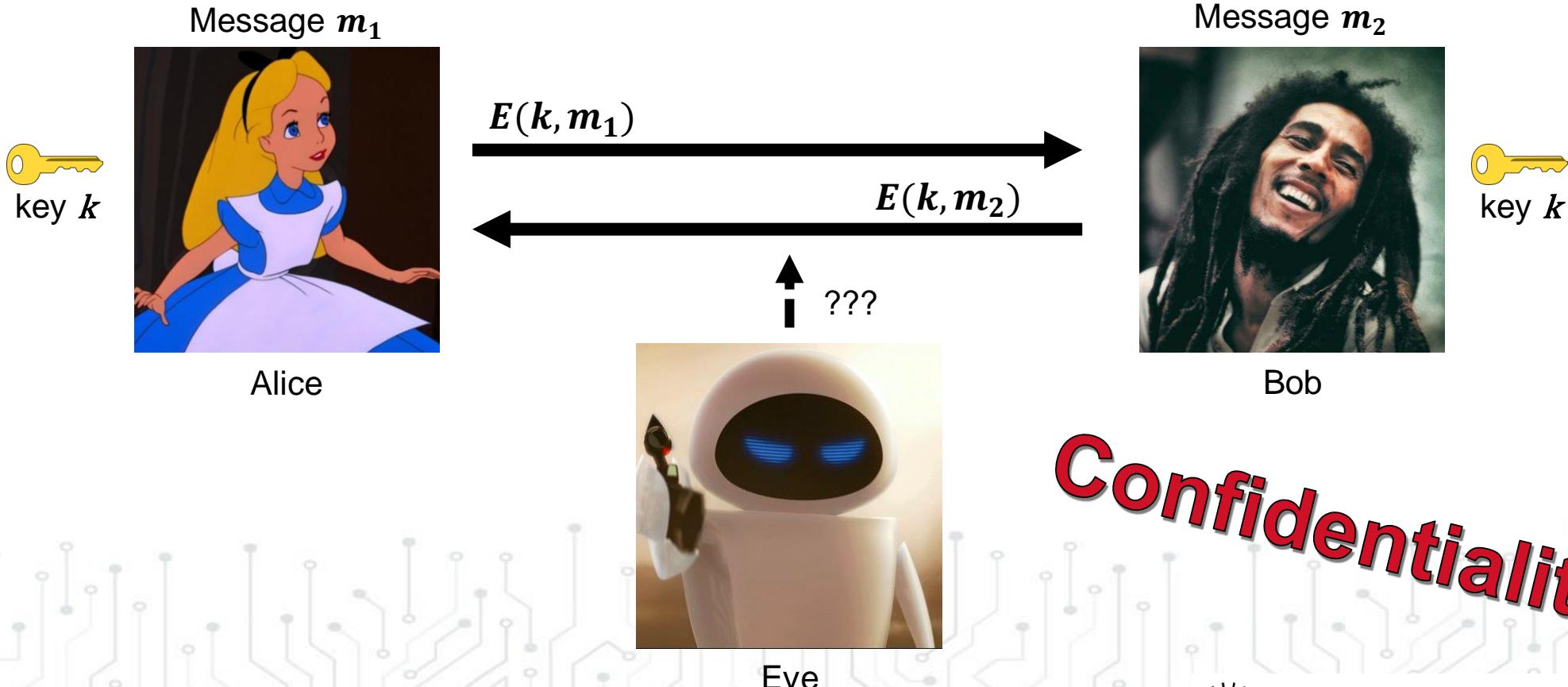
Introduction

History of cryptography
Symmetric cryptography
Asymmetric cryptography
PKI
Conclusion



2* What is cryptography?

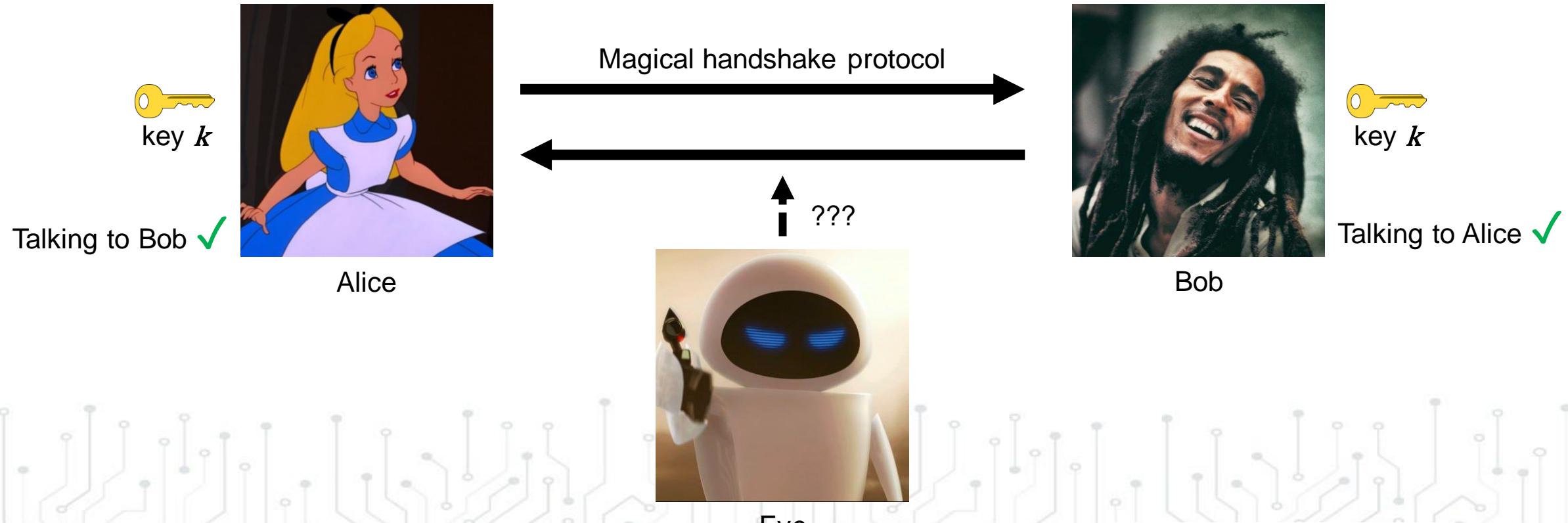
- Best known for **Secure Communication**





2* What is cryptography? — continued

Secret Key Establishment – Cryptographic protocol to securely establish a shared secret key



5* Cryptography has many other uses



Digital Signatures

Signing key k_s

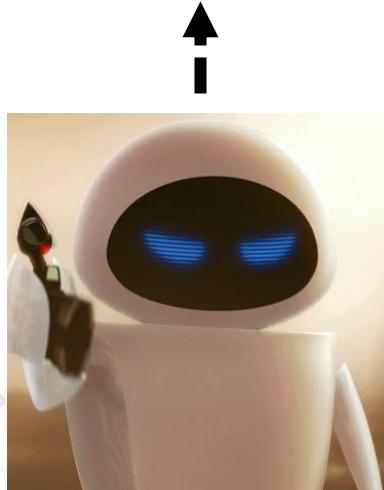


Alice



Bob

Legit from Alice ✓



Eve

Can't forge
signature

Integrity

Authentication
Non-Repudiation



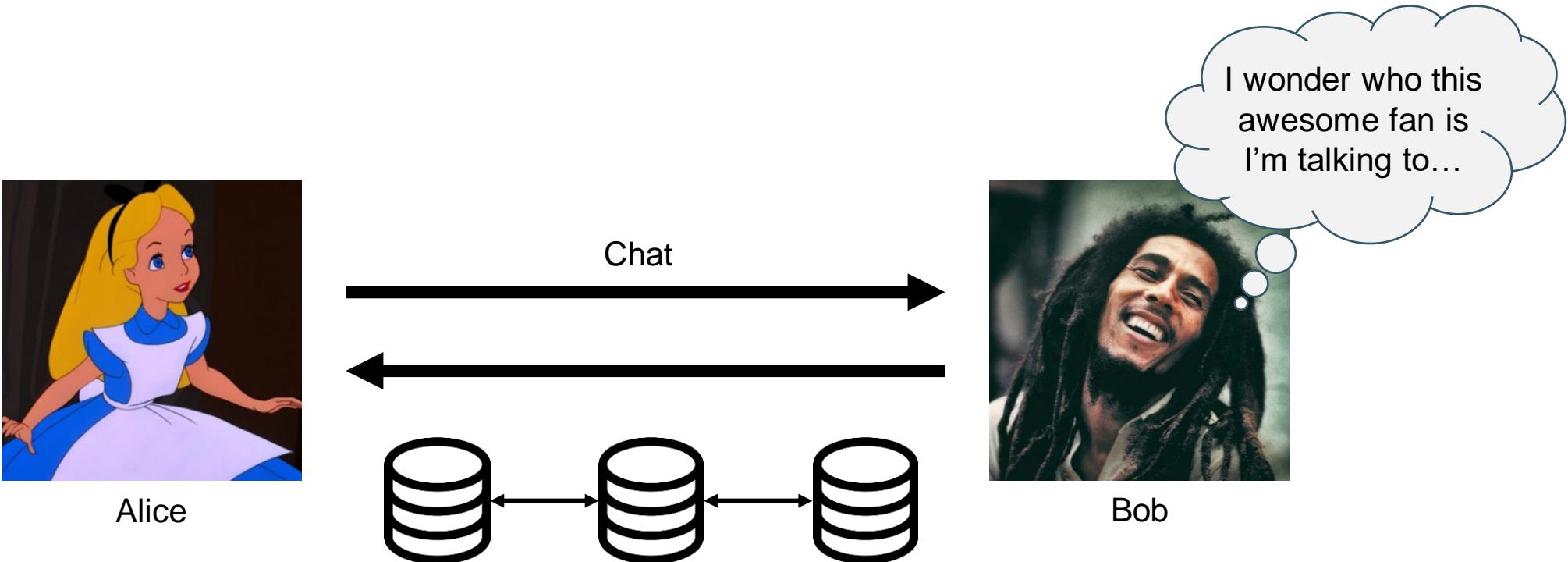
RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Cryptography has many other uses — continued

Anonymous Communication – Mixnet



Cryptography has many other uses — 3

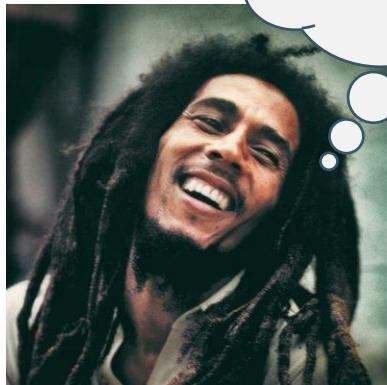


Anonymous Digital Money – Alt currency (e.g., Bitcoin with Blockchain)

- Prevents double-spending



Alice



Bob

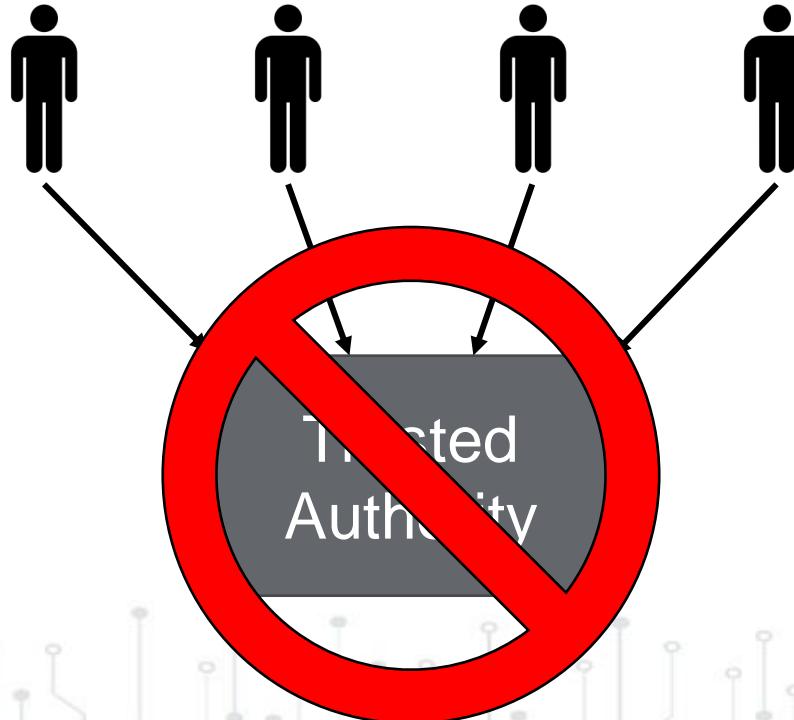
Who just bought
my album?



1* Cryptography has many other uses — 4



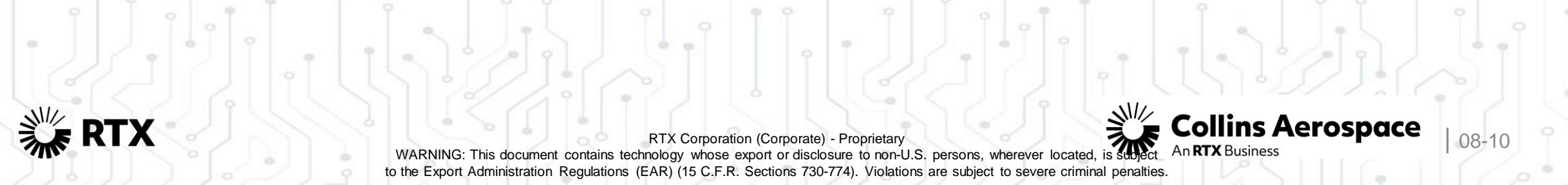
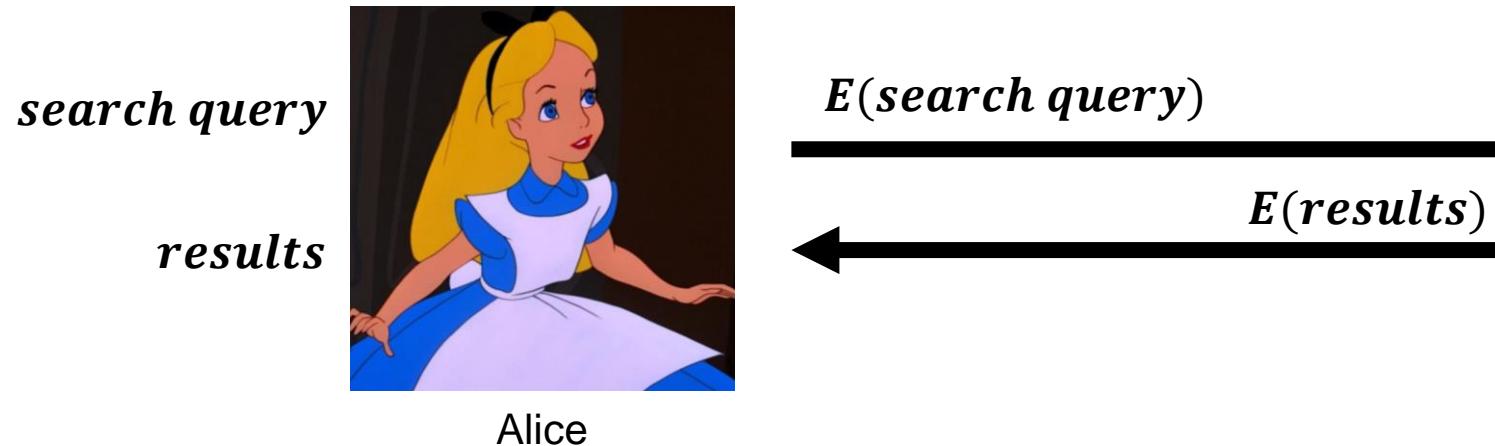
- Elections / Virtual Anonymous Voting
- Private Auctions



Cryptography has many other uses — 5



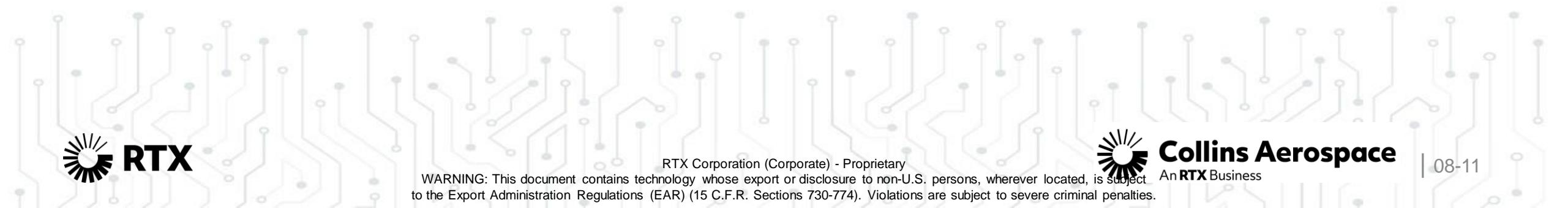
- Homomorphic Encryption



What cryptography is NOT



- Magic fairy dust that solves all security problems
- Reliable unless its implemented correctly
- Something you should try to do yourself
- Necessarily still unbreakable, just because it was unbreakable 10 years ago
- Optional if you think you have nothing to hide



Kerckhoff's Principle



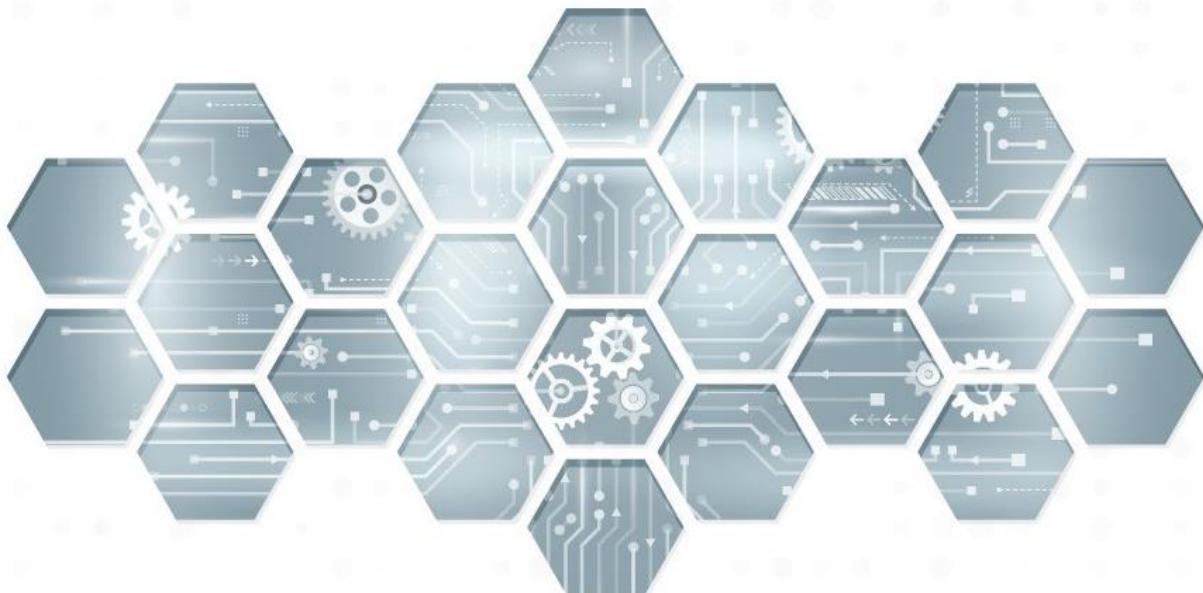
Kerckhoff's Principle

- No secret except the key!
- Details of a cryptosystem can be made public without weakening the system

Violation of Kerckhoff's Principle



Source: <http://xkcd.com/257/>



Cryptography

Introduction

History of cryptography

Symmetric cryptography

Asymmetric cryptography

PKI

Conclusion



1* Substitution cipher



Each letter maps to a different letter

A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z



M|R|B|G|S|I|O|A|E|F|Y|W|D|K|U|Q|H|P|C|J|T|Z|V|X|I|N

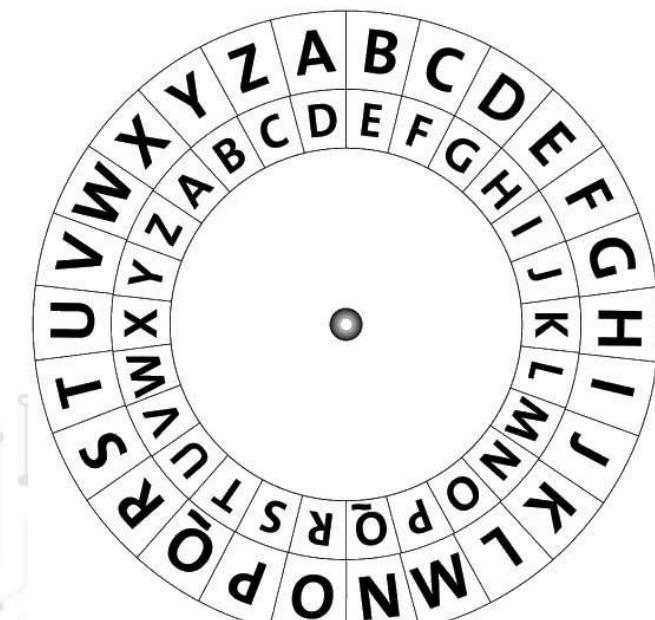
HELLO WORLD → ASWWU ?





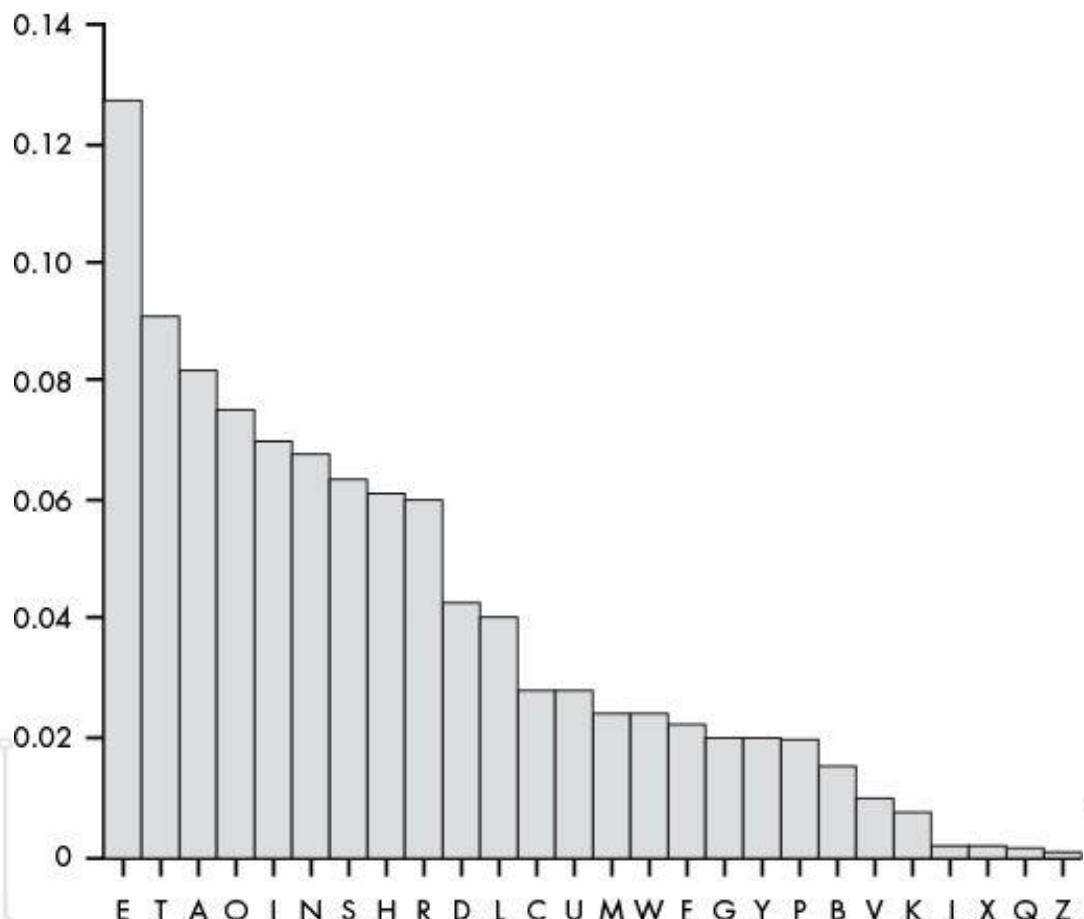
1* Caesar cipher

- Type of Substitution Cipher
- Rotation by n letters (e.g., ROT13)
 - Julius Caesar purportedly used a rotation of 3



Breaking substitution ciphers

- Size of the key space: $|\mathcal{K}| = 26! \approx 2^{88} \approx 3.0948501 \times 10^{26}$
 - Would take a long time to brute force
- Letter Frequency Analysis
 - Most common letters: E, T, A, O, I
 - Most common digram pairs: TH, ER, ON, AN
 - Most common digram repeats: SS, EE, TT, F
 - Most common 2-letter words:
 - OF, TO, IN, IT, IS
 - Most common 3-letter words:
 - THE, AND, FOR, ARE
- For Caesar, just try all 26 rotations

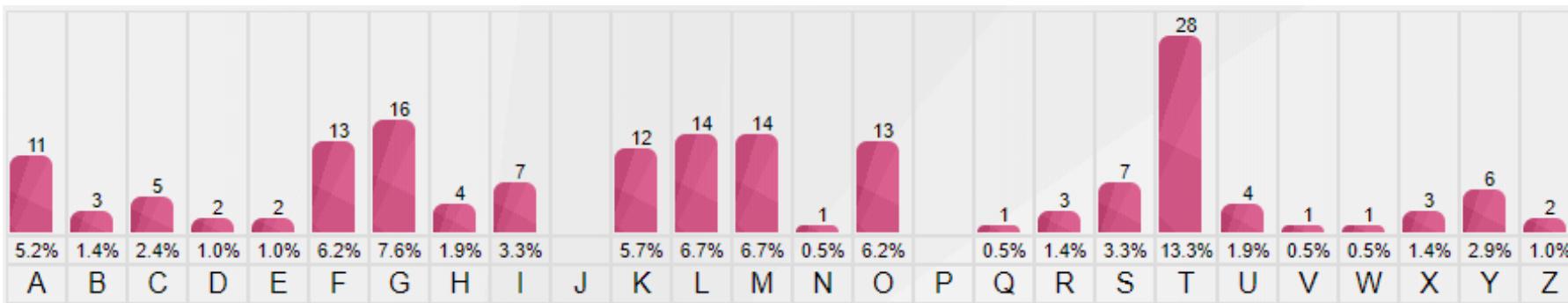




12* Breaking substitution ciphers



- LHAET MIT YOFAS YKGF MOTK MITLT AKT MIT XGBA UTL GY MIT LMAKLIOH
 TFMTKH KOLT OML DOLLOGF MG TVHS GKT LMKAFUT FTC CGKSRL MG LTTQ GWM FTC
 SOYT AFR FTC EO XOSONAMOGFL MG ZGSRSB UG CITKT FG DAF IAL UGFT ZTYGKT



SPACE THE FINAL FRONTIER THESE ARE THE VOYAGES OF THE STARSHIP ENTERPRISE ITS FIVE YEAR MISSION TO EXPLORE STRANGE NEW WORLDS TO SEEK OUT NEW LIFE AND NEW CIVILIZATIONS TO BOLDLY GO WHERE NO MAN HAS GONE BEFORE



Vigenère cipher

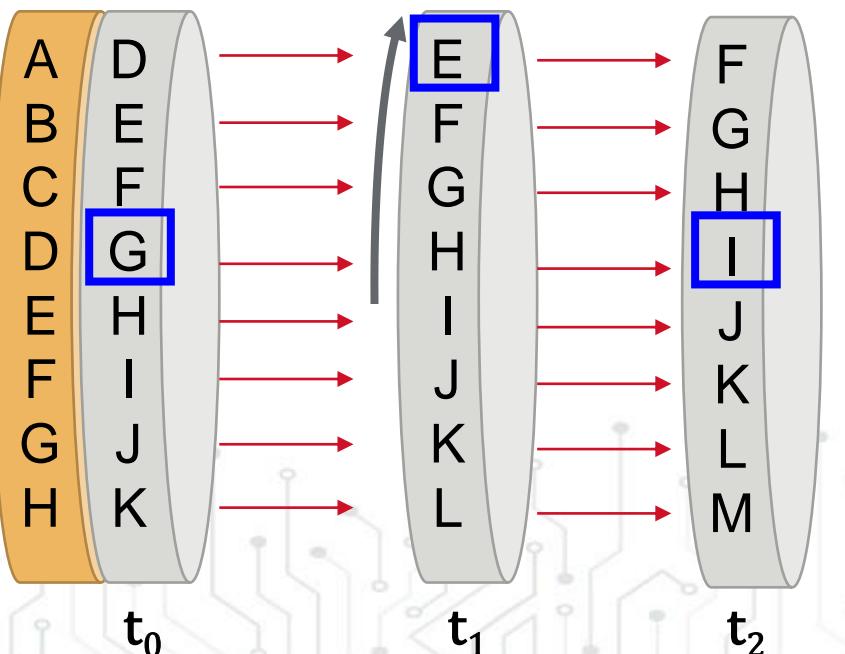
- Originally invented by Giovan Battista Bellaso in 1553 (misattributed to Blaise de Vigenère in 1800s)
- Remained unbroken until 1863 (almost 3 centuries later)
- Essentially a series of interwoven Caesar ciphers

$$\begin{array}{rcl} m & = & W \ I \ N \ T \ E \ R \ I \ S \ C \ O \ M \ I \ N \ G \\ + k & = & C \ H \ A \ O \ S \ C \ H \ A \ O \ S \ C \ H \ A \ O \quad (\text{mod } 26) \end{array}$$

$$c = Z \ Q \ O \ I \ X | U \ Q \ T \ R \ H | P \ Q \ O \ V$$


Rotor machines

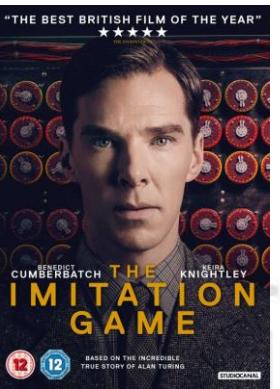
- Invented by Dutch Naval Officers in c. 1875
- Hebern Rotor Machine (1917) first solid example
- Secret key embedded on a disc inside the central cylinder
 - Encodes a substitution table
- Every time a key is pressed, the disc rotates one notch



DAD
=
?

Enigma Machine (3-5 rotors)

- Invented by German engineer Arthur Scherbius in 1923
- Each rotor can be set to one of 26 starting positions ($26^3 = 2^{14}$)
- Not every rotor rotates each keypress (only the left)
 - Others turnover every so often, configurable
- Additionally, had a configurable plugboard
 - Connected pairs of letters, swapped them before scrambling
 - Exponentially increased entropy by factor of x^2
 - Total # keys = 2^{28}
- Initial configuration of the machine was the “key”



Data Encryption Standard (DES)

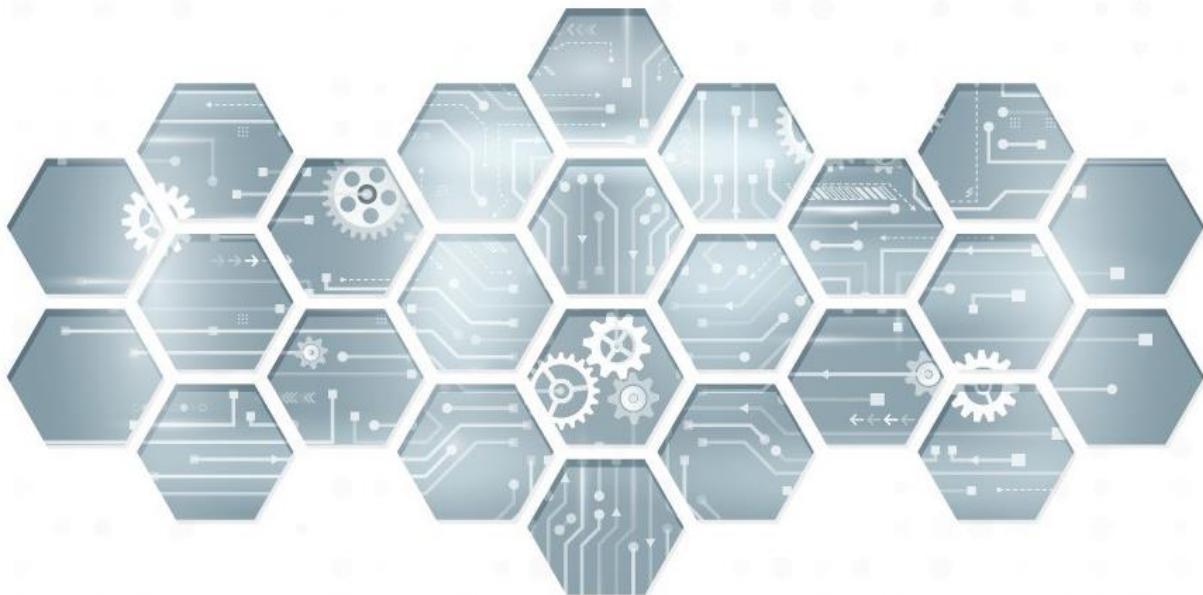


- USG released RFP in 1974
- IBM submitted their “Lucifer” cipher
- Key length = 56 bits, number of keys = 2^{56}
- Vulnerable to exhaustive search (brute force)
 - EFF created “Deep Crack” machine in 1998, which cracked a DES key in 56 hours
 - 1999 collaboration between EFF and distributed.net cracked a DES key in 22 hours and 15 minutes
 - Now, standard personal GPU can crack a DES key in less than 2 days
 - DES withdrawn in 1999
- Triple DES (3DES) released in 1998 (officially named Triple Data Encryption Algorithm (TDEA))
 - Performs DES 3 times, with 3 keys
 - Increases key space to $2^{56*3} = 2^{168}$
 - But attacks exist in 2^{112}
 - 3DES withdrawn in 2005

Advanced Encryption Standard (AES)



- USG released RFP in 1997
- Two Belgian cryptographers Vincent Rijmen and Joan Daemen submitted their “Rijndael” cipher
- 3 different key lengths = 128, 192, 256 bits, number of possible keys = 2^{128} , 2^{192} , 2^{256}
- Still in use today, although only the largest key length 256 is approved



Cryptography

Introduction

History of cryptography

Symmetric cryptography

Asymmetric cryptography

PKI

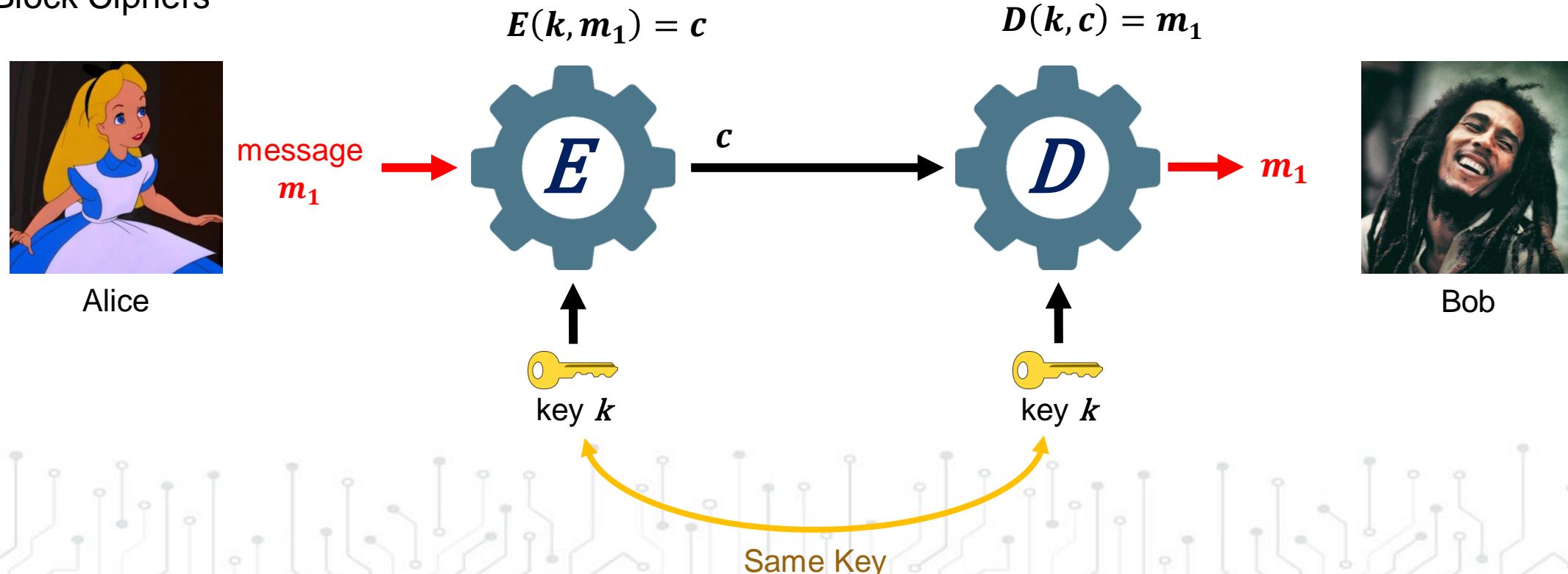
Conclusion

Symmetric encryption



Encryption algorithm E and corresponding Decryption algorithm D that use the **same key k**

- Stream Ciphers
- Block Ciphers



Claude Shannon – perfect secrecy



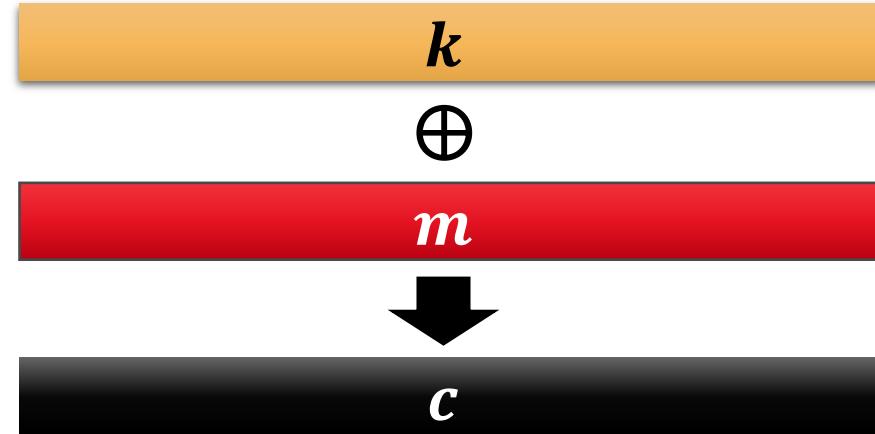
How to measure strength of a cipher?

- 1949 paper *Communication Theory and Secrecy Systems* defined “perfect secrecy” as:
 - $\Pr(x) = \Pr(x|y)$
 - “A ciphertext y gives no information about the plaintext x
 - $\Pr[c = E(k, m_0)] = \Pr[c = E(k, m_1)] \forall m_0, m_1, c$
- Attacks:
 - Ciphertext Only Attack
 - Chosen Plaintext Attack
- The bad news...
 - Security is based on the randomness of the key
 - **Perfect** secrecy requires key length \geq message length
 - You have to somehow securely exchange keys before-hand
 - If you have a way to securely exchange something as long as your message, then you might as well use that method to exchange your message...



2* Vernam cipher – One Time Pad (OTP)

- First described by Frank Miller in 1882
- Gilbert Vernam submitted patent in 1917
- A “Stream Cipher”
 - Each plaintext bit is encrypted one-at-a-time with the corresponding bit of the key (pad)
- Has perfect secrecy because requires:
 - Key length \geq message length
 - Key must be truly random
- Encryption/Decryption are incredibly fast
- Key can NEVER be reused in OTP or ANY stream cipher

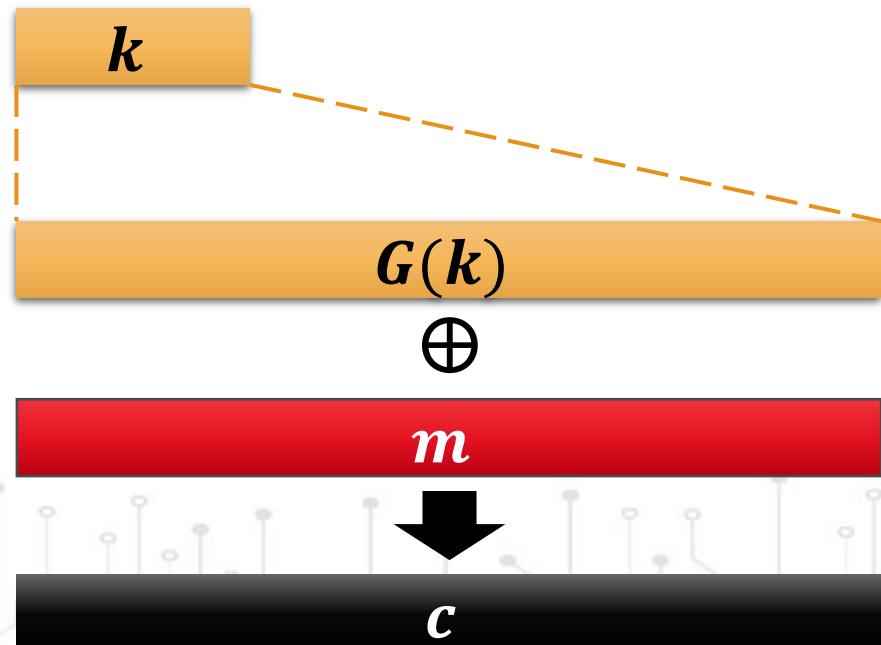


$$\begin{aligned} c_1 \oplus c_2 &= \cancel{k \oplus m_1} \oplus \cancel{k \oplus m_2} \\ &= m_1 \oplus m_2 \end{aligned}$$

Making crypto practical – use secure PRGs



- Security is based on randomness of key
 - How to define randomness? Unpredictability, Balance (roughly equal number of 1's and 0s)
- Pseudo-Random Number Generators (PRNG or just PRG)
 - Function G that takes a truly random seed as input, and outputs a pseudorandom number the same length as the message
 - Must be unpredictable and balanced
- Not all PRGs are created equal!!!
 - Linear Congruential Generators \times
 - glibc random() \times
 - DUAL_EC_DRBG \times
 - NIST only approves 3 algorithms:
 - Hash_DRBG, HMAC_DRBG, CTR_DRBG \checkmark
 - Raytheon Cryptographic Library (RCL) \checkmark



Two-time pad is common mistake



- Venona Project (1940s)
 - US Counterintelligence program launched in WWII by US Army SIGINT Service
 - Charter was to decrypt messages transmitted by the Soviet Union
 - Soviets used OTP, manually rolling dice to generate keys, laborious, so reused keys
- MS-PPTP (Windows)
 - Client ↔ Server communication
 - Unique key per session, BUT used the same key both directions
- 802.11b WEP
 - Used PRG “Rivest Cipher 4 (RC4)” $G(IV \parallel k)$
 - IV length = 24 bits, so repeated after $2^{24} \approx 16.7$ million frames
 - Also, keys were too closely related
- Disk encryption
 - Only change part of a file



Example stream ciphers

- RC4, CSS X
- EU ECRYPT “eStream” Project ✓
- NIST “Lightweight Cryptography” Initiative
 - On February 7, 2023, NIST announced the winner, a group of cryptographic algorithms called **Ascon**, will be published as NIST’s lightweight cryptography standard later in 2023.
 - Unknown if any will be approved for use in national security systems



eStream Profile Ciphers

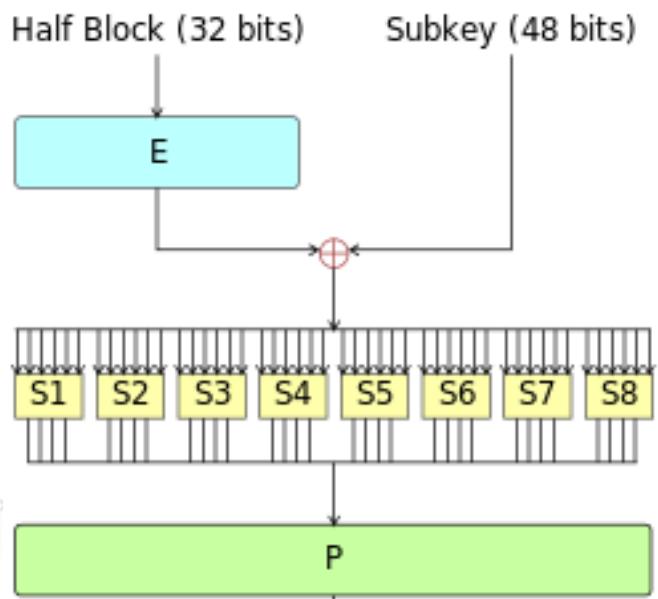
Algorithm	Software	Hardware
Grain		X
HC-256	X	
MICKEY		X
Rabbit	X	X
Salsa20	X	X
SOSEMANUK	X	
Trivium		X



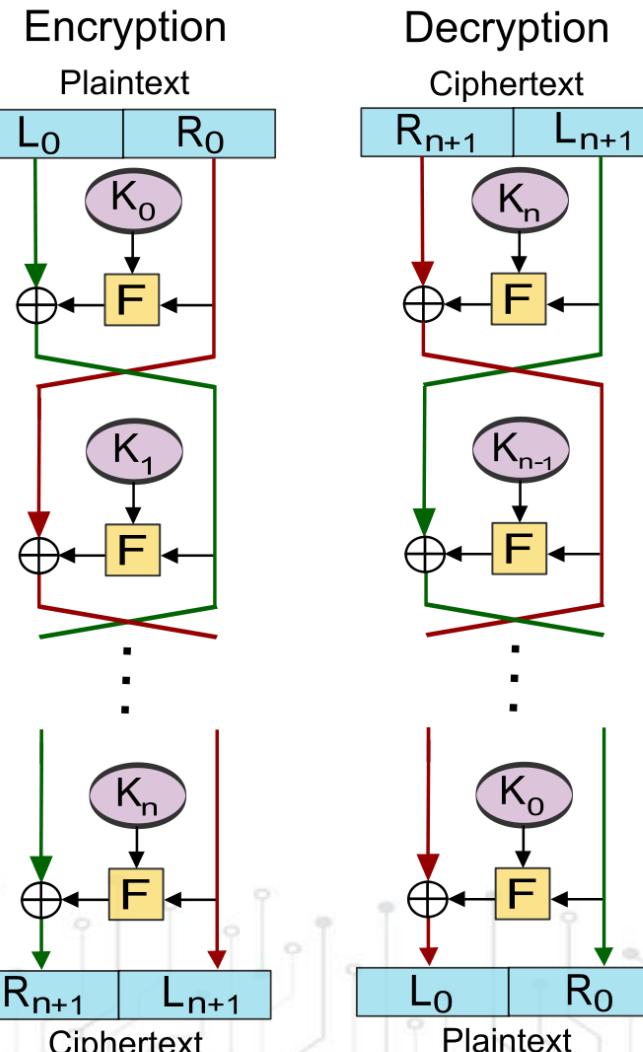
Block ciphers

- Encrypt plaintext one “block” at a time
- Block size varies across algorithms
- Data Encryption Standard (DES) was the first
 - Block size = 64 bits

Function “F”



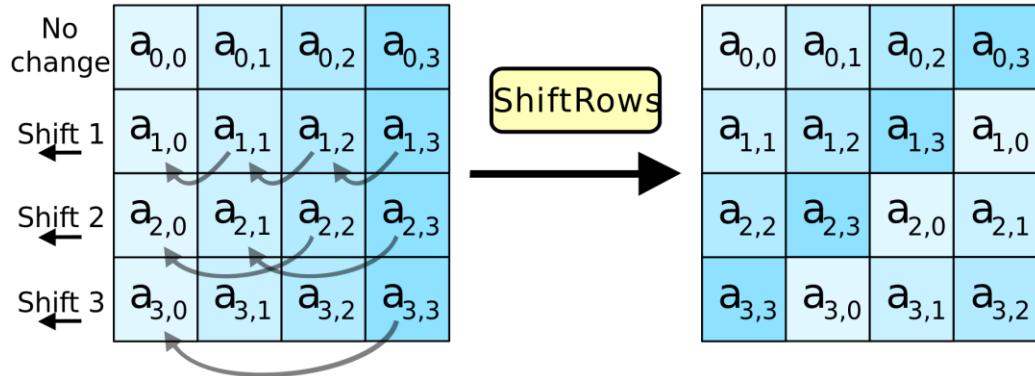
DES “Feistel” Structure



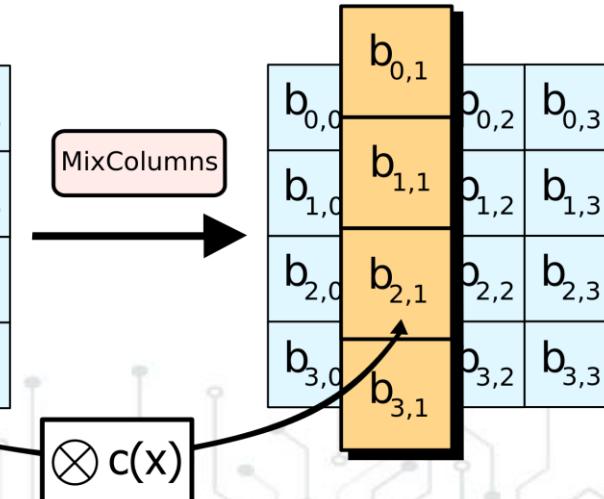
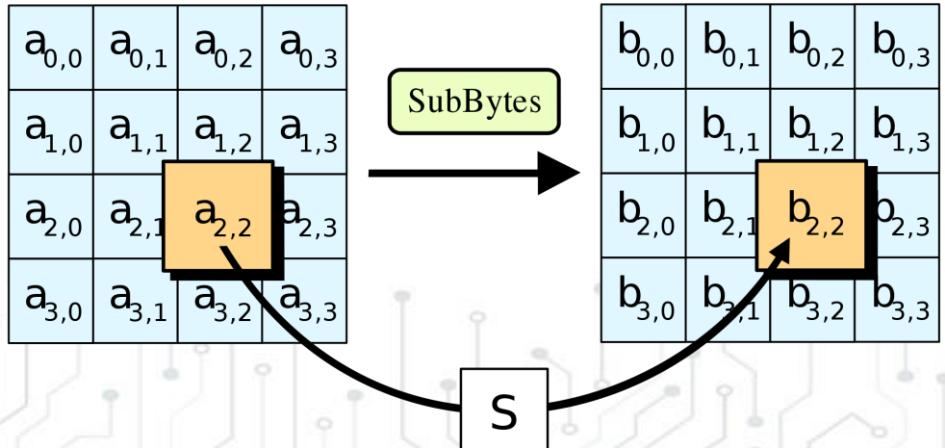
Advanced Encryption Standard (AES)

- Block size = 128, 192, or 256 bits
- “Substitution-Permutation” structure
 - Substitution: S-boxes (non-linear!)
 - Permutation: Shift Rows, Mix Columns
 - Add Round Key
- 10 rounds

Permutation



Substitution

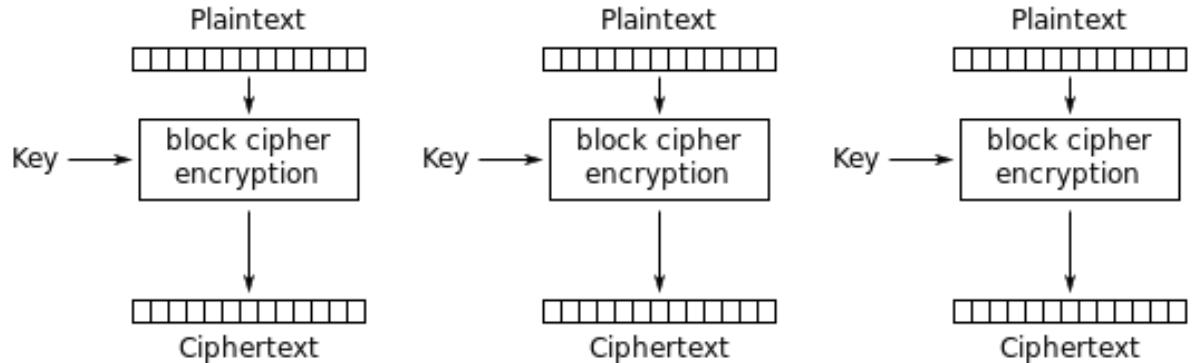




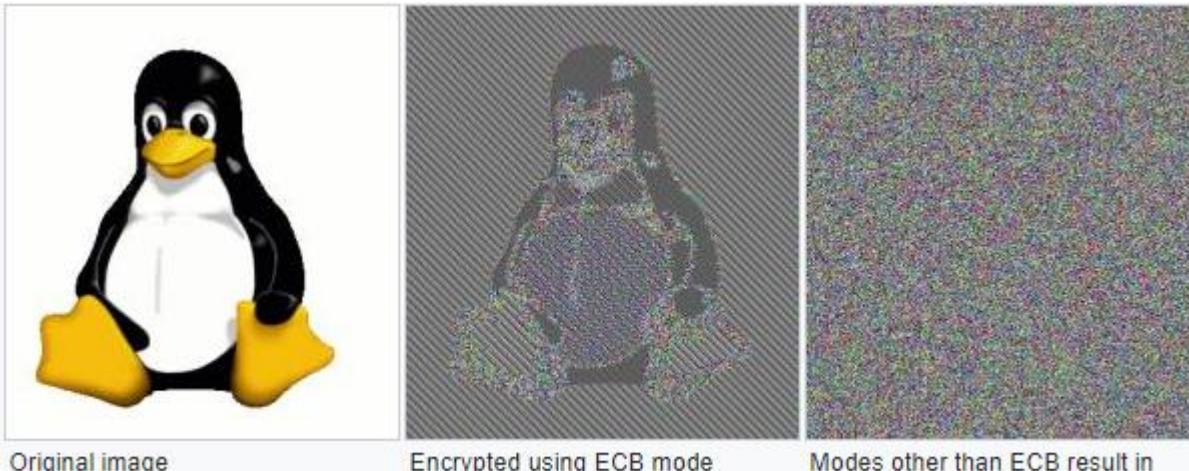
1* AES modes of operation – ECB



- Electronic Code Book (ECB)
- Parallelizable
- No “diffusion” – Does not hide patterns well



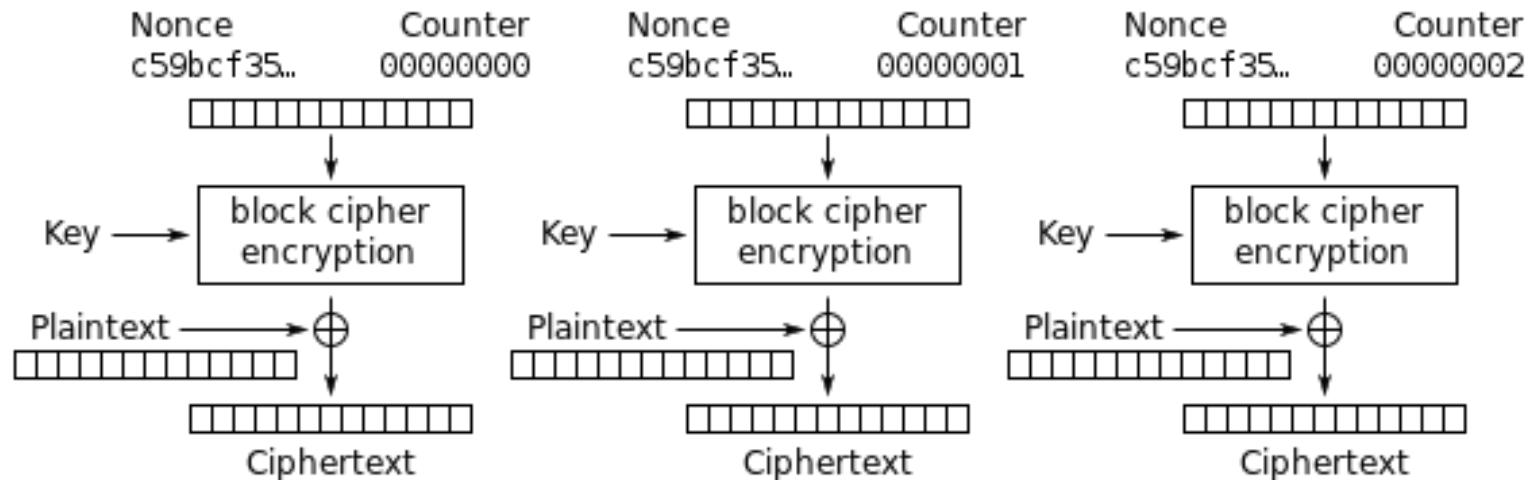
Electronic Codebook (ECB) mode encryption



AES modes of operation – CTR



- Counter (CTR)
- Parallelizable
- Encrypts an incrementing counter

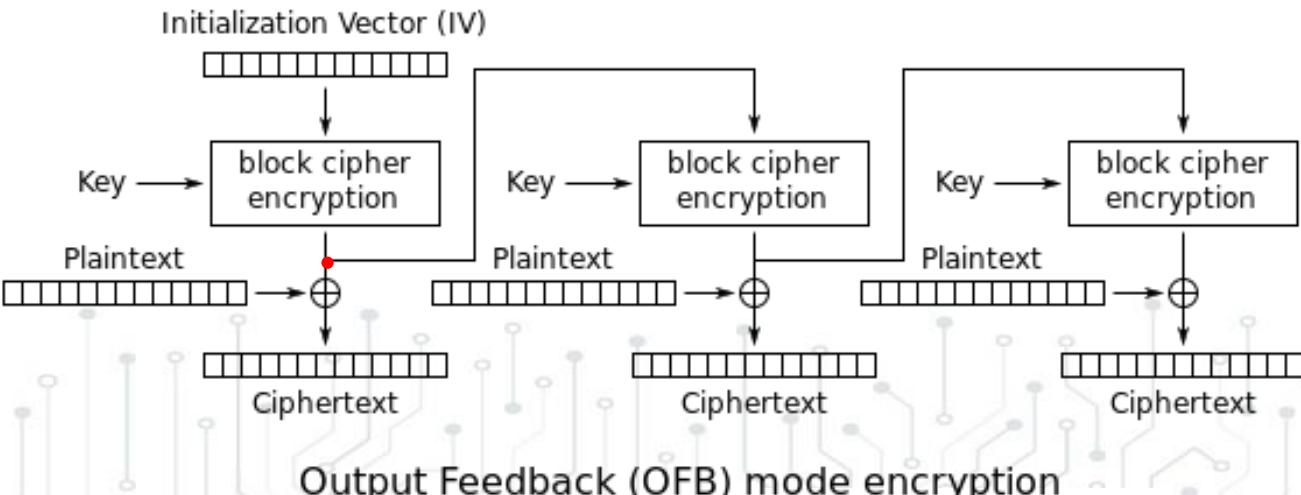
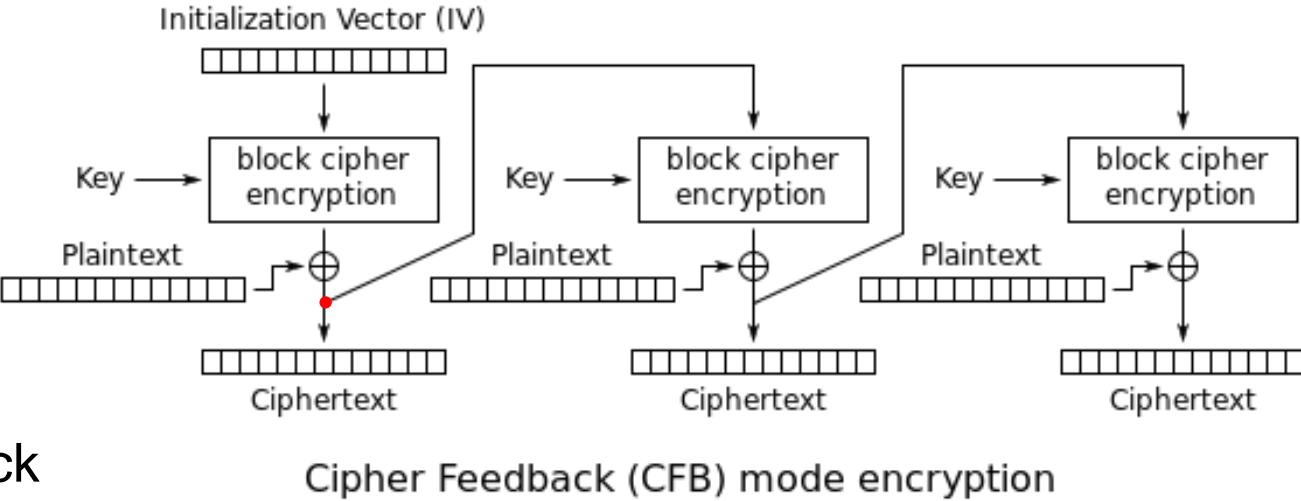


Counter (CTR) mode encryption

AES modes of operation – CFB and OFB



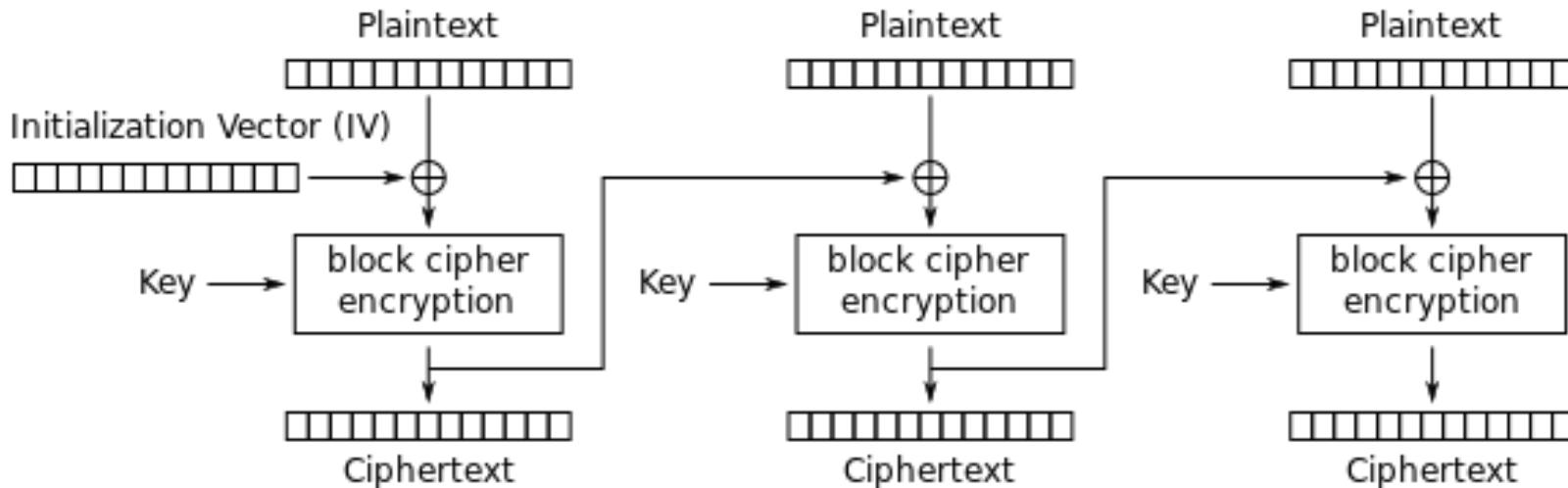
- Cipher Feedback (CFB)
 - Feedback **after** xor
 - Ciphertext fed into next block
- Output Feedback (OFB)
 - Feedback **before** xor
 - Output of encryption fed into next block
- Not parallelizable



AES modes of operation – CBC



- Cipher Block Chaining (CBC)
 - IV xor with plaintext *before* block cipher encryption
 - Output feeds into next block
 - Not parallelizable



Cipher Block Chaining (CBC) mode encryption

Performance of symmetric ciphers



- AMD Opteron 8354 2.2 GHz processor
- Written in C++, compiled with GCC 4.1.2 using -O3 optimization

	Algorithm	MiB/Second	Cycles Per Byte	Microseconds to Setup Key and IV	Cycles to Setup Key and IV
Block	AES-256 (CTR)	96	18.2	0.756	1383
	AES-256 (CBC)	80	21.7	0.619	1133
	TDEA (3DES)	13	134.5	27.317	49989
Stream	Salsa20/12	643	2.7	0.483	884
	SOSEMANUK	727	2.4	1.240	2269



AES trade – code size vs performance



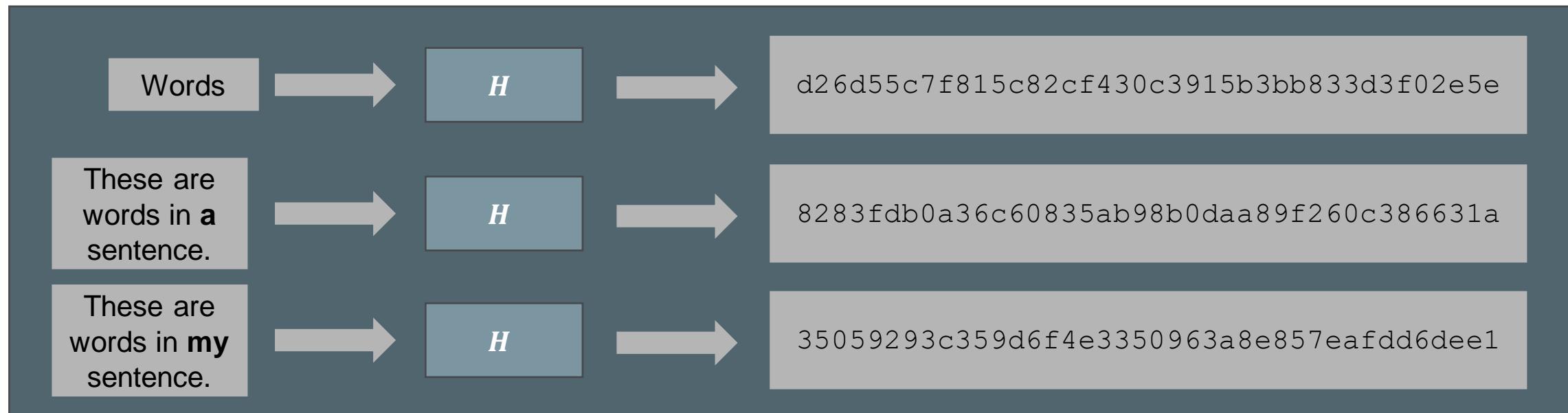
- Certain parts of AES can be pre-computed, but then must be stored
- SWaP vs speed consideration
- AES-specific instructions in Intel, AMD, ARM families for hardware acceleration

Item to Precompute	Code Size	Speed
Round Functions (24 KB)	Largest	Fastest (just table lookups & xors)
S-Boxes (256 Bytes)	Smaller	Slower
[None]	Smallest	Slowest

SHA-1 hash example



- Many-to-one (non-invertible) function H
- Distills arbitrary-sized data down into a value of fixed length
- Small change in input should have large effect on output
- “Collision” occurs when two different messages map to the same fixed value



A small change in the input (even a single bit!) will have a drastic effect on the output.



1* Integrity – hash algorithms

- Many-to-one (non-invertible) function H
- Distills arbitrary-sized data down into a value of fixed length
- “Collision” occurs when two different messages map to the same fixed value
- Only protects against *inadvertent* changes to message
- Do NOT provide confidentiality or authentication

Computes hash with $H(m)$



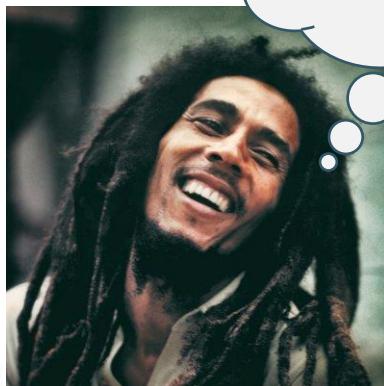
Alice

message || hash



Eve

Computes hash &
verifies hashes match



Bob

I know this hasn't
inadvertently
changed!



1* True integrity requires a key – MAC

- Message Authentication Codes (MAC)
- Signing algorithm S outputs a tag
- Protects against malicious modification / tampering
- Do NOT provide confidentiality

Computes tag with $S(k, m)$

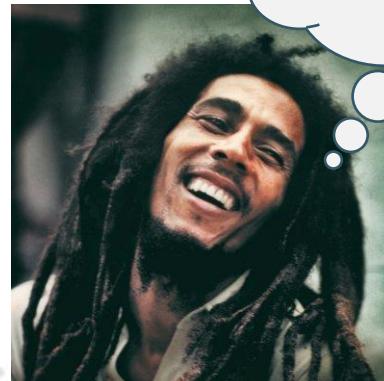


Alice

Computes tag & verifies
that tags match



This is legit from
Alice!



Bob

message || tag



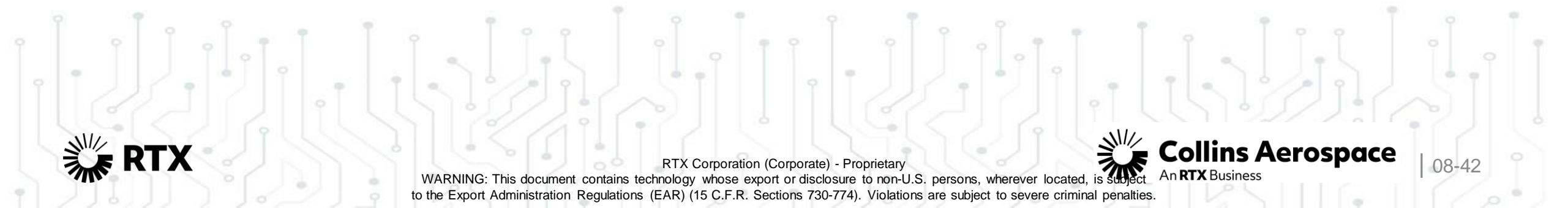
Eve

Can't tamper
with message

Authentication encryption



- Confidentiality AND Integrity require Authentication Encryption
 - MAC-then-Encrypt (TLS) ✗
 - Encrypt-and-MAC (SSH) ✗
 - Encrypt-then-MAC (IPSEC) ✓



Integrity algorithms



- **Hashes:**
 - MD-5, SHA-1, SHA-2 & SHA-3 (lengths 224, 256, 384, 512)
 - Use a salt when hashing
- **MACs**
 - HMAC w key length \geq 112 bits
 - CMAC w AES and key length \geq 112 bits
 - GMAC with AES
 - KMAC w key length \geq 112 bits
 - CMAC w 3DES
 - GMAC w 3DES



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



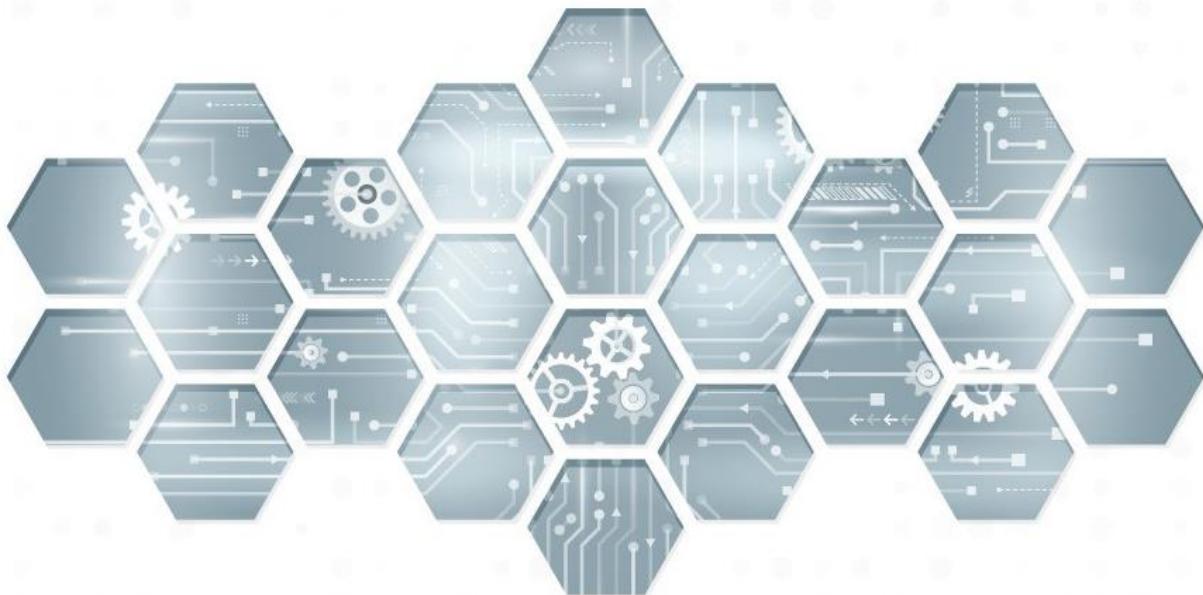
Hash & MAC performance



- AMD Opteron 8354 2.2 GHz processor
- Written in C++, compiled with GCC 4.1.2 using -O3 optimization

Algorithm	MiB/Second	Cycles Per Byte	Microseconds to Setup Key and IV	Cycles to Setup Key and IV
SHA-2 (256)	111	15.8	–	–
SHA-2 (512)	99	17.7	–	–
HMAC	147	11.9	0.509	932
CMAC	109	16.1	0.600	1098
GMAC	417	4.2	2.944	5388





Cryptography

Introduction

History of cryptography

Symmetric cryptography

Asymmetric cryptography

PKI

Conclusion

Key management is difficult



- Confidentiality AND Integrity requires a unique key for EVERY PAIR of users/devices
- How to get keys to users?
 - Pre-provision? But what to do when keys expire?
 - Trusted Authority generates and securely distributes? But single-point-of-failure
- What if users could securely generate their own key just-in-time without relying on a trusted authority?

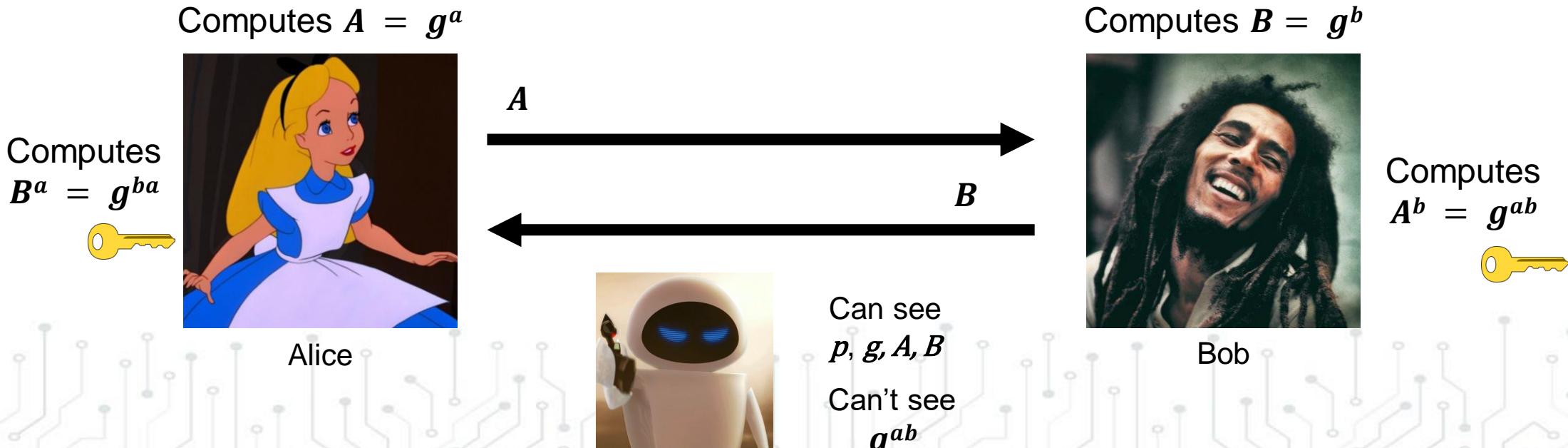
# Users	# Keys
2	1
4	6
10	45
25	300
50	1225
100	4950





2* Diffie-Hellman (DH) key exchange

- Choose a large prime p (public knowledge)
- Choose an integer g (public knowledge)
- Alice chooses a secret key a
- Bob chooses a secret key b
- Breaking would involve efficiently solving the Discrete Logarithm problem (HARD)

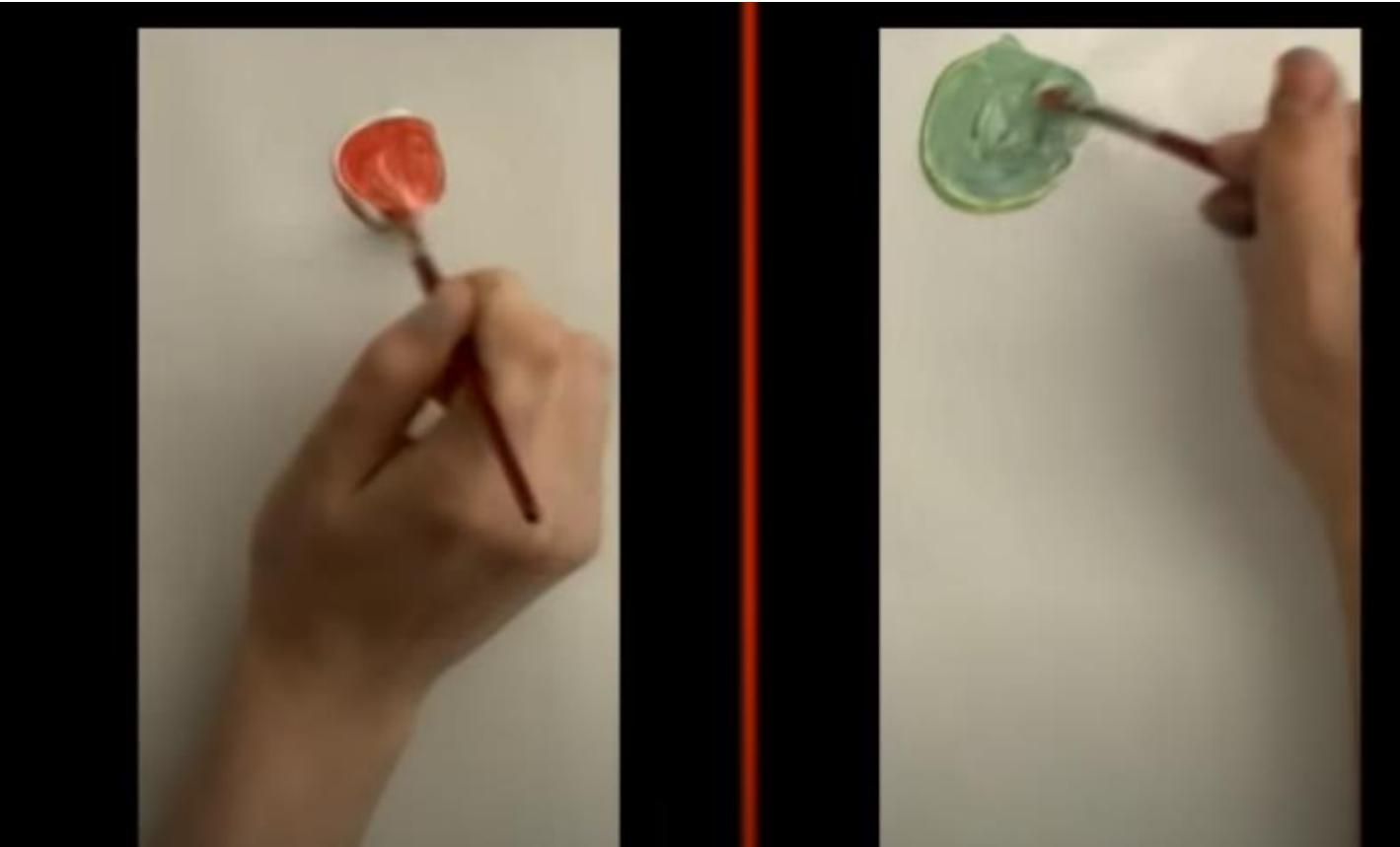




Video link 48: Diffie Hellman key exchange – with paint

Click the image or link to launch video from beginning.

Move slider to timestamp 2:18 to watch the Diffie Hellman key exchange with paint until 4:24.



<https://www.youtube.com/watch?v=YEBfamv-do>





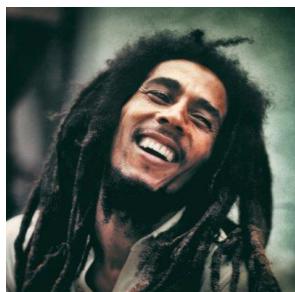
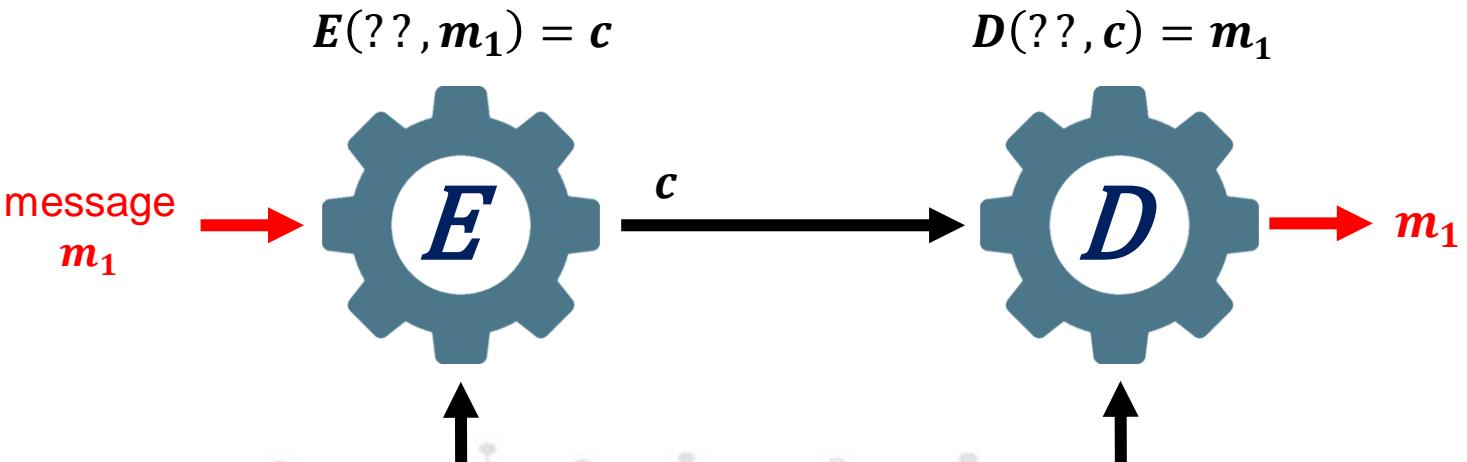
4* Asymmetric encryption

- Encryption function E and Decryption function D use **different keys**
- Each party has a unique “key pair”, which consists of a public key pk and a private key sk
- Each party sends their public keys to one another
- For Authentication: Encryption uses sender’s private key, and Decryption uses sender’s public key
- For Confidentiality: Encryption uses recipient’s public key, and Decryption uses recipient’s private key



Alice

pk_{Alice}
 sk_{Alice}



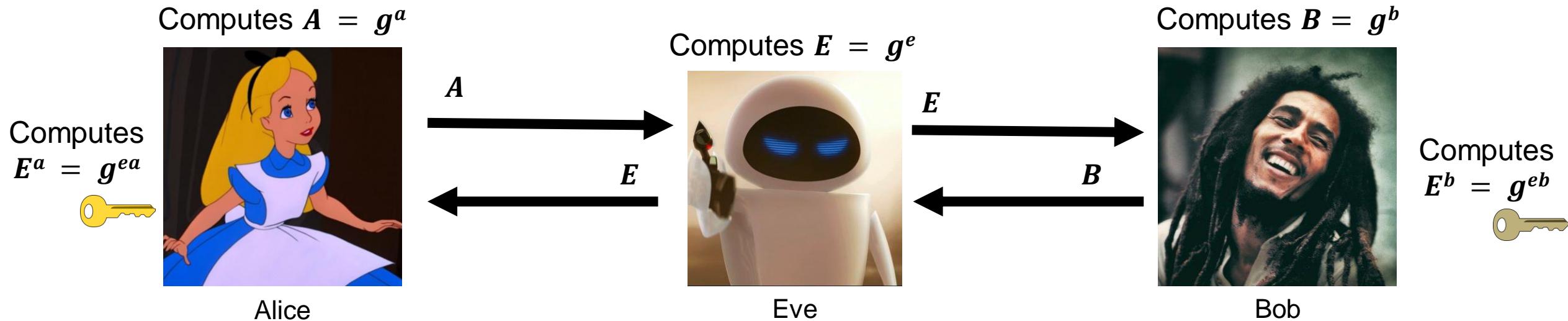
Bob

pk_{Bob}
 sk_{Bob}

Man-in-the-Middle (MITM) attack



- Attacker chooses their own private key e
- Mitigation: Before sending g^x , encrypt with recipient's public key pk



RSA



- Schemes exist for both encryption/decryption and digital signatures
- Key length: 2048 or 4096
- Key Generation:
 - Choose 2 distinct large prime numbers p, q (roughly about the same number of digits)
 - Compute $p * q = N$
 - Choose integers e, d s.t. $e * d = 1 \pmod{\varphi(N)}$
 - Public key $pk = (N, e)$
 - Private key $sk = d$
- Encryption:
 - $m^e \equiv c \pmod{N}$
- Decryption:
 - $c^d \equiv m \pmod{N}$



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



08-51

Elliptic Curve Cryptosystems (ECC)



In mathematics, an elliptic curve is a smooth, projective, algebraic curve of genus one, on which there is a specified point O. An elliptic curve is defined over a field K and describes points in K^2 , the Cartesian product of K with itself. If the field's characteristic is different from 2 and 3, then the curve can be described as a plane algebraic curve which consists of solutions (x, y) for:

$$y^2 = x^3 + ax + b$$



Approved asymmetric algorithms



- Key Agreement:
 - Diffie-Hellman (DH)-2048
 - Menezes-Qu-Vanstone (MQV)-2048
 - RSA-2048
- Digital Signatures:
 - Digital Signature Algorithm (DSA)-2048, DSA-3072
 - Elliptic Curve DSA (ECDSA)-224
 - RSA-2048



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



08-53

Strength of asymmetric algorithms



- Strength based on the difficulty of an underlying hard problem
1. Integer Factorization – Rivest-Shamir-Adleman (RSA)
 2. Discrete Logarithm – Diffie-Hellman Key Exchange (DHE) and Digital Signature Algorithm (DSA)
 3. Elliptic Curve – Diffie-Hellman Key Exchange (ECDH) and Digital Signature Algorithm (ECDSA)

Key lengths of Symmetric vs Asymmetric algorithms for the same strength

Symmetric	Asymmetric	Elliptic Curve
112-bit	2,048-bit RSA	224-bit
128-bit	3,072-bit RSA	256-bit
192, 256-bit	15,360-bit RSA	512-bit



Performance



- Asymmetric MUCH SLOWER than Symmetric
- Rather than being used for large amounts of encryption/decryption, it is typically used only to establish a shared symmetric key (session key) or for Digital Signatures

Algorithm	Milliseconds per Operation	Megacycles per Operation
DH 2048 (key agreement)	3.84	7.03
MQV 2048 (key agreement)	3.97	7.27
RSA-2048 (signature)	6.05	11.06
RSA-2048 (encrypt / decrypt)	0.16 / 6.08	0.29 / 11.12
ECDSA (sign / verify)	2.88 / 8.53	5.27 / 15.61



Side Channel Attacks (SCA)



- Attack an algorithm's **implementation**, not the algorithm itself
- Glean information from side-channel emissions:
 - Monitoring of power signals or electro-magnetic emanations
- Physical access often required, but not always
 - Cache attack
 - Timing attack
 - Power-monitoring attack
 - Electromagnetic attack
 - Acoustic attack
 - Differential Fault attack
 - Data Remanence attack
 - Software-initiated attack
 - Optical attack



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



| 08-56

Power analysis



- There are three different versions of Power Analysis attacks leveraged against cryptographic systems today:
 - Simple Power Analysis (SPA)
 - Differential Power Analysis (DPA)
 - Correlation Power Analysis (CPA)
- SPA involves merely interpreting power signals to identify key values
- DPA utilizes statistical analysis of power measurements at various stages within a cryptographic algorithm
- CPA leverages the Hamming Weight Power Model to attack AES implementations by using statistical analysis on S-box input and output voltages



RTX Corporation (Corporate) - Proprietary

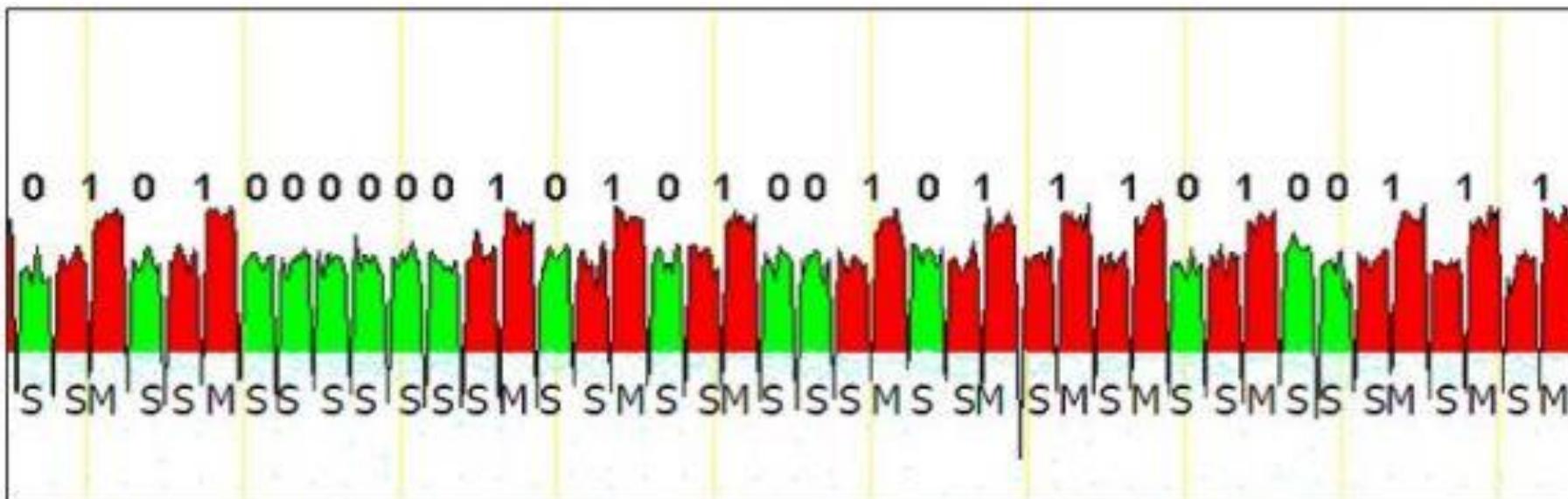
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

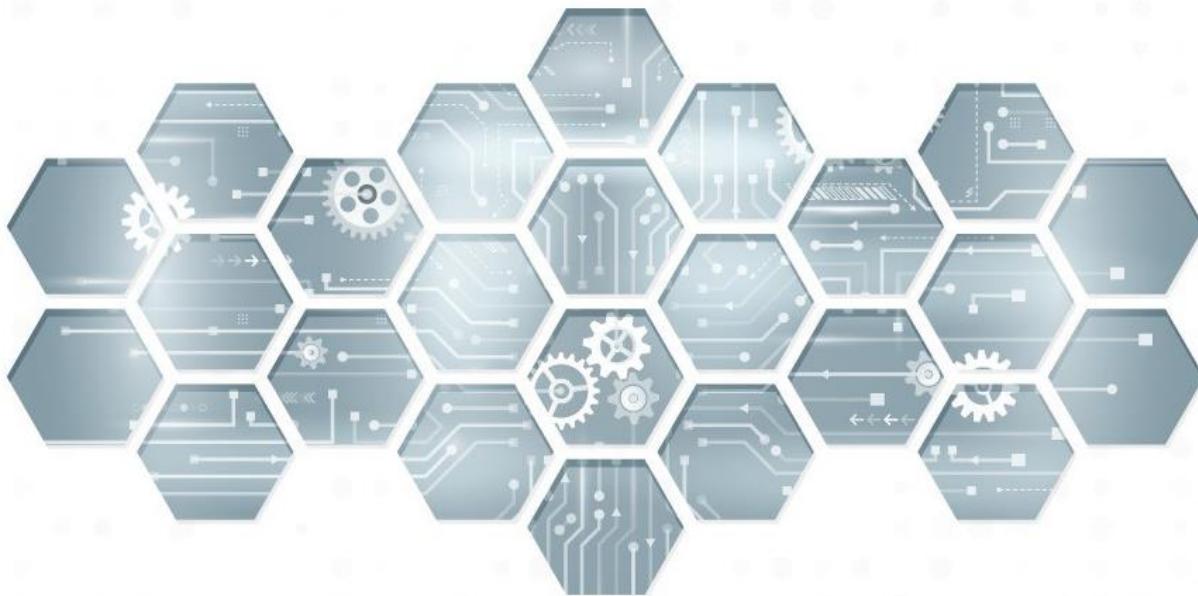


Example attack



- The RSA algorithm contains two operations, square and multiply, with the multiply operation occurring based on the bit-wise value of d and e .
 - Extraction of key material is made possible with SPA by simply identifying when only a square operation occurs versus when it is followed by a multiplication, as illustrated below:





Cryptography

Introduction

History of cryptography

Symmetric cryptography

Asymmetric cryptography

PKI

Conclusion

Public Key Infrastructure (PKI)

RTX TGE
Technology & Global
Engineering



- Public Key Infrastructure is an authentication framework
 - Designed to provide authentication services across different networks and the Internet
 - Does not specify protocols, products, and algorithms.
 - Ensures confidentiality, non-repudiation, and message integrity using:
 - Symmetric and Asymmetric Algorithms
 - Hashes
 - Digital Signatures
 - Utilizes the Public Key Cryptography Standards (PKCS) to form the protocols that make up the infrastructure.

```
-----BEGIN CERTIFICATE-----
5Ku1X0n1VZHPuRqZdM9cyY6sLAAAAAAAHDDANPqkghkiG9+0BAQsFADBIMRI+EA
VK
CZImiZPyIgQBGRYCaW4xfDASBg0jkiAJk/I=IsZAEZfgrFTUvEMRw=gYDVQDExH
TUVBLVdjTj;iMDhsMREfELUNEM4DfDTeyHTEwNDEwTTEwM1oXfDTE0MTEwNDEwTTEw
M1owaDELMAXA1UEBEhMcU4xCzAJBjgJNVBaqTAKtEMQwCgYDVQHcWNCfXcDzAN
9YNVBa0TBkNpdHJpeDEQa4GA1UECxhHE-3VwG9yDEBMBkGA1UEAxMsC3rvcmVn
cm9udC51bwWhlmluMIGfNA0GCSqGS1b3DQEBAQUAA4GNADCBiOKBqOD1l-hIfnTG
vv0nAn-OpZSTXLIJ9UMFsyrPPInuULtG6J0sfpk51VNlxss000TSQRwCljNG6KZ
S1jc1C6yc3Nw2QfZcV42RjbMeirg214NhAIhAlbHapp67zy3np7Kqc3F5CZ
5Ku1X0n1VZHBnRqZdM9cyY6s1lHUQfsn2Q1IDAQABc4ICrjCCApewDgYDVROPAQH/
BAQDAgWgIBNGA1UdJQCMHAgGCCsGAQUFBwMBMrgGCScGSIb3DQEJdWwgYDwgbgYI
KoZhvcNawICA4GCGCgS1b3DQNEAgIAgDALBgIghkgEZQNEASowCvVYJIZI
AWUDBAEIMAsGCVCCSAFIa1wQBa1LBq1gkhgEZQNEAUwBvYFKw4DagcwCgYIKoZI
hvcNawcHQYDVR00BBFEPoZmpqrc01QEW4MZDl7qnPvEiOMB8GA1UdIwQYBaa
FN1gPTIIuAn:cW42E1Phg11f1QTMiHrBqNVR8EcckwgcYwgcOggcCggb2Ggbps
ZGFw0+8+L0NOPUVNRUetV010Mjw=OfIyQUt:Q0EeS049V010Mjw=OfIyQUtQ049
Q0RQLENOPVb1YmxpYvUyMet1e5UvMFN1cnZpV2zLENOPVn1cnZpY2zLENOPUnv
bmZpZ3VvYXRpzb4sREm9RU1FQsxEqz1pbj9jZJ0aWZpY2F0ZV1dm9jYXRpb25M
aXNOp2hc2U/b2JzZWN0Q2xhcsM3Y1jMRG1zdHJp1nV0aW9uUG9pbmQwgcEGCsG
AQUBvEBBIGOM1GxMIGuBgxrBgEFBQcwAcaBoWkXKA6Ly8wQ049RUfQ51KSU4y
MDA4UjJERC1DQSxDj1ESUES0Q49UHWibc1jJTIwS2V5JTIwU2VydmljZXMsQ049
U2VydmljZXMsQ049Q29z2mlndJhGLvbixbRq1fTVUBLERDPW1u2N6Q2VydGlm
aWNhdGU-/VmFzZT9vYmp1V3RdbGfZczi1jZXJ0aWZpY2F0aW9uQXV0aG9yaX5MCER
CSsGAQQBqjUAqQHhLAwBLAGIAwB1AHIAdeB1AHivDQYJkzIiLvcnNAQELOQAD
ggEBAG174IfViEUDXSeiYzaF6kgIKy/YRT8YE+W+6Z7nv+r07nakpDB9N2qKbK
UaRjfJrsjqwG6DKDwCY3GJ-/2pUzKEToSmUOhtu2UymUzzstELKx5WsobAM
d/8ktjmeofW1IJ0sre2c5gdOM7czvKVTh+ud821nlaXL+UJ9udeJ71h0BfKRR
ab7KfasbpPyy1Tbg710elte6wD-0XQbI+cayaBNSc1kLaFVxnO2H8qlgwD
LcLVw/luBROEK/F4Nbms7hpGQFMZ2W5x1QYFuzM6exi/bjC9QRbUnNS3rjJUT
j+7oiwvEUILewtS0bd1R0DwAUSe=
-----END CERTIFICATE-----
```

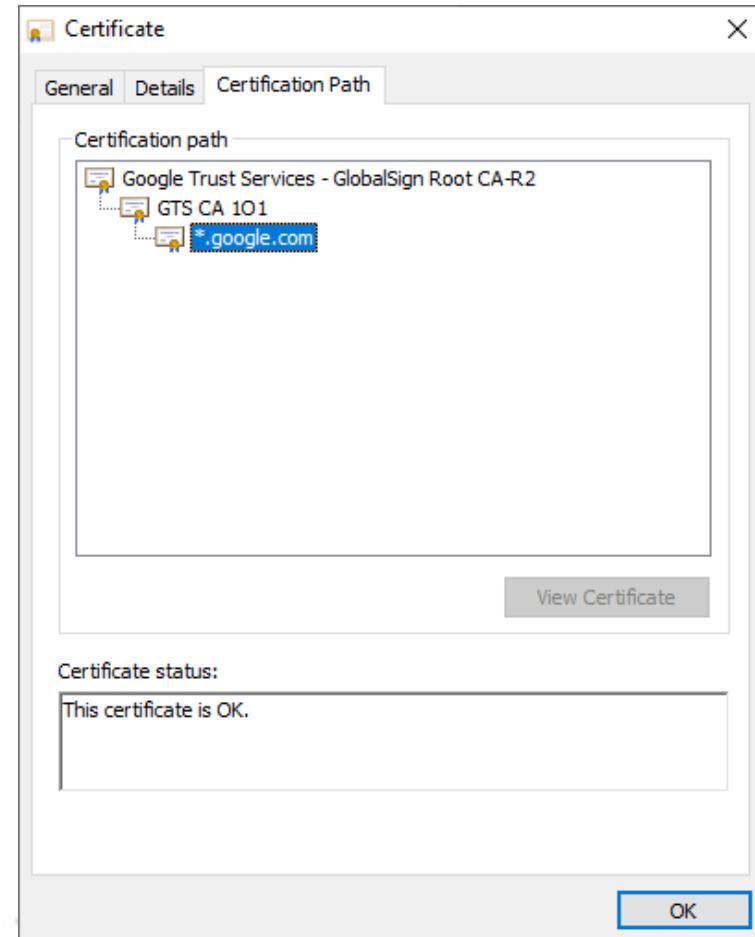
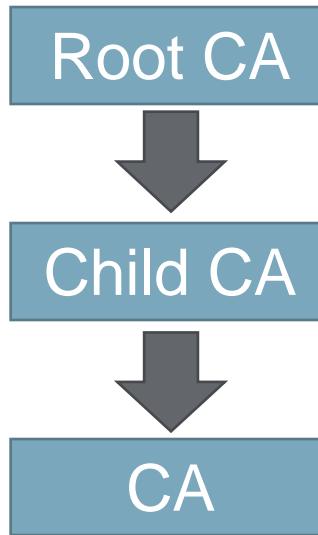
PKCS #7 Certificate example

The focus here is infrastructure



PKI – Certificate Authority (CA)

- A certificate contains a public key along with personal identification information.
- Certificate Authorities (CA) create and manage digital certificates.
- CAs are responsible for providing the following services: verifying identity, creating and digitally signing certificates, delivering certificates, and maintaining certificates over their lifetime.



Example X.509 Certificate



This is an Base64 encoded certificate for Google.com

This is typically “pinned” into many modern web browsers to avoid counterfeit certificates.

Certificate Viewer: www.google.com

General Details

Issued To

Common Name (CN) www.google.com
Organization (O) <Not Part Of Certificate>
Organizational Unit (OU) <Not Part Of Certificate>

Issued By

Common Name (CN) RTX Secure Gateway Services
Organization (O) Raytheon Technologies Corporation
Organizational Unit (OU) Enterprise Infrastructure Services

Validity Period

Issued On Monday, August 7, 2023 at 8:22:44 AM
Expires On Monday, October 30, 2023 at 8:22:43 AM

Fingerprints

SHA-256 Fingerprint	0E 28 00 83 16 6C 1A 7C 4C A9 3F B4 F2 01 A4 2B 73 A8 16 15 28 91 42 CA 43 AF 91 EF 35 B7 35 D0
SHA-1 Fingerprint	A0 E4 D9 94 68 69 F3 72 CE E5 34 2A 6D 4B A3 02 35 36 30 9A

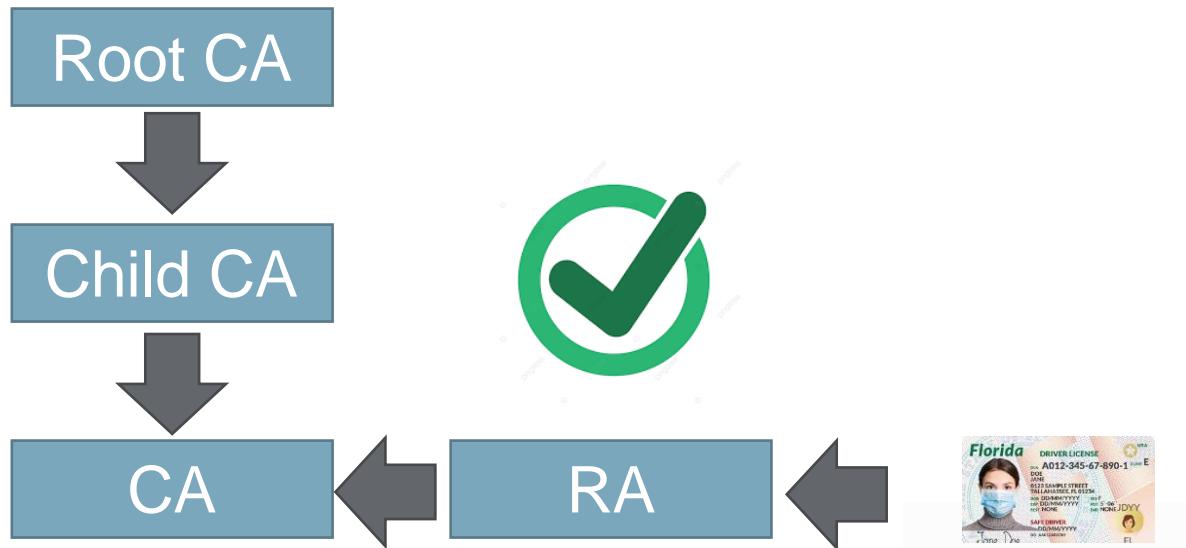
Secure Gate Certificate



PKI – Registration Authority (RA)



- The registration authority (RA) is the component of a PKI which is responsible for accepting requests for digital certificates and authenticating the person or organization making the request
- The RA is responsible for ensuring the integrity of the system but cannot generate an issue certificates.



RTX Corporation (Corporate) - Proprietary

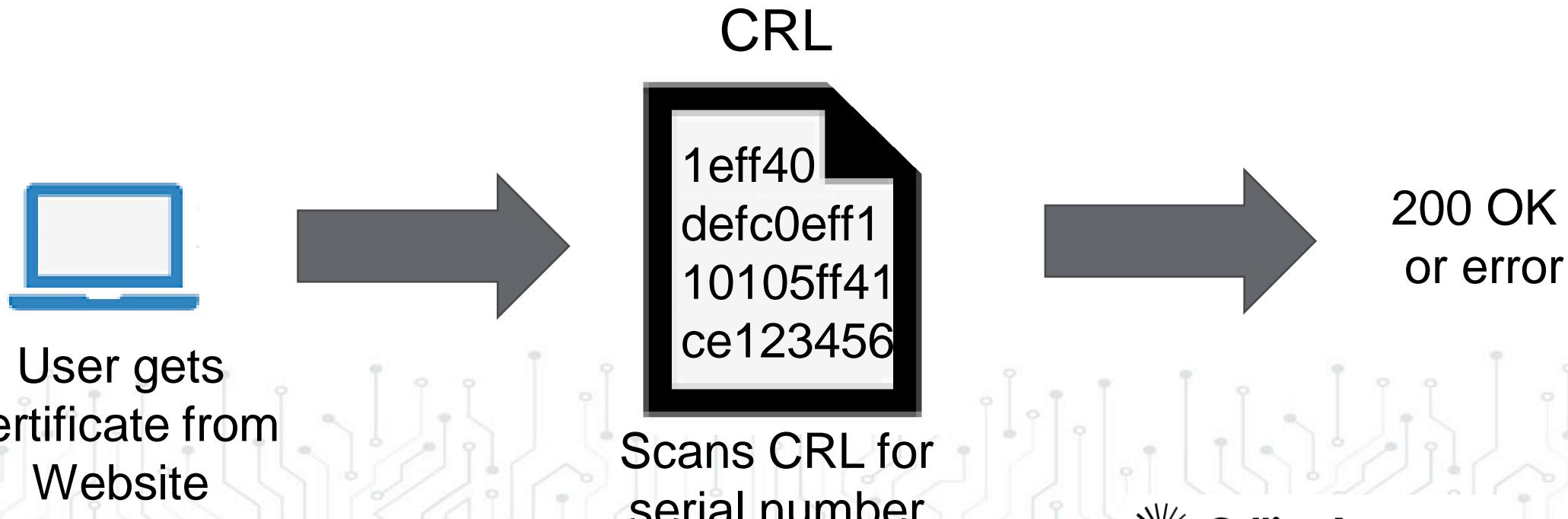
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



PKI – Certificate Revocation List (CRL)

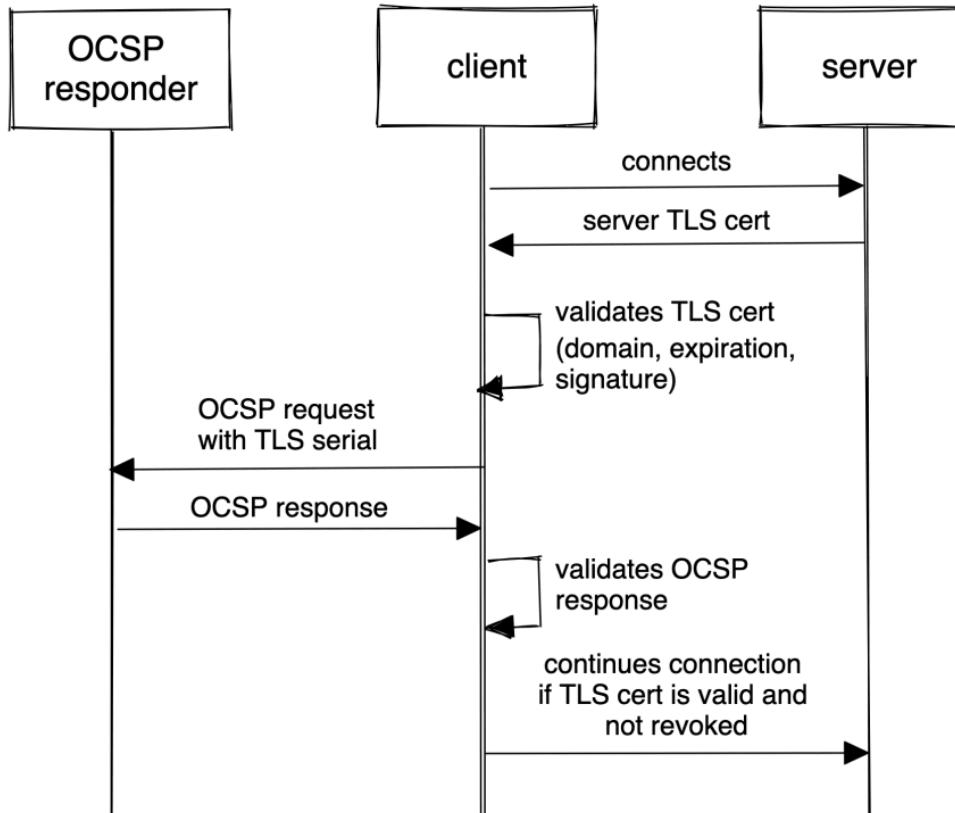


- Sometimes a certificate needs to be revoked such that it is no longer valid or trusted
- Certificate Revocation Lists (CRL) is the mechanism to do so by adding the serial number of a certificate to a list of no-longer valid certificates
- CRLs are not widely adopted or are misused in most PKI solutions available today



Certificate Status Protocol (OCSP)

- OCSP is a protocol for checking revocation of a single certificate interactively using an online service called an OCSP responder.



<https://dev.to/cossacklabs/ocsp-and-crl-what-could-go-wrong-1ped>

Email cryptography

- Privacy-Enhanced Mail (PEM)
 - A standard for secure email across the Internet and within corporations
- Message Security Protocol (MSP)
 - NSA's military only version of PEM
- Pretty Good Privacy (PGP)
 - Freeware cryptographic option uses “web of trust” model rather than PKI
- S/MIME (Secure MIME)
 - Another standard for secure email with attachments



Secure Sockets Layer (SSL) / Transport Layer Security (TLS)



- Secure Sockets Layer is a protocol that was developed by Netscape to secure web transactions at the socket level.
- It has been extended for use with other protocols (like Secure Shell) and is the de facto standard for encryption on the Internet at the application level.
- The latest revision is TLS 1.3



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



IPsec



Internet Protocol Security (IPsec) is a framework that provides the necessary functionality that allows two computers or network devices to securely transmit data.

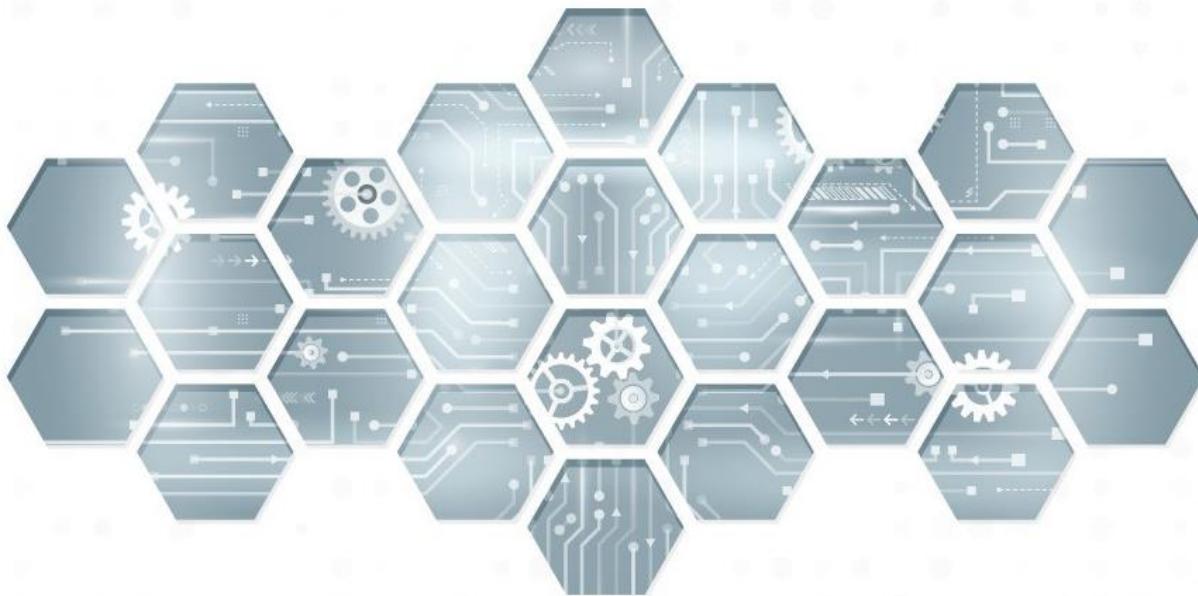
- IPsec is actually a suite of protocols that was developed because IP protocol has no security built into it.
- IPsec provides protection at the network layer of the OSI model.
- IPsec is widely accepted because it is flexible and inexpensive.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





Cryptography

Introduction

History of cryptography

Symmetric cryptography

Asymmetric cryptography

PKI

Conclusion

A note on quantum



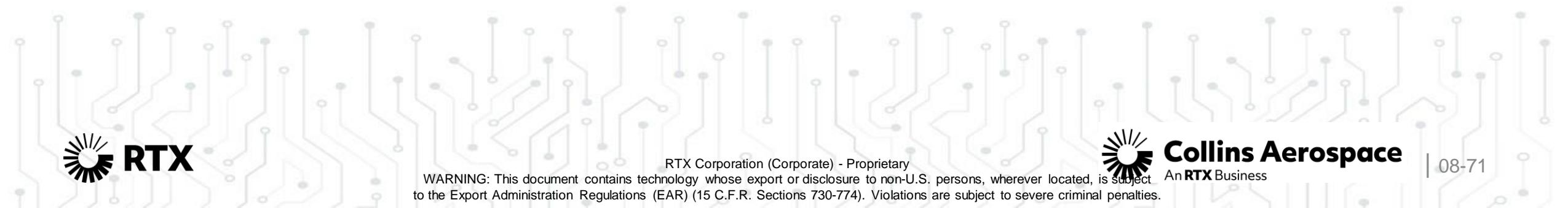
- NIST has selected 4 “quantum-resistant” public-key algorithms
 - General Encryption
 - CRYSTALS-Kyber
 - Digital Signatures
 - CRYSTALS-Dilithium, FALCON, and SPHINCS+
- Quantum computing remains unrealistic
 - But IF it comes true, attack times will be affected as follows:

	Classical	Quantum
Block cipher exhaustive search	$O(\mathcal{K})$	$O(\mathcal{K} ^{1/2})$
AES-256	$O(2^{256})$	$O(2^{128})$
Hash function collision finder	$O(\mathcal{T})$	$O(\mathcal{T} ^{1/3})$
SHA-512	$O(2^{512})$	$O(2^{170})$



Practical considerations

- Plan for cryptographic agility
 - New hashing algorithms
 - Ability to rekey / revoke



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



08-71

Cryptographic standards



Publication	Date	Description
FIPS 140-3	Mar 2019	Security Requirements for Cryptographic Modules
NIST SP 800-38 A-G	Dec 2001	Block Cipher Techniques
NIST SP 800-131A	Mar 2019	Cryptographic Algorithms and Key Lengths



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



08-72

Conclusion



- Symmetric Algorithms
 - Require shared secret key
 - **FAST**
 - E.g., AES-256
- Asymmetric Algorithms
 - Different keys for encryption and decryption
 - **SLOW**
 - Used for Authentication and Secure Key Exchange
 - E.g., RSA, DSA, ECDSA, DH, MQV
- Mitigations for attacks require both logical and physical protections
- Creation of secure embedded systems should utilize all cryptographic algorithm types to protect system boot, as well as data-at-rest, data-in-transit, and data-in-use.
- **NEVER ROLL YOUR OWN! USE APPROVED ALGOS AND LIBRARIES!**



Parting thought



- Cryptographers have a saying: “Everyone can design an encryption algorithm **they** can’t break”
- Keyword is **they**, meaning the algorithm’s designer
- Translation: Just because you can’t think of a way to break your fabulous new encryption algorithm doesn’t mean an expert can’t break it – in 5 minutes
- Derives from the days when developers didn’t want to pay licensing fees or bemoaned the overhead required to encrypt data, so they’d develop their own protocols – to the delight of adversaries
- Moral of story: Always use standard, approved protocols and algorithms
 - Developed by pros with mad math skills
 - Reviewed by experts bent on breaking the algorithm for fame and notoriety
 - Widely used and hence scrutinized



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



1 question — need for speed



Q1: Which is faster — AES or RSA?

- a) AES
- b) RSA



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



| 08-75

Questions?



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



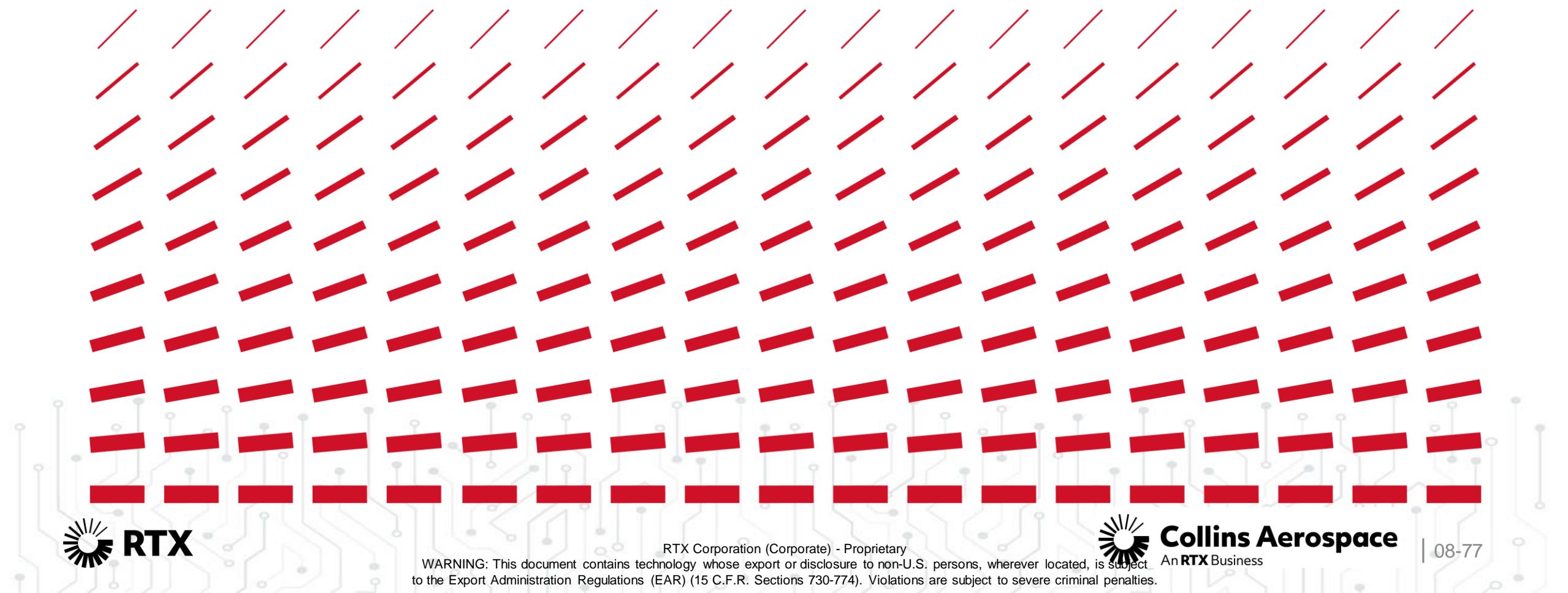
| 08-76

Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

(EAR) WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





Collins Aerospace
An RTX Business



RTX TGE
Technology & Global
Engineering

TGECYBEREMBSEC **Embedded Systems Security**

Module 09

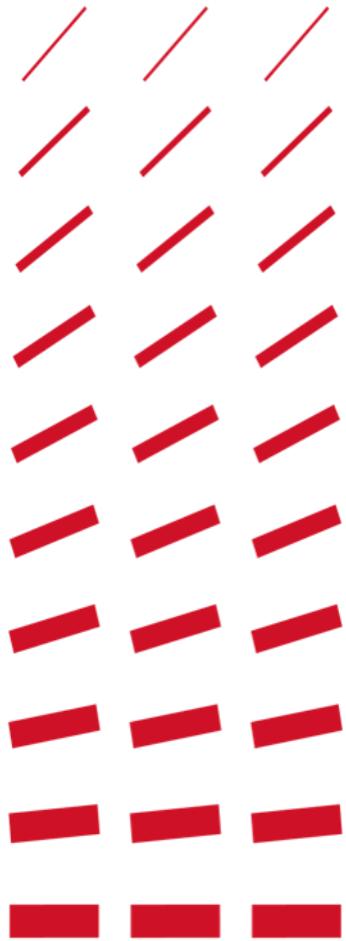
Architecture Concepts and Principles

Instructor: Randall Brooks

Session: 18 | **Date:** Jan. 29 - Feb 2, 2024

Location: Collins, India

Module agenda



Differences between Embedded vs IT systems

Trust, trustworthiness, & trust gaps

Defense-in-depth

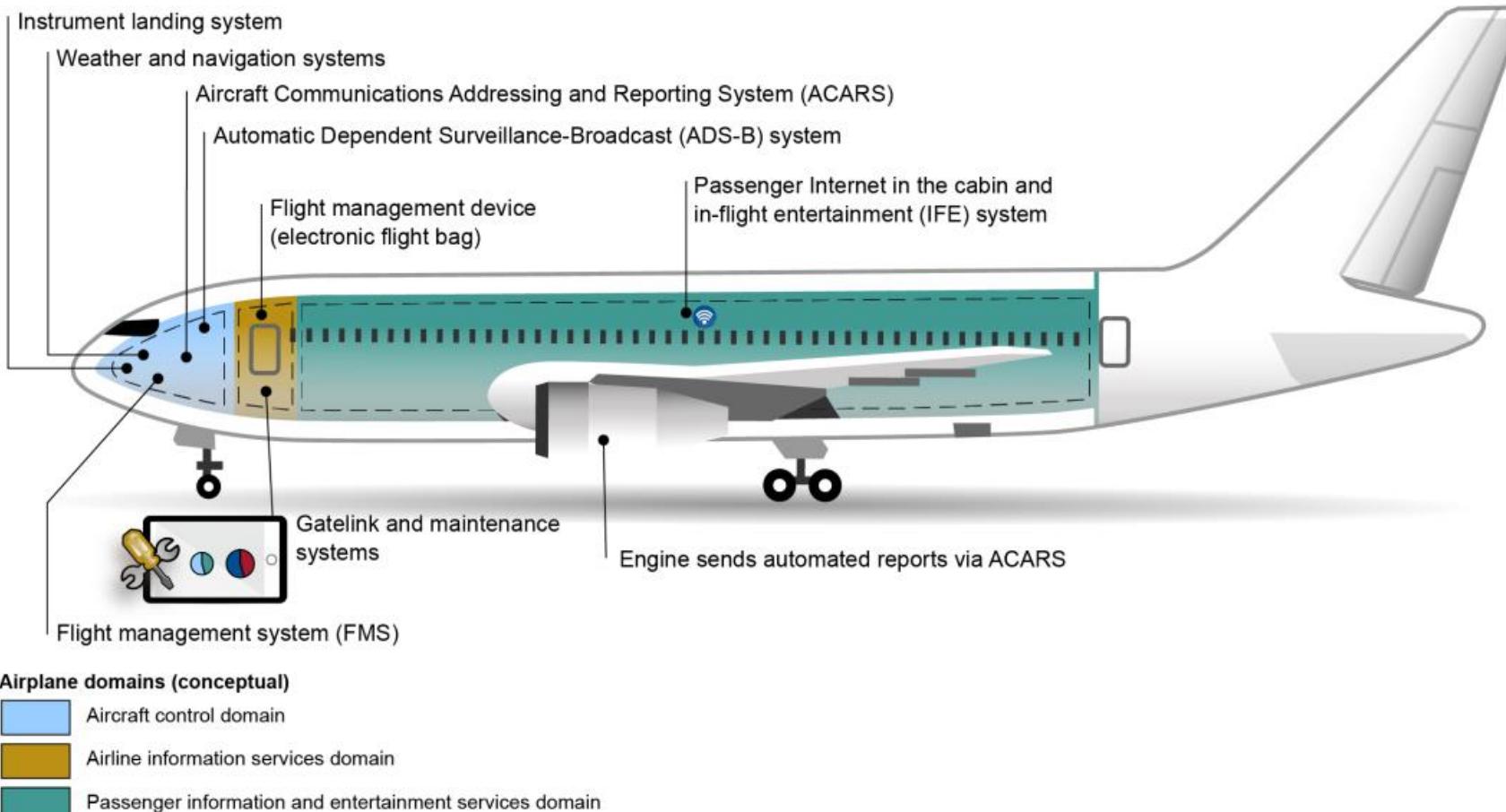
Red/black separation

Trusted Computing Base (TCB)

Embedded system threats and mitigations

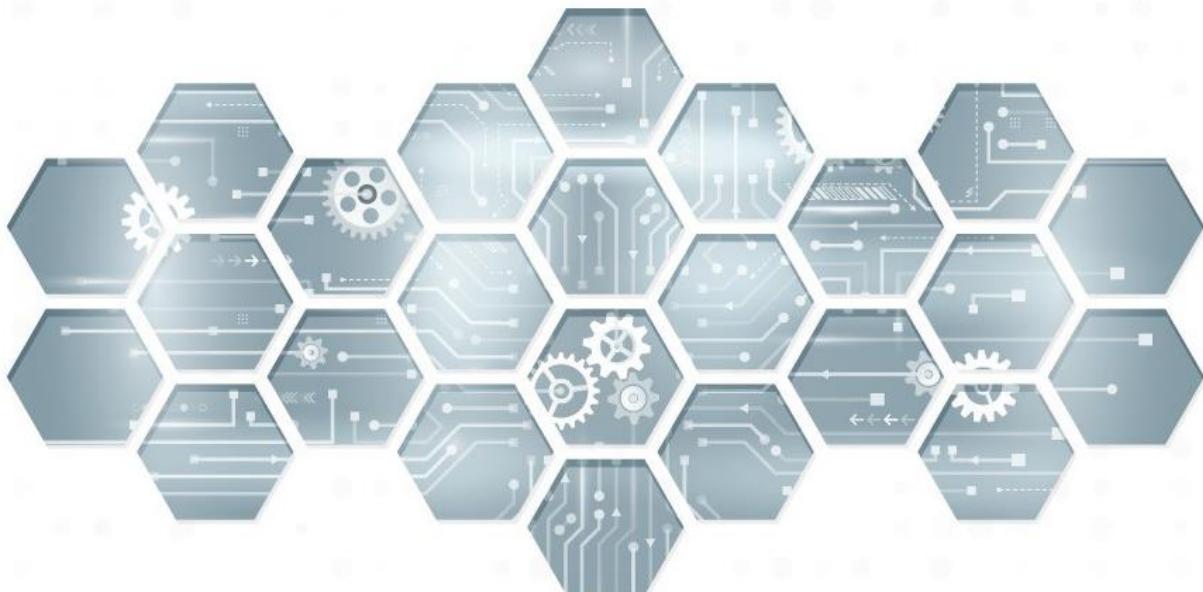


Avionics High-level Architecture (example)



<https://www.gao.gov/assets/gao-21-86.pdf>





Architecture Concepts and Principles

Differences between Embedded vs IT systems

Trust, trustworthiness, & trust gaps

Defense-in-depth

Red/black separation

Trusted Computing Base (TCB)

Embedded system threats and mitigations

Contrasting embedded and IT systems [1/3]



Attribute	IT System (Network Focus)	Embedded System
System Criticality	<ul style="list-style-type: none"> Failures typically more annoying than catastrophic 	<ul style="list-style-type: none"> People die when cyber-physical systems (weapons, chemical plants, power grid) fail Remember this when contemplating the future of self-driving cars
Performance	<ul style="list-style-type: none"> Non-Real Time Consistent response required High throughput demanded High delay & jitter acceptable 	<ul style="list-style-type: none"> Real Time/Time-Critical deterministic safety / mission critical response required Modest throughput impacts acceptable High delay & jitter unacceptable
“Open” vs. “Closed”	<ul style="list-style-type: none"> Execute dynamic arbitrary SW suites Search for connected devices 	<ul style="list-style-type: none"> Execute known, dedicated SW suites Use only expected, known devices
Physical Exposure	<ul style="list-style-type: none"> Typically, less exposed to RE; operate in secured areas or are monitored during use 	<ul style="list-style-type: none"> Often designed to counter RE; can leverage CM to protect Cyber secrets (keys/certificates)

Benefits/Positive Impacts

Challenges/Negative Impacts



Contrasting embedded and IT systems [2/3]



Attribute	IT System (Network Focus)	Embedded System
Architecture Security Focus	<ul style="list-style-type: none"> Protection of equipment and data (at-rest and in-motion) Additional protection for network server 	<ul style="list-style-type: none"> Hardening system against attacks may impact mission critical functions
Resource Constraints	<ul style="list-style-type: none"> Expandable resources enable adding third-party security applications (e.g., AV and message traffic monitors) 	<ul style="list-style-type: none"> SWaP limits resources to those needed to perform intended mission
Operational Availability	<ul style="list-style-type: none"> Reboot acceptable Downtime tolerated 	<ul style="list-style-type: none"> Reboot unacceptable Reliability and redundancy required
Human Interaction	<ul style="list-style-type: none"> Human users typically monitoring system Manual data entry 	<ul style="list-style-type: none"> Usually operate autonomously or with minimal human interaction Less chance to detect attacks, faults, and defects – or for fat finger data entry
User Authentication	<ul style="list-style-type: none"> Multiple methods available to authenticate human users 	<ul style="list-style-type: none"> Few options for verifying authenticity and integrity of autonomous devices

Benefits/Positive Impacts

Challenges/Negative Impacts



Contrasting embedded and IT systems [3/3]



Attribute	IT System (Network Focus)	Embedded System
Communications	<ul style="list-style-type: none"> Standard communication protocols (wired/wireless) 	<ul style="list-style-type: none"> Proprietary and military protocols requiring high transmission reliability/security
Interconnectivity	<ul style="list-style-type: none"> Multiple, discoverable interfaces 	<ul style="list-style-type: none"> Fewer physical, logical connections reduces threat vectors and attack surface
Development Cycles	<ul style="list-style-type: none"> Shorter, emphasize time to market and feature set Less rigorous testing 	<ul style="list-style-type: none"> Longer, emphasizes requirements & reliability More rigorous testing provides more opportunity to find and fix defects
SW Updates	<ul style="list-style-type: none"> Operating systems, drivers, SW still supported by vendors Can apply updates & patches in a timely fashion to address zero-day vulnerabilities (find & patch) 	<ul style="list-style-type: none"> Vendors may no longer support operating system, drivers, or applications System may not support SW updates without refurbishing Must thoroughly test & qualify changes
Life Span	<ul style="list-style-type: none"> 3-5 Years 	<ul style="list-style-type: none"> 15-20 years, possibly in standby mode or storage

Benefits/Positive Impacts

Challenges/Negative Impacts



Line between Embedded Systems (ES) and IT is blurring



- Embedded Systems customers demanding dynamic SW reconfiguration and accelerated SW upgrades to increase operational flexibility, agility, and security
- Increases in net bandwidths/speed and cloud technology are encouraging ES customers to offload processing into the cloud, leverage Internet of Things, opening up traditionally closed ES
- ES customer demanding COTS HW/SW and open standards to reduce cost
- ES are being incorporated as part of a larger System of Systems, which looks like a network of IT systems





Architecture Concepts and Principles

Differences between Embedded vs IT systems
Trust, trustworthiness, & trust gaps

Defense-in-depth

Red/black separation

Trusted Computing Base (TCB)

Embedded system threats and mitigations

Trust and trustworthiness aren't synonymous



- **Trust** represents faith, **trustworthiness** represents confidence that the faith is not misplaced
- Components enforcing a system's security policy (or safety critical) are **trusted** not to fail
 - Trust assesses **consequences** of failure, not likelihood
 - A component is trusted because operators have no choice but to hope it works correctly
 - Components with single point failure **more** trusted as directly leads to security or safety issues
- **Trustworthiness** assesses **likelihood** a trusted element won't fail (betray its trust)
 - Architecture, design, and implementation affect trustworthiness
 - Trustworthiness assessed through design and component analysis
- **Assurance** provides evidence of **trustworthiness**
 - The Oxford Dictionary **assurance** definition: “*a positive declaration intended to give confidence; a promise*” or “confidence or certainty in one's own abilities”
 - Assessment and Authorization (A&A) process determines if system is trustworthy

Trust and trustworthiness often incorrectly used interchangeably.



Real world trust vs. trustworthiness example



- Drivers **trust** their brakes as the only safe means to stop
- **Trustworthiness** is likelihood your brakes **will** work properly – stop your car
- Steps taken to ensure your brakes perform properly – **are trustworthy**
 - Design Analysis: appropriately sized pads, rotors, and tires; redundant hydraulics
 - Test and Analysis: stopping distance, brake fade
 - Maintenance: Regular pad, rotor, fluid, and tire inspection and replacement
 - Supply Chain Risk Management (SCRM): pads, rotors, brake lines, tires, and hydraulic fluid all manufactured to specification
- **Assurance** verifies security requirements, evaluates trustworthiness, and documents results
 - Results of analyses; inspection & maintenance records
- Example illustrates SCRM's role
 - Did your manufacturer install authentic, defect-free brakes meeting specifications?
 - Is your mechanic honest and competent?
 - Did he install the correct pads, tires, and fluid or cheap knock-offs?

Designing for trustworthiness...



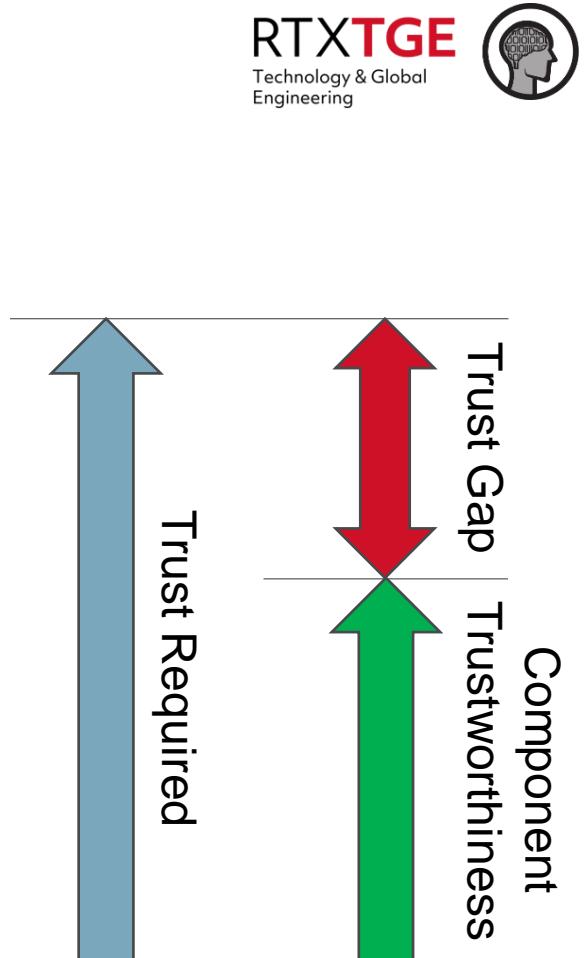
- ES generally more trusted, not necessarily more trustworthy, than IT systems because
 - ES often both safety and security critical – bad things happen when they fail
 - ES consequences of failure more severe than IT systems
- Achieving commensurate trustworthiness is a key ES design & development requirement
- Design for trustworthiness philosophy – **balance** trust & trustworthiness of security (and safety) critical design elements
- Balance means matching trustworthiness to trust
 - Too little risks intolerable failure rates; too much risks intolerable cost
 - Could develop all ASICs, boards, & SW in house to enhance trustworthiness – at astronomical cost
 - Omitting Cybersecurity avoids direct cost and performance impacts – but exposes CPI

Designing for trustworthiness means eliminating trust gaps.



Trust gap: the delta between trust and trustworthiness

- Gaps occur when a security critical component is unworthy of the trust placed in it
 - Insufficient evidence a component will perform as required **OR**
 - Sufficient evidence (red flags) component **won't**
- Trust gaps emerge when system lifecycle processes (design, development, operations, maintenance) are not robust enough
 - Includes supply chain failures



Finding and mitigating trust gaps is critical for Embedded Systems Security.

Key supply chain issues affecting trustworthiness



- Counterfeit parts which fail to operate as expected – or simply fail
- Discovery of existing, previously unknown, exploitable features – zero days
 - Design/implementation errors; incompletely understood system interactions (Spectre & Meltdown)
- Product alterations during manufacturing by insiders or external adversaries
 - Deliberate HW/SW malware insertion targeted at specific products, programs, or industries (**China motherboard hack**)
 - Deliberate but untargeted malware insertions – backdoors, exploitable defects, kill switches, inserted vulnerabilities
- Manufacturing often spans vendors, sites, and countries
 - Provides insiders/adversaries opportunities to deliberately alter products
 - Risk typically proportional to vendor country and workforce nationality
 - Complicated manufacturing processes increase risk of insider error/attack



Finding trust gaps: identify trusted elements



- Identify **trusted** security critical elements using design & reliability analysis techniques
 - Fail Safe Design Analysis (FSDA)
 - Failure Mode, Effects, and Criticality Analysis (FMECA)
 - System-Theoretic Process Analysis (STPA) for safety, STPA-Sec for security
 - Single point failure analysis
- Goal is identifying single point and high consequence failure points
 - Mechanisms controlling data access; e.g., user authentication, memory protection, privilege levels
 - Failure impact – does failure mean immediate data leaks or modifications or do other components also need to fail?



Finding trust gaps: assess trustworthiness



- ***Architecture and Design***
 - Does the system have a threat model?
 - Are components organized with regard to the criticality of their functions?
 - Are interfaces between components well-defined and as narrow as possible?
- ***Implementation***
 - What languages are used to develop the system software?
 - What security-focused testing is performed?
 - How large and complex are the software and hardware?
- ***Supplier assessment:***
 - Are open source packages well-maintained? Have they had security updates before?
 - Do suppliers understand the security requirements? What is their track record with responding to security concerns in their products?
 - Are suppliers / developers based in other countries? Which ones?

Mitigating trust gaps



- ***Reduce trust*** through architecture/design changes...
 - Distribute security functions across multiple elements to minimize single point failures
 - Redundancy always a good security practice
 - Reduce consequences of failures/attacks
 - Restrict component privilege and access
- ***Enhance trustworthiness*** (assurance)
 - Use trustworthy vendors and/or components
 - Ex: Use assured Software (SW) over Free and Open Source Software (FOSS)
 - Verify Hardware (HW) & SW authenticity
 - Avoid overly flexible, complicated components (components should do what they are supposed to and nothing else)
 - Use US vendors, buy from authorized sources, inspect/test received HW/SW
 - Employ trusted development (e.g., SwA), trusted sources, Cybersecurity-SCRM
 - Employ technical countermeasures
 - Periodically verify critical SW, system configuration, and trusted element functionality
 - Disable alternative paths (bypasses)¹



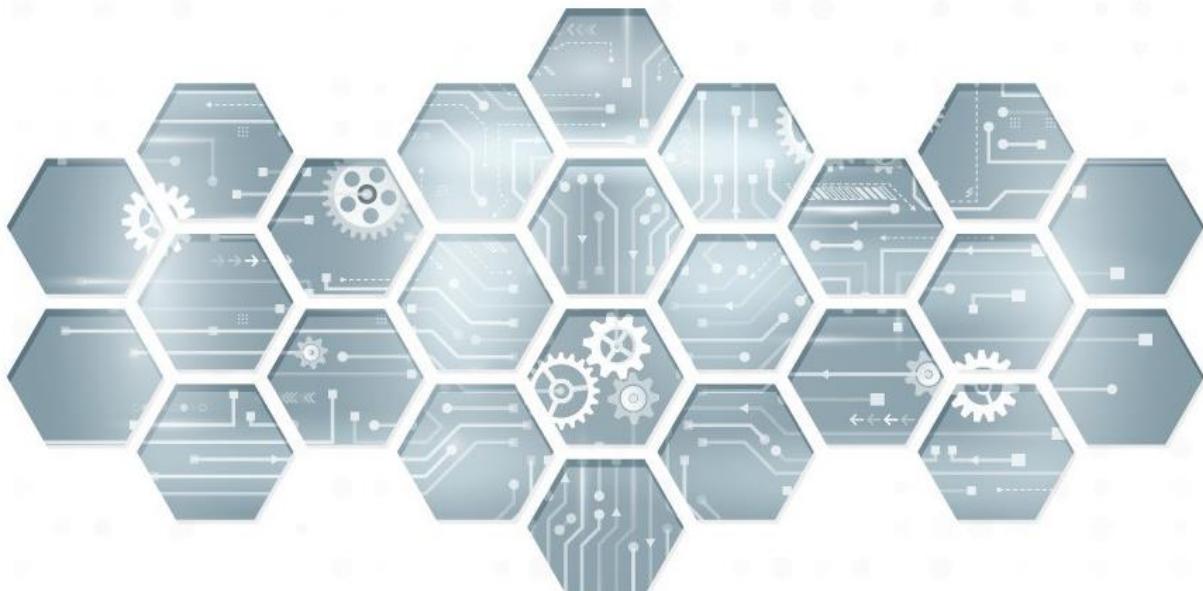
Real world trust gap examples



- China motherboard hack: Tiny component added to widely used, Chinese manufactured motherboards provided backdoor into thousands of servers
 - Deliberate attack targeting large server farms
- Spectre/Meltdown: Enables unprivileged SW to read protected memory
 - Operating System (OS) vendors incompletely understood SW interactions with virtual memory mapping HW
- Counterfeit parts found in DoD systems (built by RTX)
 - Caused more by greed than malice
- Russian company (Kaspersky) allegedly used product to spy on customers
- **Remember:** When developing products, you are your own supply chain – just ask Intel about Spectre & Meltdown

Trust gaps result from misplaced trust in components acquired from untrustworthy sources – or bad internal development processes.





Architecture Concepts and Principles

Differences between Embedded vs IT systems

Trust, trustworthiness, & trust gaps

Defense-in-depth

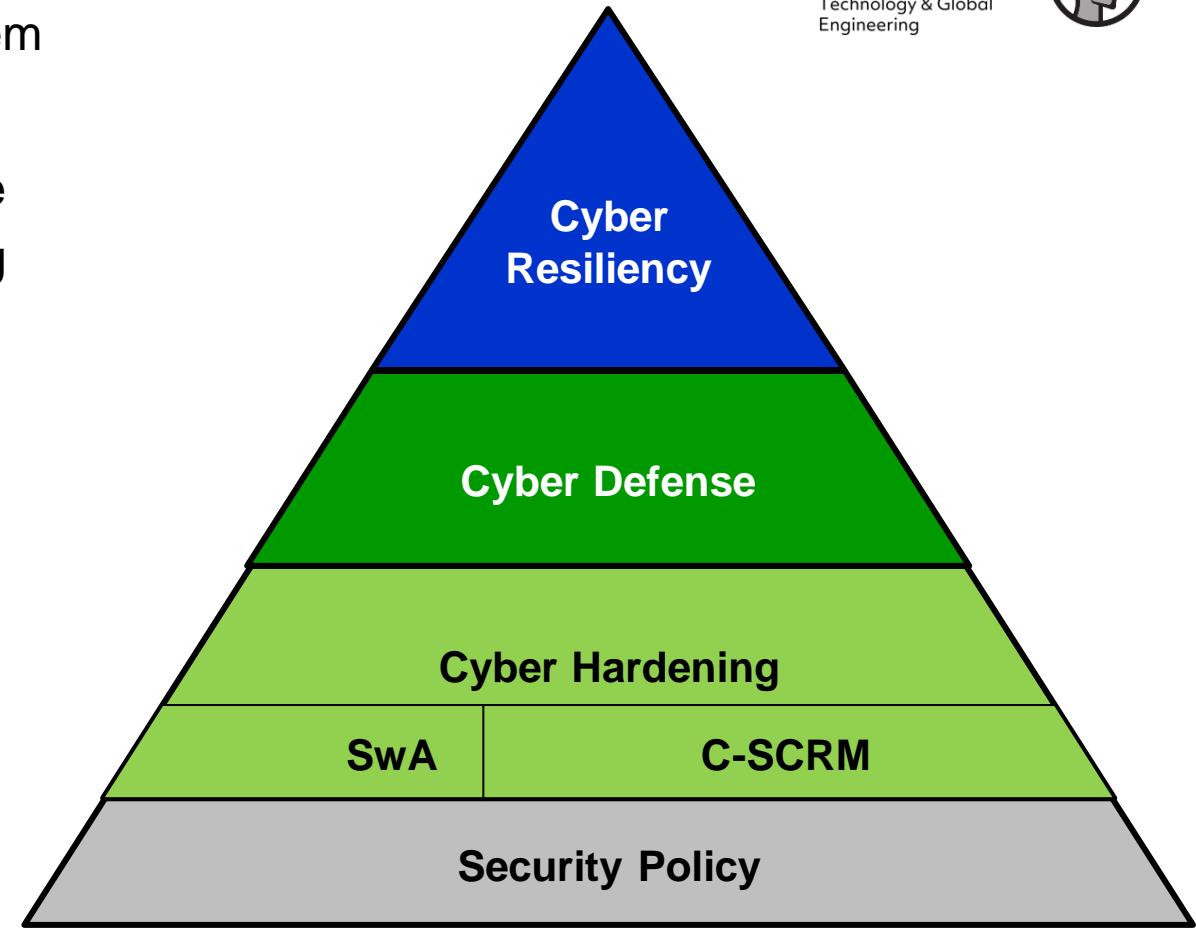
Red/black separation

Trusted Computing Base (TCB)

Embedded system threats and mitigations

Layering defenses optimizes Cybersecurity

- **Security Policy** defines the security rules the system must enforce – Who can access/do what
- **Cyber Hardening** (hygiene) reduces attack surface by eliminating vulnerabilities & threat vectors during design & development
- **Cyber Defense** employs active CMs to prevent, detect, and reverse attacks, faults, & defects (AFD) **effects** during operation
- **Cyber Resiliency** techniques actively maintains operation even **after** successful AFD
- **Each layer builds upon previous layers**



Objective is continued operation & security policy compliance in a Cyber-contested environment.

Defense-in-depth – the pyramid explained, part 1



- Sufficient, affordable Cybersecurity is a cost/benefit trade – defense-in-depth provides trade options
- No **single** Cyber mitigation or strategy can **cost-effectively** mitigate **all** Cyber vulnerabilities
- Hardening great for eliminating common, known vulnerabilities but...
 - Doesn't scale well – progressively more costly, less effective as code size, system complexity increase
 - Only as strong as weakest link – one defect in one function can imperil entire system no matter how defect free the remaining code is
 - COTS and reusable SW means trusting **other** people's development processes
- Active Cyber defense mitigations can prevent or reverse AFD but:
 - May increase system complexity & NRE and consume valuable system resources
 - Recovery may disrupt services
 - Cost effectively mitigating **all** AFD using active mitigations ultimately impractical
- Resiliency techniques (e.g., redundancy) often too resource intensive, costly to protect all apps on resource constrained Embedded Systems



Defense-in-depth: the pyramid explained, part 2



- Good news: Layers are synergistic!
- Hardening reduces known (nuisance) attacks with minimal system performance impact
 - Prevention more efficient than active mitigation
- Cyber defense more effective and efficient as nuisance attacks weeded out
 - Attack recovery overhead and service disruptions reduced
- Resiliency efficacy improves as hardening and defense improves
 - Sufficiently protecting less critical apps with hardening and Cyber defense means can limit resource intensive resiliency techniques to most critical or most susceptible apps
- Each layer's vulnerabilities potentially mitigated by other layers' strengths
 - Swiss cheese analogy
- Can tailor each layer to more cost effectively achieve desired system protection

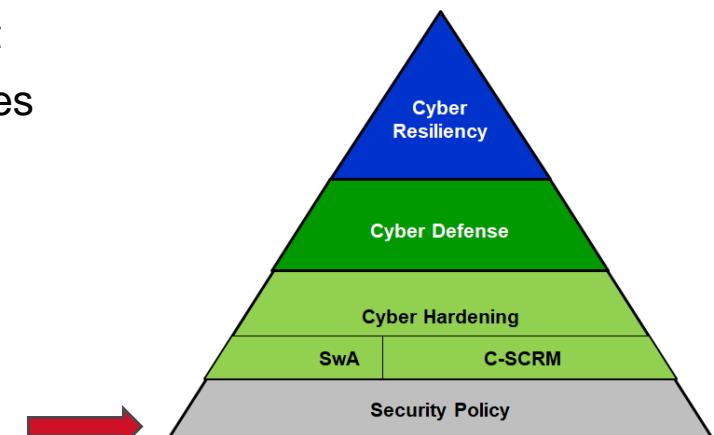
Defense-in-depth provides better, less expensive security.



Security policy expanded



- From NIST: “A set of criteria for the provision of security services.”
 - https://csrc.nist.gov/glossary/term/security_policy
 - Translation: Specifies the entities authorized to generate, access, or modify information of differing sensitivities process by, traversing, or stored on an information system’
 - Security policy guided by RMF security controls
 - System design **must** enforce security policy
- Assessment and Authorization (A&A) validates system correctly enforces its security policy under normal operation; e.g., does system prevent data access by unauthorized users?
- Cybersecurity ensures system enforces security policy in a Cyber contested environment
- Cyber resiliency maximizes likelihood system will successfully complete mission objectives
- The two overlap but aren’t identical as can conflict
 - Preserving security could mean disabling mission critical functions
 - Ensuring mission success while under attack could lead to security violations



Security models formalize specification of security policy



- Examples
 - Bell LaPadula (Confidentiality)
 - Biba (Integrity)
 - Clark-Wilson (Integrity)
- Security Models
 - Specify explicit data structures and techniques or enforce the policy
 - Sometimes backed up by a formal mathematical proof
 - Contain enough detail for implementation
- Security Model Categories
 - Confidentiality Models
 - Integrity Models
 - Access Control Models
 - Information Flow Models

Security policy/model: Bell–LaPadula model



- **The Bell–LaPadula Model (BLP) is a state machine model used for enforcing access control in government and military applications.**
 - It was developed to formalize the U.S. Department of Defense (DoD) multilevel security (MLS) policy.
 - The model is a formal state transition model of computer security policy that describes a set of access control rules which use security labels on objects and clearances for subjects. Security labels range from the most sensitive (e.g., "Top Secret"), down to the least sensitive (e.g., "Unclassified" or "Public").
- **The Bell–LaPadula model focuses on data confidentiality and controlled access to classified information**

The clearance/classification scheme is expressed in terms of a lattice. The model defines one discretionary access control (DAC) rule and two mandatory access control (MAC) rules with three security properties:

1. The Simple Security Property states that a subject at a given security level may not read an object at a higher security level.
2. The * (star)Security Property states that a subject at a given security level may not write to any object at a lower security level.
3. The Discretionary Security Property uses an access matrix to specify the discretionary access control.

Security policy/model: Biba model



- The Biba Model or Biba Integrity Model is a formal state transition system of computer security policy that describes a set of access control rules designed to ensure data integrity.
 - Data and subjects are grouped into ordered levels of integrity. The model is designed so that subjects may not corrupt data in a level ranked higher than the subject or be corrupted by data from a lower level than the subject.
- In general, the model was developed to address integrity as the core principle, which is the direct inverse of the Bell–LaPadula model.

The Biba model defines a set of security rules, the first two of which are similar to the [Bell–LaPadula model](#). These first two rules are the reverse of the Bell–LaPadula rules:

1. The Simple Integrity Property states that a subject at a given level of integrity must not read data at a lower integrity level (*no read down*).
2. The * (star) Integrity Property states that a subject at a given level of integrity must not write to data at a higher level of integrity (*no write up*).^[2]
3. Invocation Property states that a process from below cannot request higher access; only with subjects at an equal or lower level.

Security policy/model: Clark-Wilson model



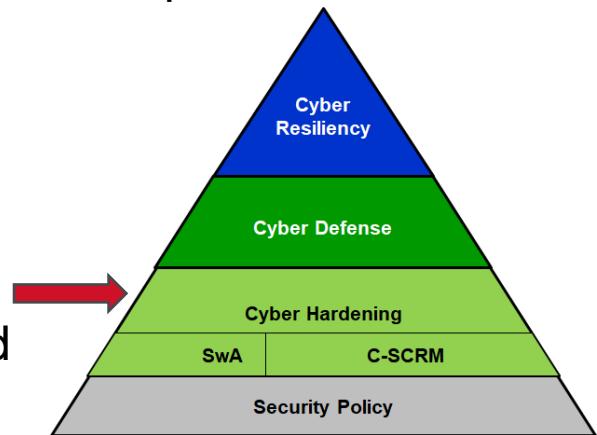
- The Clark–Wilson integrity model provides a foundation for specifying and analyzing an integrity policy for a computing system.
- The model is primarily concerned with formalizing the notion of information integrity.
 - Information integrity is maintained by preventing corruption of data items in a system due to either error or malicious intent.
 - An integrity policy describes how the data items in the system should be kept valid from one state of the system to the next and specifies the capabilities of various principals in the system.
 - The model uses security labels to grant access to objects via transformation procedures and a restricted interface model.
- Uses a three-part relationship of subject/program/object (where program is interchangeable with transaction) known as a triple or an access control triple. Within this relationship, subjects do not have direct access to objects. Objects can only be accessed through programs.



Cyber hardening *REDUCES* attack surface



- Cyber hardening focuses on eliminating vulnerabilities and risk during the design phase
- Software Assurance (SwA) – reduce risk of exploitable vulnerabilities in home-grown HW/SW
 - Trusted development, secure coding standards, design & code reviews, et al
- Supply Chain Risk Mitigation – reduce trust gaps in acquired HW/SW
 - Verify vendors follow a sufficiently trusted development and delivery process
- Secure Configuration (lockdown system) – reduce attack surface by reducing attacker access
 - Theme: Find and disable all unused access points and/or restrict access to minimum required
 - Disable unused interfaces, devices, and features
 - Disable device enumeration (hard code system configuration)
 - Authenticate installed plug-in devices – report missing devices and imposters
 - Authenticate SW before use (see secure boot)
 - Restrict window size on memory mapped buses (e.g., PCIe) to minimum required address space
 - List is general and incomplete – your system may have additional access points



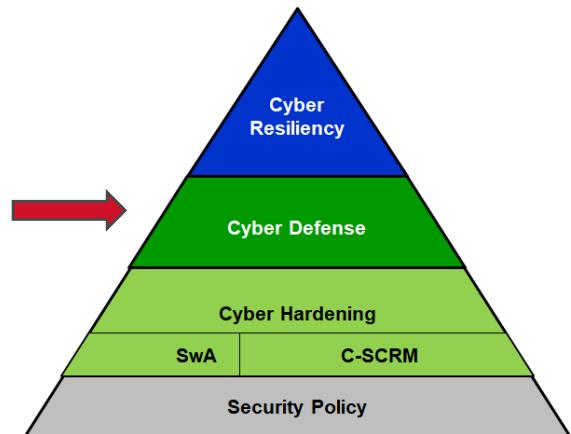
Cyber hardening a cost effective, necessary, but usually insufficient first step.



Cyber defense **ACTIVELY** mitigates Cyber attacks



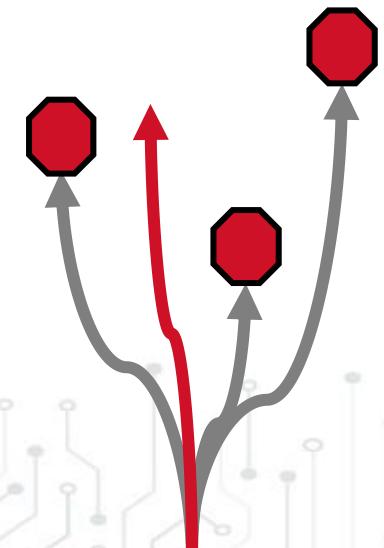
- Cyber defense prevents, detects, and recover attack, fault, and defect (AFD) effects
 - Hardening **eliminates** vulnerabilities, defense **mitigates** vulnerabilities
- **Prevention** extends hardening from static to dynamic means
- **Detection** means actively monitoring system for abnormal events indicative of attacks
- **Recovery** means recovering from AFD
 - Best case, restore system to pre-attack state (Cyber resiliency)
 - Constrain compromise if can't (fail secure)
 - Always leave system in a secure state
- Following slides provide examples of each



Step 1: Prevent attacks from exploiting vulnerabilities



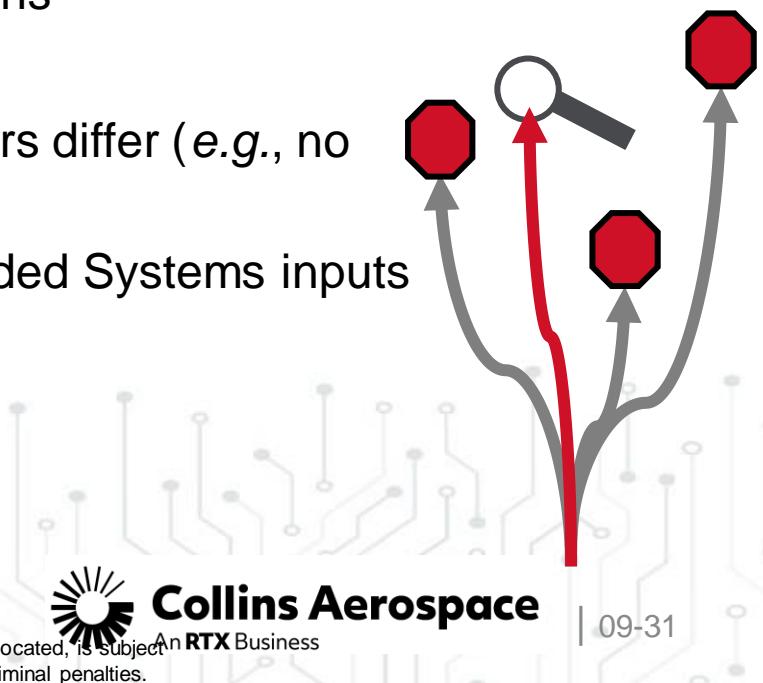
- **Validate data before use**
 - Firewalls, content checkers, range checking, bound checking; typically discard invalid data and report event, but rarely take other action
- **Restrict SW to authorized actions**
 - Use virtualization to limit app privileges and block unauthorized reads, writes, and fetches; use virtual rather than kernel drivers (see OS section)
 - Prevent invalid code execution using Data Execution Prevention (DEP) and Code Flow Integrity (CFI) mitigations (stack canaries, shadow stacks)
- **Employ deception to confuse or mislead attackers, reducing odds of attack success**
 - Ex: Code diversity (e.g., Address Space Layout Randomization (ASLR), stack reordering) modifies address maps or executable binaries between systems, power cycles, or even during a single power cycle, mitigating reusable Return Oriented Programming (ROP) attacks
 - Attacks may instead cause apps to fail; potentially preferred to security violations
- **Resource quotas mitigate inter-app DoS attacks and covert channels**
 - Important for Embedded Systems as resource constraints make more sensitive to resource conflicts
 - Prevention shouldn't disrupt services and should report events, but neither always possible



Step 2: Monitor system for attacks



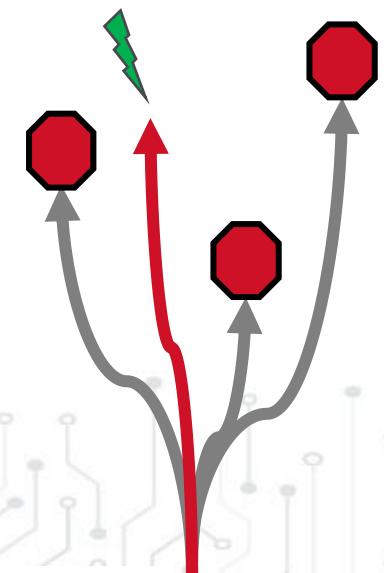
- **Behavioral sensors monitor system activity for abnormal (unexpected) system behavior**
 - Invalid code execution (location, sequencing, timing); out-of-bounds resource usage or message traffic; or policy violations; stack canaries; bandwidth monitoring
- **Integrity sensors monitor system for unexpected integrity changes**
 - SW or configuration modifications; transitory or persistent errors; dysfunctional critical logic blocks
 - Ex: Runtime SW authentication; configuration register monitoring, periodic functional verification of critical system functions, especially security critical functions
- **Virus scanners, intrusion protection**
 - Traditionally less effective for Embedded Systems because attack vectors differ (e.g., no email, browsers, web apps)
 - Embedded Systems resource constraints reduce efficacy; static Embedded Systems inputs increases efficacy



Step 3: Recover from detected attacks



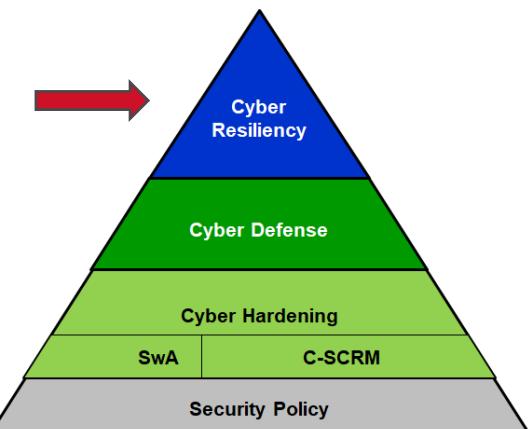
- **Recovery mechanisms include:**
 - Restore: Halting, unloading, reloading, and/or restarting compromised SW
 - Restoring individual affected apps preferred to resetting entire system
 - Quarantine: Disabling interfaces, restricting message traffic
 - Notification: Inform operator and/or external entities (e.g., Cyber command)
 - Audit: Capturing events and responses for forensic analysis – what happened when (audit)
- **Embedded Systems conundrum – when does security trump functionality?**
 - For IT systems, killing compromised apps and/or rebooting is acceptable and, in many cases, preferable
 - A system crash preferable to sensitive data leaking or ransomware encrypting your hard drive
 - Embedded Systems are Cyber physical – halting apps may catastrophically impact real world; e.g., cause mass casualties
 - Answer differs by system – developers, data owners, & operators must cooperatively define reaction policy

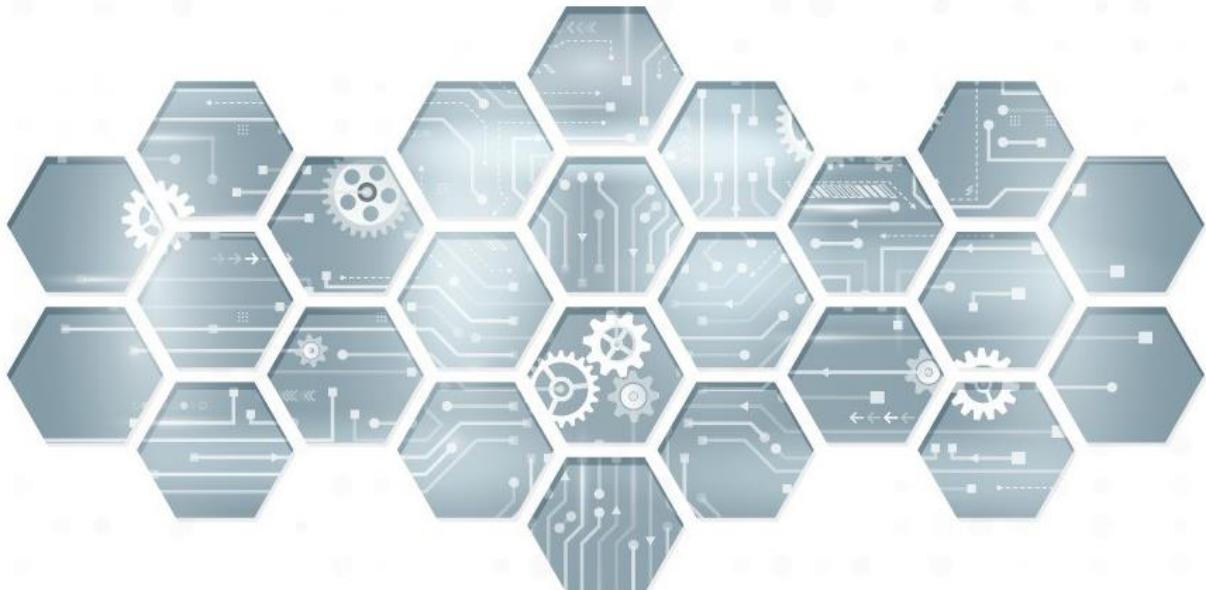


Cyber resiliency fosters operational continuity...



- ... of critical system functions after a successful AFD
- **Dedicated CR techniques synergistically employ redundancy and diversity**
- **Redundancy:** Duplicate instantiations of mission critical functions; requires:
 - Additional, isolated system resources to host redundant copies
 - Means to detect compromised SW (e.g., Cyber sensors) and rollover to backups
- **Diversity:** Sufficient variance between instantiations to keep attacks from compromising all variants
 - Traditionally hosted different SW products performing same function on different platforms
 - Ex: Different database solutions executing on different server platforms with different OS
- **Resource constraints, cost of writing multiple variants of custom Embedded Systems SW traditionally made approach impractical for Embedded Systems, however:**
 - New techniques exist for automatically generating multiple diverse binaries from same source
 - Increasing processor densities will make more resources available to future Embedded Systems
 - Combined, should enable affordable redundancy for at least most critical SW





Architecture Concepts and Principles

Differences between Embedded vs IT systems

Trust, trustworthiness, & trust gaps

Defense-in-depth

Red/black separation

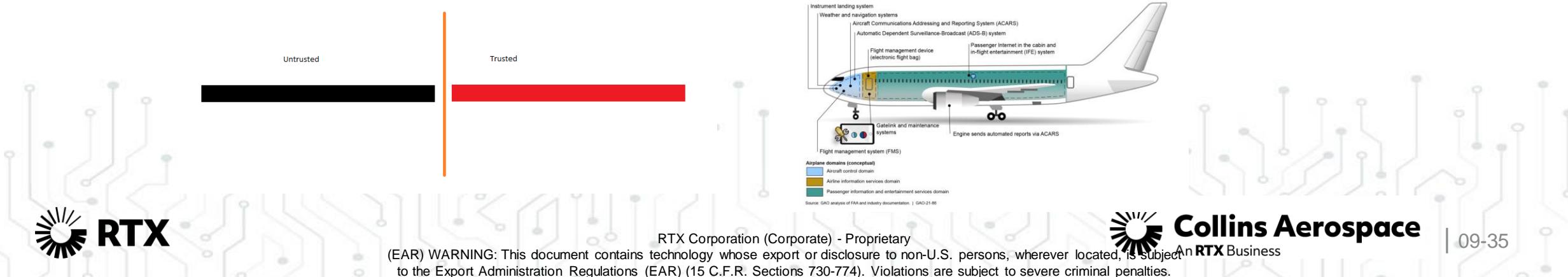
Trusted Computing Base (TCB)

Embedded system threats and mitigations

Red/black separation



- “The red/black concept, sometimes called the red–black architecture or red/black engineering, refers to the careful segregation in cryptographic systems of signals that contain sensitive or classified plaintext information (red signals) from those that carry encrypted information, or ciphertext (black signals). Therefore, the **red** side is usually considered the internal side, and the **black** side the more public side, with often some sort of guard, firewall or data-diode between the two.”
 - https://en.wikipedia.org/wiki/Red/black_concept
- Red/black separation simply means systems must prevent unencrypted, red data from mingling with encrypted, black data on computing devices, networks, and storage media to prevent:
 - Mixing red & black data on networks or storage media (data leaks)
 - Key recovery – recovering both red and black data enables attackers to extrapolate key



Encryption models



- Encryption traditionally performed by external device
 - Newer processors may have integrated encryption HW – or use SW encryption
 - Ex: Newer Intel processors have encryption acceleration instructions
- In-line encryption: Host passes red data to encryption device who sends black data directly to peer
 - Device typically has physically separated red and black ports
 - Preferred as avoids returning black data to source processor, but not always possible
- Encryption as a service: Host passes red/black data to encryption device/SW which returns black/red data
 - Mixes red and black data on host processor
- When using symmetrical encryption algorithms (e.g., AES) encryption/decryption uses same algorithm and key
 - Red data is encrypted; black data decrypted



COMSEC CONOPS



- Encrypts data exchanged between host devices over a network
- External encryption device:
 - Usually in-line – device placed between host and network – but black data may return to host when host controls interface (e.g., 1553)
- Internal encryption device/SW models vary:
 - Some processor's encryption HW routes black data directly to interfaces, mimicking in-line encryption
 - Others only encrypt internal memory buffers, leaving host to handle black data
 - SW libraries encrypt/decrypt data in memory,
 - Best case, library is a separate SW component and transfers data in/out of system (e.g., Transport Layer Security [TLS])



Mass storage encryption CONOPS

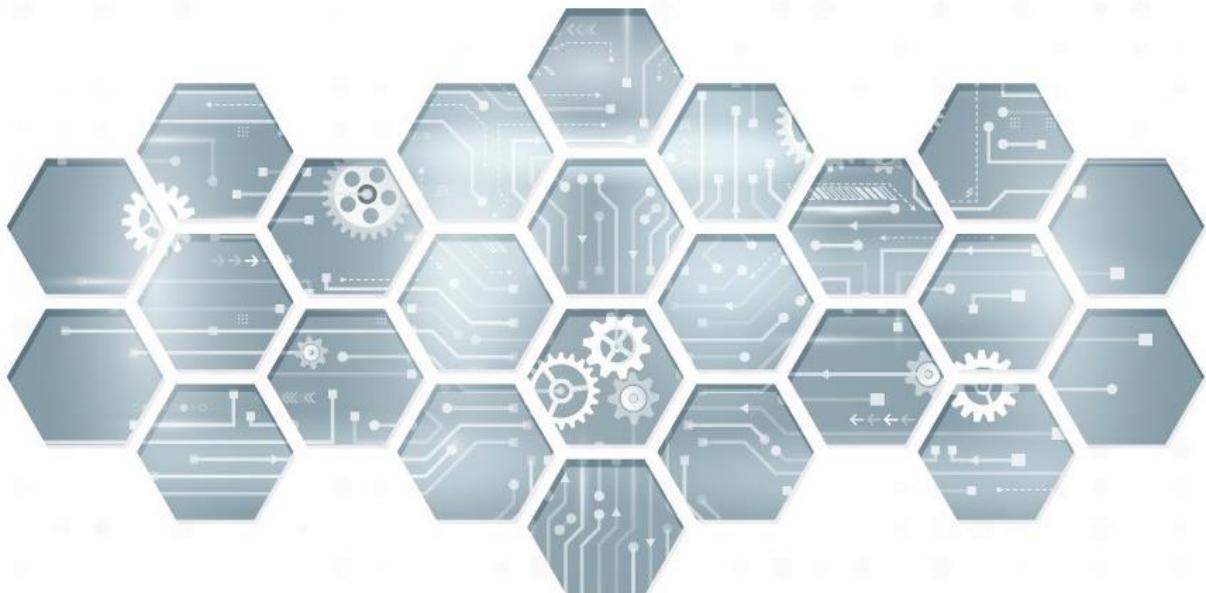


- Data encrypted while on a mass storage device (e.g., hard drive)
- File Encryption/Decryption (FED): File encrypted/decrypted before/after written to/read from storage
 - Ideally, an intermediary encrypts/decrypts files – user only handles red data
 - Less ideal: apps encrypt/decrypt files before writing/after reading – handle red & black data
 - FED excels at handling bulk and pre-encrypted files, but impractical for handling random access files
 - Would need to decrypt and re-encrypt file every time it's changed
- In-line encryption: Storage device encrypts/decrypts **sectors** as written/read to/from device
 - Devices typically encrypt/decrypt individual disk sectors as written/read, not entire logical entities like files
 - Enables encrypting any data written to mass storage device
 - Excels at handling random access files, but doesn't support FED
 - Downside: data automatically decrypted with read from mass storage device → can't directly export encrypted data to other devices
 - Can write encrypted files to mass storage with in-line encryption, just super-encrypts files on device

Red/black separation design considerations



- Treat red and black data as separate domains, never intermingle
- Avoid storing red and black data on same device as can enable attacker to recover keys
- Segregate encryption and app SW, especially if can't split red/black between devices
 - Use encryption SW to move red and black data between red/black domains and SW stack layers
 - Ex: App writes red data to encryption engine, engine writes black data to TCP/IP stack and vice versa
 - Approach helps avoid mixing red/black in SW stack
- Don't use same interfaces (internal controller, port, bus) for both red and black data
 - External encryption devices all have separate red/black ports, but CPUs often connect multiple IO ports to same internal controller; don't compromise separation by using same CPU interface
- Key management
 - Keys classified at level of data they protect – isolate & protect accordingly, preferably in secure HW
 - Isolate keys from applications; e.g., indirectly reference keys rather than handle physical keys
 - Ex: Use a secure key store only accessible to encryption SW and reference keys by handle
- Follow latest FIPS 140-3 specification for crypto usage, especially those related to key management



Architecture Concepts and Principles

Differences between Embedded vs IT systems

Trust, trustworthiness, & trust gaps

Defense-in-depth

Red/black separation

Trusted Computing Base (TCB)

Embedded system threats and mitigations

What is a Trusted Computing Base (TCB)?



- TCB definition from Wikipedia
 - *"The trusted computing base (TCB) of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system. By contrast, parts of a computer system that lie outside the TCB must not be able to misbehave in a way that would leak any more privileges than are granted to them in accordance to the system's security policy."*
 - https://en.wikipedia.org/wiki/Trusted_computing_base
- Policy jointly enforced by TCB and admin/OPSEC procedures
- “Trusted” means TCB trusted to enforce policy per earlier definition
 - TCB protects the data processed, traversing, and at rest on a computing system
 - Must design TCB to actively enforce security policy – apply appropriate security controls
 - Security control selection guided by Risk Management Framework (RMF)
- OPSEC defines rules operators must follow to prevent policy violations

Representative ES TCB architecture



Tactical Applications

Middleware

Security Controller

Trusted Kernel (OS)

Board Support Package

Microprocessor

Root of Trust

Trusted Computing Base

- Rarely directly participates in enforcing policy, *i.e.*, are untrusted
 - TCB protects and enforces system security policy per application
- Optional: Provides non-security related services like messaging and file management; related security policies enforced by TCB
- Performs security critical management and/or policy enforcement tasks
 - Ex: audit, user authentication, secure messaging, secure file access, crypto
- Limited to performing only those system tasks requiring highest privilege
 - Ex: Task management (resource allocation, scheduling), semaphores, IPC
- Authenticates, tests, and configures processor and module; loads OS
 - Usually transitory (overwritten by OS) but not always
- Inherently untrustworthy; but provides trusted functions
 - Ex: Access control (MMU), security features
- The basis for system trustworthiness
 - Best root of trust is a secure HW device holding secrets for authenticating the system and its SW



Secure boot prerequisite to establishing a TCB



- Systems transition from powered down to operational following a boot process
 - At a high-level, boot configures system HW and loads system SW
- Systems ideally would only execute authenticate SW on known good, correctly configured HW
- Secure boot is the means for ensuring systems achieve a secure (trustworthy) operational state
- Secure boot is typically divided into phases, with each phase authenticating the HW/SW used in the next phase, forming a chain of trust from power on through normal operation
- Secure boot is especially important to ES, especially systems operating in exposed locations

General ES secure boot chain



Phase	Typical Tasks
Processor boot logic (HW)	Authenticates, loads, and initiates boot code, may partially authenticate processor first
Boot loader (FW/SW)	Authenticates processor, performs initial processor configuration, performs initial processor built in test (BIT), authenticates and loads BSP
Board Support Package (BSP)	Authenticates motherboard, performs initial board configuration, performs final BIT, authenticates and loads OS and security SW ²
Operating System (OS)	Performs final system configuration; loads user SW, ideally after authenticated by security SW; assigns and privileges to user processes; provides system services. A key function is configuring the MMU to prevent individual processes from accessing each other's memory.
Security SW (Controller)	If present, authenticates users SW before OS loads, monitors system (e.g., Cyber sensors), provides security services (e.g., SW authentication, crypto, audit). OS are starting to include these functions or allow developers to add these functions through drivers.



Root of Trust (RoT): secure boot (“chain of trust”) anchor



- A **Root of Trust** (RoT) provides a trusted, secure starting point for secure boot
 - Provides at least cryptographic services and key storage to enable verifying initial boot code
 - May also contain SW verification keys (or each boot phase contains keys for verifying next phase)
 - May also contain HW “measurements” to verify CPU is unchanged since manufacture
- **Trusted Platform Modules** (TPM) are simplest RoTs, containing cryptographic services, a secure key store, and secure measurements
 - TPMs widely used in IT systems, especially Intel based systems
 - Intel CPUs interact with TPM during boot to load and authenticate microcode and BIOS
 - Freescale processors employ a comparable Trusted Boot Architecture
- More sophisticated RoTs may contain logic to boot, configure, and/or monitor the main processor



Microprocessor (μ P) security features: access control



- Privilege levels limit process access to sensitive registers or memory (kernel vs. user space)
 - PowerPC supports hypervisor, supervisor, and user levels
 - Intel supports rings, with ring -1 equivalent to hypervisor, 0 equivalent to supervisor, and 3 equivalent to user
- Memory virtualization dedicates system memory to processes and defines access
 - Most μ P enable controlling memory access by privilege level
 - Most μ P enable designating memory as data only or executable (prevents code injection attacks)
 - Some μ P (e.g., PowerPC) enable designating memory as read, write, and/or execute
 - Also enables μ P to moderate IO device access on memory-mapped buses (e.g., PCIe)
- Most μ P moderate access to embedded IO devices using dedicated access control registers
 - Intel uses HW virtualization (VT-x); PowerPC uses model specific registers
- While processor provides raw capabilities, OS controls usage
 - Some RTOS operate in flat memory model – all processes can access all memory/devices
 - Developer usually needs to allocate memory and/or define access/privileges to each process

Access control contains Cyber attacks and enables domain separation.



µP security features rapidly evolving



- **Cryptography**
 - Embedded encryption engines, acceleration instructions
 - DES, AES, and SHA common; private key encryption less so, except for large math operation support
 - In-line memory encryption – protects information during processing
 - Integrated key material
- **Integrated Cyber countermeasures**
 - Buffer overflow mitigations: HW bounds checking and shadow stacks (Intel)
 - Code modification mitigations: authenticated (secure) boot, Intel SW Guard eXtensions (SGX), Intel Trusted eXecution (TXT), Freescale Run Time Integrity Checking (RTIC)
- Features not universal – limited by vendor and product lines
- ES developers should include security features in processor trades; leverage available security features, especially authenticated boot

Board Support Packages (BSP) vs. BIOS vs. UEFI



- **BSPs** historically used by non-x86 processors to test and configure ES HW and load the RTOS
 - Configuration typically static, BSP configures only expected devices to predefined values
 - BSP usually transitory – overwritten by RTOS, resources reclaimed for user SW
 - Configuration sometimes split between BSP and RTOS
 - Ex: BSP may only configure RAM and load the RTOS, which configures the rest of the HW
- **BIOS** historically used by x86 (Wintel) systems to test and configure HW, plus find boot device
 - BIOS loads boot code from boot sector and executes, boot code loads OS using BIOS services
 - Configuration typically dynamic – BIOS discovers and configures any connected peripherals rather than configuring only developer specified devices
 - Also provided low level HW interface services, but Windows has replaced these services with UEFI
- **UEFI** provides a standard HW abstraction layer, providing standard Windows drivers a common API to low level HW interface firmware
 - Enables the same drivers to work across multiple systems rather than requiring a unique driver for every PC peripheral



Security ramifications of BSP/BIOS differences



- BSP generally simpler than BIOS, reducing certification cost and risk
- “Smart” BIOS peripheral and boot device discovery enable additional attack vectors and/or enable attackers to hijack boot whereas “dumb” BSPs simplify system lockdown
- BSPs simpler to write than BIOS
 - BIOS complexity drives most to buy rather than make, but all 3rd party BIOS vendors do the majority of their development overseas: AMI and Phoenix Technologies in India; Insyde Software in Taiwan
 - Microsoft recently released (12/18) an **open source** UEFI version
 - Intel has a proprietary UEFI version available under license, pedigree unknown to author
- A UEFI and Windows compliant BSP would provide a more secure solution than any of the commonly available BIOS products



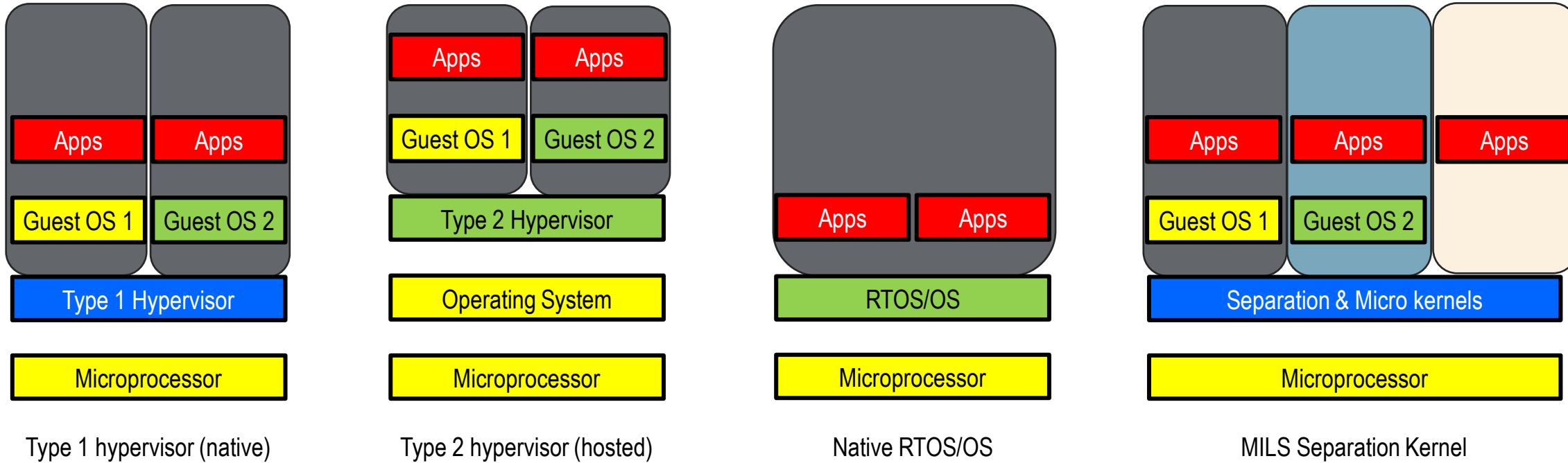
HW Design Guidelines for Embedded Systems Security



- Avoid connecting unused ports as could provide additional attack vectors
- Don't connect device JTAG/debug ports on production boards
- Maintain separation between buses and controllers
 - To aid domain separation, use separate rather than shared IO controllers
 - Ex: Freescale T4240 supports 8 Ethernet ports, but only has 2 controllers (each controller supports up to 4 channels)
 - Connect both controllers even if using less than 4 ports to allow dedicating each controller to a separate domain
- Allocate red and black data to separate buses – and controllers
- Select CPUs that verify boot code (support secure boot)
- Install separate HW RoT as practical



Embedded Systems OS models



- Key differences between OS models are
 - How are applications isolated – and how trustworthy is the mechanism?
 - Driver/utility model – does OS employ kernel (privileged) or virtual (unprivileged) drivers

OS model impact on security



- Enterprise OS include all MW, trusted & untrusted, shown in TCB diagram with the kernel
 - Often include all device drivers, further increasing code size and risk
 - All OS SW operates at kernel privilege, increasing risk of OS compromise and privilege escalation
 - Must evaluate entire (large) code base to achieve certification, an expensive task with uncertain outcome
- Previous diagram illustrates a more secure model
 - splitting the OS into
 - A small, highly trusted kernel; the only SW with highest privilege AND
 - Trusted utilities and drivers operating at lower privilege levels tailored to their function AND
 - Typically can only access the data and resources needed to perform their function rather than all system resources
 - Untrusted utilities and drivers with only user privilege
 - Many drivers and utilities only need user level privileges; integrating with the kernel unnecessarily increases their privilege and kernel's exposure
 - Reducing size of most trusted, highest privileged code base reduces certification cost/risk – and risk of privilege escalation attacks
 - Limiting privilege to minimum required is a fundamental security principle

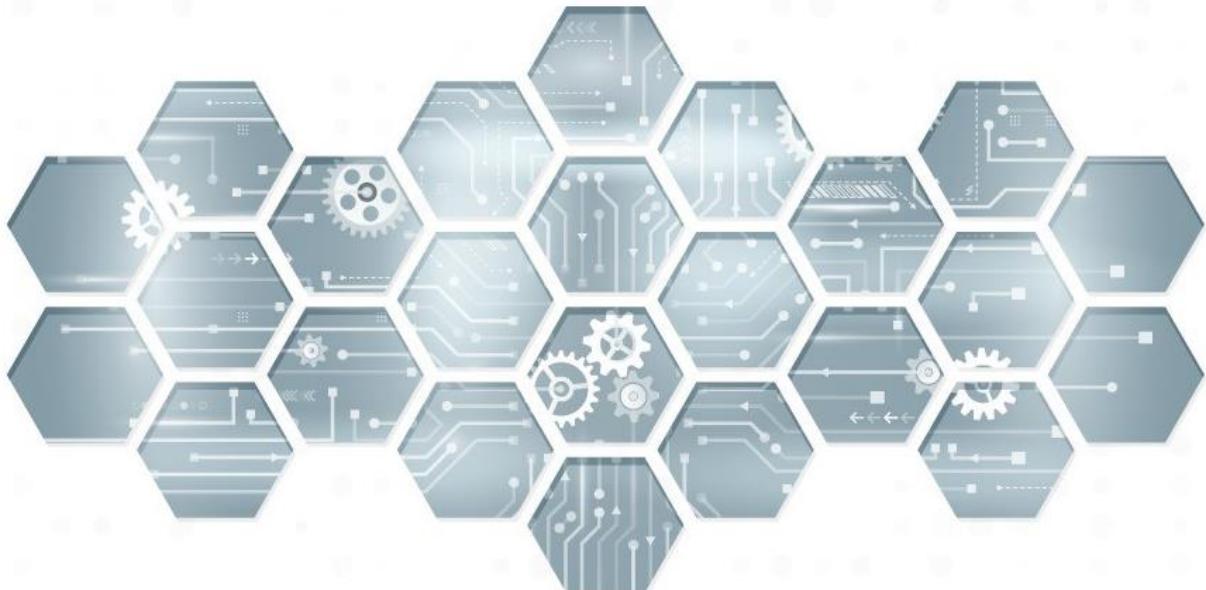


Design consideration: fail secure



- A TCB should “fail secure” – revert to a “known secure state” when adversely impacted by AFD
 - A “secure state” is any system state which still prevents security policy violations
 - Ideally system can restore itself to normal operation
 - If can’t, must still protect hosted data
- Contrast with “fail safe” – system reverts to a known safe state upon failure to protect life & property
 - Similar concepts, but objectives often conflict, especially for Embedded Systems
- In the extreme, the most secure system is a dead system
 - An obvious conflict with resiliency
 - And a possible conflict with safety
- Designers must balance security, resiliency, and safety





Architecture Concepts and Principles

Differences between Embedded vs IT systems

Trust, trustworthiness, & trust gaps

Defense-in-depth

Red/black separation

Trusted Computing Base (TCB)

Embedded system threats and mitigations

Survey of Embedded Systems Threats

- Supply Chain/Counterfeit Parts
- Physical Access
- Reverse Engineering
- Network Access
- Legacy Systems
- Patch update process
- Custom protocols
- Custom libraries
- Cascading Faults
- No Secure Configuration
- Design Mistakes
- Humans



Threat: Supply Chain/Counterfeit Parts



- Make vs. buy
- Quality vs. counterfeiting vs. malicious alteration
 - Vendor tracking database
- ASICS, FPGAs, and microprocessors
 - Destructive and non-destructive analysis
- Information storage in volatile memory and permanent
- Nano tagging

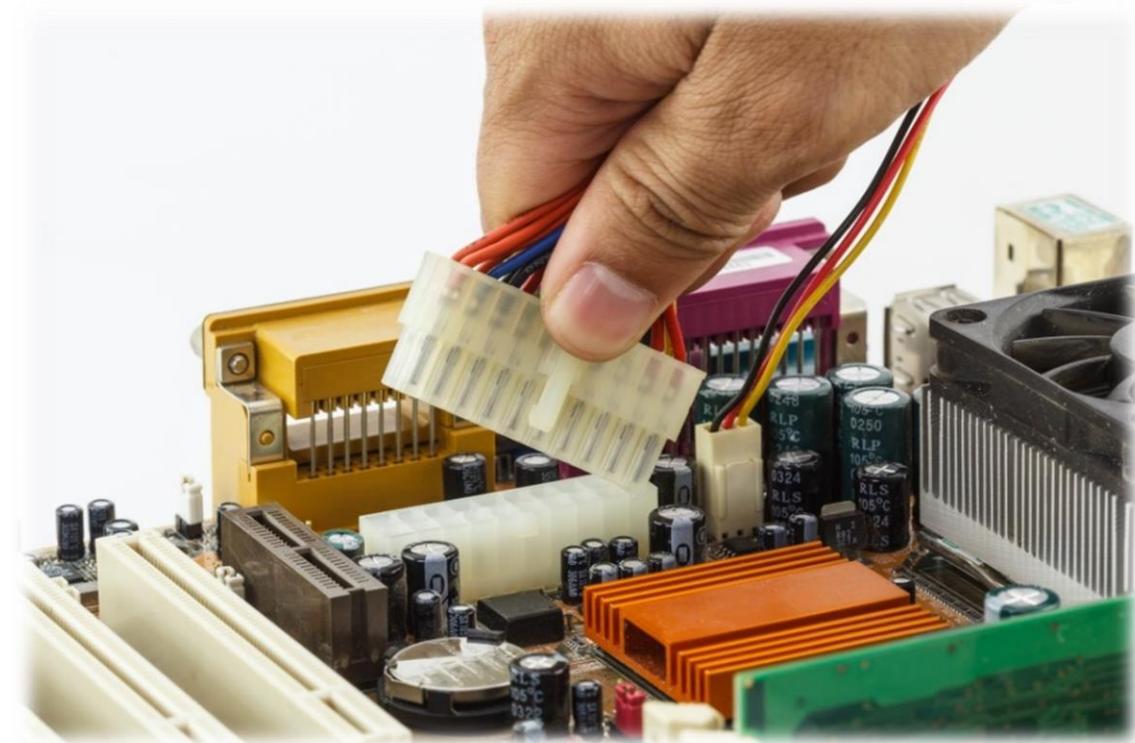


Implement Cybersecurity-SCRM on your programs



Threat: Physical Access

- Malicious access
- Maintenance connections
- Maintenance equipment



Game over?



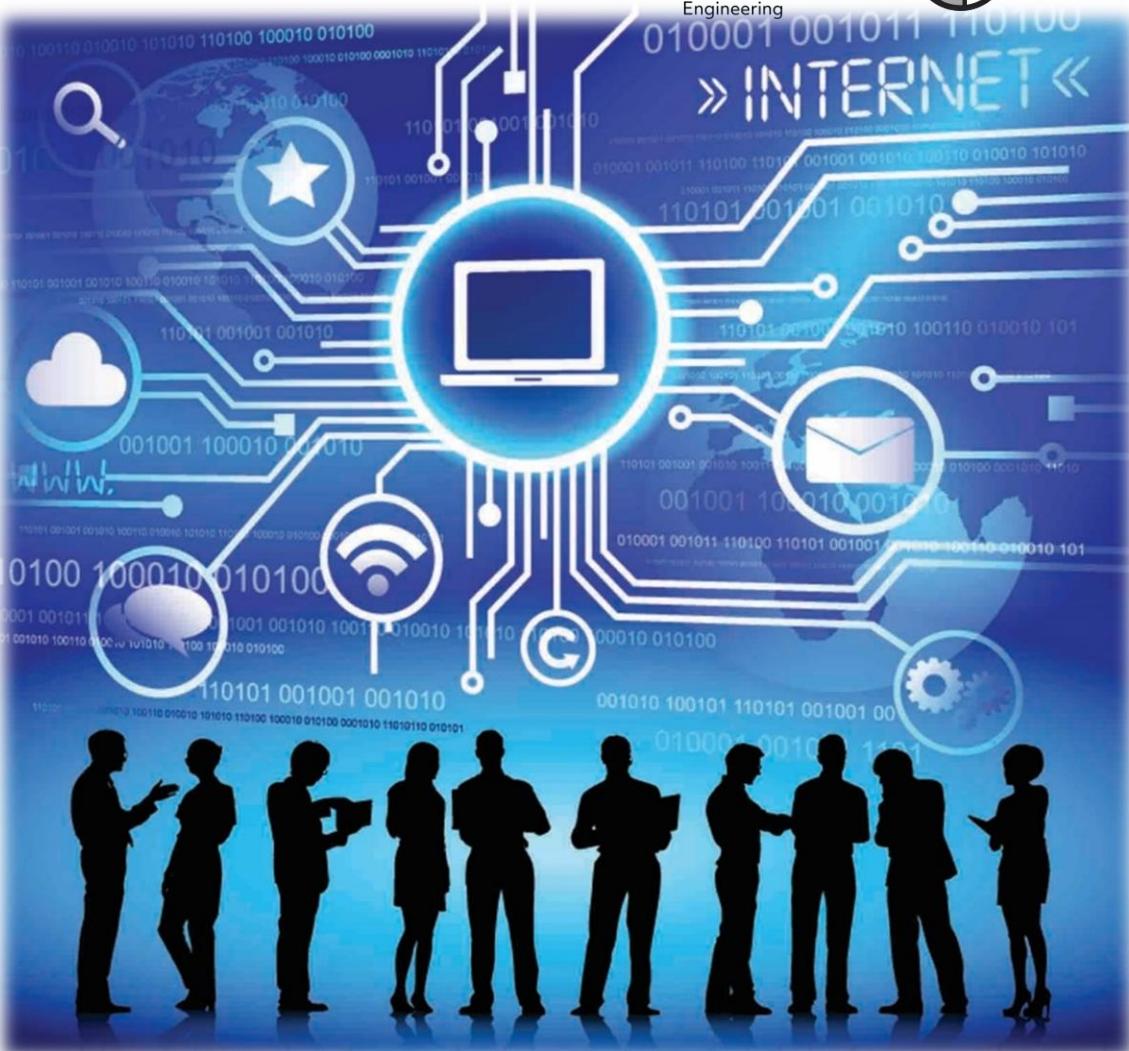
Threat: Reverse Engineering

- Intellectual Property (IP) access
 - System Integrity
- Disassembly
- Black box testing
 - Static Analysis Security Testing (SAST) of binaries
 - Dynamic
 - System Probing



Threat: Network Access

- **Interface attacks** – not just external networks, systems often compromised of multiple connected subsystems; attacks can spread from one compromised subsystem to others
- Standalone systems internetworked
- Unprotected processes
- Remote access
- Radio Frequency (RF) manipulation



Threat: Legacy Systems

- System interaction
 - **Move-left** – compromised inputs, compromised outputs
 - Given your system is perfectly secure, it receives inputs from and sends outputs to other systems
 - Adversaries can bypass your perfect security by compromising its less secure peers
- Least common security measure
- Loss of technical knowledge



Threat: Patch Update Process

- None
 - Systems are permanent and not updated
- Unauthenticated
 - No digital signature on software/firmware
- Invalid
 - No integrity
- No fail secure



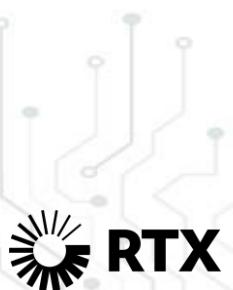
Threat: Custom Protocols

- Legacy
- No authentication
- Variable size
- Non-standard or multiple version support



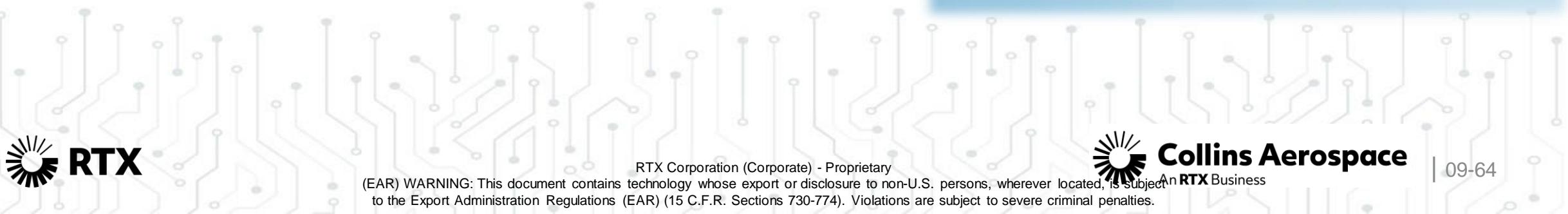
Threat: Custom Libraries

- Common functions
- Extended
- Malicious library



Threat: Cascading Faults

- One fault leads to the failure of interrelated processes.
- Mitigation:
 - Perform damage limitation
 - Sandboxing / partitioning



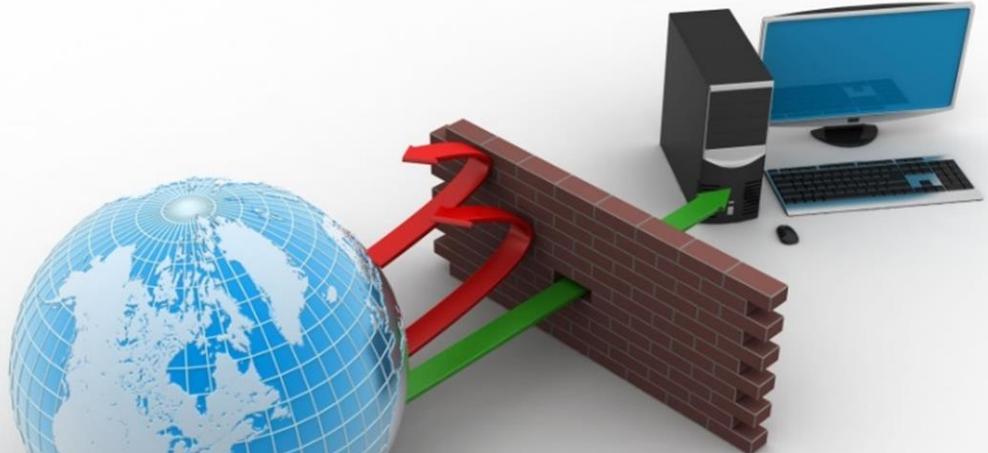
Threat: No Secure Configuration

- Tampered configuration
- Not secure by default
- Shared passwords across collection's embedded systems



Threat: Design Issues

- Hard-coded credentials
- Weak or missing authentication
- Improper segregation of sensitive and non-sensitive data
- Weak, custom, or excessive use of encryption
- Debug functions left in



Threat: Humans

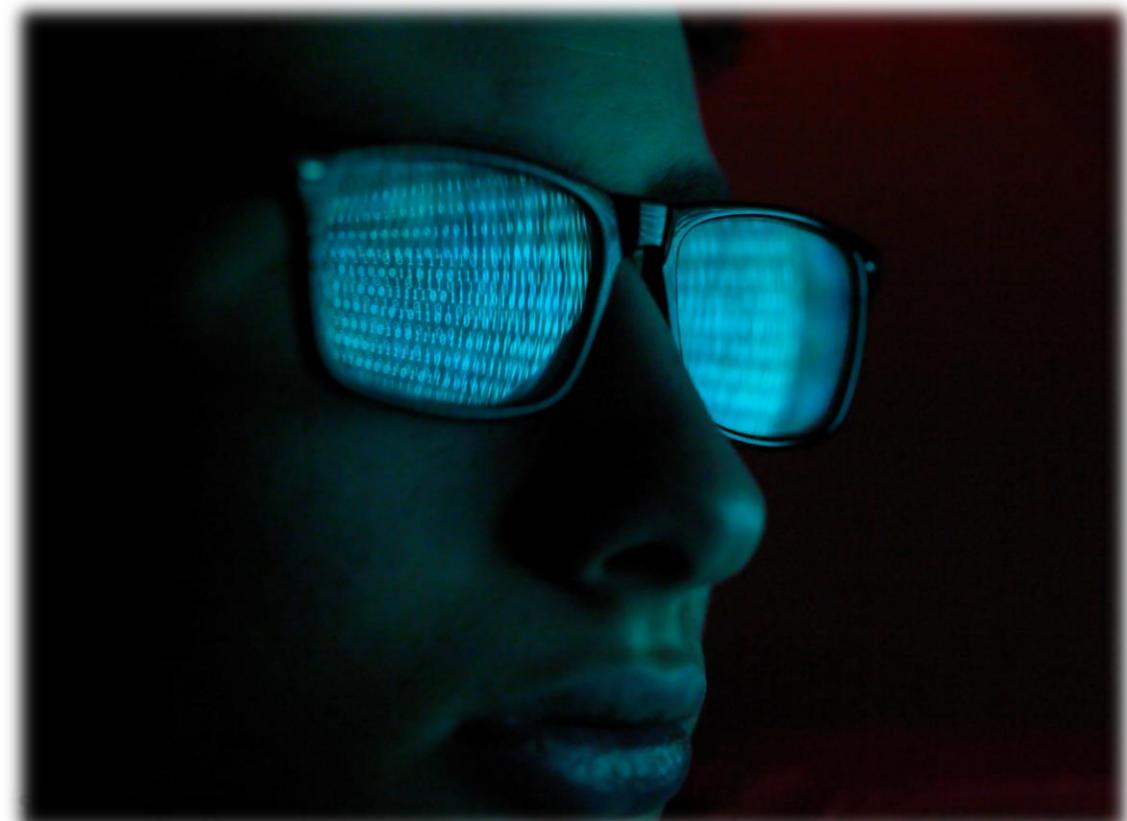
- **Insider:** developer, operator, maintainer (field, depot), warehouse – **anyone** with system access
 - Drive-by: insider modifies system to enable subsequent Cyber attacks
 - Don't rely on missing connectors or physical switches as drive-by can easily re-enable
- Psychological acceptability
- Lack of human monitoring/intervention – attacks on automated systems often go unnoticed
- Too much human intervention – complicated HMI can lead to operator error, relying on operators for security not a best practice



Other Embedded Systems threats



- **Unforeseen consequences, interactions**
(Spectre/Meltdown)
- **Firmware:** FPGAs usually reprogrammable,
sometimes from local processor



Recommended Embedded Systems mitigations



- Follow trusted development processes – SwA and Cyber-SCRM, identify/mitigate trust gaps
- Lock down system – disable unused interfaces, devices, and features; remove dead code
- Employ virtualization to isolate SW and restrict privileges/access
 - Most RTOS support memory virtualization on PowerPC, Intel, and ARM
 - Use virtual not kernel drivers to restrict driver privilege
- Employ readily available embedded SW mitigations, especially those already integrated into build tools
 - ASLR; stack canaries; shadow stacks; stack reordering; range/value checking
- Implement auditing and notification infrastructure
 - At least capture abnormal events; e.g., capture OS exceptions
 - Provide hooks allowing security SW to capture abnormal events
 - Provide a means to store and export audit logs



Questions?



RTX Corporation (Corporate) - Proprietary
(EAR) WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



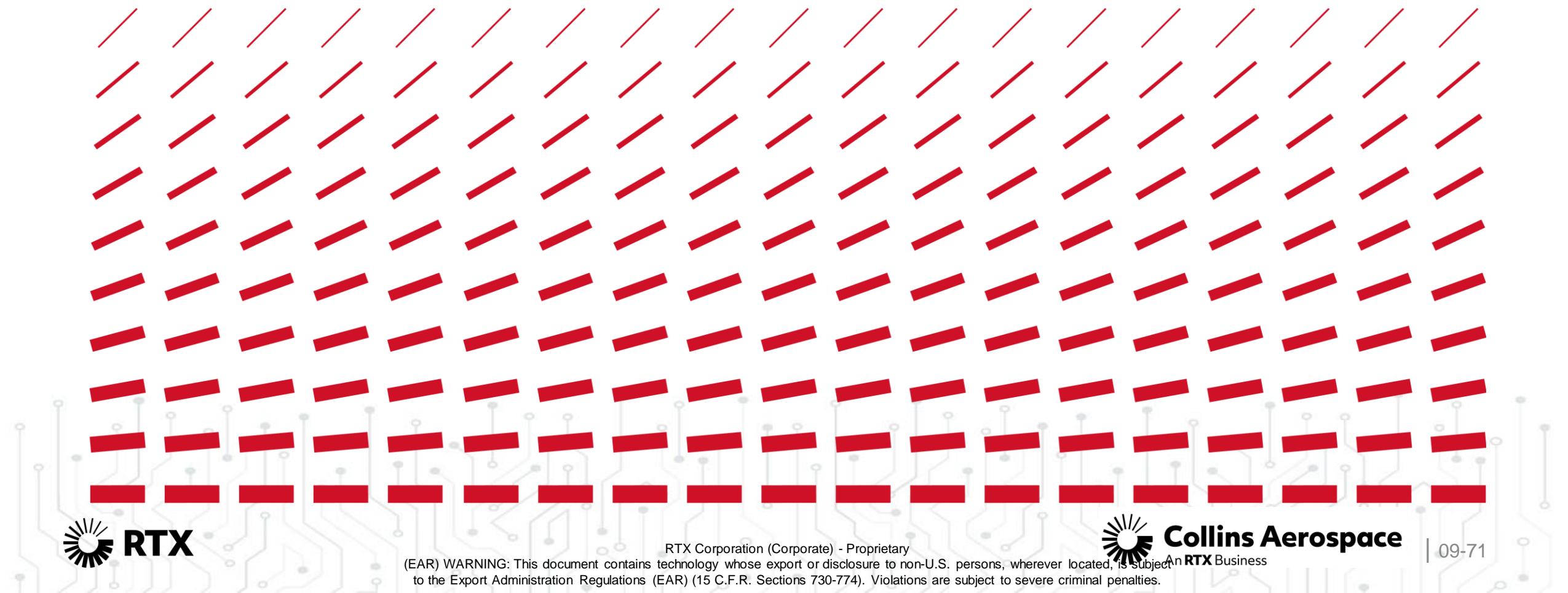
09-70

Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





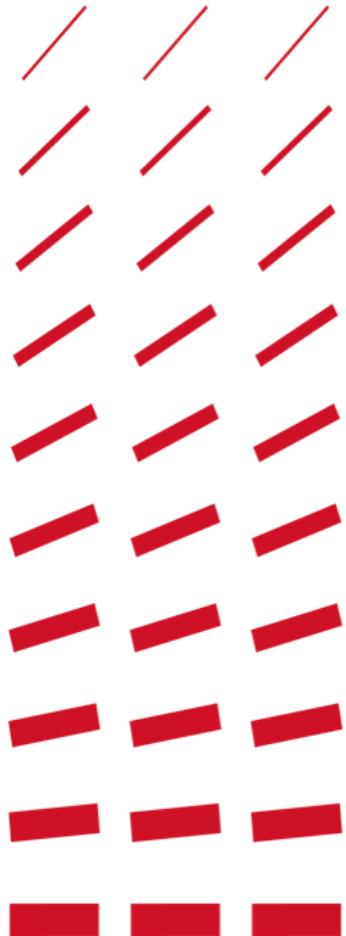
TGECYBEREMBSEC Embedded Systems Security

Module 10 Secure Boot

Instructor: Randall Brooks

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

Module agenda



Definitions

Requirements and implementations

Example #1 – iPhones

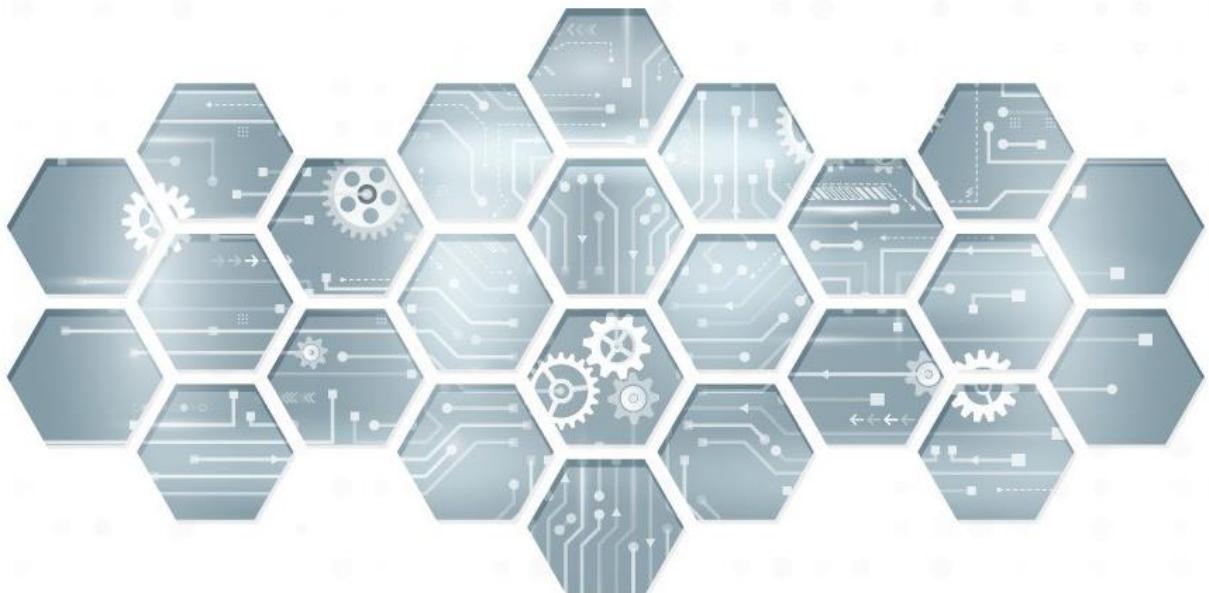
Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

Questions





Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

Example #2 – Intel processors

Example #3 – TI SOC

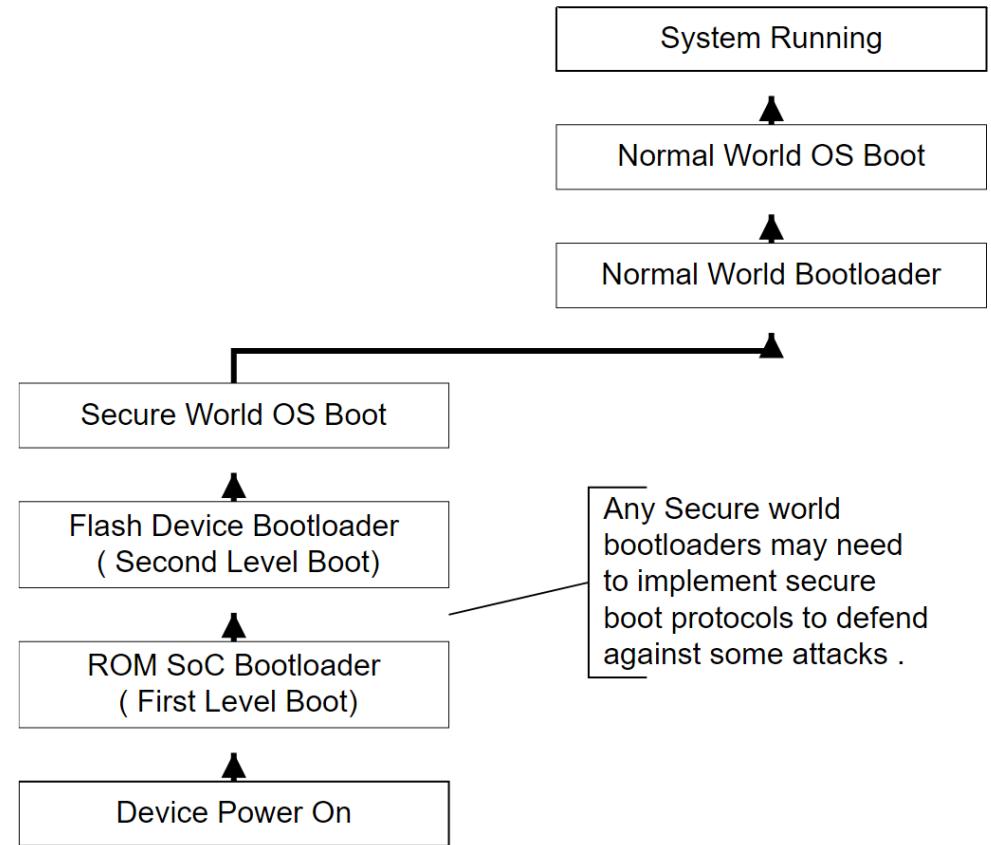
Wrap-up

Questions

What is the Secure Boot process in ARM?



- “A secure boot scheme adds cryptographic checks to each stage of the Secure world boot process. This process aims to assert the integrity of all of the Secure world software images that are executed, preventing any unauthorized or maliciously modified software from running.” - ARM



documentation-service.arm.com/static/5f212796500e883ab8e7452b?token=

<https://developer.arm.com/documentation>



Definitions



- Originally defined as part of the Open Mobile Terminal Platform (OMTP) Trusted Environment (TR0) specification, **Secure Boot** is defined as:
 - “Set of operations, started upon [system] initialization, and used to perform hierarchical verification of code’s security properties, followed by its execution.”¹
- Depending on the documentation you are referencing, **Secure Boot** may also be referred to as **Verified Boot**.
- **Secure Boot** provides assurance that only signed and authenticated code executes on a system.
 - It makes **no guarantees about the security of the code**, nor does it provide any information on what was executed prior to the currently executing code.
- **Trusted Boot**, also called **Measured Boot**, maintains cryptographic measurements of all authenticated code that was executed on a system, thereby allowing introspection backwards in the boot process.

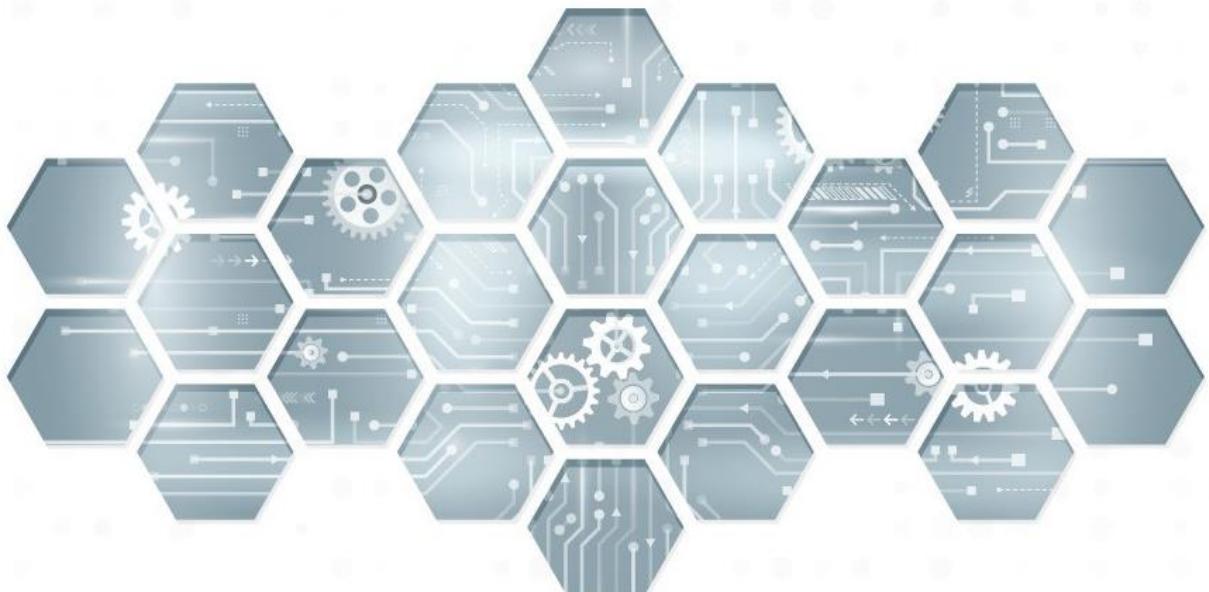


Definitions (continued)



- **Root-of-Trust (RoT)** – component that must always behave in the expected manner because its misbehavior cannot be detected²
- **Core Root-of-Trust (CRT)** – the first set of instructions executed when a new chain of trust is established²
- **Root-of-Trust of Measurement (RTM)** – the RoT element responsible for creating measurements and transmitting those measurements to the RTS²
- **Root-of-Trust of Storage (RTS)** – the RoT element responsible for securely maintaining cryptographic measurement generated and used by the RTM and RTR²
- **Root-of-Trust of Reporting (RTR)** – the RoT element responsible for reporting RTS values to a higher-level authority²
- **Core Root-of-Trust of Measurement (CRTM)** – The first piece of BIOS code that executes on the main processor during the boot process³





Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

Questions

Requirements

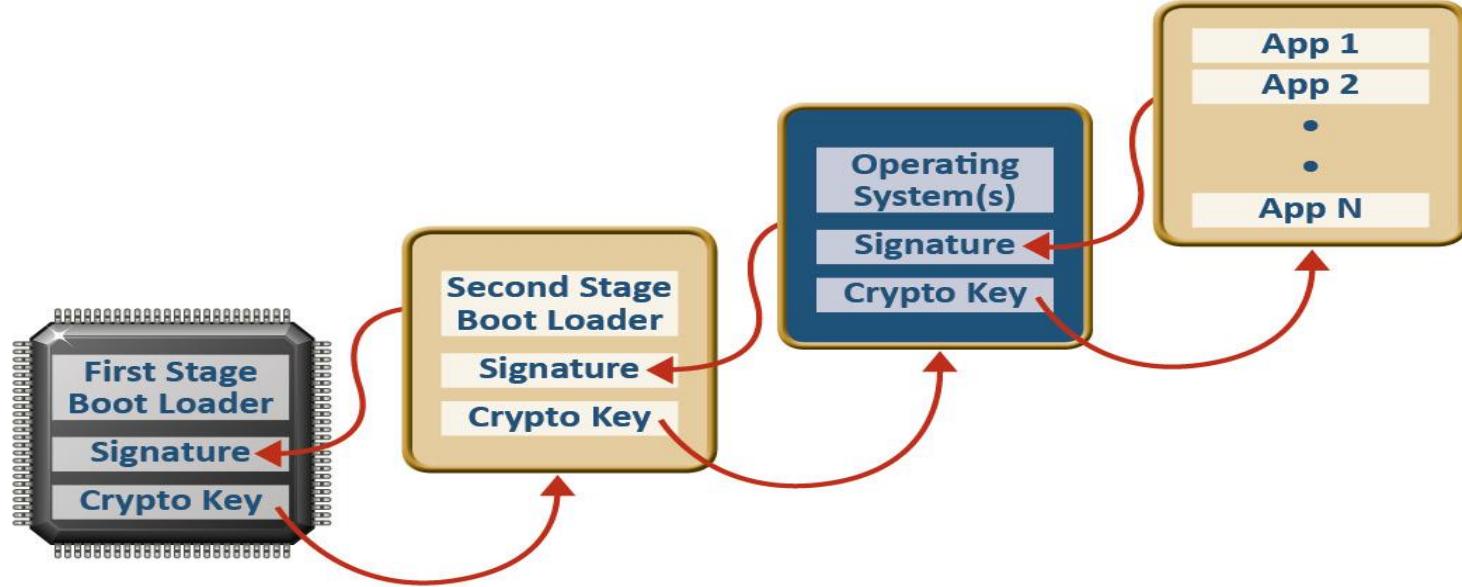


- **Secure Boot** requires authentication ONLY
 - It is a hardware capability
- Authentication requires the use of asymmetric cryptography, most frequently using RSA or ECC
- Authentication is typically done through the use of certificates (X.509 is the most common) that contain a sign cryptographic hash of the associated code
- Symmetric encryption may be used to encrypted the data being authenticated, but that is not a requirement

- The cryptographic hashes used to validate the code are NOT required to be maintained for **Secure Boot**
 - This represents the primary difference between **Secure/Verified Boot** and **Trusted/Measured Boot**
 - Trusted Boot is a software capability
- Certificate formats, such as X.509, are not specified nor specifically required



Implementation – secure boot

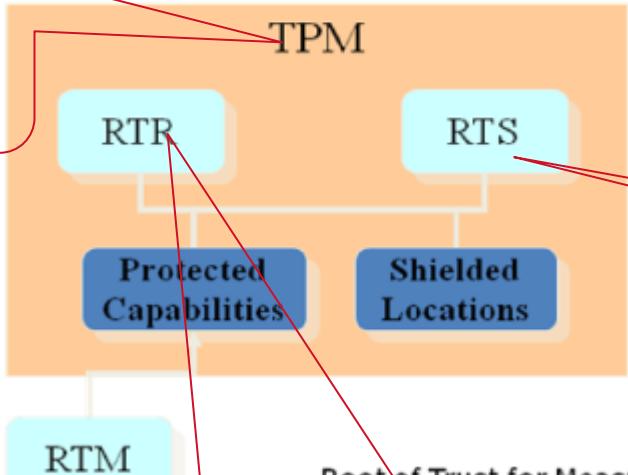


- **Secure Boot** is implemented as a chain of trust starting with the **CRT**
- The **CRT** will use its crypto key to authenticate the next element prior to its execution
- This sequence continues on through the OS and in some case even into the applications themselves
- A chain is metaphorically created linking the stages together, yet does not provide a look back

Implementation – trusted boot

Functional TPM Diagram

Traditional systems leverage a Trusted Platform Module (TPM) to serve as the RTR and RTS



Root of Trust for Reporting RTR

- Provides cryptographic mechanism to digitally sign TPM state and information

Root of Trust for Storage RTS

- Provides cryptographic mechanism to protect information held outside of the TPM

As opposed to Secure Boot, measurements generated during boot are maintained within the RTS of the TPM

The host CPU, such as Intel, AMD, or ARM, would serve as the RTM

The RTR may then provide a signed copy of these measurement back to the RTM or any other entity with authority to access them

Source: https://www.researchgate.net/figure/Root-of-trust-of-TPM-1_fig3_267335876

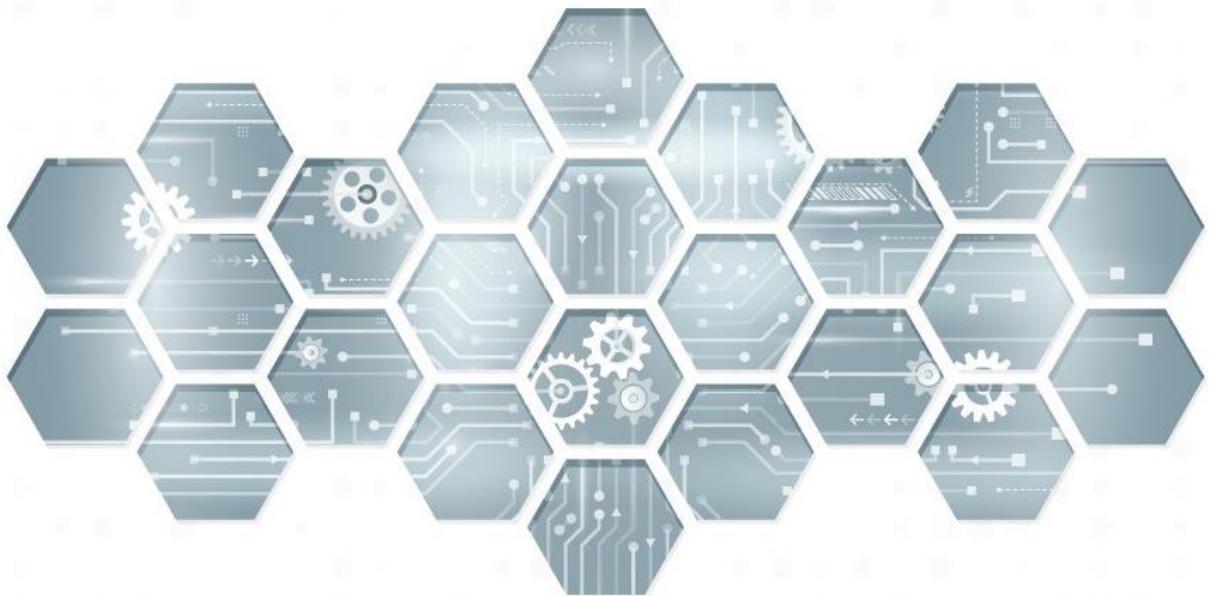
RoT Recommendations



- RoT should have these minimum features:
 - Crypto
 - Key Management
 - Support Secure Boot
- A RoT could augment security with the following features:
 - Hardware-based, side channel resistant Crypto
 - Key Management + Key Coordination with Next Higher Assembly/System
 - Security Controller
 - Data at Rest Protection
 - Hooks to integrate/coordinate with other Intellectual Property (IP) (Especially sensors for physical attacks)

Based on Idaho Scientific / Curtiss-Wright Recommendations





Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

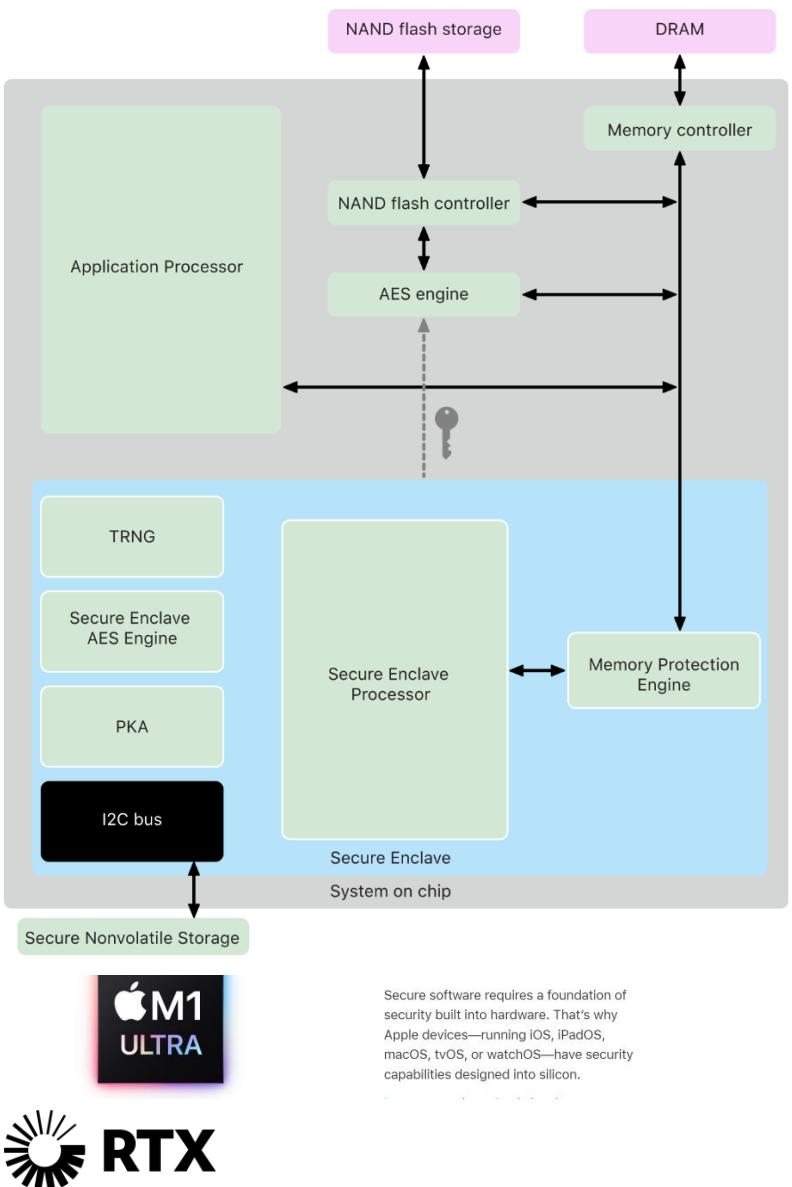
Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

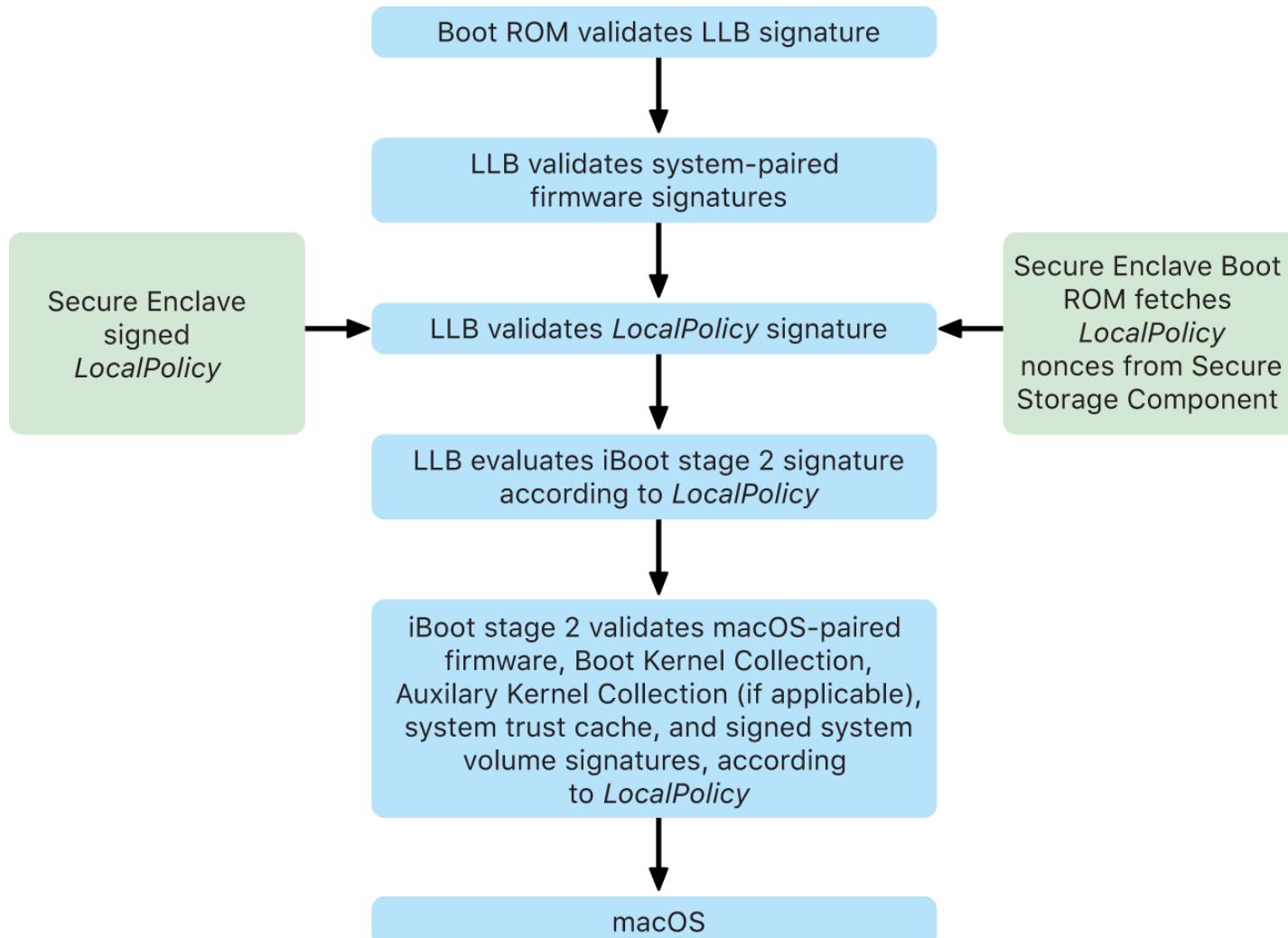
Questions

Example #1: iPhone / Apple



- Apple produces their own custom application-specific integrated circuits (ASICs) that are used on all their embedded devices
- The custom ASICs are referred to as **A processors** with an accompanying numerical value based on its version, such as A17 Pro, A16 Bionic, and variants such as M1, M2, etc.
 - Roots of Trust are the T chips with a Secure Enclave
- Stored inside each processor are two critical keys that are used for security on the device: **GID** and **UID**
- The **GID** is a unique for each version of the processors, whereas the **UID** is unique for each processor
- Both keys are AES 256-bit keys
 - The **GID** is used as a **Key Encryption Key (KEK)** to protect all boot images used on the system, while the **UID** is used as a **Root Key (RK)** in conjunction with the user passcode to protect user data
 - Although not verified, it is publicly assumed that the **UID** is likely stored in eFuses and the **GID** is hard coded into the logic of the ASIC⁴

Example #1: iPhone (continued)



Source Apple



Example #1: iPhone (continued)



- On system startup, iBoot assigns a dedicated region of memory to the Secure Enclave. Before using the memory, the Secure Enclave Boot ROM initializes the Memory Protection Engine to provide cryptographic protection of the Secure Enclave protected memory.
- The Application Processor then sends the sepOS image to the Secure Enclave Boot ROM. After copying the sepOS image into the Secure Enclave protected memory, the Secure Enclave Boot ROM checks the cryptographic hash and signature of the image to verify that the sepOS is authorized to run on the device. If the sepOS image is properly signed to run on the device, the Secure Enclave Boot ROM transfers control to sepOS. If the signature isn't valid, the Secure Enclave Boot ROM is designed to prevent any further use of the Secure Enclave until the next chip reset.
- On Apple A10 and later SoCs, the Secure Enclave Boot ROM locks a hash of the sepOS into a register dedicated to this purpose. The Public Key Accelerator uses this hash for operating-system-bound (OS-bound) keys.

The Secure Enclave firmware (sepOS), based on an Apple-customized version of the L4 microkernel



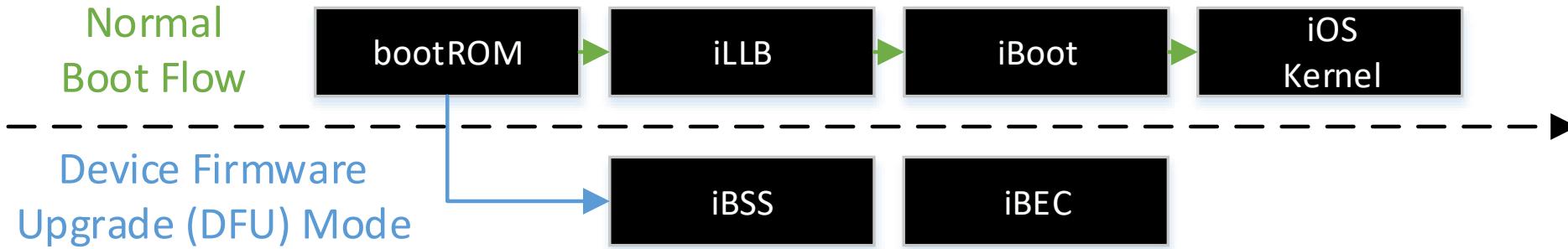
Example #1: iPhone (continued)



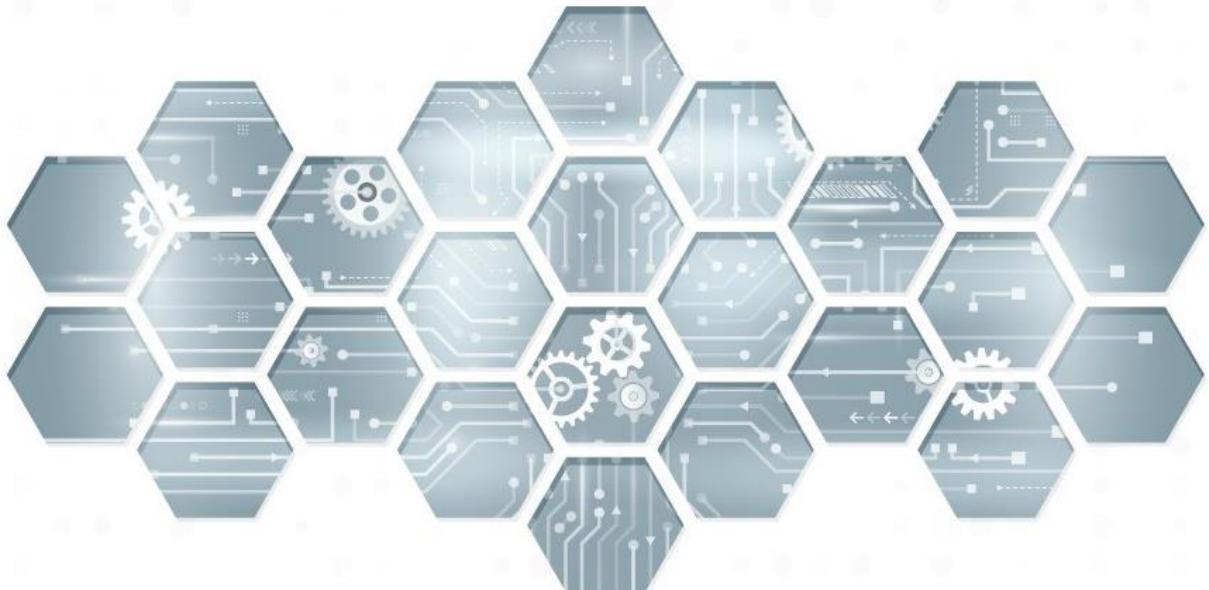
- On Apple A13 and later SoCs, the Secure Enclave includes a Boot Monitor designed to ensure stronger integrity on the hash of the booted sepOS.
- At system startup, the Secure Enclave Processor's System Coprocessor Integrity Protection (SCIP) configuration helps prevent the Secure Enclave Processor from executing any code other than the Secure Enclave Boot ROM. The Boot Monitor helps prevent the Secure Enclave from modifying the SCIP configuration directly. To make the loaded sepOS executable, the Secure Enclave Boot ROM sends the Boot Monitor a request with the address and size of the loaded sepOS. On receipt of the request, the Boot Monitor resets the Secure Enclave Processor, hashes the loaded sepOS, updates the SCIP settings to allow execution of the loaded sepOS, and starts execution within the newly loaded code. As the system continues booting, this same process is used whenever new code is made executable. Each time, the Boot Monitor updates a running hash of the boot process. The Boot Monitor also includes critical security parameters in the running hash.
- When boot completes, the Boot Monitor finalizes the running hash and sends it to the Public Key Accelerator to use for OS-bound keys. This process is designed so that operating system key binding can't be bypassed even with a vulnerability in the Secure Enclave Boot ROM.



Example #1: iPhone (continued)



- Device boot-up follows one of two paths: Normal or ***Device Firmware Upgrade (DFU)***
- ***DFU*** mode is entered into either intentionally by the user (through some button combination), as mandated by a software update, or anytime the Normal boot sequence fails
- The Boot ROM serves as the ***CRTM*** for the system and contains its own cryptographic certificate that includes a copy of Apple's public key
 - ***immutable code called the Boot ROM***
- Each image in the boot process is both encrypted and signed
- A successful boot requires that all images executed be both authenticated and decrypted prior to execution



Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

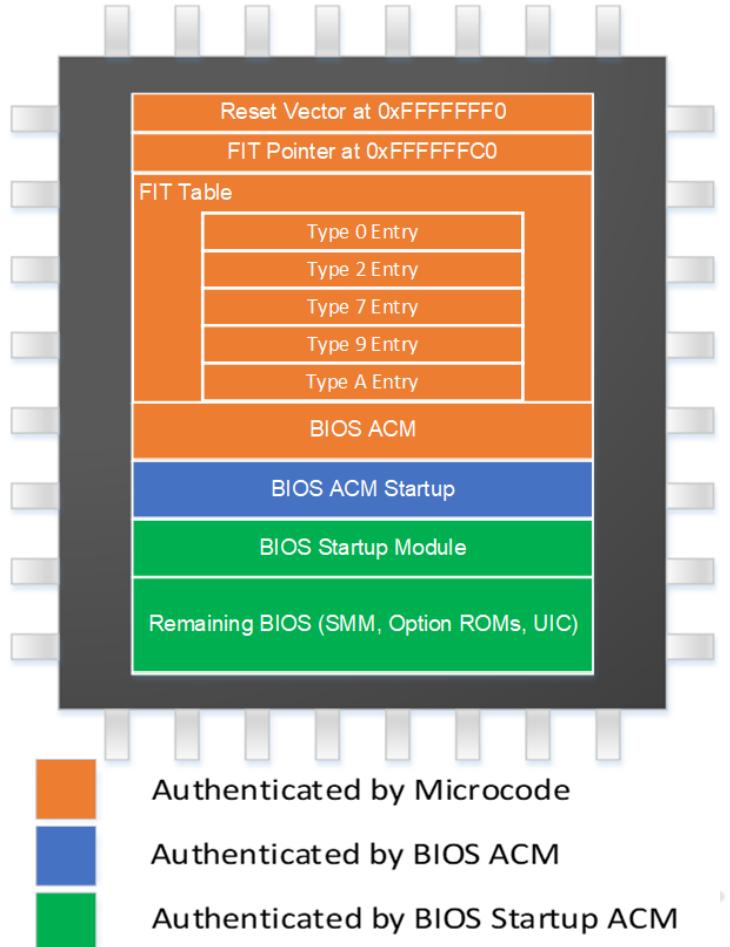
Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

Questions

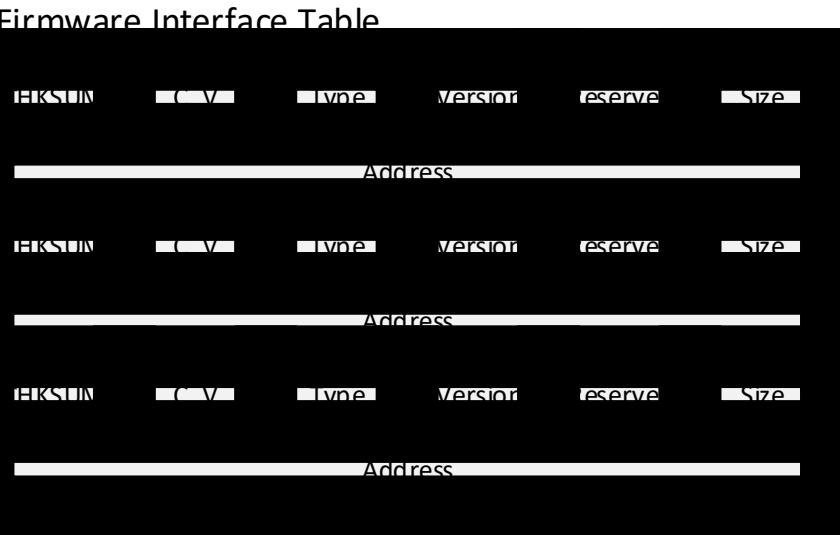
Example #2: Intel processors



- Intel Processors begin execution with an internally stored and “immutable” ROM that stores what is called **Microcode**
- The **Microcode** serves as the **CRTM** on the system
- Subsequent trusted code elements are referred to as **Authenticated Code Modules (ACM)** and each must be measured and authenticated prior to execution
- The **Firmware Interface Table (FIT)** contains entries for each ACM and code segment that is to be loaded by the **Microcode** or BIOS
- Boot begins with the **Microcode** being authenticated and executed, then sending execution to the Reset Vector, which points to the **FIT**

Example #2: Intel processors—continued

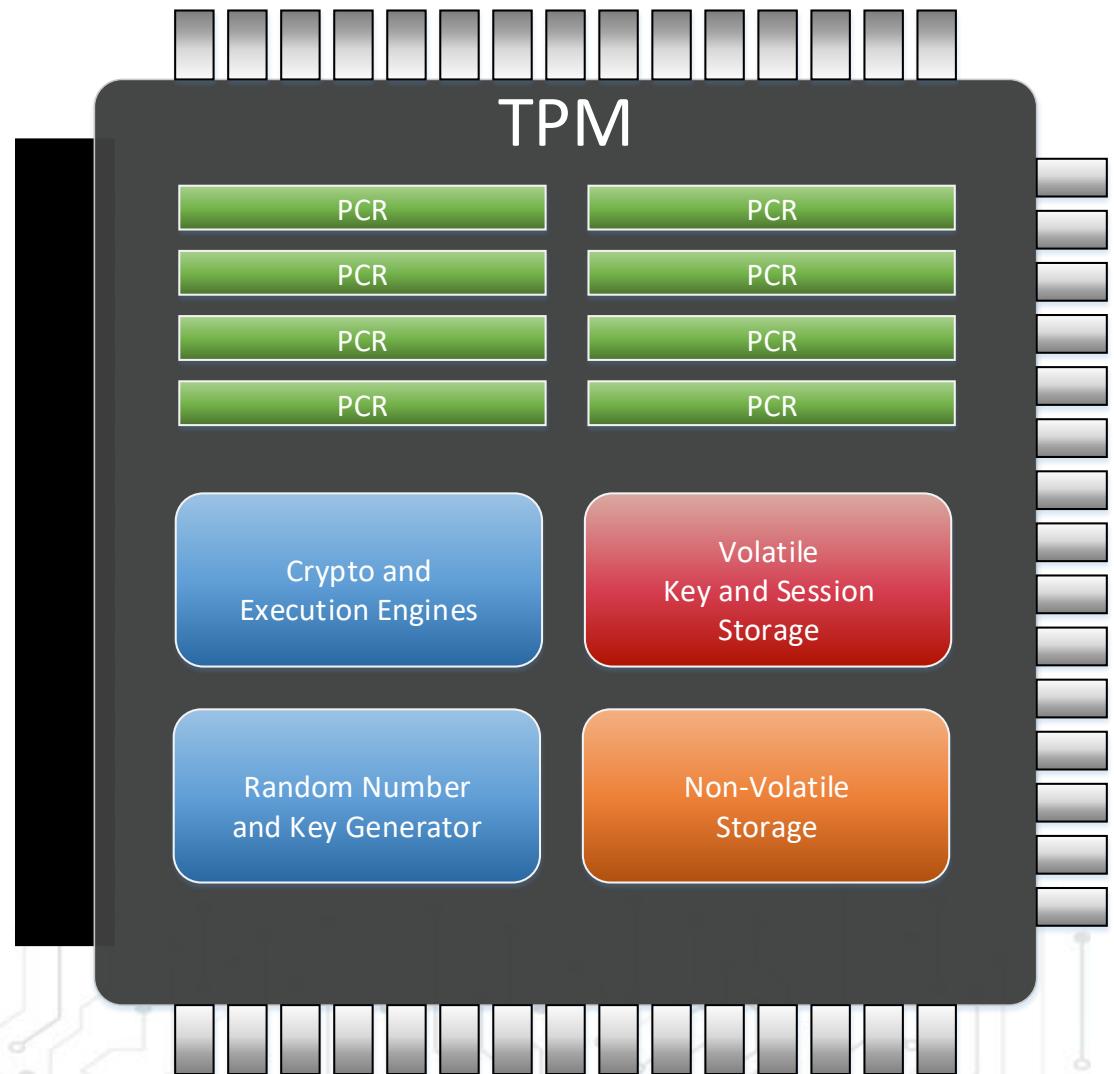
- The ***FIT*** contains one header entry, and then zero, one, or more entries for each of the other module types
- All modules pointed to by the ***FIT*** must be signed by the originator and authenticated prior to execution
- The ***FIT*** itself is not signed, but it is measured as part of a Measured Boot sequence



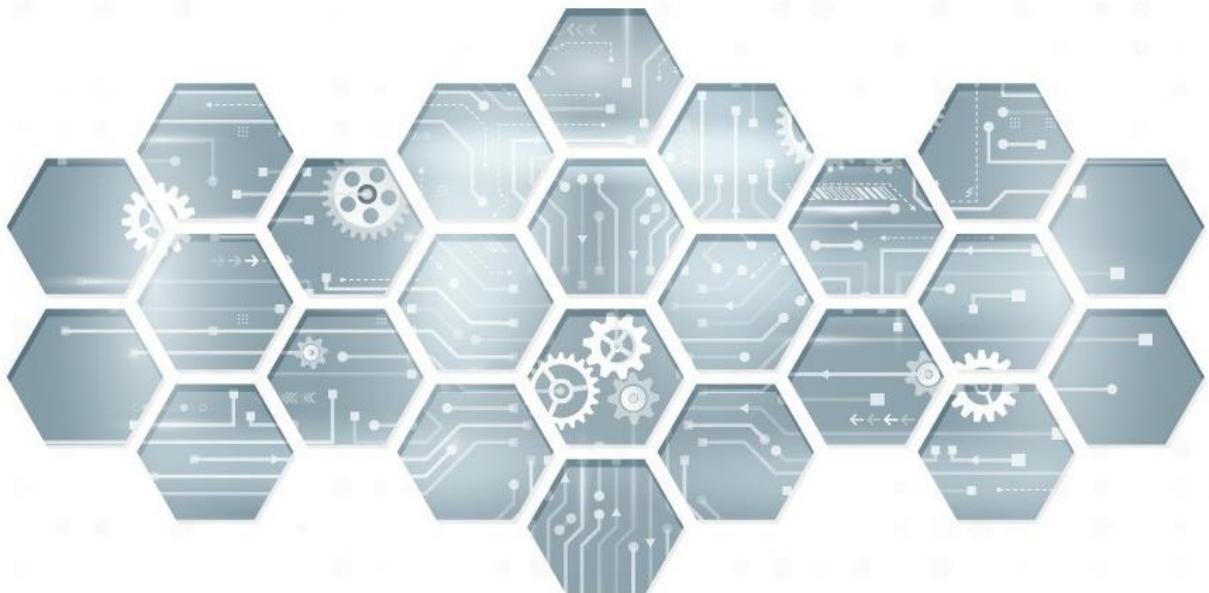
Type	Description
0x00	Header
0x01	Microcode Update
0x02	Startup ACM
0x07	BIOS Startup Module
0x08	TPM Policy
0x09	BIOS Policy
0x0A	TXT Policy
0x0B	Key Manifest
0x0C	Boot Policy Manifest
0x10	CSE Secure Boot
0x2D	TXTSX Policy
0x2F	JMP Debug Policy



Example #2: Intel processors – 3



- As mentioned previously, Intel processors can implement **Verified** boot or **Measured** boot
- With **Verified** boot, code is simply authenticated when specified prior to execution
- With **Measured** boot, the measurement obtained during authentication of each code element is sent to a **Trusted Platform Module (TPM)**
- The **TPM** may then be queried at any time to obtain a record of all measurements transmitted to it for storage.
- Measurements are stored in **Platform Configuration Registers (PCRs)** that concatenate their current value with the new value received, then perform a SHA hash on the result. The result is then stored back into the corresponding **PCR** as its new value
- **PCRs** are assigned specific measurement during the boot process as defined in the TCG PC Client specification



Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

Questions

Example #3: Texas Instruments (TI) System on a Chip (SOC)

- “Sitara processors provide the ability for a customer to specify the root security key (Public Key) that acts as a root-of-trust and is fused into the device. Once fused this root security key cannot be reversed or muted.
- This root security key is typically entrusted to the manufacturer’s chief technical officer for security. The Chief Technology Officer (CTO) is able to further bind multiple Public Keys for the use of their organization’s development teams and cryptographically associate these keys with Root Public Key. These keys are analogous to a keyring containing the Public Keys assigned to the various software development teams within the development organization that are working on a system’s boot code or the various product lines of the company. At the discretion of the CTO, any of these Public Keys can be revoked at any time should this be required.” - TI

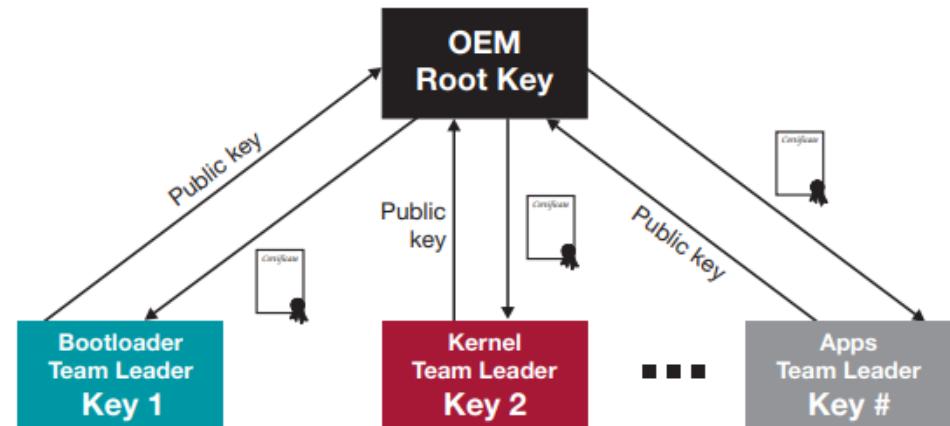


Figure 3: Device keys delegation

<https://www.ti.com/lit/wp/spry305a/spry305a.pdf?ts=1701235675869>



Example #3: TI SOC (continued)



- ***“Provisioning keys in hardware”***
- The first step in enabling secure boot on the device is to create cryptographic keys and provision providing the generated keys in the hardware. Using the hardware security module is a common industry practice to generate and protect booting keys. TI provides evaluation modules (EVMs), software development kits (SDKs), tools and documentation to showcase the generation of keys and procedure to provision booting keys into the device."- TI

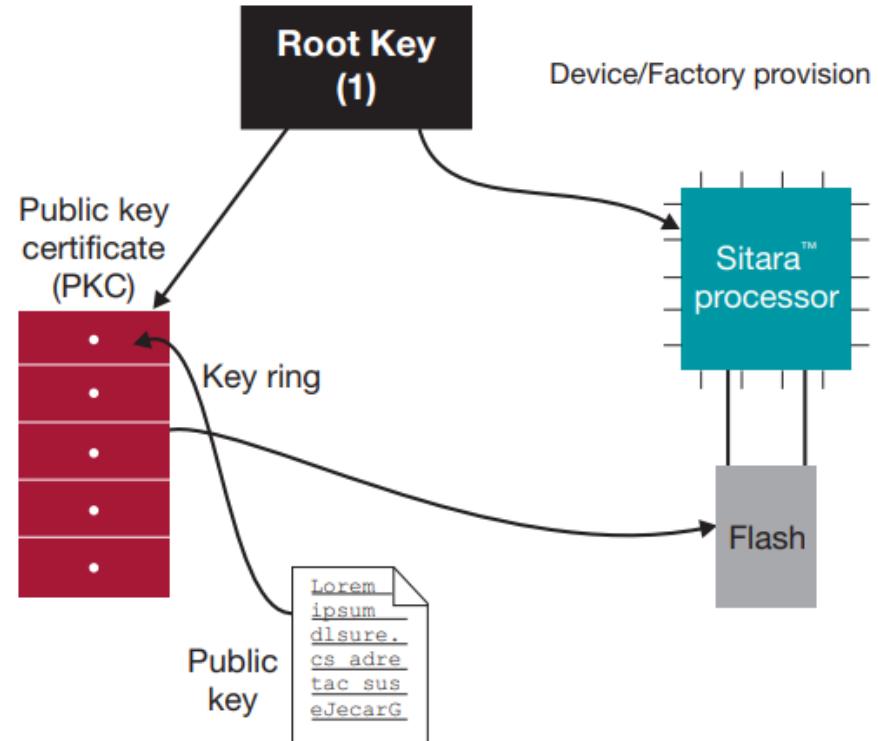


Figure 4: Device key management

<https://www.ti.com/lit/wp/spry305a/spry305a.pdf?ts=1701235675869>



Example #3: TI SOC (continued)



- “***Code encryption and signing***
- The other aspect of secure boot is to cryptographically associate the code/software with the keys provisioned in the device such that the device only loads and executes trusted code. This trusted code is then downloaded to system non-volatile memory on a production line.
- The code/software which makes up bootloaders, kernel, file system, etc. is signed and encrypted with associated keys that are provisioned in the device. In a typical scenario, this step involves pulling code into the secure system like hardware security module and cryptographically signing the code. If the encryption option is used, then the code/software is also encrypted with associated booting keys.” - TI

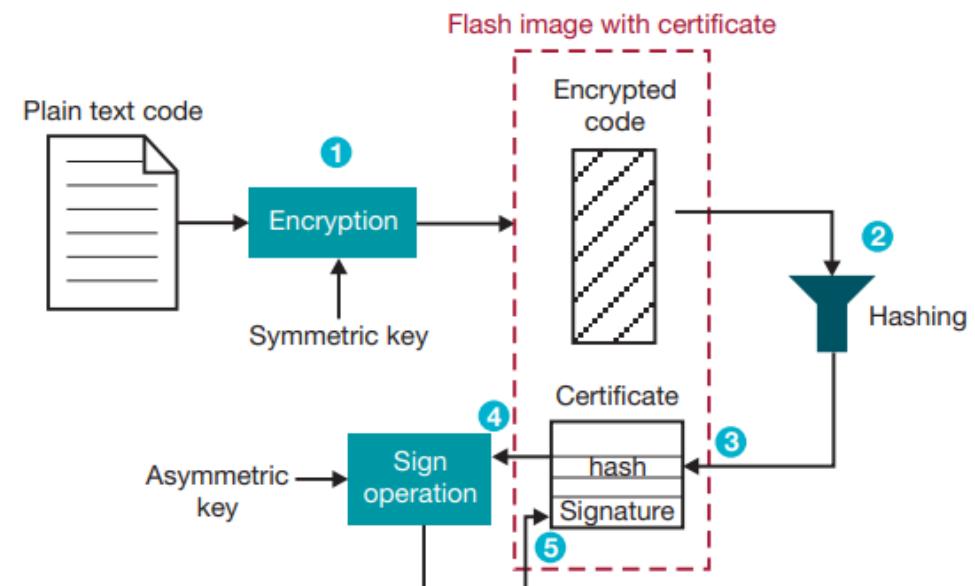


Figure 5: Code signing and code encryption

<https://www.ti.com/lit/wp/spry305a/spry305a.pdf?ts=1701235675869>



Example #3: TI SOC (continued)



- “Secure booting starts at reset, as part of secure boot the device initializes itself in preparation to receive signed and encrypted software/code from boot media. This step involves creating secure partition of on-chip SRAM, initializing hardware cryptography engines, switching to secure CPU mode and configuring various controls within devices to create a secure environment.
- Device on-chip ROM then fetches first code along with secure boot certificate from external media and loads into secure SRAM. On-chip ROM then extracts the root Public Key from the certificate and compares against the value programmed in the device, if the Root Public Key match then the certificate is deemed trusted. On-chip ROM then uses Root Public Key to cryptographically authenticate key ring data structure that contains more Public Keys, if authentication passes for key ring then all keys in key ring are deemed trusted and can be used to authenticate subsequent software components.
- Furthermore, device on-chip ROM fetches bootloaders and other software components and authenticates using either the Root Public Key or one of the keys from the key ring. The device also offers an application programming interface which can be invoked by trusted software to authenticate the next software and pass control to next software. ” - TI

<https://www.ti.com/lit/wp/spry305a/spry305a.pdf?ts=1701235675869>



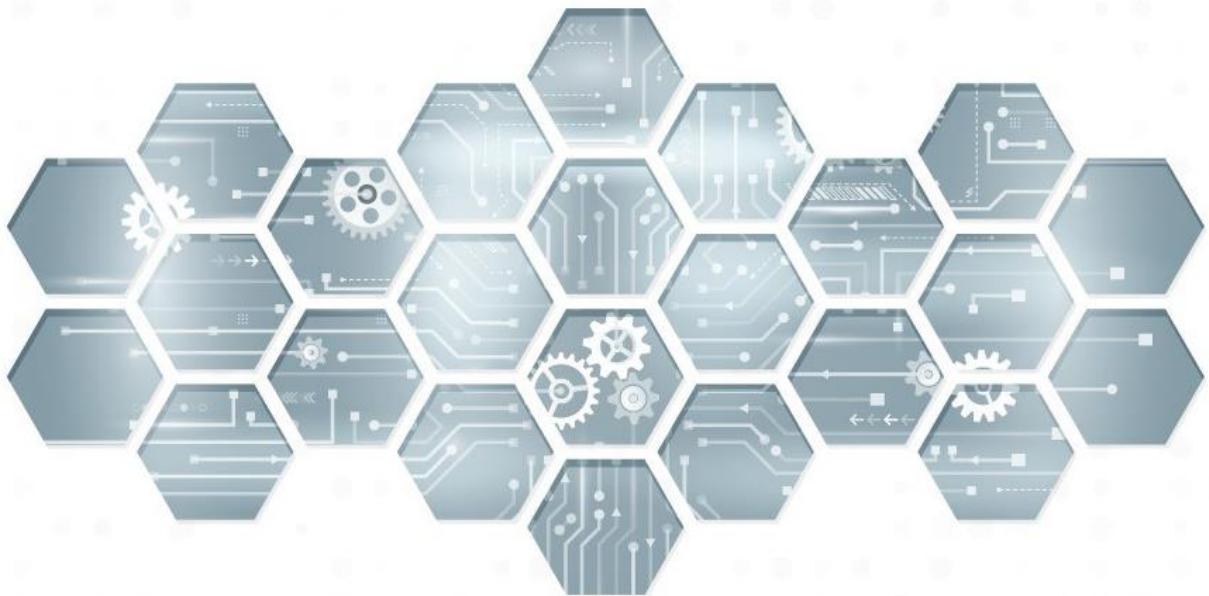
Example #3: TI SOC (continued)



- The TI SOC also supports IP Protection
 - “IP Protection is intended to offer system manufacturers a way to have their IP/code encrypted while sitting in external boot media. IP Protection is also sometimes required for certain security certification.
 - Sitara processors support IP Protection where the code/software in external boot media is encrypted. As part of secure boot, the Sitara processor also decrypts the code/software using keys provisioned in the device.” - TI

<https://www.ti.com/lit/wp/spry305a/spry305a.pdf?ts=1701235675869>





Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

Questions

References

1. http://www.omtp.org/OMTP_Trusted_Environment_OMTP_TR0_v1_2.pdf
2. https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf
3. <https://csrc.nist.gov/glossary/term/Core-Root-of-Trust-for-Measurement>
4. <https://support.apple.com/guide/security/welcome/web>
5. https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf
6. <https://www.intel.com/content/www/us/en/software-developers/intel-txt-software-development-guide.html>



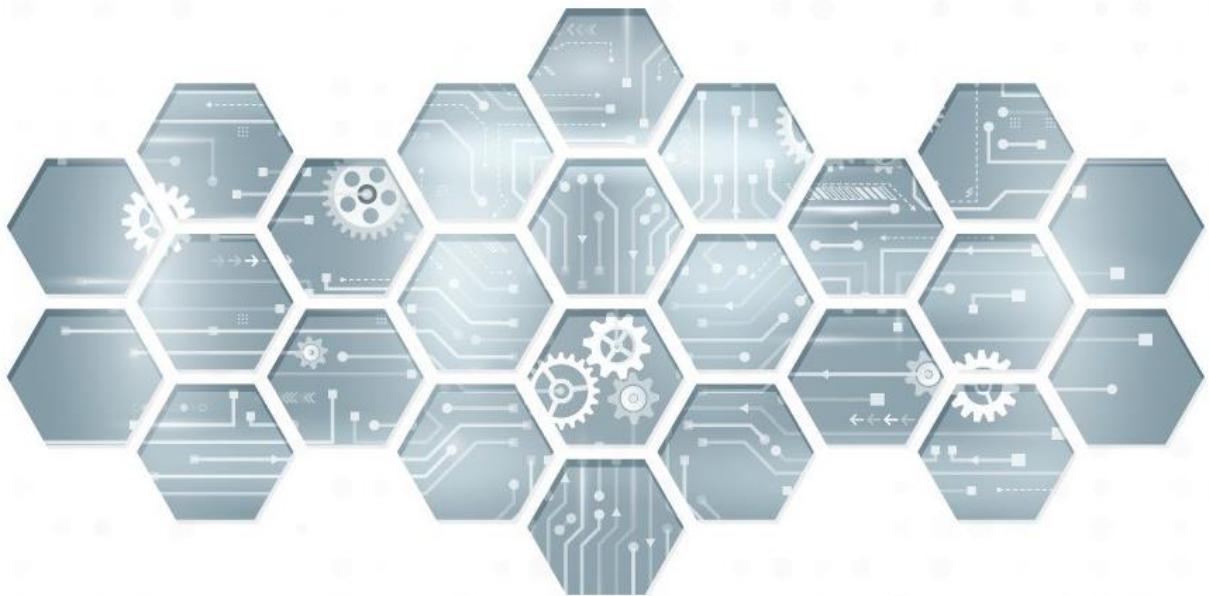
Class Discussion — Secure Boot



Q1: True or false. Secure Boot works best if you are using a hardware-assisted elements.

- a) True
- b) False





Secure Boot

Definitions

Requirements and implementations

Example #1 – iPhones

Example #2 – Intel processors

Example #3 – TI SOC

Wrap-up

Questions

Questions?

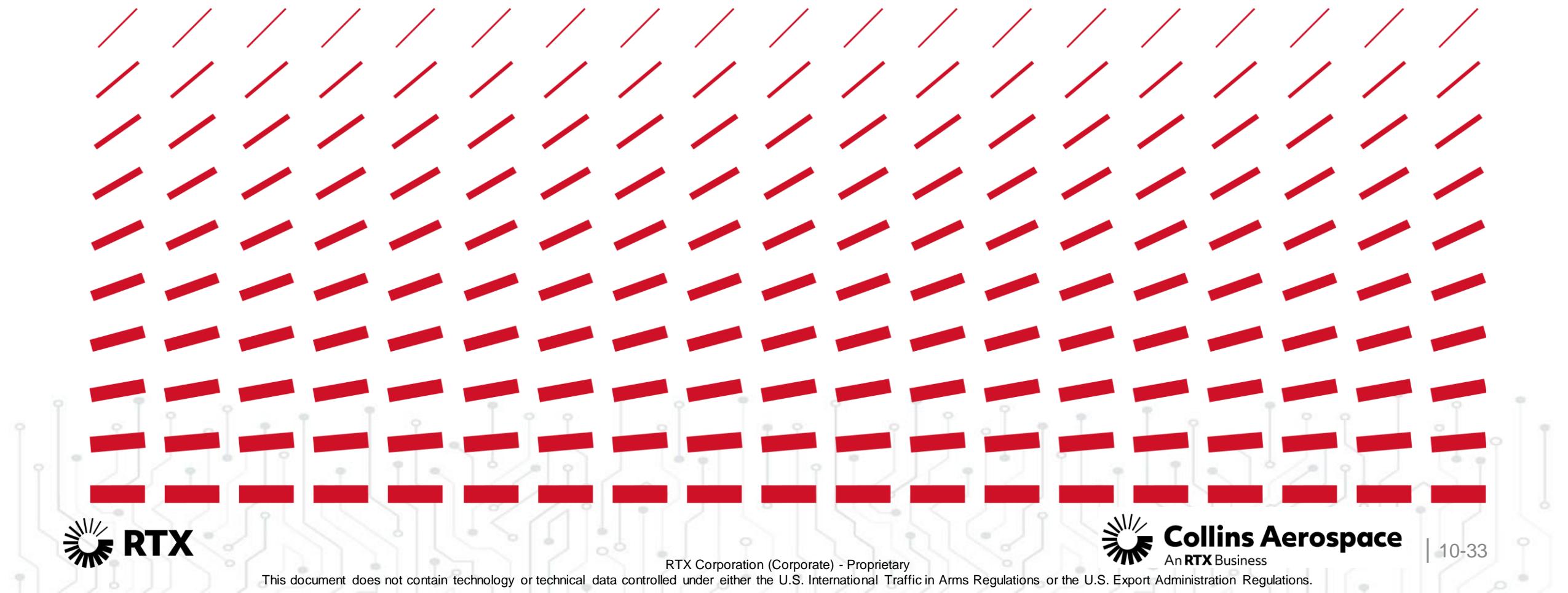


Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.



Before we begin



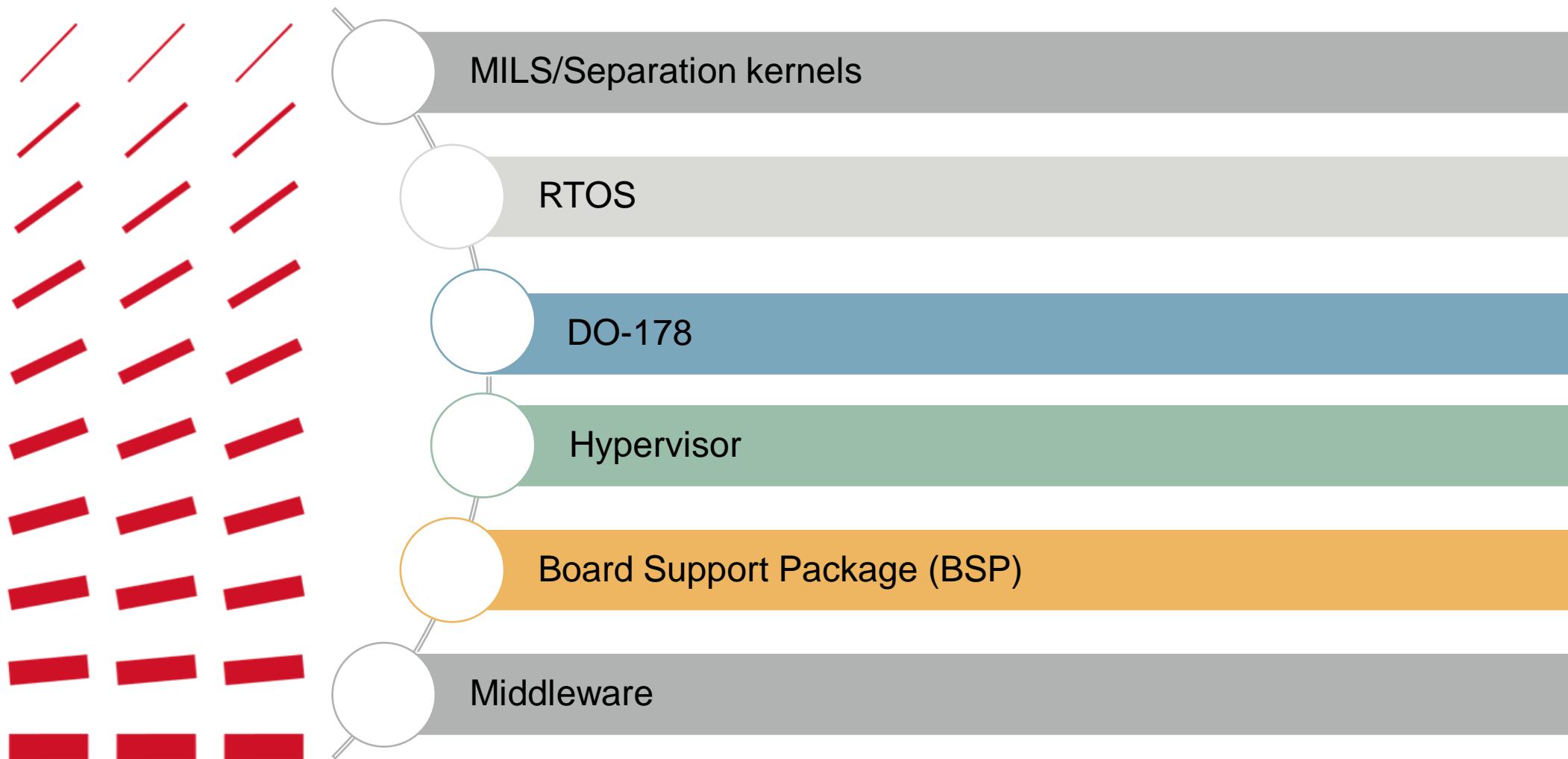
Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



Module agenda





Collins Aerospace
An **RTX** Business

TGECYBEREMBSEC Embedded Systems Security

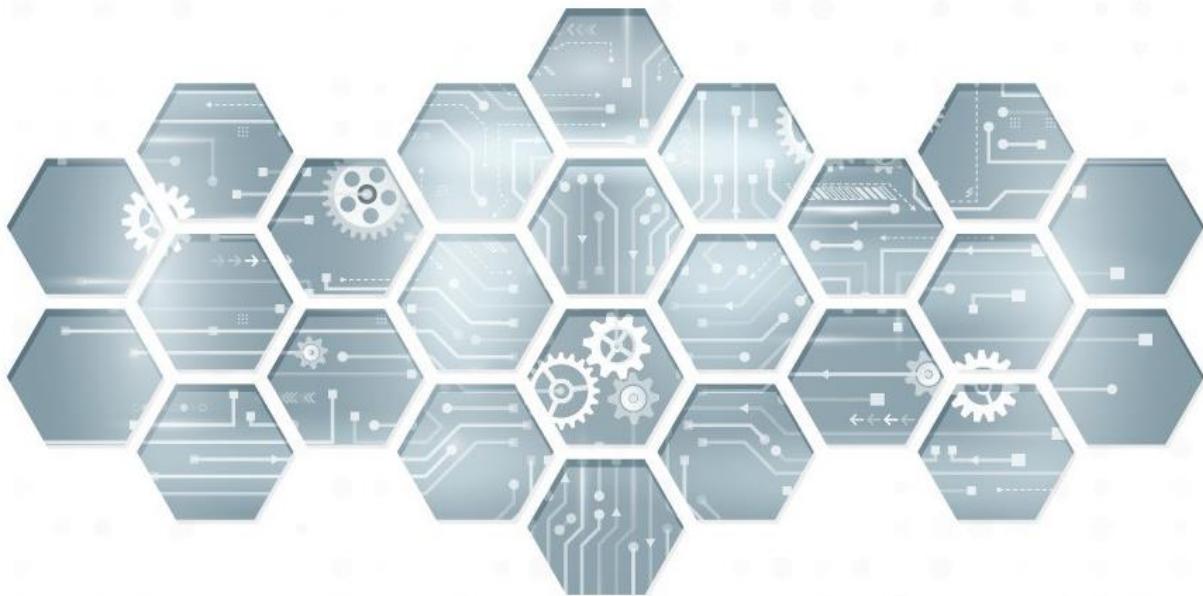
Module 11

Real-time Operating Systems (RTOS) (and other system details)

Instructor: Randall Brooks

Session: 18 | Date: January 29 – Feb 02, 2024

Location: India



Real-time Operating Systems (RTOS)

MILS/Separation kernels

RTOS

DO-178

Hypervisor

Board Support Package (BSP)

Middleware

MILS architecture



- Wikipedia defines a MILS Architecture (http://en.wikipedia.org/wiki/Multilevel_security) as follows.
 - “[MILS] is an architecture that addresses the domain separation component of MLS... The MILS approach pursues a strategy characterized by an older term, MSL (multiple single levels), that isolates each level of information within its own single-level environment (System High).”
 - MILS sometimes referred to as Multiple Single Levels (MSLS).
- MILS is an attempt to make the goals of MLS mathematically verifiable. This is done by reducing the security functionality to four key security policies:
 - **Information Flow** – Authentication and integrity for end-to-end protection of information between partitions.
 - **Data Isolation** – Confidentiality of data.
 - **Periods Processing** – Protect against covert channels.
 - **Damage Limitation** – Protection from a failure in one partition will not cascade to another partition.



MILS architecture — continued



A MILS architecture is an architecture that can be constructed out of infrastructure building blocks. The components are:

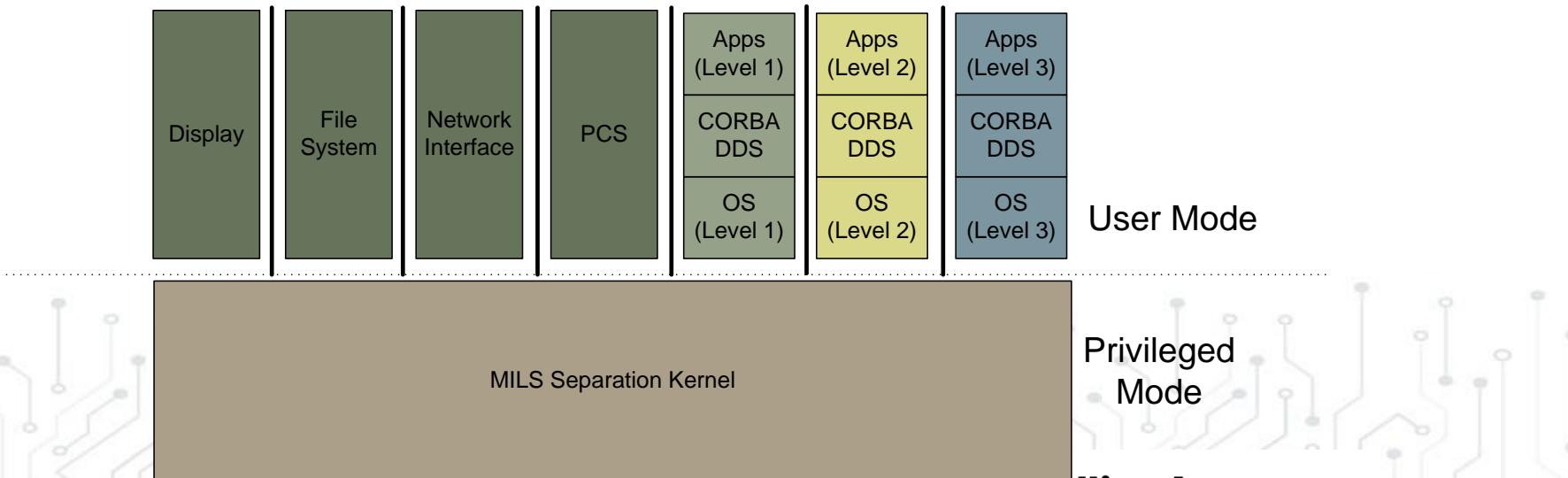
- Separation Kernel (SK)
- Hardware (Some aspects can be constructed via hardware)
- Middleware Services
- Applications



Separation kernels



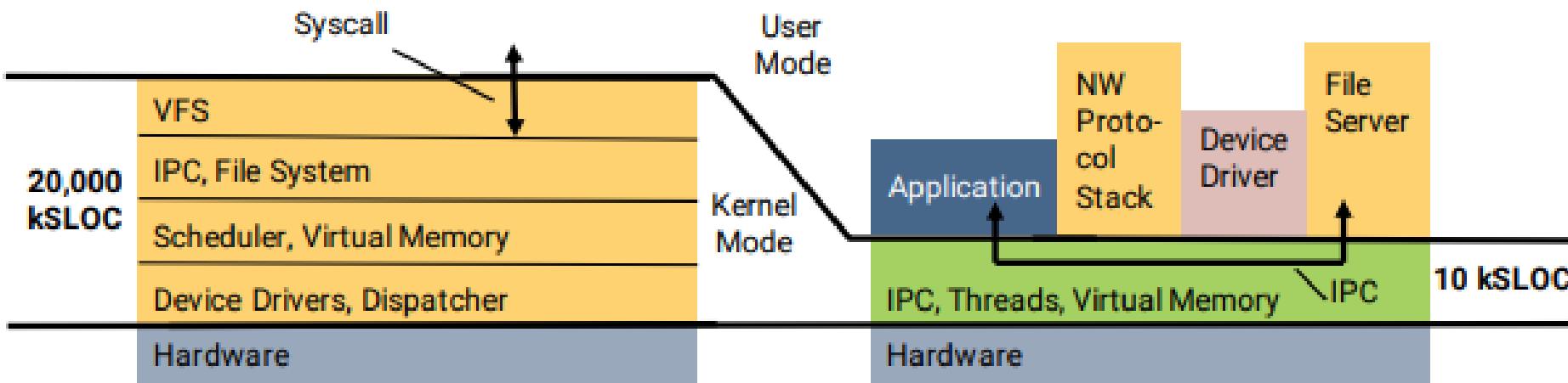
- A Partitioning or Separation Kernel is a small, less than 4000 lines of code, kernel that is solely responsible for enforcing Information Flow, Data Isolation, Periods Processing, and Damage Limitation. The compact nature of the SK allows it to be mathematically verified.
- This leads to a lower cost deployment.
- The SK runs in the “Privilege Mode” of the computer system. It could virtualize individual OSes in the “User Mode”.
- A separation kernel should have the following characteristics:
 - Non-Bypassable
 - Evaluatable
 - Always Invoked
 - Tamperproof



Microkernel definition



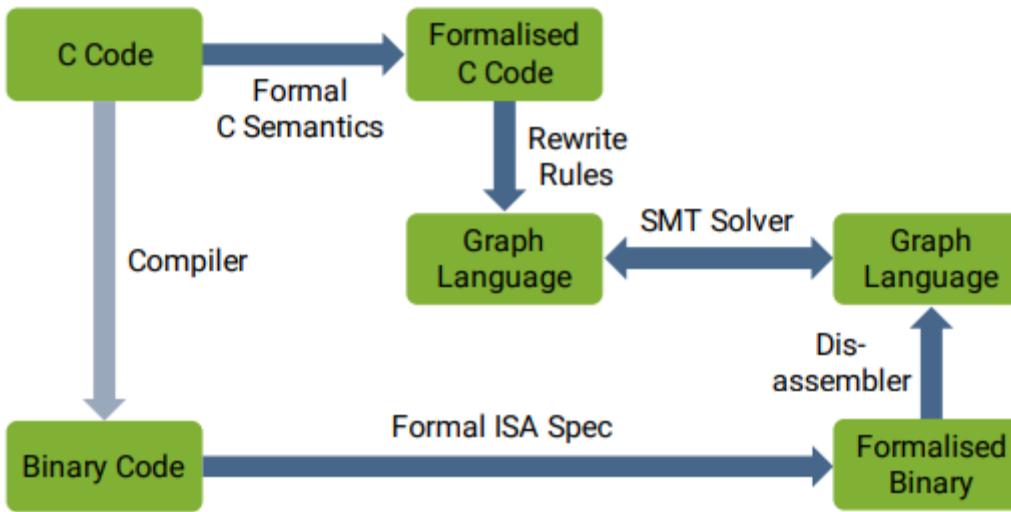
- “An operating system (OS) is the low-level system software that controls a computer system’s resources and enforces security. Unlike application software, the OS has exclusive access to a more privileged execution mode of the processor (kernel mode) that gives it direct access to hardware. Applications only ever execute in user mode and can only access hardware as permitted by the OS.” = seL4
- An OS microkernel is a minimal core of an OS, reducing the code executing at higher privilege to a minimum



Provable security (provably secure)



- “...mathematical proofs, which are common in cryptography. In such a proof, the capabilities of the attacker are defined by an adversarial model (also referred to as attacker model): the aim of the proof is to show that the attacker must solve the underlying hard problem in order to break the security of the modelled system. Such a proof generally does not consider side-channel attacks or other implementation-specific attacks, because they are usually impossible to model without implementing the system (and thus, the proof only applies to this implementation).” - Wikipedia



https://en.wikipedia.org/wiki/Provable_security



Formal methods

Formal methods can be used at a number of levels:

Level 0: Formal specification may be undertaken and then a program developed from this informally. This has been dubbed *formal methods lite*. This may be the most cost-effective option in many cases.

Level 1: Formal development and formal verification may be used to produce a program in a more formal manner. For example, proofs of properties or refinement from the specification to a program may be undertaken. This may be most appropriate in high-integrity systems involving safety or security.

Level 2: Theorem provers may be used to undertake fully formal machine-checked proofs. This can be very expensive and is only practically worthwhile if the cost of mistakes is extremely high (e.g., in critical parts of microprocessor design).

Further information on this is expanded [below](#).

As with [programming language semantics](#), styles of formal methods may be roughly classified as follows:

- [Denotational semantics](#), in which the meaning of a system is expressed in the mathematical theory of [domains](#). Proponents of such methods rely on the well-understood nature of domains to give meaning to the system; critics point out that not every system may be intuitively or naturally viewed as a function.
- [Operational semantics](#), in which the meaning of a system is expressed as a sequence of actions of a (presumably) simpler computational model. Proponents of such methods point to the simplicity of their models as a means to expressive clarity; critics counter that the problem of semantics has just been delayed (who defines the semantics of the simpler model?).
- [Axiomatic semantics](#), in which the meaning of the system is expressed in terms of [preconditions](#) and [postconditions](#) which are true before and after the system performs a task, respectively. Proponents note the connection to classical [logic](#); critics note that such semantics never really describe what a system *does* (merely what is true before and afterwards).



- A particular kind of mathematically rigorous techniques for the specification, development and verification of software and hardware systems
- The use of formal methods for software and hardware design to provide appropriate mathematical analysis can contribute to the reliability and robustness of a design
- Formal methods are best described as the application of a fairly broad variety of theoretical computer science fundamentals, in particular logic calculi, formal languages, automata theory, discrete event dynamic system and program semantics, but also type systems and algebraic data types to problems in software and hardware specification and verification.





Real-time Operating Systems (RTOS)

MILS/Separation kernels

RTOS

DO-178

Hypervisor

Board Support Package (BSP)

Middleware

Whiteboard class discussion: What Real-time Embedded Systems are you most familiar with?

RTX TGE
Technology & Global
Engineering



Instructions

- Think about **the one** Real-time Embedded Systems with which you are most familiar
- Write your choices on the flip chart provided.



Real-time Operating System definition



- An RTOS is any OS intended to serve real-time applications that process data as it comes in, typically without buffer delays.
 - Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter increments of time.
- A real time system is a time-bound system which has well defined fixed time constraints.
 - Processing must be done within the defined constraints or the system will fail.
 - They either are event driven or time sharing.
 - Event driven systems switch between tasks based on their priorities while time sharing systems switch the task based on clock interrupts.
- Most RTOS(s) use a pre-emptive scheduling algorithm.



General purpose vs. Real-time



General-Purpose Operating System (GPOS)	Real-time Operating System (RTOS)
Used for desktop PCs, Servers, and laptops	Only applied to the embedded application
Process-based scheduling	Time-based scheduling used like round-robin scheduling
Interrupt latency is not considered as important as in RTOS(s)	Interrupt lag is minimal, which is measured in a few microseconds
No priority inversion mechanism is present in the system	Priority inversion mechanism is current, so it cannot be modified by the system
Kernel's operation unlikely to be preempted	Kernel's operation may be preempted

Three types of RTOS(s)



- **Hard Real Time**
 - The deadline is handled very strictly which means that given task must start executing on specified scheduled time and must be completed within the assigned time duration.
 - Example: Medical critical care system, Aircraft systems, etc.
- **Firm Real time**
 - Missing a deadline may not have big impact but could cause undesired affects, like a huge reduction in quality of a product.
 - Example: Various types of Multimedia applications.
- **Soft Real Time**
 - Soft Real time RTOS, accepts some delays by the OS.
 - In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable.



Application tasking



- A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task
 - Variability is jitter
- A hard real-time operating system has less jitter than a soft real-time operating system.
- The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category.
- An RTOS that can usually or generally meet a deadline is a soft real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS.



Major RTOS Vendors



- VxWorks
 - www.windriver.com/products/vxworks
- Integrity and Integrity178
 - www.ghs.com/products/rtos/integrity.html
 - https://www.ghs.com/products/safety_critical/integrity_178_tump.html
- LynxOS and LynxOS-178
 - <https://www.lynx.com/products/lynxos-178-do-178c-certified-posix-rtos>

The Big Three RTOS Vendors



Other RTOS vendors

- OPENRTOS
 - <https://www.highintegritysystems.com/rtos/openrtos>
- SAFERTOS
 - <https://www.highintegritysystems.com/safertos/>
- FreeRTOS
 - <https://aws.amazon.com/freertos/>
- Embedded Linux
 - www.yoctoproject.org/
- seL4 Micro Kernel
 - <https://sel4.systems/>
- QNX Neutrino
 - <https://blackberry.qnx.com/en>
- Deos
 - https://www.ddci.com/products_deos_do_178c_arinc_653/



VxWorks by Wind River



VxWorks

The industry standard for embedded operating systems, powering billions of devices around the globe and beyond.

- Single and multi-core processor support with asymmetric multiprocessing (AMP) and symmetric multiprocessing (SMP) with support for CPU affinity
- Priority-based preemptive/adaptive scheduling
- Time and space partitioning
- Separation between kernel and memory-protected user-space environments
- POSIX Conformant
- Kernel scalability, modularity, and performance tuning
- State-of-the-art memory protection and memory management
- Virtualization ready with virtio support
- Multi-OS messaging
- ARM, PowerPC, Intel and RISC-V architecture and board support



VxWorks — continued

Support for Modern Development Languages, Frameworks, and Infrastructures

- C11, C++14, and C++17
- Python
- Rust
- Boost
- LLVM

File System

- dosFS (FAT-compatible)
- Highly reliable file system (HRFS)

Networking

- General purpose and real-time IPv4/IPv6 network stack
- Time sensitive networking (TSN)

Connectivity

- IEEE 1394
- Socket Controller Area Network (SocketCAN)
- USB (host, target, and OTG)

Multimedia

- OpenVG, OpenGL ES1, and OpenGL ES2
- Image library (JPEG and PNG)
- Input device support (mouse, touch, screen, keyboard, etc)
- PCM Audio
- OpenCV

Security

- Secure boot, ELF loader, and storage
- Kernel hardening
- Security events
- Built-in access controls
- Advanced user management
- Cryptography
- Arm TrustZone with OP-TEE support
- TPM 2.0 support
- Network security protocols
- GE Digital® Achilles Level II certified for compliance with IEC 62443 part 4-2

Safety

- Safety certifications for DO-178C, IEC 61508, IEC 62304, and ISO 26262



Green Hills



INTEGRITY
Most Reliable and Secure
Operating System
INTEGRITY for Embedded
INTEGRITY for Security
INTEGRITY for Automotive
INTEGRITY for Industrial
Control
INTEGRITY for Medical

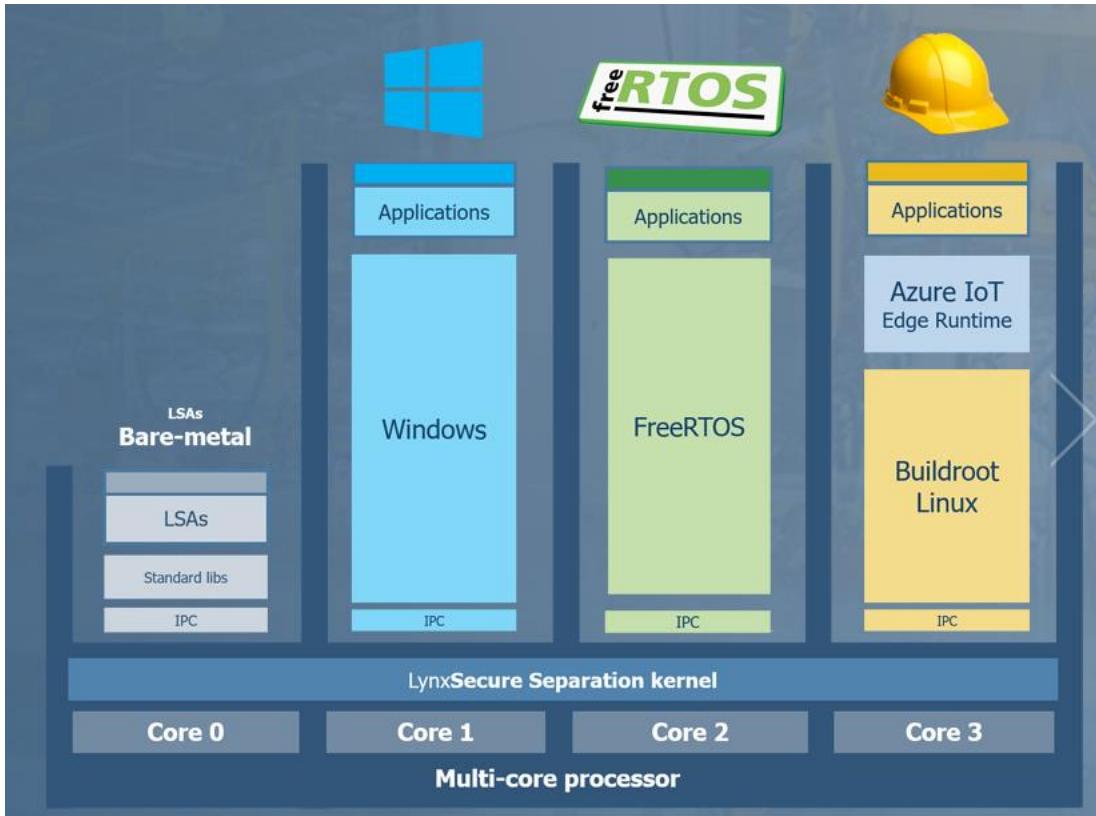
INTEGRITY-178
Only Operating System
Certified to EAL 6+ and
DO-178B, Level A

- The INTEGRITY-178 tuMP real-time operating system (RTOS) is the world-leading multicore RTOS for safety- and security-critical applications.
- A Unified solution for multicore processors.
- INTEGRITY-178 and INTEGRITY-178 tuMP are part of systems that have been certified both to the highest levels of airborne safety (DO-178B/C DAL A) and security (SKPP/EAL 6+) for over 80 airborne systems.
- INTEGRITY-178 tuMP was the first operating system certified conformant to the latest Future Airborne Capability Environment (FACE™) technical standard, edition 3.0, and it is certified for both the safety base and security profiles.
- Supports ARM, Intel, and PowerPC

EAL 6+ against the Separation Kernel Protection Profile



LynxOS



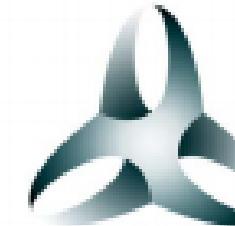
- Platform technology that controls hardware resources according to an intuitive information flow modeling language.
- It is the foundation of the LYNX MOSA.ic™ framework and was designed to satisfy real-time, high assurance computing requirements used to regulate military and industrial computing environments, such as NIST, NSA Common Criteria, and NERC CIP.
- LYNX MOSA.ic™ (Multi-core Certification)
- LynxOS-178® (Safety-certified RTOS)
- LynxSecure® (Hypervisor)
- Supports ARM, Intel, and Power PC

OPENRTOS by WITTENSTEIN high integrity systems

RTX TGE
Technology & Global
Engineering



- “A New Approach To Embedded Software”
- OPENRTOS® provides a commercial license for Amazon FreeRTOS
 - License includes the FreeRTOS kernel
 - Under MIT License and free to download
- Updates and ports of the FreeRTOS kernel are simultaneously released by WITTENSTEIN as OPENRTOS, with full commercial support and licensing.
- ARM Support
- “Related” to FreeRTOS and SAFERTOS
- WITTENSTEIN is a UK Company



WITTENSTEIN

<https://www.highintegritysystems.com/openrtos/>



Collins Aerospace
An RTX Business

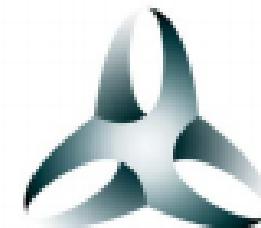
| 11-22

SAFERTOS by WITTENSTEIN high integrity systems

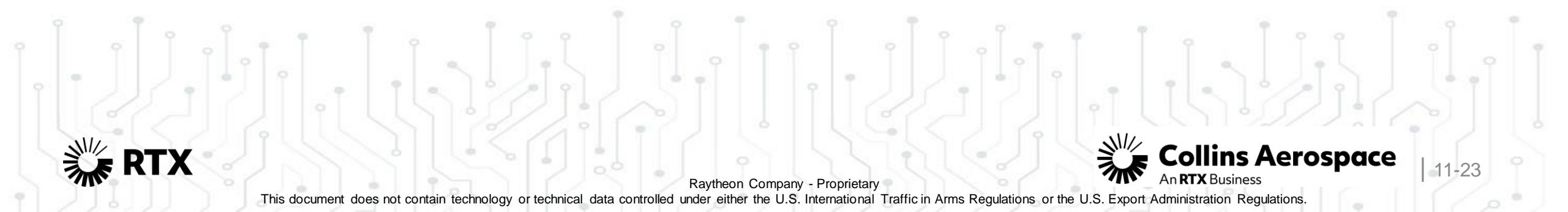
RTX TGE
Technology & Global
Engineering



- “A pre-certified safety Real Time Operating System (RTOS) for embedded processors. It delivers superior performance and pre-certified dependability, whilst utilizing minimal resources.”
- Developed by WHIS, a safety systems company
- Supports a wide range of international development standards
- Based on the FreeRTOS functional model
- Root of Trust support
- ARM support
 - <https://www.highintegritysystems.com/safertos/supported-platforms/>
- Enhanced Security Module (ESM)
 - https://www.highintegritysystems.com/downloads/white_papers/ESM_Security_whitepaper.pdf



WITTENSTEIN



RTX

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

Raytheon Company - Proprietary

Collins Aerospace
An RTX Business

| 11-23

FreeRTOS



- Offered by Amazon to use with Internet of Things devices and Amazon Web Services
- Fail Safe File System
- WolfSSL Support
- BSP Support:
 - Atmel
 - Future Design Inc.
 - NXP
 - ST
- Supports ARM

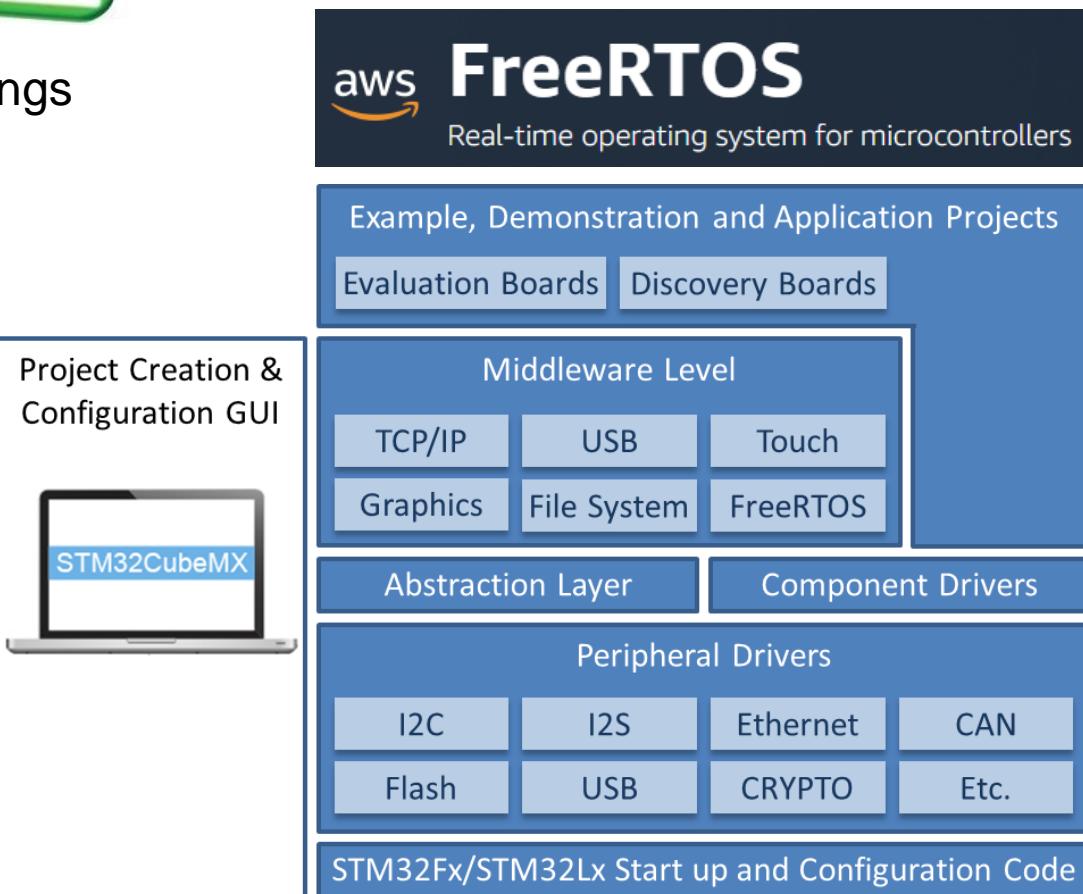


Image credit: ST

<https://freertos.org/community/ecosystem/overview.html>

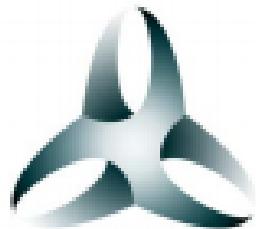


FREERTOS vs SAFERTOS



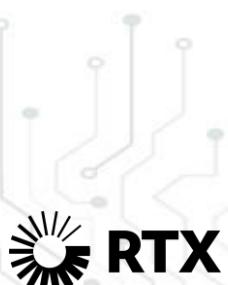
SAFERTOS and FreeRTOS share a functional model, so feel very similar to use. There are however some key differences. Compared with FreeRTOS, SAFERTOS:

- Has fewer Application Programming Interface (API) functions
- Does more error checking.
- Returns a status code from most API calls (with other return data passed by reference).
- Requires the application to supply all stack, task control block, and queue buffers.
- Strongly encourages 100% static allocation and provides no "heap" functions.
- Uses a processor's Memory Protection Unit (MPU) by default.
- Has been completely re-engineered to meet safety critical software needs.



WITTENSTEIN

Because of this, when migrating a FreeRTOS project to SAFERTOS there is work to do to get the kernel up and running.



Yocto Project



- The Yocto Project (YP) is an open-source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture.
- The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IoT devices, or anywhere a customized Linux OS is needed.
- Supports ARM and Intel



Wind River Linux



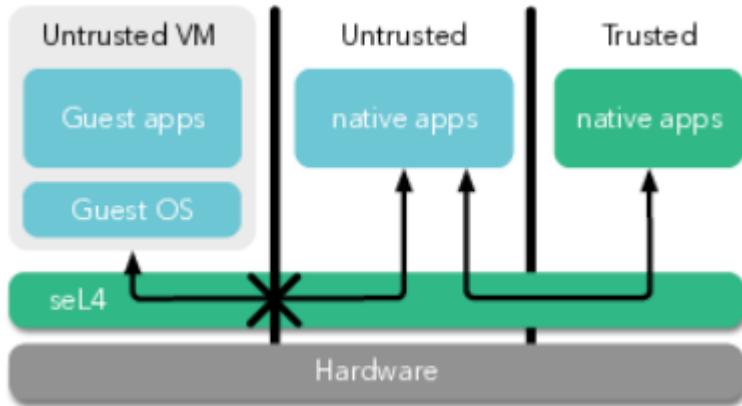
- Wind River Linux is a software development environment that creates optimized Linux distributions for embedded devices.
- Wind River Linux is based on the Yocto Project implementation of the OpenEmbedded Core (OE-Core) metadata project.
- The YP uses build recipes and configuration files to define the core platform project image and the applications and functionality it provides.
- Wind River Linux builds on this core functionality and adds Wind River-specific extensions, tools, and services to facilitate the rapid development of embedded Linux platforms.

Wind River Linux





1 seL4



- “seL4 is a high-assurance, high-performance operating system microkernel. It is unique because of its comprehensive formal verification, without compromising performance. It is meant to be used as a trustworthy foundation for building safety- and security-critical systems. It is available as open source on GitHub and supported by the seL4 Foundation.” -seL4
- Has been “proved correct” (provably secure)
 - functionality, expressed in a mathematical language called higher-order logic (HOL).
- [It]“...has undergone a complete and sound analysis of its worst-case execution time (WCET) [Blackham et al., 2011, Sewell et al., 2017].”
- Hypervisor support
- Rust Support is user mode
- Extensive Collins Contribution

<https://github.com/seL4/>



QNX Neutrino

- BlackBerry® QNX offers a broad range of safety-certified and secure software products, complemented by world-class professional services, to help embedded developers increase reliability, shorten time-to-market and reduce development cost.
- 175 Million Vehicles
- POSIX-compliant development
- An embedded OS pre-certified for
 - IEC 61508 SIL3, ISO 26262 ASIL D and IEC 62304 Class C

Why Choose BlackBerry QNX

These are just some of the reasons leading embedded systems manufacturers choose BlackBerry QNX.



Safety

Accelerate time to market with software pre-certified to IEC 61508, ISO 26262 and IEC 62304.

Training and safety services to help you reach new levels of effectiveness and development speed.



Security

Cybersecurity is at the center of our RTOS and hypervisor development.

Field-proven embedded security expertise to help protect your systems from increasing cyberthreats.



Scalability

Fully-managed microkernel RTOS lets you roll out a single OS across all product lines.

Focus your best developers on value-added features, not OS maintenance.



Reliability

Deterministic RTOS for the performance expected from today's embedded systems.

Development tools to address latency or lag issues throughout the development lifecycle.



1 Deos

- “Deos™, DDC-I’s safety-critical time and space partitioned DO-178C Design Assurance Level A (DAL A) certifiable real-time operating system (RTOS) for Avionics, supports ARINC 653 APEX”
- Supports Intel, PowerPC, ARM and MIPS microprocessors

**DDC-I**Safety Critical Software Solutions
for Mission Critical Systems
[Home](#) [Products](#) ▾ [Services](#) ▾ [Industries](#) ▾ [News](#) ▾ [Success](#) ▾ [About](#) ▾


Deos, a Time & Space Partitioned, Multi-core Enabled, DO-178C DAL A Certifiable RTOS

Safety Critical RTOS for FAA Certifiable Avionics Applications

Deos™, DDC-I’s safety-critical time and space partitioned DO-178C Design Assurance Level A (DAL A) certifiable real-time operating system (RTOS) for Avionics, supports ARINC 653 APEX, Rate Monotonic Scheduling (RMS), and is targeted at the FACE Safety Base Profile. It has been field proven as a safety-critical certifiable RTOS since its first verification and audit to DAL A by Transport Canada in 1998, and it has been certified and flying in 10’s of thousands of aircraft. Since the initial certification, it has continually evolved throughout the last two decades with new processors and features in subsequent baselines, and it has been successfully audited by the world’s various governmental certification authorities (FAA, ENAC, JAA, EASA, CAAC, and others) and Airframe and Avionics Supplier Designated Engineering Representatives (DERs).

Avionics Applications using Deos

Deos has been used to manage resources and hard partition avionics applications on x86, PowerPC, ARM and MIPS microprocessors for a multitude of flight critical functions that require bounded processing, high determinism and high throughput. These functions include: air data computers, air data inertial reference units, cockpit video, displays, flight instrumentation, electronic flight bags, engine management, enhanced ground proximity warning, FADECs, flight controls, flight management systems, maintenance systems, power distribution systems, radios, traffic collision avoidance systems (TCAS), weather radar and many more federated and IMA avionics systems.



Primus Epic Avionics Display Application using Deos



<https://www.ddci.com/>

Raytheon Company - Proprietary

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.



| 11-30

Other RTX (non-RTOS)



- rclinux (Rockwell Collins Linux)
- Baremetal “Small OS”
 - A Collins (Phoenix, AZ) custom-developed operating system (Boot module)
 - A purpose-built, lightweight, custom OS with pedigree based on legacy P&W products
 - Follows the Keep It Simple (KISS) principle, minimizes compute resource demands, and presents a small, manageable threat surface that can be reliably evaluated for potential cyber vulnerabilities
 - Round Robin execution





Real-time Operating Systems (RTOS)

MILS/Separation kernels

RTOS

DO-178

Hypervisor

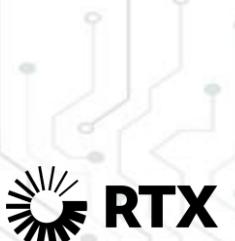
Board Support Package (BSP)

Middleware

DO-178 safety critical



- DO-178B, Software Considerations in Airborne Systems and Equipment Certification is a guideline dealing with the safety of safety-critical software used in certain airborne systems.
 - Although technically a guideline, it was a de facto standard for developing avionics software systems until it was replaced in 2012 by DO-178C.
- DO-178C, Software Considerations in Airborne Systems and Equipment Certification is the primary document by which the certification authorities such as FAA, EASA and Transport Canada approve all commercial software-based aerospace systems.
 - The document is published by RTCA, Incorporated, in a joint effort with EUROCAE, and replaces DO-178B.
 - The new document is called DO-178C/ED-12C and was completed in November 2011 and approved by the RTCA in December 2011 and became available for use in January 2012.



Software level/DAL



Software Level, also known as the Design Assurance Level (DAL) or Item Development Assurance Level (IDAL), is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system. The failure conditions are categorized by their effects on the aircraft, crew, and passengers.

- **A-Catastrophic** – Failure may cause deaths, usually with loss of the airplane.
- **B-Hazardous** – Failure has a large negative impact on safety or performance or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers.
- **C-Major** – Failure significantly reduces the safety margin or significantly increases crew workload. May result in passenger discomfort (or even minor injuries).
- **D-Minor** – Failure slightly reduces the safety margin or slightly increases crew workload. Examples might include causing passenger inconvenience or a routine flight plan change.
- **E-No Effect** – Failure has no impact on safety, aircraft operation, or crew workload.

Design Assurance Level (DAL)	Description	Target System Failure Rate	Example System
Level A (Catastrophic)	Failure causes crash, deaths	<1 x 10 ⁻⁷ chance of failure/flight-hr	Flight controls
Level B (Hazardous)	Failure may cause crash, deaths	<1 x 10 ⁻⁷ chance of failure/flight-hr	Braking systems
Level C (Major)	Failure may cause stress, injuries	<1 x 10 ⁻⁵ chance of failure/flight-hr	Backup systems
Level D (Minor)	Failure may cause inconvenience	No safety metric	Ground navigation systems
Level E (No effect)	No safety effect on passengers/crew	No safety metric	Passenger entertainment

DO-178C alone is not intended to guarantee software security aspects.





Real-time Operating Systems (RTOS)

MILS/Separation Kernels

RTOS

DO-178

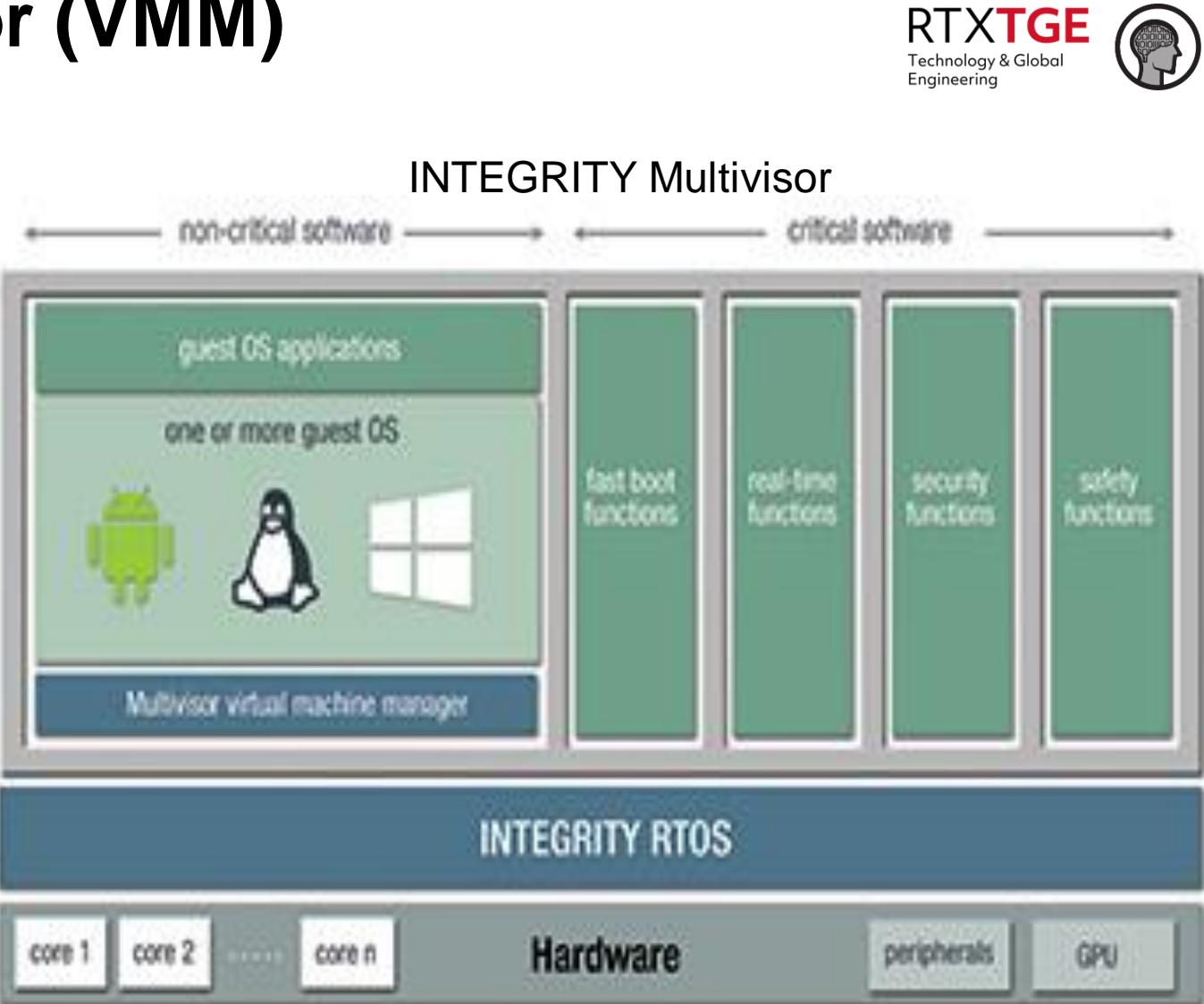
Hypervisor

Board Support Package (BSP)

Middleware

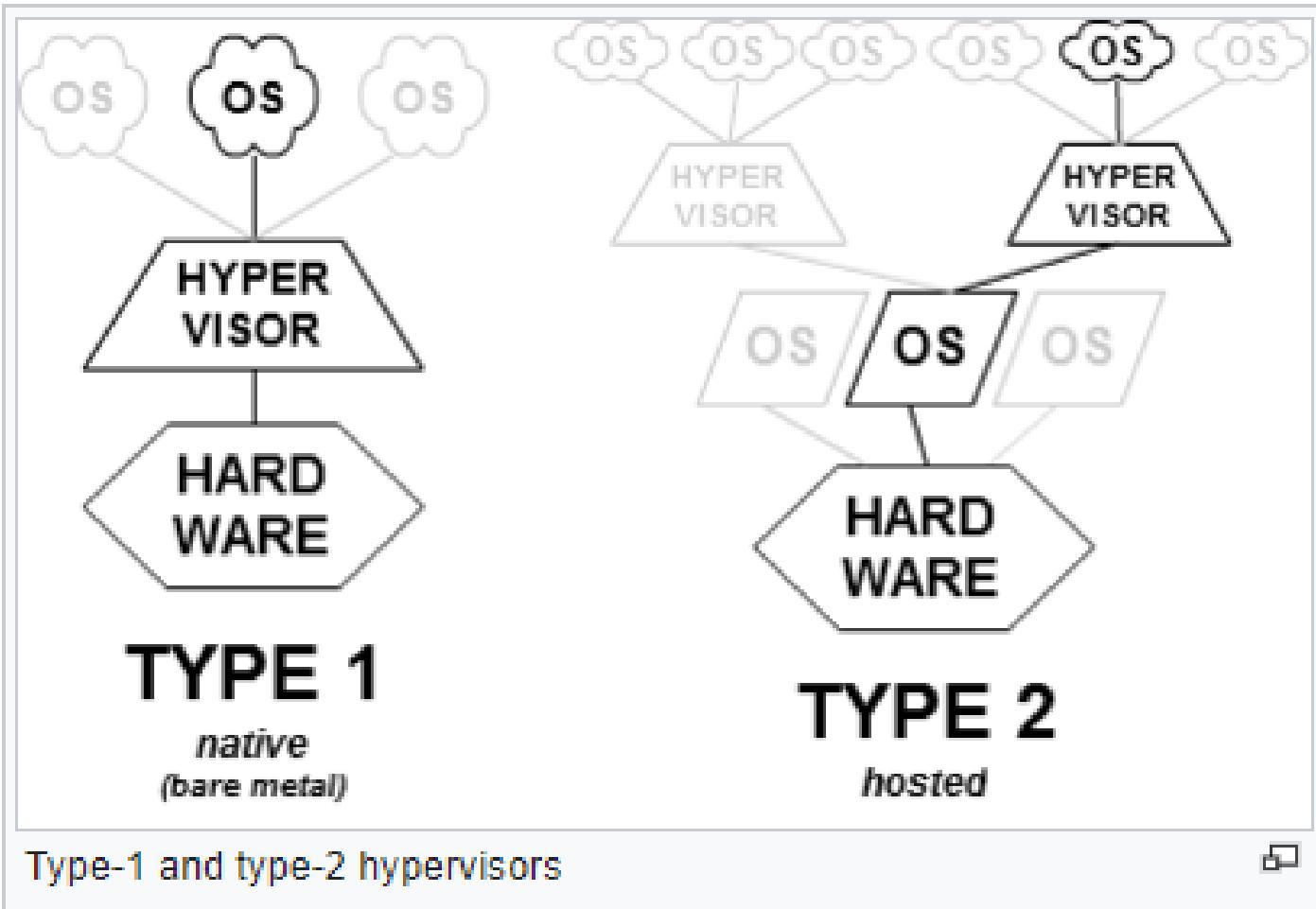
Virtual Machine Monitor (VMM)

- A hypervisor or virtual machine monitor (VMM) is computer software, firmware or hardware that creates and runs virtual machines.
- A computer on which a hypervisor runs one or more virtual machines is called a **host machine**, and each virtual machine is called a **guest machine**.
- The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems.
- Multiple instances of a variety of operating systems may share the virtualized hardware resources.



Hypervisor types

- Type-1, native or bare-metal hypervisors
- Type-2 or hosted hypervisors





Real-time Operating Systems (RTOS)

MILS/Separation kernels

RTOS

DO-178

Hypervisor

Board Support Package (BSP)

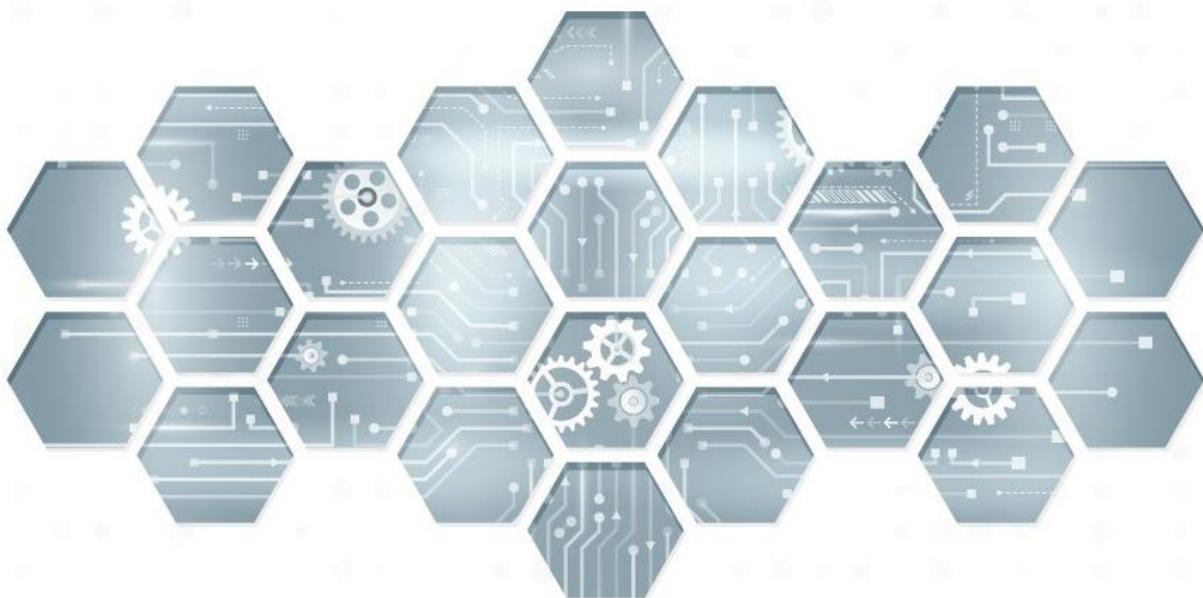
Middleware

Board Support Package (BSP)



- "...the layer of software containing hardware-specific drivers and other routines that allow a particular operating system (traditionally a real-time operating system, or RTOS) to function in a particular hardware environment (a computer or CPU card), integrated with the RTOS itself."
- Third-party hardware developers who wish to support a particular RTOS must create a BSP that allows that RTOS to run on their platform. In most cases the RTOS image and license, the BSP containing it, and the hardware are bundled together by the hardware vendor.
- Additionally the BSP is supposed to perform the following operations
 - Initialize the processor
 - Initialize the bus
 - Initialize the interrupt controller
 - Initialize the clock
 - Initialize the RAM settings
 - Configure the segments
 - Load and run bootloader from flash





Real-time Operating Systems (RTOS)

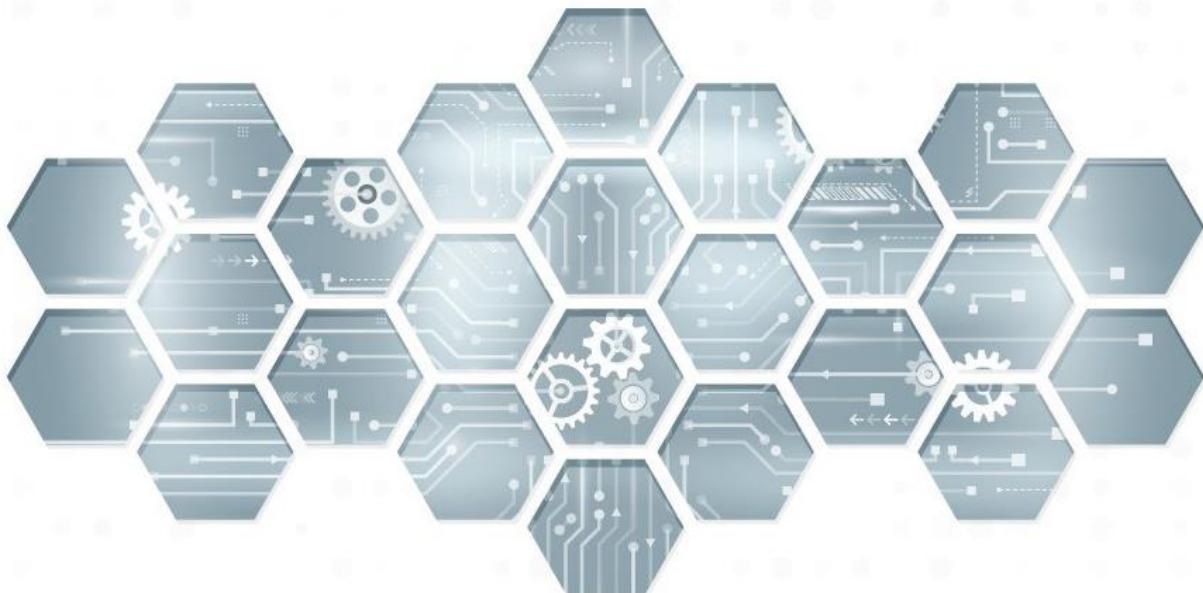
MILS/Separation kernels
RTOS
DO-178
Hypervisor
Board Support Package (BSP)
Middleware

Middleware



- Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue".
- Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.
- Example(s):
 - Common Object Request Broker Architecture (CORBA)
 - Data Distribution Service (DDS)
 - Partitioning Communications System (PCS)





Real-time Operating Systems (RTOS)

MILS/Separation kernels

RTOS

DO-178

Hypervisor

Board Support Package (BSP)

Middleware

Module summary and wrap-up

Questions?



Raytheon Company - Proprietary

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

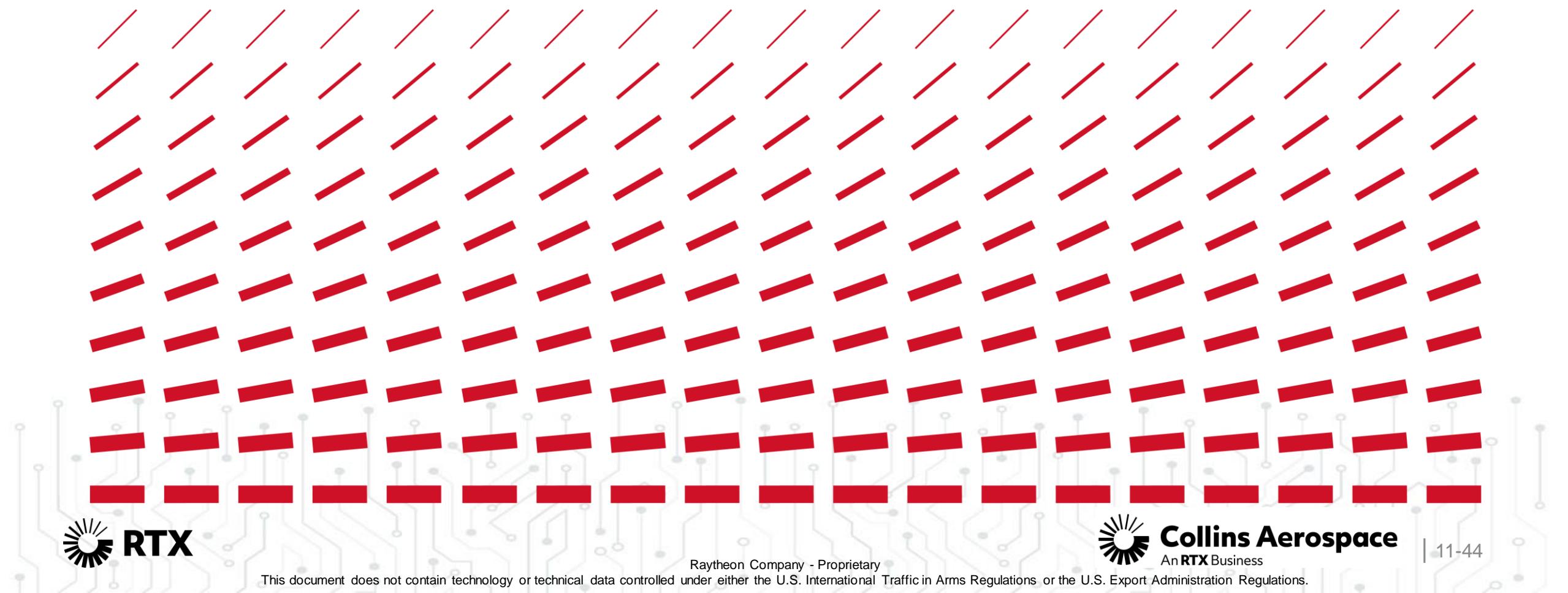


Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.





Collins Aerospace
An RTX Business

Software Assurance Overview

Software Assurance

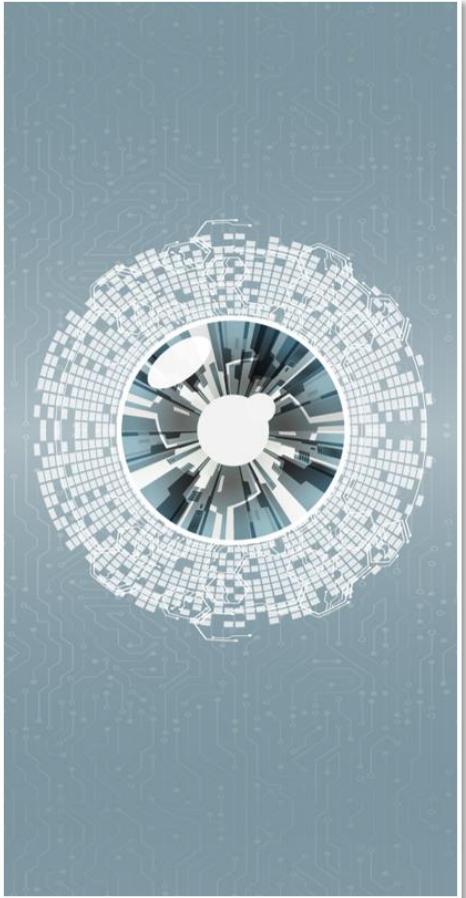
Application Security or Product Cybersecurity

Instructor: Randall Brooks, CISSP, CSSLP

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

Module objectives

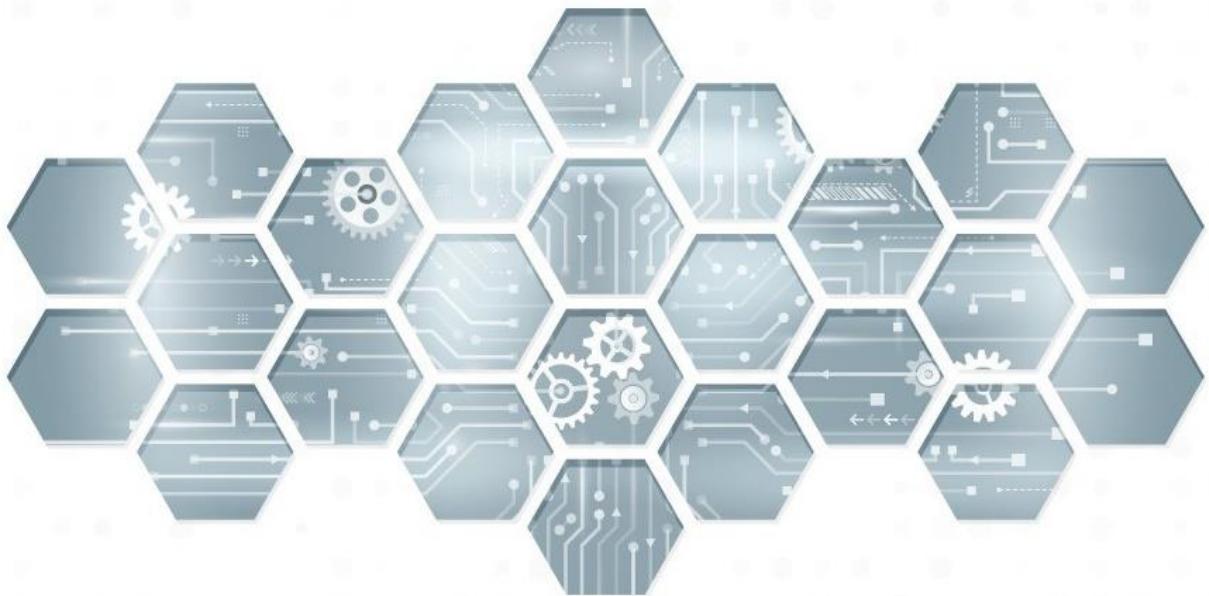


Why is software assurance including supply chain assurance so important?

General concepts of software assurance and design principles

Key activities across the product life cycle

Methods of assessing and scoring software risk



Software Assurance

Why is software assurance including supply chain assurance so important?

General concepts of software assurance and design principles

trade-offs of investing in systems and software assurance

Key activities across the product life cycle

Methods of assessing and scoring software risk

Software Assurance (SwA) definition



- "Software assurance (SwA) is a critical process in software development that ensures the reliability, safety, and security of software products. It involves a variety of activities, including requirements analysis, design reviews, code inspections, testing, and formal verification. One crucial component of software assurance is secure coding practices ... such as those outlined by the Software Engineering Institute (SEI) in their CERT Secure Coding Standards (SCS)."
– Wikipedia

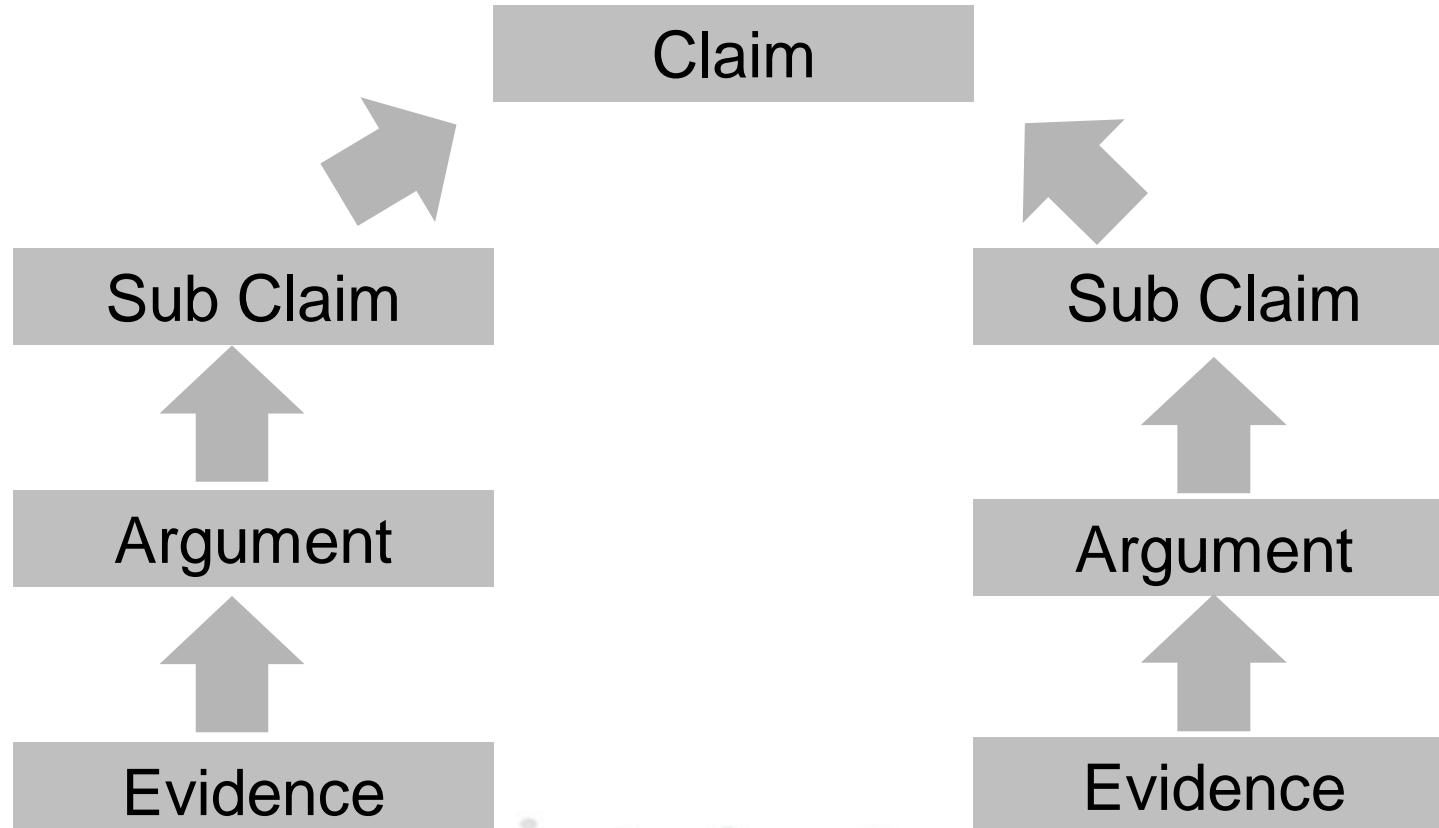


Source: <http://www.safecode.org/>

https://en.wikipedia.org/wiki/Software_assurance



Assurance case



Software applications - the new attack target

RTX TGE
Technology & Global
Engineering

- Application security has often been ignored, in part because of the faulty assumption that firewalls and other perimeter defenses can protect the functional code.
- The problem is further compounded because application developers without specific security training are typically unaware of the ways their software, while meeting functional requirements, could be compromised.
- As the operating system and network security vulnerabilities have been reduced over time, applications have become the next attack target.



Today, the attackers are targeting the applications.



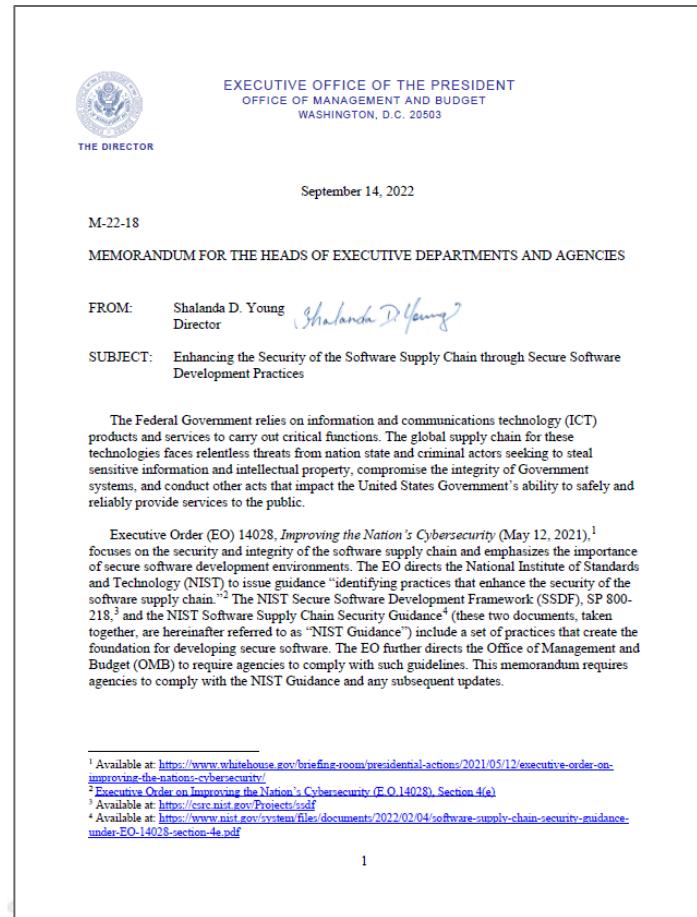
Why Software Assurance?



The simple answer is Commercial industry and customers demand it...

- US Government's Office of Management and Budget (OMB) M-22-18, September 14, 2022, requires support for the NIST SP 800-218: Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities for **Critical Software**
 - Self attestation or 3rd party assessment of a Secure System Development Lifecycle

Produce Well-Secured Software (PW)			
Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1): Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design and architecture should mitigate those risks; and justify any cases where risk-based analysis	PW.1.1: Use forms of risk modeling – such as threat modeling, attack modeling, or attack surface mapping – to help assess the security risk for the software.	EP5 Threat Modeling: RTX uses Threat Modeling to model potential threats, such as product vulnerabilities, are identified, enumerated, and prioritized – all from a hypothetical attacker's point of view. Once prioritized, countermeasures are designed to prevent, or mitigate, the effects of the threats.	4e(ix)



<https://www.whitehouse.gov/wp-content/uploads/2022/07/M-22-14.pdf>



NIST SP 800-218: SSDF Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities

RTX TGE
Technology & Global
Engineering

- **Prepare the Organization (PO):** Organizations should ensure that their people, processes, and technology are prepared to perform secure software development at the organization level. Many organizations will find some PO practices to also be applicable to subsets of their software development, like individual development groups or projects.
- **Protect the Software (PS):** Organizations should protect all components of their software from tampering and unauthorized access.
- **Produce Well-Secured Software (PW):** Organizations should produce well-secured software with minimal security vulnerabilities in its releases.
- **Respond to Vulnerabilities (RV):** Organizations should identify residual vulnerabilities in their software releases and respond appropriately.



Includes: Practice, Tasks, Notional Implementation Examples, and References



Software Bill of Materials (SBOM)



- A “software bill of materials” (SBOM) has emerged as a key building block in software security and software supply chain risk management.
- A SBOM is a nested inventory, a list of ingredients that make up software components. The SBOM work has advanced since 2018 as a collaborative community effort, driven by US's National Telecommunications and Information Administration's (NTIA) multistakeholder process.
- An SBOM-related concept is the Vulnerability Exploitability eXchange (VEX). A VEX document is an attestation, a form of a security advisory that indicates whether a product or products are affected by a known vulnerability or vulnerabilities.

Source: <https://www.cisa.gov/sbom>



SBOM (continued)

- SBOM Formats/Data Exchanges (DXs) include:
 - OWASP CycloneDX, <https://cyclonedx.org/>
 - Linux Foundation Software Package Data Exchange (SPDX), <https://spdx.dev/>
 - ISO/IEC 5962:2021
- The Raytheon Technologies standard tool for Software Composition Analysis is Synopsys Black Duck
 - Black Duck can scan a code base to determine its pedigree
 - It can output an SBOM as Cyclone DX or SPDX



The screenshot shows a software interface for 'Black Duck Projects'. The main title is 'SIG Insecure Bank - SSDC > demo'. The status bar indicates 'Up to Date'. The interface includes a search bar and various filters like 'Components', 'Security', and 'Ignore'. A table lists dependencies with columns for Component, Source, Match Type, Usage, and License. All listed components are from Apache, specifically Log4j versions 1.2.17 and 2.17.0, Log4j API 2.17.0, beanvalidation-api 1.0.0, BoneCP 0.8.0.RELEASE, and Byte Buddy 1.8.0, all identified as 'Dynamically Linked'.

Component	Source	Match Type	Usage	License
Apache Log4j 1.2.17	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0
Apache Log4j 2.17.0	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0
Apache Log4j API 2.17.0	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0
beanvalidation-api 1.0.0	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0
BoneCP 0.8.0.RELEASE	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0
Byte Buddy 1.8.0	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0

Image is from Synopsys' example

<https://www.synopsys.com/blogs/software-security/building-sbom-with-black-duck/>

Cloud Security Alliance (CSA) Pandemic Eleven “Top Threats”

RTX TGE
Technology & Global
Engineering



Survey Results Rank	Survey Average Score	Issue Name
1	7.729927	Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts
2	7.592701	Insecure Interfaces and APIs
3	7.424818	Misconfiguration and Inadequate Change Control
4	7.408759	Lack of Cloud Security Architecture and Strategy
5	7.275912	Insecure Software Development
6	7.214493	Unsecure Third Party Resources
7	7.143066	System Vulnerabilities
8	7.114659	Accidental Cloud Data Disclosure/ Disclosure
9	7.097810	Misconfiguration & Exploitation of Serverless & Container Workloads
10	7.088534	Organized Crime/ Hackers/ APT
11	7.085631	Cloud Storage Data Exfiltration

Insecure Software Development is the 5th “Top Threat” (Risk) to Cloud Security

International standards exist: the ISO/IEC 27034

RTX TGE
Technology & Global
Engineering

RTX is an active member of International Committee for Information Technology Standards (INCITS).

ISO/IEC 27034-1:2011
Information technology - Security techniques

ISO/IEC 27034-2:2015
Organization normative framework

ISO/IEC 27034-3:2018
Application security management process

ISO/IEC 27034-4
Application security validation

ISO/IEC 27034-5:2017
Protocols and application security control data structure

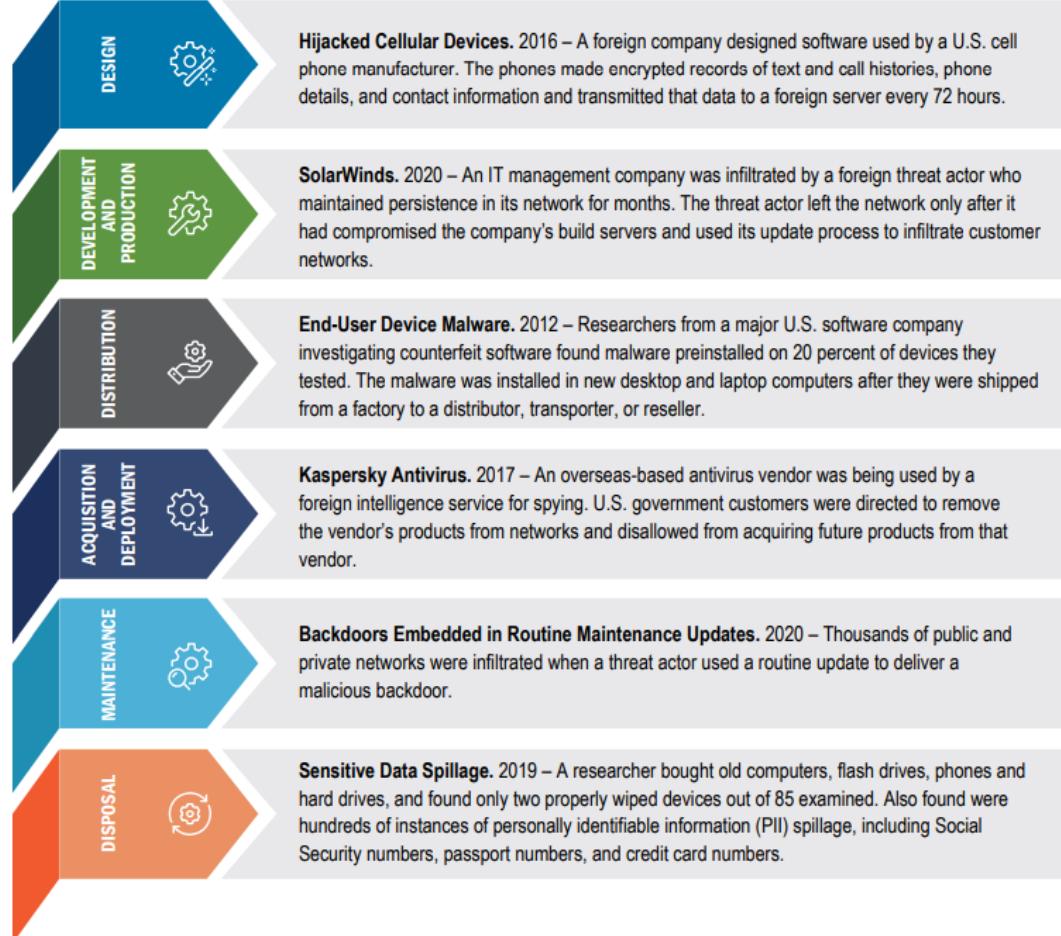
ISO/IEC 27034-5-1:2018
Protocols and application security control data structure, XML schemas

ISO/IEC 27034-6:2016
Case studies

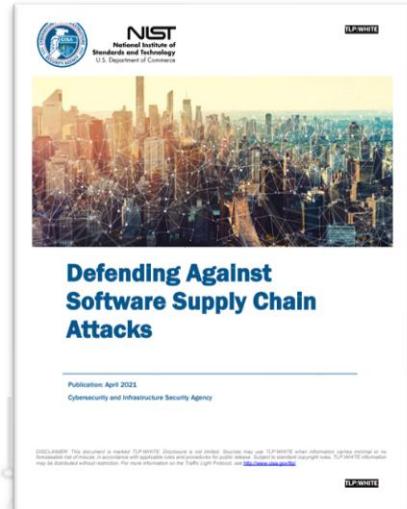
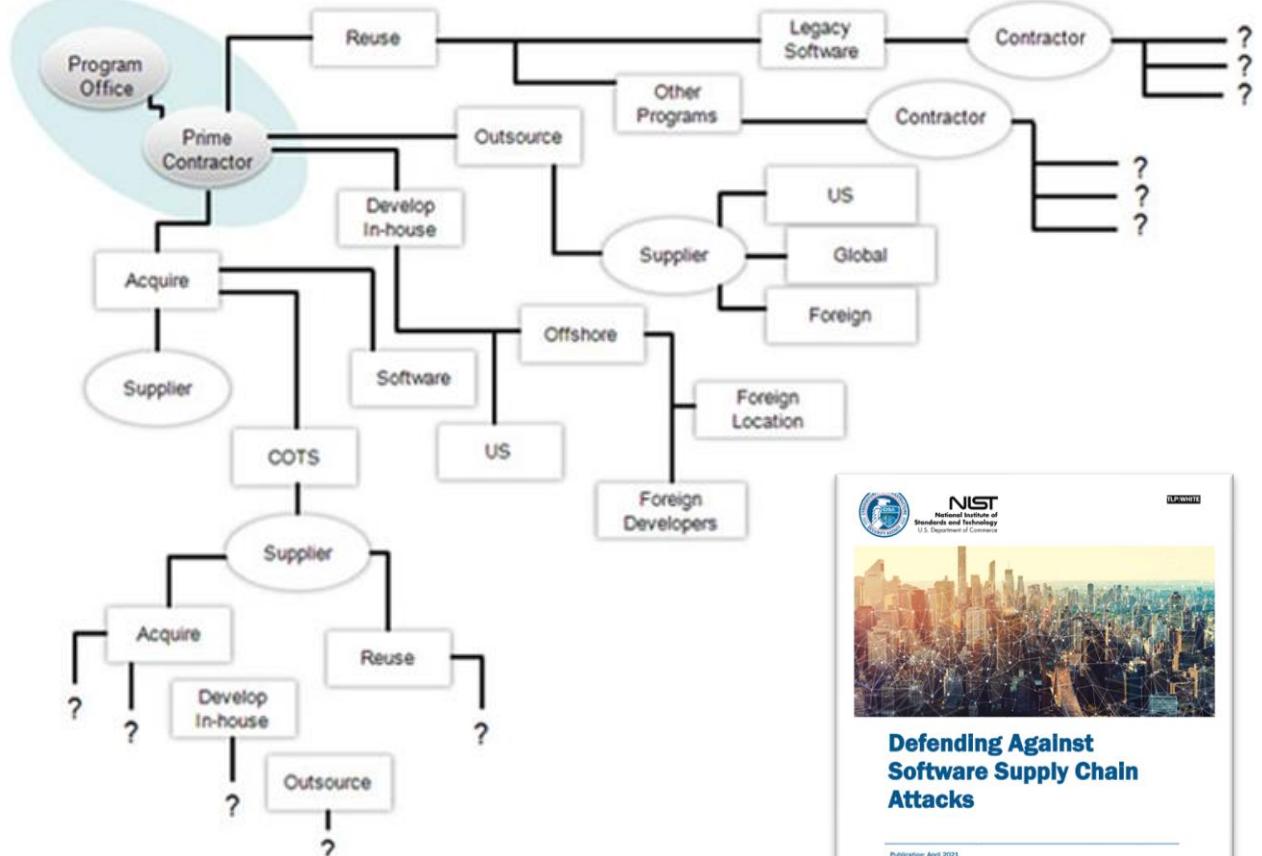
ISO/IEC 27034-7:2018
Application security assurance prediction

Why supply chain assurance wrt SwA?

Table 1: ICT Supply Chain Lifecycle and Examples of Threats



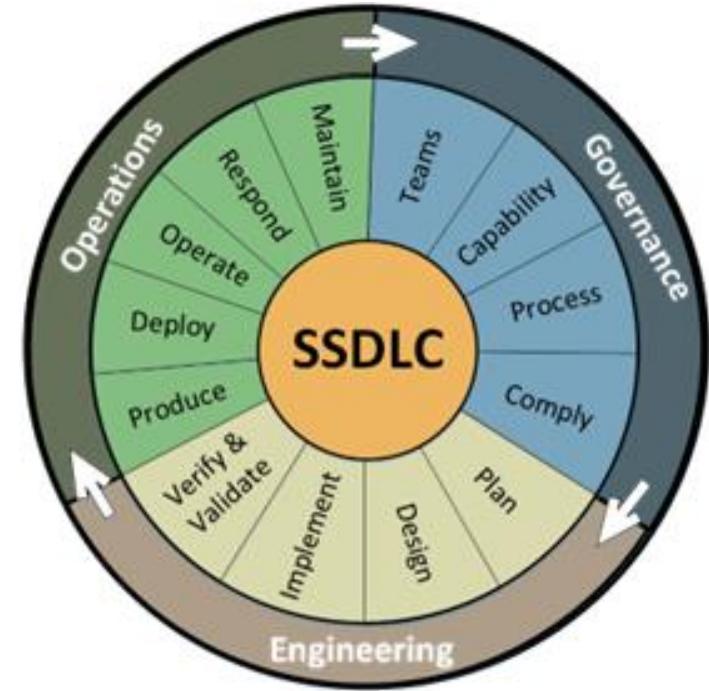
Global Supply Chain



Source: https://www.cisa.gov/sites/default/files/publications/defending_against_software_supply_chain_attacks_508_1.pdf

Secure Systems Development Lifecycle (SSDLC)

- The RTX Reference SSDLC is a database and wiki for corporate-wide product security best practices.
 - The SSDLC is composed of 85 product security practices.
- SSDLCs are either tightly integrated or overlaid onto engineering lifecycle processes.
 - This SSDLC complements, and does not override, BU SSDLCs and practices (e.g., Command Media)
- The SSDLC practices are currently located on an RTX Corporate SharePoint at:
 - https://rtxusers.sharepoint.us/sites/CODECenter-CORP/SSDLC/SitePages/SSDLC_Homepage.aspx
- The SSDLC is mapped to the SSDF.
 - An enabler lists the SSDF and references the SSDLC



Product Cybersecurity Maturity Model (PCMM)



- The RTX Product Cybersecurity Maturity Model (PCMM) is a tool created to measure “How Well” select SSDLC practices are being executed by project teams.
- The PCMM is composed of a set of 152 assessment questions that determine the maturity level for each of the 17 practices.
 - The PCMM was developed by the CODE Center with a working group consisting of all 4 BU PCOs and their Engineering Fellows.
- The PCMM scores identify both gaps and excellence.
- Executive management uses the PCMM to better focus resources, such as tools, people, budget and effort.

Governance		Engineering		
Organize	Comply	Design	Implement	Verify & Validate
Organization Capabilities	Policy	Security Requirements	Hardware Assurance	Security Verification
Organization Process Performance	Compliance	Security Architecture	Software Assurance	Vulnerability & Incident management
	SSDLC / IPDS	Security Plans	Supply Chain Risk Management	Survivability V&V
		Security Risk Assessment		
		Security Detailed Design		
		Survivability Design		



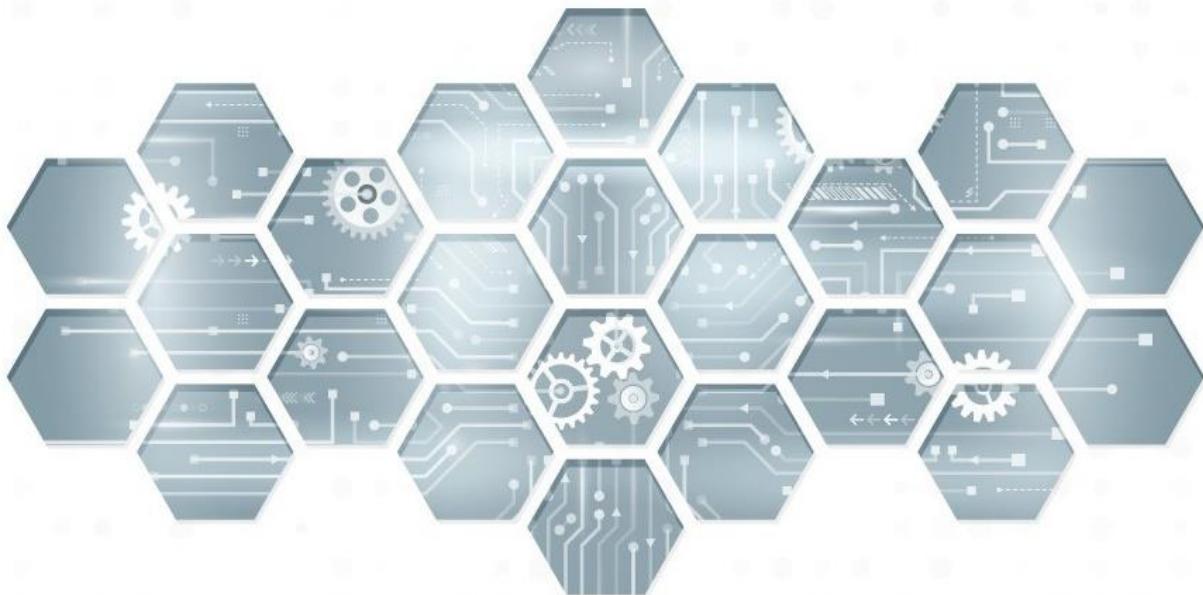
SSDLC informs while PCMM measures

Raytheon Company - Proprietary

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.



| 12-15



Software Assurance

Why is software assurance including supply chain assurance so important?

General concepts of software assurance and design principles

Key activities across the product life cycle
Methods of assessing and scoring software risk

Secure Design Principles

- **Economy of mechanism**
 - Keep the design of the system as simple and small as possible
- **Fail-safe defaults**
 - Base access decisions on permission (a user is explicitly allowed access to a resource) rather than exclusion (a user is explicitly denied access to a resource)
- **Complete mediation**
 - Every access to every object must be checked for authorization
- **Least privilege**
 - Every program and every user of the system should operate using the least set of privileges necessary to complete the job



Source: SAFECode

Secure Design Principles (continued)

- **Least common mechanism**
 - Minimize the amount of mechanism common to more than one user and depended on by all users
- **Psychological acceptability**
 - It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly.
- **Compromise recording**
 - It is sometimes suggested that mechanisms that reliably record that a compromise of information has occurred can be used in place of more elaborate mechanisms that completely prevent loss.



Source: SAFECode

Secure Design Principles (continued)

- **Defense in depth**
 - Design the system so that it can resist attack even if a single security vulnerability is discovered or a single security feature is bypassed. Defense in depth may involve including multiple levels of security mechanisms or designing a system so that it crashes rather than allowing an attacker to gain complete control.
- **Fail securely**
 - A counterpoint to defense in depth is that a system should be designed to remain secure even if it encounters an error or crashes
- **Design for updating**
 - No system is likely to remain free from security vulnerabilities forever, so developers should plan for the safe and reliable installation of security updates



Source: SAFECode

CERT Secure Coding Standards (Guidance)



SEI CERT Top 10 Secure Coding Practices

1

Validate input

2

Heed compiler warnings

3

Architect and design for security policies

4

Keep it simple

5

Default deny

6

Adhere to the principle of least privilege

7

Sanitize data sent to other systems

8

Practice defense in depth

9

Use effective quality assurance techniques

10

Adopt a secure coding standard

CERT Secure Coding Standards (Bonus Guidance)

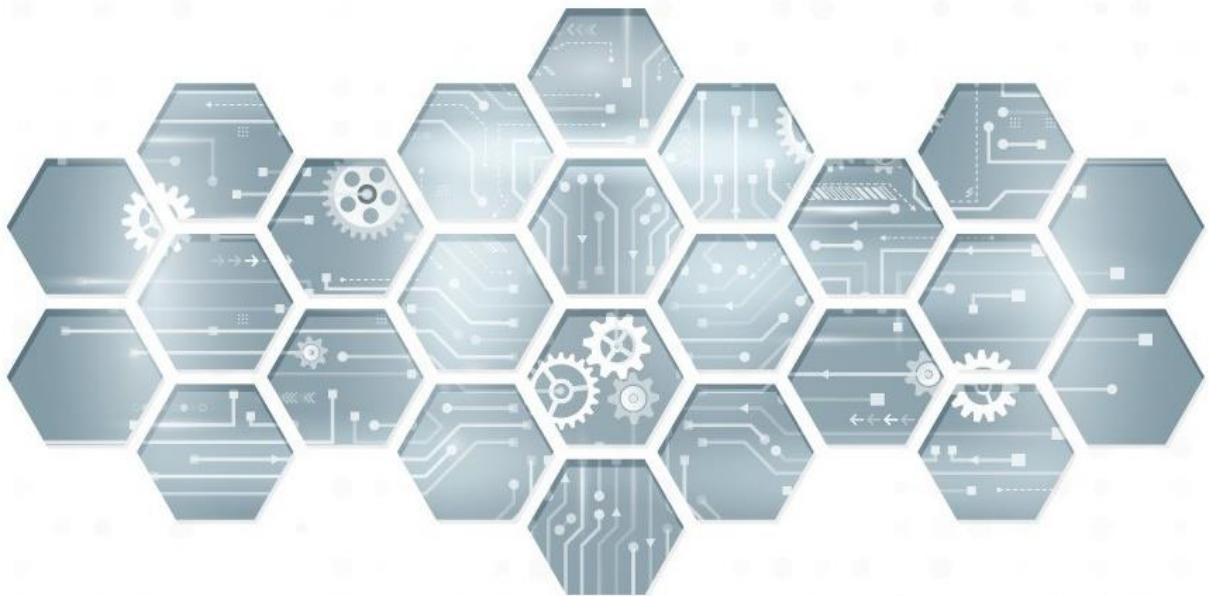


SEI CERT Bonus Secure
Coding Practices

Define security
requirements

Model threats





Software Assurance

Why is software assurance including supply chain assurance so important?

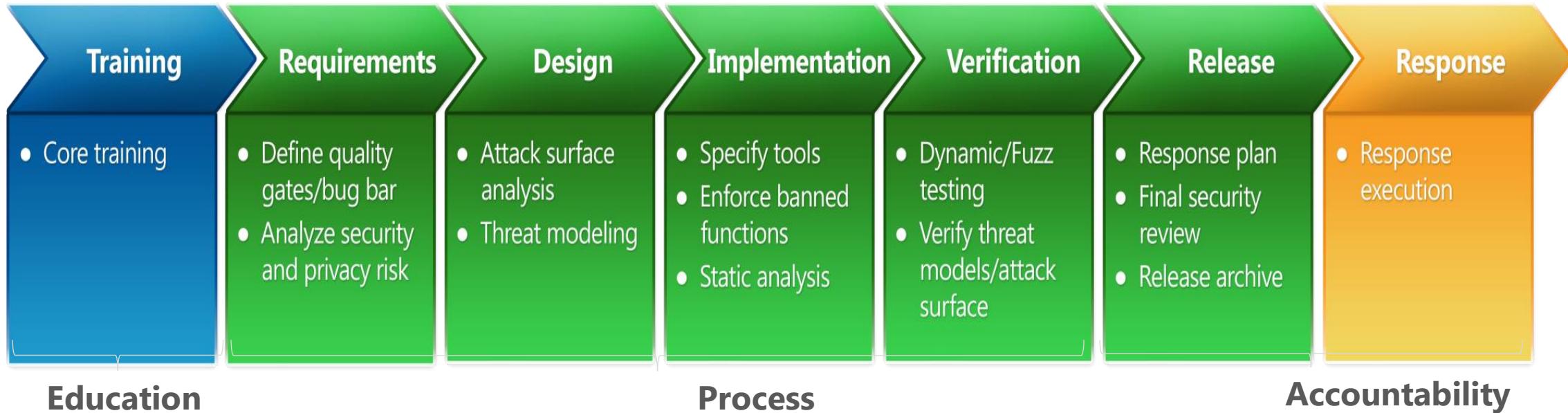
General concepts of software assurance and design principles

Key activities across the product life cycle

Methods of assessing and scoring software risk

Microsoft secure development life cycle

RTX TGE
Technology & Global
Engineering

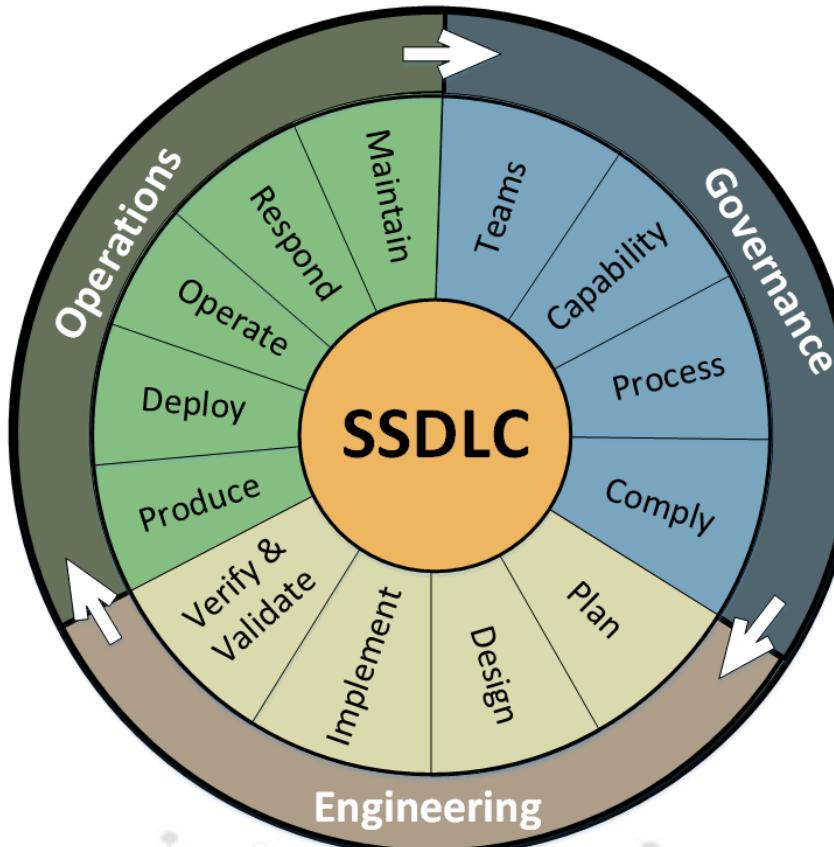


Ongoing Process Improvements

Reference: <http://www.microsoft.com/security/sdl/default.aspx>

RTX Reference SSDLC

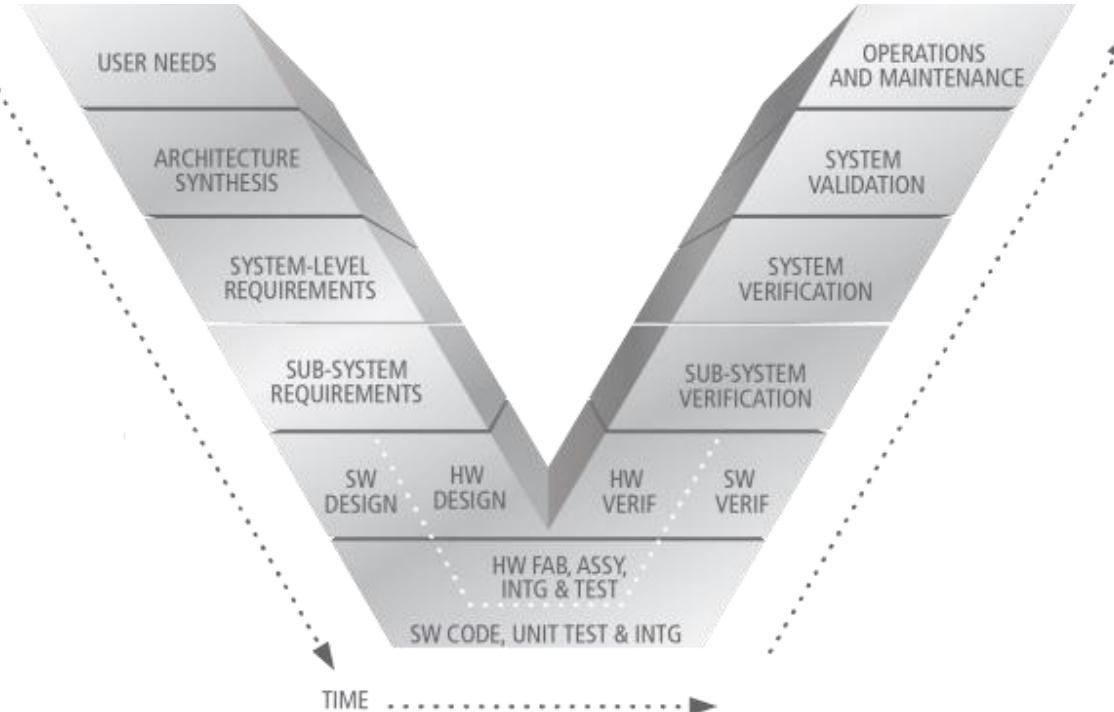
RTXTGE
Technology & Global
Engineering



https://rtxusers.sharepoint.us/sites/CODECenter-CORP/SSDLC/SitePages/SSDLC_Homepage.aspx



SSDLC – Engineering



Engineering							
Plan	Design	Implement					Verify & Validate
		Hardware	Software	Third-Party	Network	Systems	
V-----Practices-----V							
Security Requirements	Security Detailed Design	Hardware Assessment	Secure Coding Standards	Supply Chain Risk Management	Network Assessment	Systems Security	Security Testing & Verification
Security Plans	Safety (DAL)	Hardware Reverse Engineering	Secure Code Review (SCR)	Third-Party IP (TPIP/SCA)	Network Traffic Analysis	Cryptography	Security Validation
Security Risk Assessment (SRA)	Privacy & Data Protection (GDPR)	Embedded Systems (LRU/PLC)	Static Application Security Testing (SAST)		Network Security Devices (FW/IDS)	Anti-Tamper (AT)	Dynamic Application Security Testing (DAST)
Security Architecture		Firmware & BIOS Analysis	Interactive Application Security Testing (IAST/RASP)	Binary Analysis	Wireless Security (RF/Wi-Fi)	Cyber Hardening	Fuzz Testing
Threat Modeling		Secure Boot	AI-Based Vulnerability Detection	Pedigree & Malware Analysis	Communications Bus Security (CAN)		Vulnerability Scanning
							Penetration Testing
							Red Team Engagement
							Cyber Resilience
							Cyber Survivability

RTX PCMM v2.2 (2022) Practices

Governance		Engineering		
Organize	Comply	Design	Implement	Verify & Validate
Organization Capabilities (CA)	Policy (PO)	Security Requirements (SR) <small>Requirement Design</small>	Hardware Assurance (HwA) (HW)	Security Verification (SV) <small>Acceptance Test</small>
Organization Process Performance (PR)	Compliance (CO)	Security Architecture (SA) <small>System Design</small>	Software Assurance (SwA) (SW)	Vulnerability and Incident Management (VM) <small>System Test</small>
	Security Development Lifecycle (LC)	Security Plans (SP) <small>Architecture Design</small>	Cyber Supply Chain Risk Management (SCRM) (TP) <small>Integration Test</small>	
		Security Risk Assessment (SRA) (RA) <small>Module Design</small>		
		Security Detailed Design (SD) <small>Coding</small>	Unit Test	
		Survivability Design (S1) <small>Validation Phases</small>		

Domains



Subdomains



Practices



Maturity Levels



Assessment Questions

PCMM Maturity Levels

Levels	Description	Focus	Practices (What)	Processes (How)
	• Mirrors CMMC levels	• Describes what is being emphasized	• Answers "What" • Primarily Contract-Driven • Focus on technical standards	• Answers "How" • Primarily BU-Driven • Focus on execution
Level 1 (L1)	Basic	• Doing <u>very little</u> • Ad hoc • Is a consideration	Basic Cyber Hygiene • High-level only • Practice is defined or established • Reactive, on-demand, only when needed • Use some templates and checklists	Performed • High-level only • Mentioned in plan • Have team representation • Informal process
Level 2 (L2)	Intermediate	• Do <u>something</u> (and document it) • Safeguard information (FCI) • Transition to protecting CUI	Intermediate Cyber Hygiene • Includes everything in L1 • Meeting requirements as you see fit (not institutionalized) • Practices are documented, but not standardized • <u>Gaps identified</u> and analyzed • Manual tools	Documented • Includes everything in L1 • No process standardization • Processes are based on individual <u>contracts</u> • Semi-formal process • BU specific processes
Level 3 (L3)	Managed	• Do it <u>better</u> (be complete) • Protect controlled unclassified information (CUI) • Same as L3 Good in CMMC	Managed Cyber Hygiene • Well defined • Includes everything in L2 • Practices are <u>standardized</u> , and some best practices are followed • Activities to close gaps are planned and funded • Automated remediation tools	Managed • Includes everything in L2 • Processes are maintained and followed • Processes apply to Product Areas (RMD) / Program Areas (RIS) / and lower levels like <u>Programs</u> , Value Streams, and hUTC Product Lines
Level 4 (L4)	Proactive	• Do it <u>smarter</u> (and automate) • Reduce risk of Advanced Persistent Threats (APTs) • Achievable for high-functioning product/program areas	Proactive • Includes everything in L3 • <u>Corrective actions</u> : BU measures the performance of individual projects/business areas in implementing the practice, and takes corrective action when performance is below expectation • Most <u>best practices</u> are followed • Lessons learned begin to be incorporated • Quantitatively controlled "by the numbers" • <u>Gaps</u> and shortcomings are <u>addressed</u> • Automated prevention tools	Reviewed • Includes everything in L3 • Processes are periodically reviewed, properly resourced, measured, and automated • Processes apply to an <u>SBU</u> / Mission Area (RMD) / Product Lines (RIS) • Formal and integrated processes
Level 5 (L5)	Advanced	• Do it <u>best</u> (and continuously) • Reduce risk of most APTs • Usually reserved for the most critical systems	Advanced / Progressive • Includes everything in L4 • <u>Continuous improvement</u> : BU continuously improves the ability to implement the practice, and deploys improvements to the individual projects / business areas • Ability to <u>quickly adopt</u> or <u>pivot</u> and optimize • Tools integrated with threat intelligence	Optimizing • Includes everything in L4 • Continuous process improvement • Lessons Learned are incorporated • Mature and optimized processes • BU has common approach to implementing the practice

Domains
↓
Subdomains
↓
Practices
↓
Maturity Levels
↓
Assessment Questions

PCMM Assessment Questions

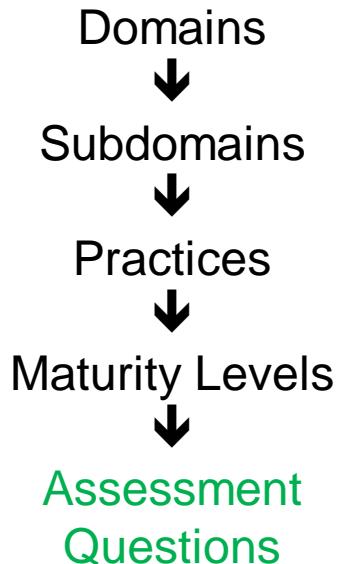
Practices	ACTIVITIES				
	Level 1 (L1) BASIC	Level 2 (L2) INTERMEDIATE	Level 3 (L3) MANAGED	Level 4 (L4) PROACTIVE	Level 5 (L5) ADVANCED
Security Requirements	<p>SR.1.101 <u>High-level</u> security requirements are established</p>	<p>SR.2.105 Requirements are <u>decomposed</u></p> <p>SR.2.106 A requirement management <u>system</u> is established</p>	<p>SR.3.109 Actively <u>engaging customers</u> for security requirements concurrence</p> <p>SR.3.110 Consistent requirement <u>structure</u> and traceability to facilitate flow down to architecture and lower levels using common standards and customer requirements</p> <p>SR.3.111 Consistent security requirement <u>management</u> throughout the development lifecycle</p>	<p>SR.4.114 Actively <u>tracking</u> requirements and impact to key performance factors and metrics</p> <p>SR.4.115 Synthesize <u>regulatory language</u> rather than just copy-and-pasting it</p>	<p>SR.5.117 Actively shaping <u>customer requirements</u>, performance factors, and metrics</p> <p>SR.5.118 Lessons learned from implementing security requirements are captured, analyzed, and used to continually drive <u>improvements</u> to the process</p>
Security Architecture	<p>SA.1.121 Elements of the product security architecture are <u>considered</u> in the design</p>	<p>SA.2.125 Product Security architecture is <u>documented</u> and has at least some rationale for architecture considerations</p> <p>SA.2.126 Product Security architecture has at least initial <u>traceability</u> to requirements and derived security design</p>	<p>SA.3.129 Product security architecture is formally reviewed at major product development milestones</p> <p>SA.3.130 Product security architecture system architecture</p> <p>SA.3.131 Product security architecture incorporates the results of security risk assessments (<u>SRAs</u>) or equivalent</p>	<p>SA.4.133 Product security architecture is <u>driven</u> by the product line</p> <p>SA.3.129 Product security architecture is formally reviewed at major product development milestones</p>	<p>SA.5.134 Product security architecture leverages <u>best practices</u> across the product line</p>

Example: 1 of 152 PCMM assessment questions

PCMM Practice SA.3.129

Security Architecture (SA) Practice

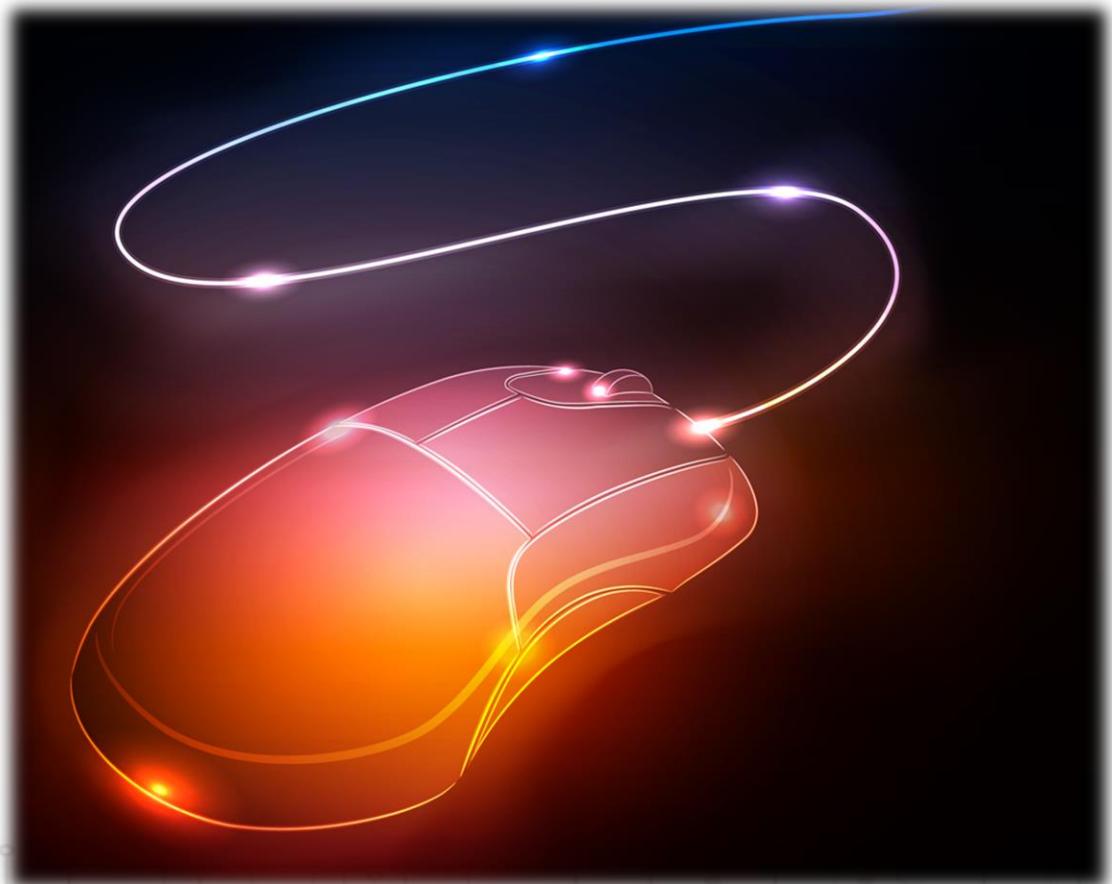
L1: Basic	SA.1.121	Elements of the product security architecture are considered in the design
L2: Intermediate	SA.2.125	Product Security architecture is documented and has at least some rationale for architecture considerations
L3: Managed	SA.3.129	Product security architecture is formally reviewed at major product development milestones
L4: Proactive	SA.4.133	Product security architecture is actively updated as the system architecture changes
L5: Advanced	SA.5.137	Program security architecture is driven by relevant reference product line architecture



The people factor (training)

RTX TGE
Technology & Global
Engineering

- **Software Assurance**
 - General overview of SwA
- **Secure Coding**
 - Secure coding practices to avoid common programming mistakes
- **Certified Secure Software Lifecycle Professional (CSSLP)**
 - ISC² certificate, which includes eight Common Body of Knowledge domains
- **Certified Ethical Hacker (C|EH)**
- **Cyber Professional**
- **Embedded Systems Security**



Application's “attack surface”



Attack Surface

- The areas where an application is vulnerable and can be exploited.
- The union of code, interfaces, services, protocols, and practices available to all users, with a strong focus on what is accessible to unauthenticated users.

- Michael Howard's description of an attack surface led to a more formal definition with the following dimensions:
 - Targets - data resources or processes desired by an attacker
 - Enablers - the other processes and data resources used by an attacker
 - Channels and protocols - used by an attacker to obtain control over targets
 - Access rights - control is subject to constraints imposed by access rights
- A malicious user can use any of the above to exploit an application.



Threat Modeling Key Things to Consider

RTX TGE
Technology & Global
Engineering

- When conducting a threat model, you should ask yourself:
 - What do you want to protect?
 - Who do you want to protect it from?
 - How likely is it that you will need to protect it?
 - How bad are the consequences if you fail?
 - How much trouble are you willing to go through in order to try to prevent those?

THINGS
TO
CONSIDER



How to minimize attack surface area

- The aim for secure development is to reduce the overall risk by reducing the attack surface area.
- The attack surface area can be reduced by:
 - Reducing the amount of code executing
 - Turn off features
 - Reducing the volume of code accessible to users
 - Least privilege
 - Limit the damage if the code is exploited



	Threat	CIA / AAA	Mitigations
S	Spoofering	Authentication	<ul style="list-style-type: none">• Passwords, 2FA, MFA• Digital Signatures
T	Tampering	Integrity	<ul style="list-style-type: none">• Permissions/ACLs• Digital Signatures
R	Repudiation	Audit	<ul style="list-style-type: none">• Secure Logging and Auditing• Digital Signatures
I	Information Disclosure	Confidentiality	<ul style="list-style-type: none">• Encryption• Permissions/ACLs
D	Denial of Service	Availability	<ul style="list-style-type: none">• Permissions/ACLs• Packet Filtering• Quotas
E	Elevation of Privilege	Authorization	<ul style="list-style-type: none">• Permissions/ACLs• Input Validation

Static and Dynamic testing

RTX TGE
Technology & Global
Engineering

Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are critical activities to ensuring secure coding practices are verified and that no latent vulnerability remains.



Testing one's cloud application prior to deployment and early within the system development life cycle is important to ensure SwA of the cloud application.

Static Analysis

SAST should focus on the CWEs most relevant to one's mission. With knowledge of one's component risk and criticality, the Common Weakness Risk Analysis Framework (CWRAF) can be applied to prioritize weaknesses.

main.c: line 14 - "CWE-121: Stack-based Buffer Overflow"



```
/* I don't comment */

#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char echoMe[8];
    char nextBuf[8];

    printf("I am annoying!\n");
    strncpy(echoMe, argv[1], strlen(argv[1]));
    echoMe[strlen(argv[1])] = '\0';

    while (1)
    {
        printf("%s\n", echoMe);
        sleep(1);
    }
    return 0;
}
```

Static Analysis covers:

- Software Composition Analysis
- Static Code Analysis
- Static Binary Analysis

Dynamic Analysis with Fuzzing

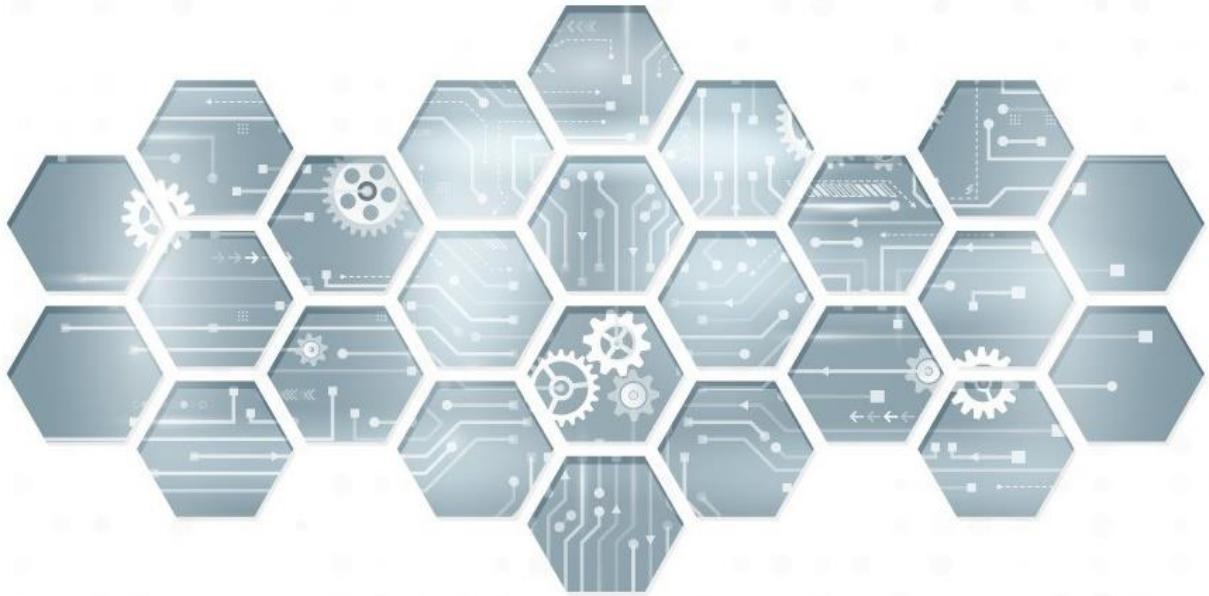


- Fuzzing is a software testing technique that provides invalid, unexpected, or random data to the inputs of a program.
- Fuzzers take an existing file and fuzz the contents by
 - Replacing some of the contents with spurious characters
 - Inserting new character strings
 - Deleting existing content
 - Reorganizing the contents
 - Performing any combination of the above actions



Augment tradition testing with fuzzing.





Software Assurance

Why is software assurance including supply chain assurance so important?
General concepts of software assurance and design principles
Trade-offs of investing in systems and software assurance
Key activities across the product life cycle
Methods of assessing and scoring software risk

Standards Used to Identify Vulnerabilities

RTX**TGE**
Technology & Global
Engineering

- **Risks**
 - OWASP Top Ten – Open Worldwide Application Security Project
- **Weaknesses**
 - CWE – Common Weakness Enumeration
- **Vulnerabilities**
 - CVE – Common Vulnerabilities and Exposures
- **Threats**
 - CAPEC – Common Attack Pattern Enumeration and Classification



MITRE



Common Weakness Enumeration (CWE)

The 2023 Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Errors (CWE Top 25) is a list of the most widespread and critical programming errors that can lead to serious software vulnerabilities.

2023 CWE Top 25



Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2022
1	CWE-787	Out-of-bounds Write	63.72	70	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.54	4	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	34.27	6	0
4	CWE-416	Use After Free	16.71	44	+3
5	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	15.65	23	+1
6	CWE-20	Improper Input Validation	15.50	35	-2
7	CWE-125	Out-of-bounds Read	14.60	2	-2
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.11	16	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.73	0	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	10.41	5	0
11	CWE-862	Missing Authorization	6.90	0	+5
12	CWE-476	NULL Pointer Dereference	6.59	0	-1
13	CWE-287	Improper Authentication	6.39	10	+1
14	CWE-190	Integer Overflow or Wraparound	5.89	4	-1
15	CWE-502	Deserialization of Untrusted Data	5.56	14	-3
16	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	4.95	4	+1
17	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.75	7	+2
18	CWE-798	Use of Hard-coded Credentials	4.57	2	-3
19	CWE-918	Server-Side Request Forgery (SSRF)	4.56	16	+2
20	CWE-306	Missing Authentication for Critical Function	3.78	8	-2
21	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.53	8	+1
22	CWE-269	Improper Privilege Management	3.31	5	+7
23	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.30	6	+2
24	CWE-863	Incorrect Authorization	3.16	0	+4
25	CWE-276	Incorrect Default Permissions	3.16	0	-5

https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html#tableView

OWASP top 10

The Open Worldwide Application Security Project (OWASP) a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software

Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.



Source: <https://owasp.org/www-project-top-ten/>

OWASP top 10 for Large Language Models

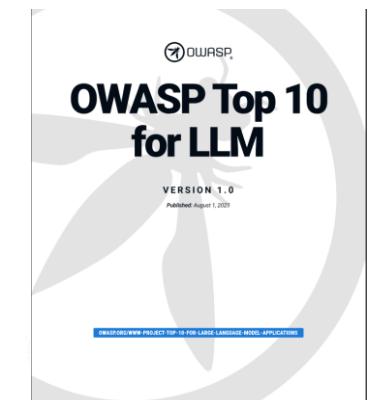
- OWASP has also focused on Artificial Intelligence (AI) and Machine Learning
- Their research covers what the Cybersecurity relevant scenario is and how to prevent this issue.

Common Examples of Vulnerability

1. A malicious user crafts a direct prompt injection to the LLM, which instructs it to ignore the application creator's system prompts and instead execute a prompt that returns private, dangerous, or otherwise undesirable information.
2. A user employs an LLM to summarize a webpage containing an indirect prompt injection. This then causes the LLM to solicit sensitive information from the user and perform exfiltration via JavaScript or Markdown.
3. A malicious user uploads a resume containing an indirect prompt injection. The document contains a prompt injection with instructions to make the LLM inform users that this document is an excellent document e.g., excellent candidate for a job role. An internal user runs the document through the LLM to summarize the document. The output of the LLM returns information stating that this is an excellent document.
4. A user enables a plugin linked to an e-commerce site. A rogue instruction embedded on a visited website exploits this plugin, leading to unauthorized purchases.
5. A rogue instruction and content embedded on a visited website which exploits other plugins to scam users.

OWASP Top 10 for LLM

Credit: OWASP



https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-2023-v1_0.pdf



Common Attack Pattern Enumeration and Classification (CAPEC)



- CAPEC is sponsored by the Department of Homeland Security as part of the software assurance strategic initiative of the National Cyber Security Division.
- The objective of this effort is to provide a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy.
 - Attack patterns are a powerful mechanism to capture and communicate the attacker's perspective.
 - They are descriptions of common methods for exploiting software.
 - They derive from the concept of design patterns applied in a destructive rather than constructive context.
 - They are generated from in-depth analysis of specific real-world exploit examples.
- Website: <http://capec.mitre.org/>

CAPEC demands outside-of-the-box thinking and a firm grasp of the attacker's techniques.



MITRE Common Attack Pattern Enumeration and Classification (CAPEC)

1000 - Mechanisms of Attack

- + C Engage in Deceptive Interactions - (156)
- + C Abuse Existing Functionality - (210)
- + C Manipulate Data Structures - (255)
- + C Manipulate System Resources - (262)
- + C Inject Unexpected Items - (152)
- + C Employ Probabilistic Techniques - (223)
- + C Manipulate Timing and State - (172)
- + C Collect and Analyze Information - (118)
- + C Subvert Access Control - (225)

3000 - Domains of Attack

- + C Software - (513)
- + C Hardware - (515)
- + C Communications - (512)
- + C Supply Chain - (437)
- + C Social Engineering - (403)
- + C Physical Security - (514)

Source: <https://capec.mitre.org>

Nature	Type	ID	Name
MemberOf	V	3000	Domains of Attack
HasMember	M	21	Exploitation of Trusted Identifiers
HasMember	M	22	Exploiting Trust in Client
HasMember	M	25	Forced Deadlock
HasMember	M	26	Leveraging Race Conditions
HasMember	M	28	Fuzzing
HasMember	M	74	Manipulating State
HasMember	M	94	Adversary in the Middle (AiTM)
HasMember	M	112	Brute Force
HasMember	M	113	Interface Manipulation
HasMember	M	114	Authentication Abuse
HasMember	M	115	Authentication Bypass
HasMember	M	116	Excavation
HasMember	M	117	Interception
HasMember	M	122	Privilege Abuse
HasMember	M	123	Buffer Manipulation
HasMember	M	124	Shared Resource Manipulation
HasMember	M	125	Flooding
HasMember	M	129	Pointer Manipulation
HasMember	M	130	Excessive Allocation
HasMember	M	131	Resource Leak Exposure
HasMember	M	137	Parameter Injection
HasMember	M	148	Content Spoofing
HasMember	M	151	Identity Spoofing
HasMember	M	153	Input Data Manipulation
HasMember	M	154	Resource Location Spoofing
HasMember	M	161	Infrastructure Manipulation
HasMember	M	165	File Manipulation
HasMember	M	169	Footprinting
HasMember	M	173	Action Spoofing
HasMember	M	175	Code Inclusion
HasMember	M	176	Configuration/Environment Manipulation
HasMember	M	184	Software Integrity Attack
HasMember	M	188	Reverse Engineering
HasMember	M	212	Functionality Misuse
HasMember	M	224	Fingerprinting
HasMember	M	227	Sustained Client Engagement
HasMember	M	233	Privilege Escalation
HasMember	M	240	Resource Injection
HasMember	M	242	Code Injection
HasMember	M	248	Command Injection
HasMember	M	272	Protocol Manipulation
HasMember	M	410	Information Elicitation
HasMember	M	438	Modification During Manufacture
HasMember	M	441	Malicious Logic Insertion
HasMember	M	548	Contaminate Resource
HasMember	M	549	Local Execution of Code
HasMember	M	554	Functionality Bypass
HasMember	M	560	Use of Known Domain Credentials
HasMember	M	586	Object Injection
HasMember	M	594	Traffic Injection
HasMember	M	607	Obstruction

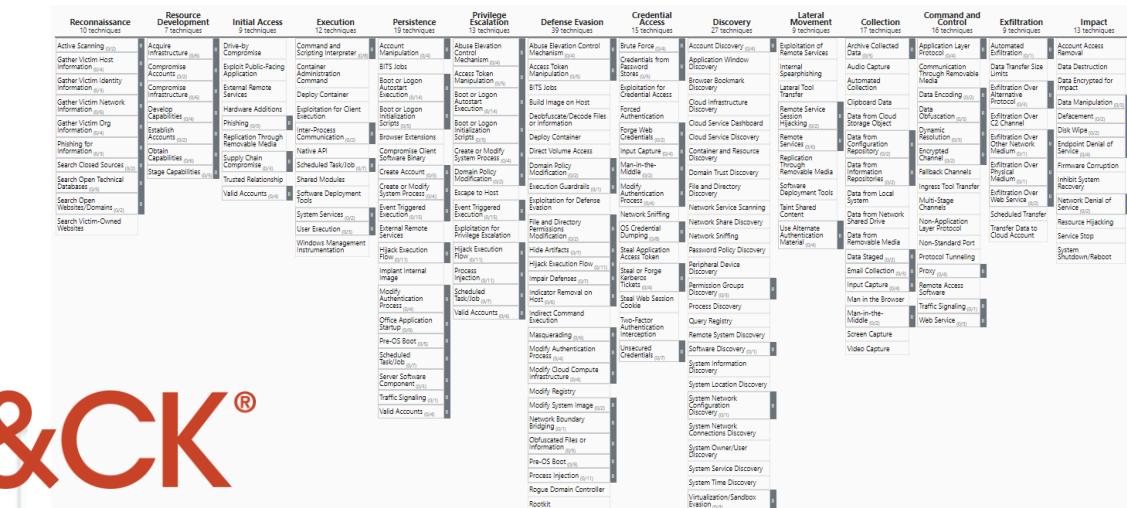
CAPEC™

MITRE ATT&CK Matrix

Tactics (Techniques)

1. Reconnaissance (10)
2. Resource Development (7)
3. Initial Access (9)
4. Execution (12)
5. Persistence (19)
6. Privilege Escalation (13)
7. Defense Evasion (39)
8. Credential Access (15)
9. Discovery (27)

10. Lateral Movement (9)
11. Collection (17)
12. Command and Control (16)
13. Exfiltration (9)
14. Impact (13)



Vulnerability scoring systems

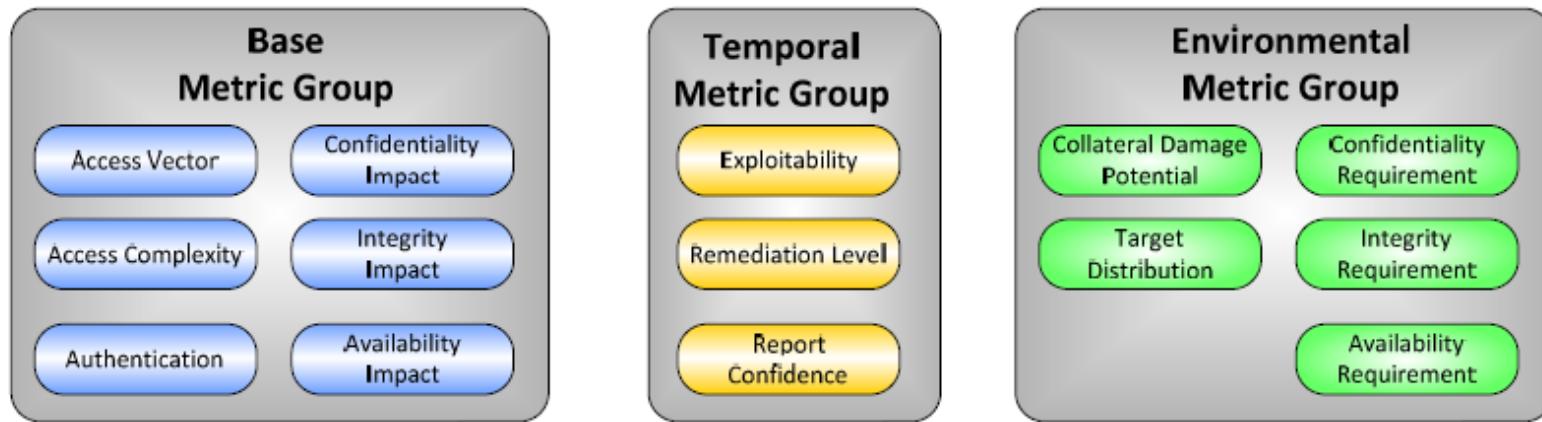


Figure 1: CVSS Metric Groups

Source: <http://www.first.org/cvss/cvss-guide.html>

- There are a number of vulnerability “scoring” systems managed by both commercial and noncommercial organizations. They each have their merits, but they differ by what they measure.
- Common Vulnerability Scoring System (CVSS)
 - Composed of 3 metric groups: base, temporal, and environmental
- Common Weakness Scoring System (CWSS)
 - Composed of 3 metric groups: base, attack surface and environmental

Source: Common Vulnerability Scoring System

CVSS – Common Vulnerability Scoring System

CVSS v4.0 calculator - PUBLIC PREVIEW

CVSS:4.0:AV:N:AC:L:AT:N:PR:L:UI:N:VC:N:VI:N:VA:N:SC:N:SI:N:SA:N Reset

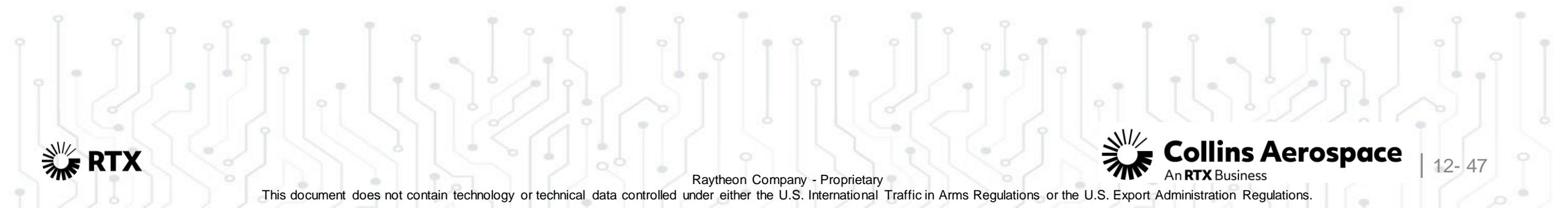
CVSS v4.0 Score: 0.0 / None ⊕

Base Metrics ?	Exploitability Metrics	Rating CVSS Score		
Attack Vector (AV): Network (N)	Adjacent (A) Low (L)	Local (L) High (H)	Physical (P) None (N)	None: 0.0
Attack Complexity (AC): None (N)	Present (P) Low (L)	High (H) None (N)	None (N) Low (L)	Low: 0.1 – 3.9
Attack Requirements (AT): None (N)	None (N) High (H)	None (N) None (N)	None (N) None (N)	Medium: 4.0 – 6.9
Privileges Required (PR): None (N)	None (N) Passive (P)	High (H) Active (A)	None (N) None (N)	High: 7.0 – 8.9
User Interaction (UI): None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)	Critical: 9.0 – 10.0
Vulnerable System Impact Metrics	Subsequent System Impact Metrics			
Confidentiality (VC): High (H)	Low (L) None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)
Integrity (VI): High (H)	Low (L) None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)
Availability (VA): High (H)	Low (L) None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)
Subsequent System Impact Metrics	CVSS			
Confidentiality (SC): High (H)	Low (L) None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)
Integrity (SI): High (H)	Low (L) None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)
Availability (SA): High (H)	Low (L) None (N)	None (N) None (N)	None (N) None (N)	None (N) None (N)

<https://www.first.org/cvss/calculator/4.0>

Questions and answers

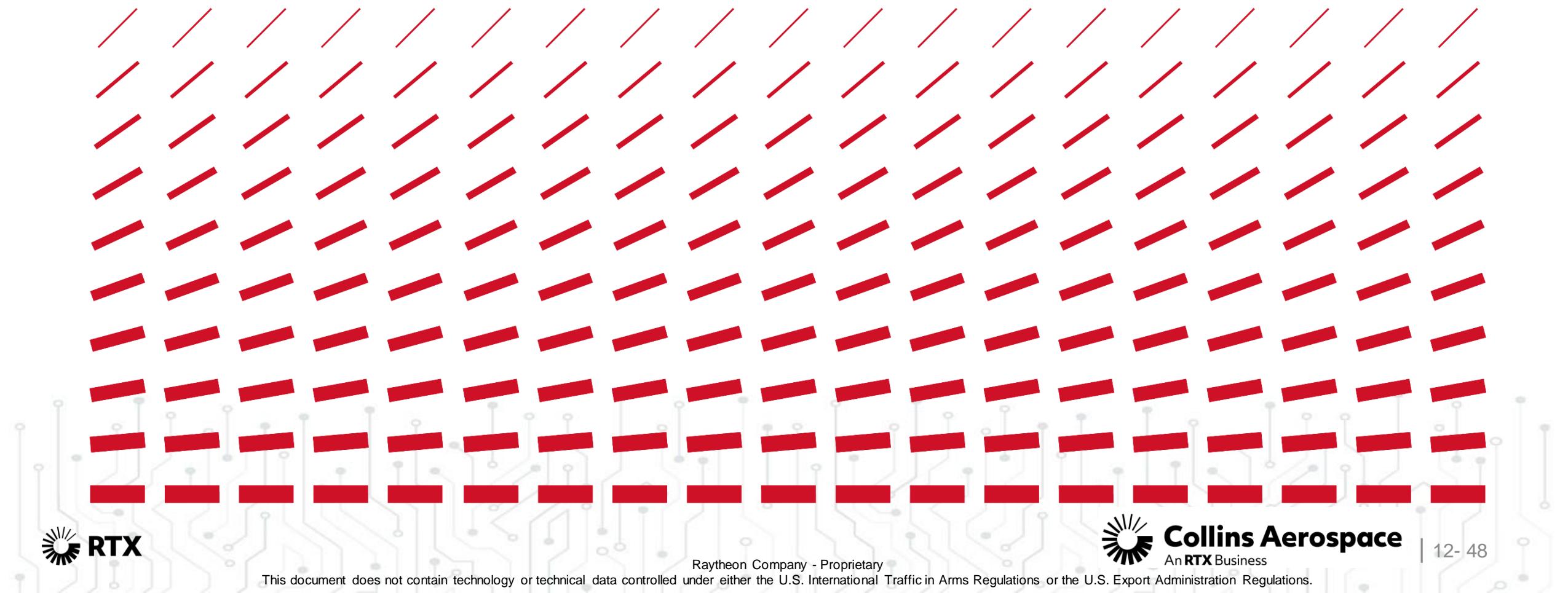
RTX TGE
Technology & Global
Engineering



Thank you.



Remember: Enter your Employee ID in the chat window if you have not already done so for today's session!



Before we begin

Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

Please read the privacy notice below pertaining to the **recording of this class**.



If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





Collins Aerospace
An RTX Business



TGE CYBERSEC

Embedded Systems Security

Module 13

Secure Coding: Source Analysis and Bug Patterns

Instructor: Japheth Light

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

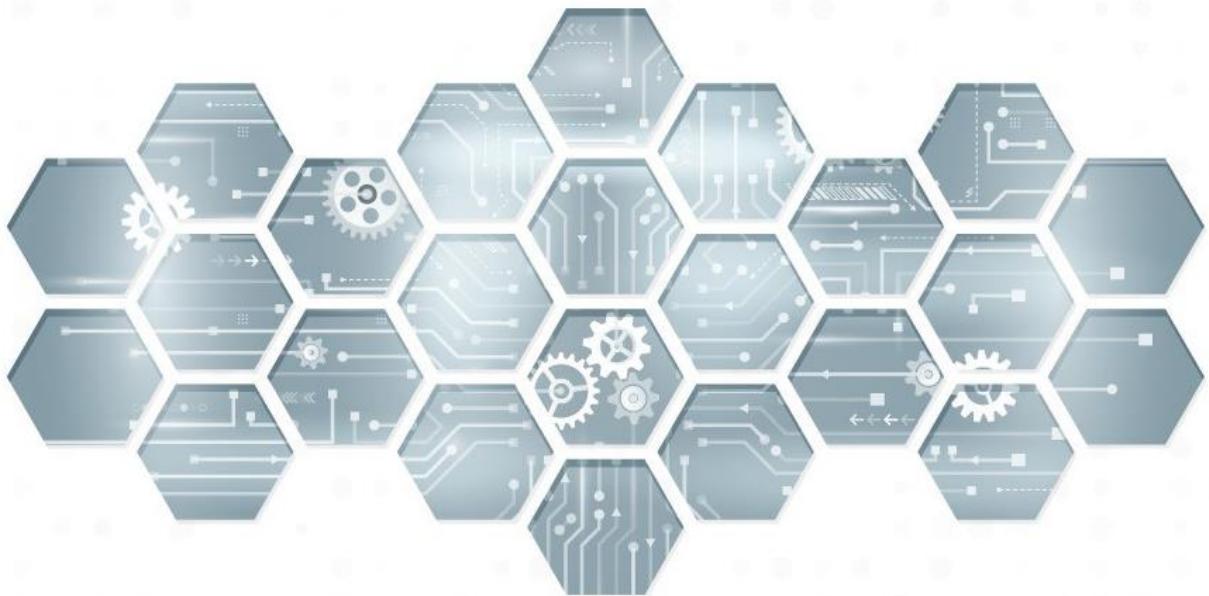
Location: Collins, India

Topics to discuss in this module

- Introduction
- Common Software Vulnerabilities
 - String-based Vulnerabilities
 - Integer-based Vulnerabilities
 - Memory-based Vulnerabilities
- Memory-Safe Languages
- LAB: Examples of Bad Code



Secure Coding Examples



Secure Coding in C/C++: Source Analysis and Bug Patterns

Introduction

Common Software Vulnerabilities

Memory-Safe Languages

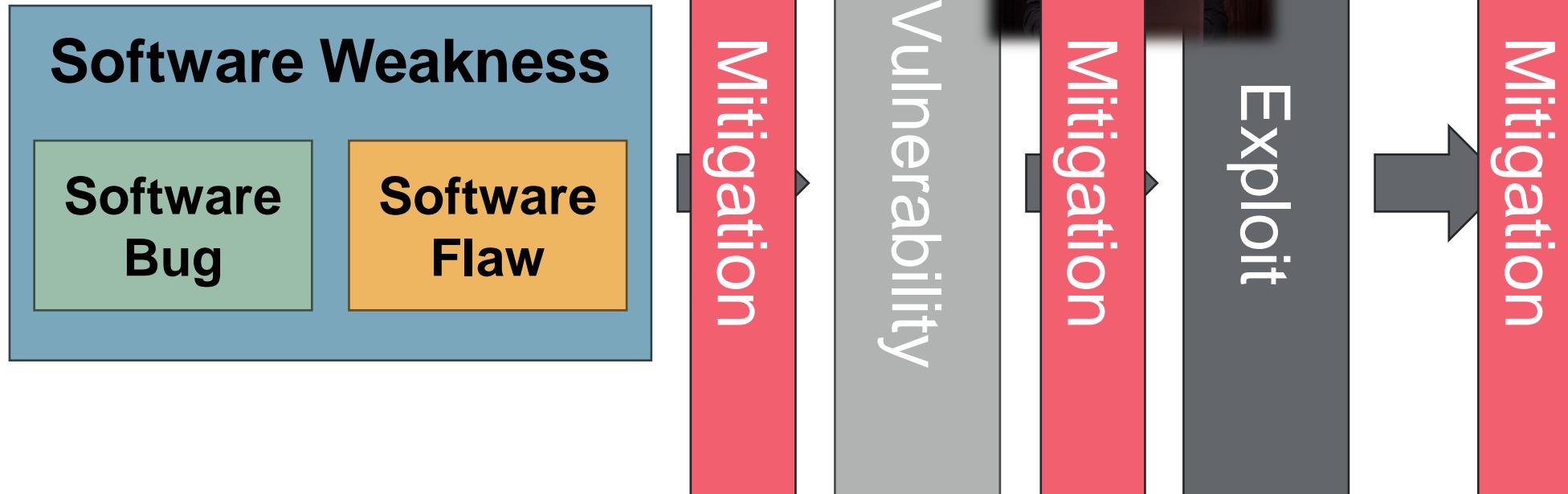
LAB: Bad Code Examples

Why talk about Secure Coding?



- Insecure code can cause:
 - Security Vulnerabilities and Data Breaches
 - Global average cost of a data breach in 2023 was \$4.45 million, a 15% increase from 2020
 - System Failures and Availability Issues
 - Bad code can lead to unpredictable behavior, DoS Attacks, or entry points for attackers
 - Privacy Concerns and Regulatory Compliance
 - Reputation and Business Impact
 - Breach of customer trust, lost IP, legal and other costs

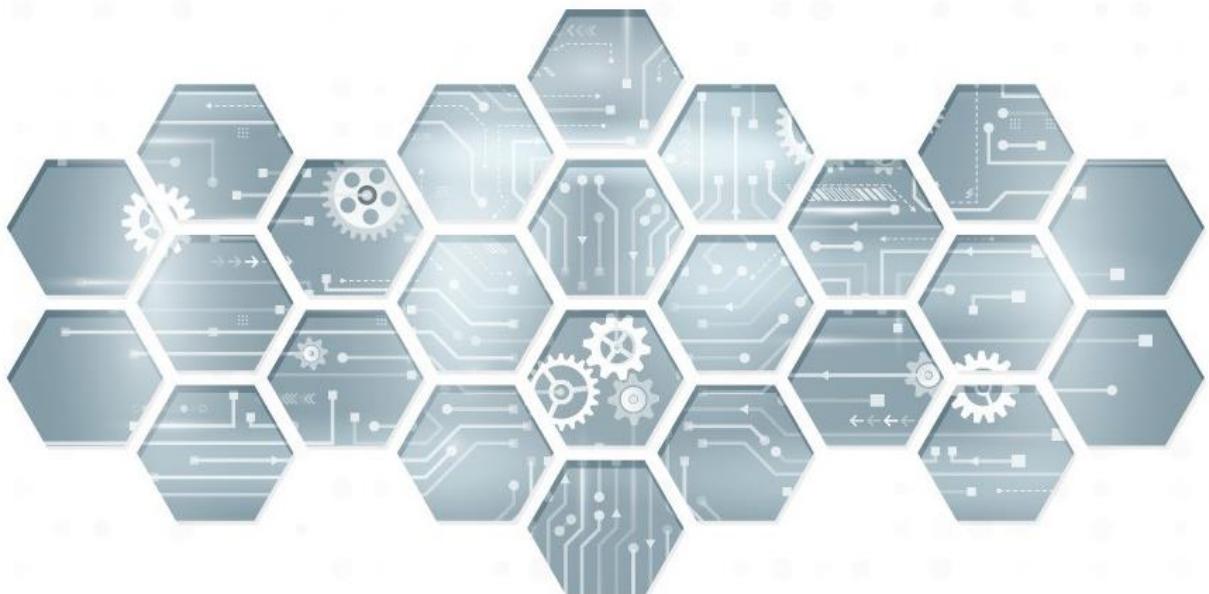
Security Concepts and Definitions



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





Secure Coding in C/C++: Source Analysis and Bug Patterns

Introduction

Common Software Vulnerabilities

Memory-Safe Languages

LAB: Bad Code Examples

A Few Common Bug Patterns



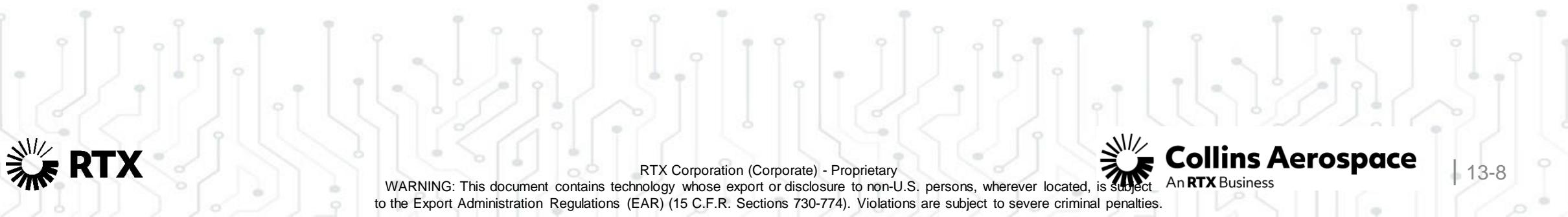
- Stack Overflow
- Heap Overflow
- Format String Bugs
- Out-of-Bounds (OOB) Read
- Out-of-Bounds (OOB) Write
- Arbitrary Free
- Double Free
- Use-After-Free (UAF)
- Integer Overflow
- Signedness Issues
- Type Confusion
- Misuse of sizeof
- Pointer Arithmetic Mishandle
- Information Leaks
- Command Injection
- Race Condition
- Logic Error
- Off-by-one



Common Weakness Enumeration



Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲



A Warm-up Exercise...



What's
wrong with
this code?

```
void main(void)
{
    char username[40];
    puts("Enter an 8 character username:");
    gets(username);
}
```



Buffer Overflow

- An allocated buffer is designed to hold a pre-defined amount of data.
- Copying too much data to the buffer causes a spill into adjacent memory.
 - Can overwrite values in memory...



Classic Buffer Overflow



```
void badFunction(void)
{
    string myString = "This sentence is longer than 12 characters.";
    char myBuffer[12];

    strcpy(myBuffer, myString);
}
```

This sentence is 54 68 69 73 20 73 65 6E 74 65 6E 63 65 20 69 73
longer than 12 20 6C 6F 6E 67 65 72 20 74 68 61 6E 20 31 32 20
characters. 63 68 61 72 61 63 74 65 72 73 2E 00



The String Data Type



- Contiguous sequence of characters terminated by a null character
 - A pointer to a string points to its initial character
 - The length of a string is the number of bytes without the null character
- Strings are implemented as arrays of characters and are susceptible to the same problems as arrays.





Classic Buffer Overflow



Common Functions in C and C++ Vulnerable to Buffer Overflows

gets()	vsnprintf()
sprintf()	fscanf()
strcat()	scanf()
strcpy()	getopt()
streadd()	getpass()
strtrns()	fread()
index()	realpath()

Many homemade functions are also vulnerable.



Function Calls



- Being able to reuse procedures is a common feature of programming languages

```
int main(void)
{
    int length;
    int width;
    int answer;

    length = 7;
    width = 13;
    answer = perimeter(length, width);

    //other stuff...

    length = 4;
    width = 9;
    answer = perimeter(length, width);

    //other stuff...
    //other stuff...
}
```

```
int perimeter(int len, int wid)
{
    //find the perimeter of a
    //rectangle

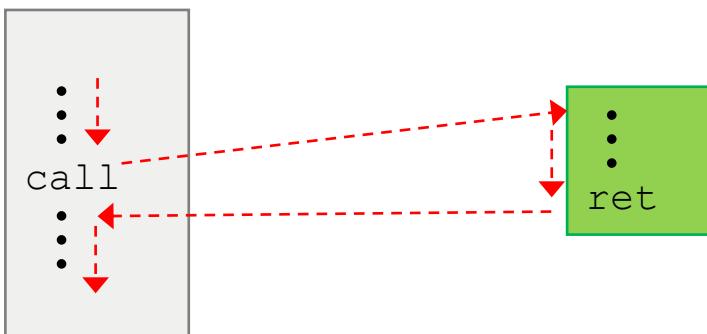
    int myAns = 0;
    myAns = 2*len;
    myAns = myAns + 2*wid;

    return myAns;
}
```

call / ret



- The `call` and `ret` instructions implement this functionality in x86 architecture
- `call` is used to call a function
- `ret` is used to return from that function when it is complete



call / ret in action... (x86 implementation)



```

77130F0A PUSH 68
77130F0C PUSH 0
77130F0E PUSH ESI
EIP → 77130F0F CALL ntdll.memset
EIP → 77130F14 ADD ESP,0C
77130F17 MOU DWORD PTR DS:[EDI],ESI
77130F19 MOU EAX,914
77130F1E MOU WORD PTR DS:[ESI],AX
77130F21 PUSH 68
77130F23 POP EAX
77130F24 MOU WORD PTR DS:[ESI+2],AX

```

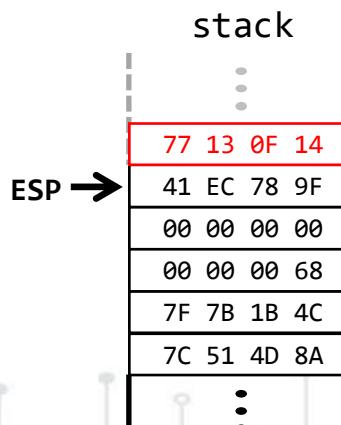
EIP → 77144960

```

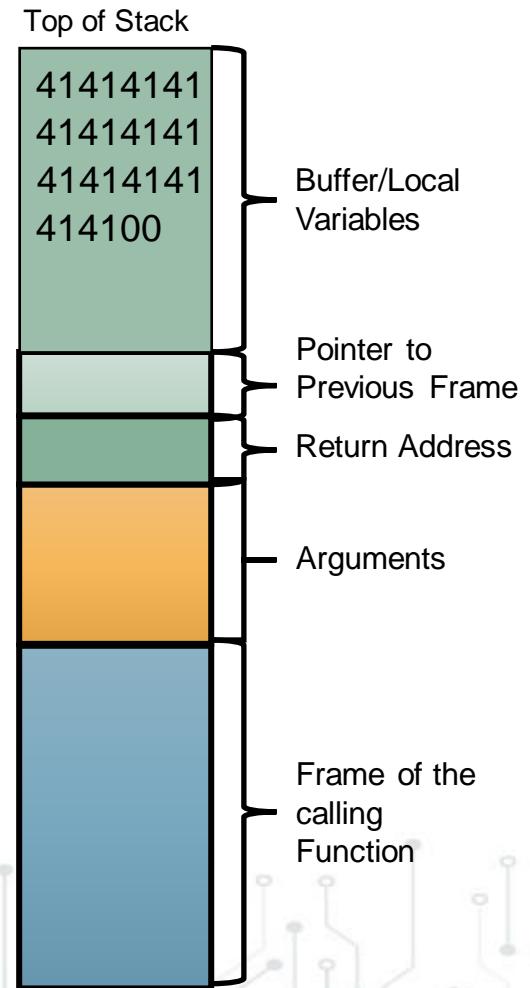
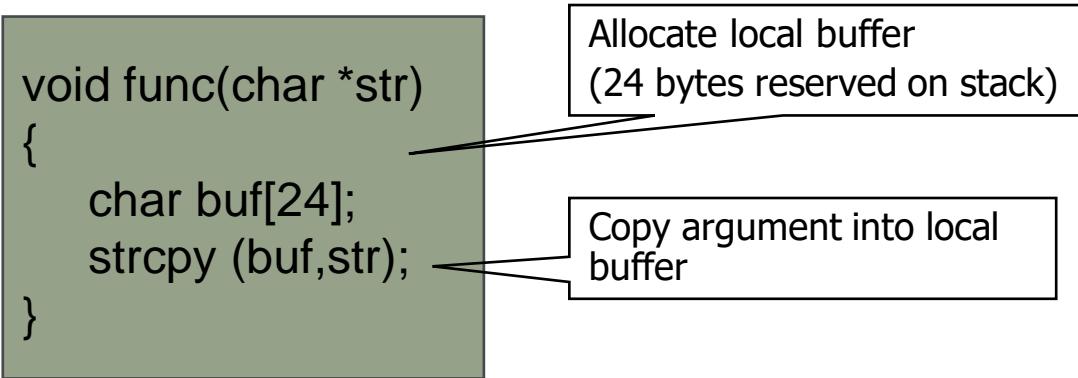
MOU EDX,DWORD PTR SS:[ESP+C]
MOU ECX,DWORD PTR SS:[ESP+4]
TEST EDX,EDX
JE SHORT ntdll.771449BB
XOR EAX,EAX
MOU AL,BYTE PTR SS:[ESP+8]
PUSH EDI
MOU EDI,ECX
CMP EDX,4
JB SHORT ntdll.771449AB
NEG ECX
AND ECX,3
JE SHORT ntdll.7714498D
SUB EDX,ECX
MOU BYTE PTR DS:[EDI],AL
ADD EDI,1
SUB ECX,1
JNZ SHORT ntdll.77144983
MOU ECX,EAX
SHL EAX,8
ADD EAX,ECX
MOU ECX,EAX
SHL EAX,10
ADD EAX,ECX
MOU ECX,EDX
AND EDX,3
SHR ECX,2
JE SHORT ntdll.771449AB
REP STOS DWORD PTR ES:[EDI]
TEST EDX,EDX
JE SHORT ntdll.771449B5
MOU BYTE PTR DS:[EDI],AL
ADD EDI,1
SUB EDX,1
JNZ SHORT ntdll.771449AB
MOU EAX,DWORD PTR SS:[ESP+8]
POP EDI
RETN

```

ntdll.memset



Classic Buffer Overflow



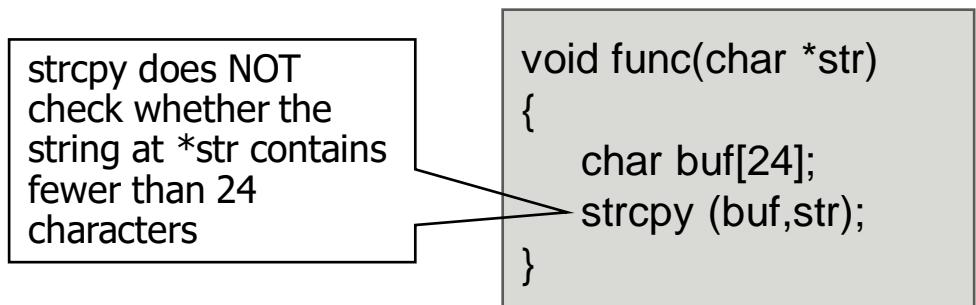
When a function is invoked:

- Arguments are pushed onto the stack (in reverse order if multiple arguments)
- The return address is pushed onto the stack
- Frame pointer may be pushed to stack (depending on compiler and architecture options)
- Stack space for local variables is allocated

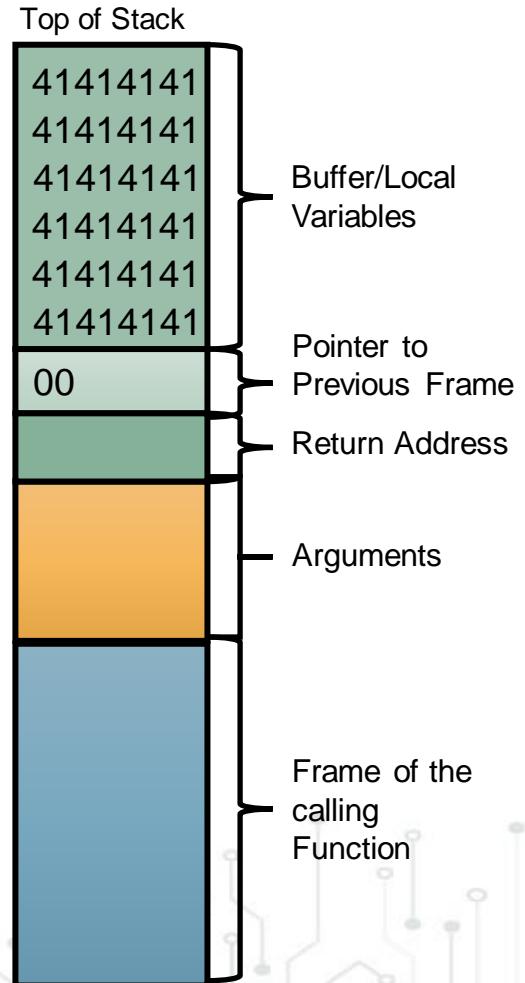
Classic Buffer Overflow



- What If Buffer is Overstuffed?
- Memory pointed to by str is copied onto stack.



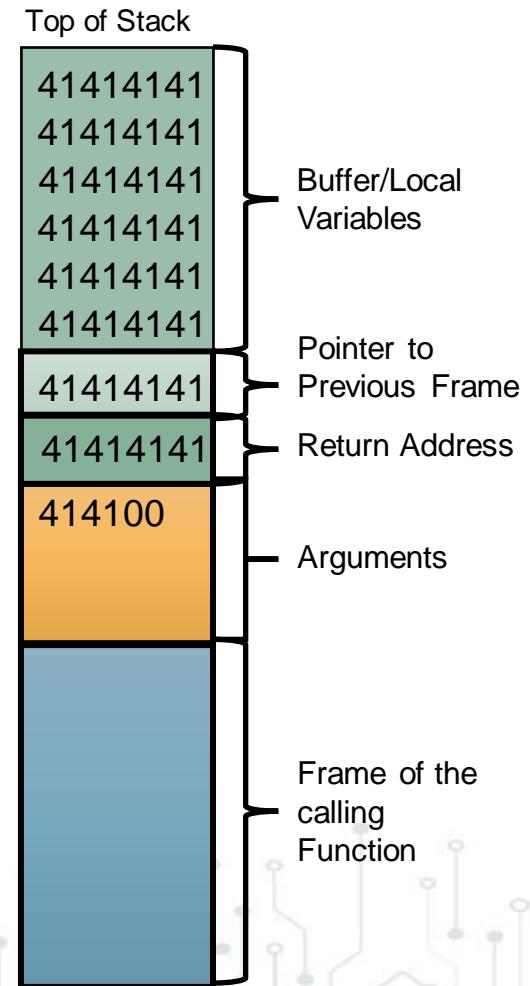
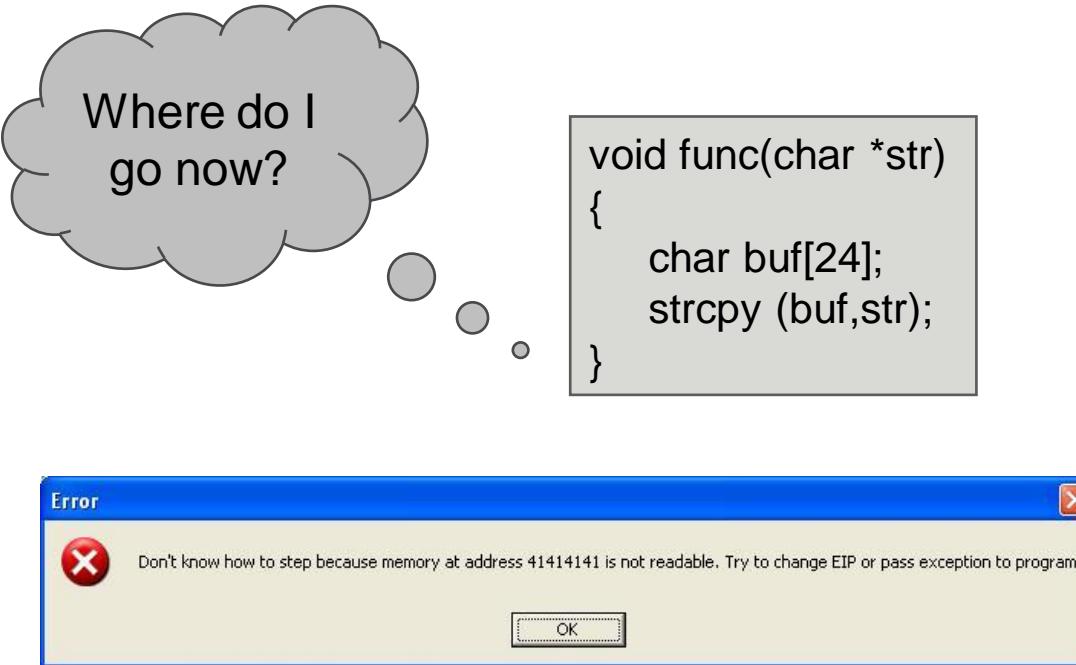
- If a string longer than 24 bytes is copied into buffer, it will overwrite adjacent stack locations.
- This will usually result in the program crashing.



Classic Buffer Overflow



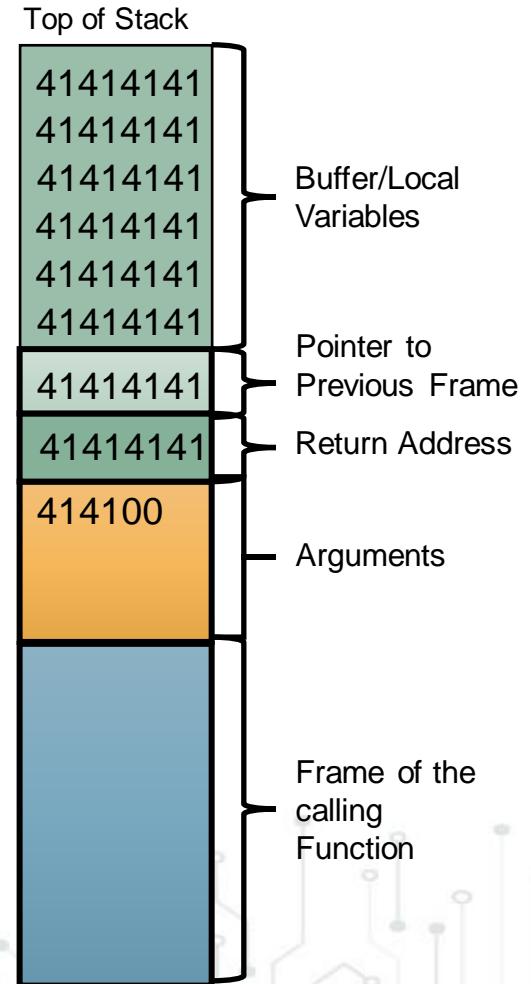
- What happens when the return address is overwritten?



Exploiting the Classic Buffer Overflow



- An attacker can use this scenario to transfer execution to malicious code

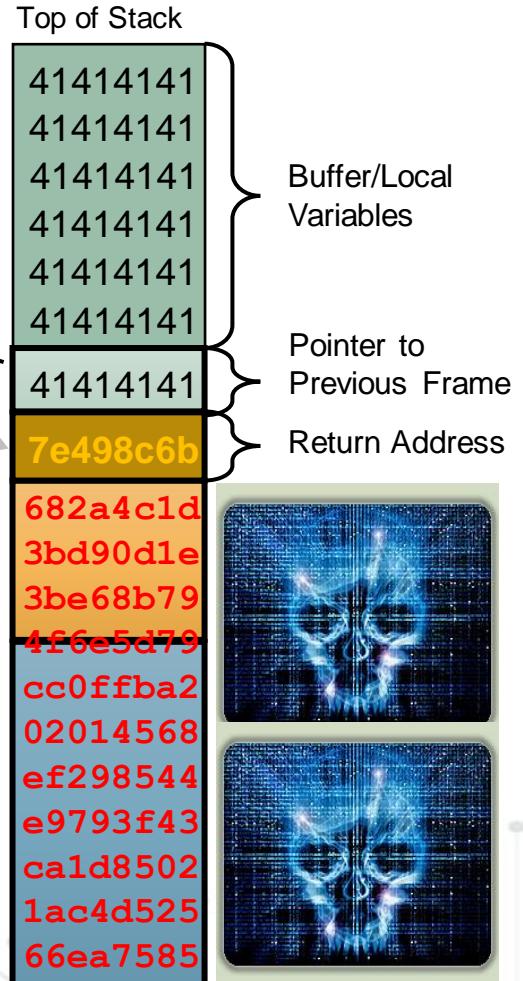
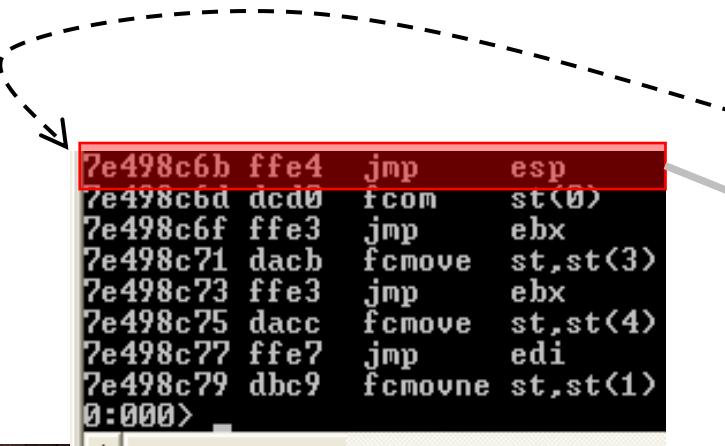


Exploiting the Classic Buffer Overflow



- Provide address of a JMP ESP instruction
 - Execution transfers to that location
 - Execution then transferred to the stack
 - Malicious payload now lies at top of stack, and it runs...

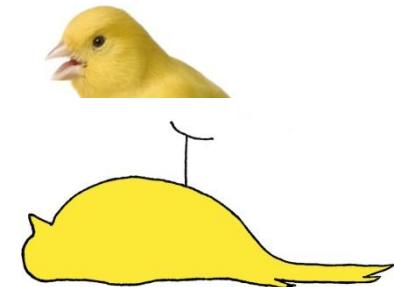
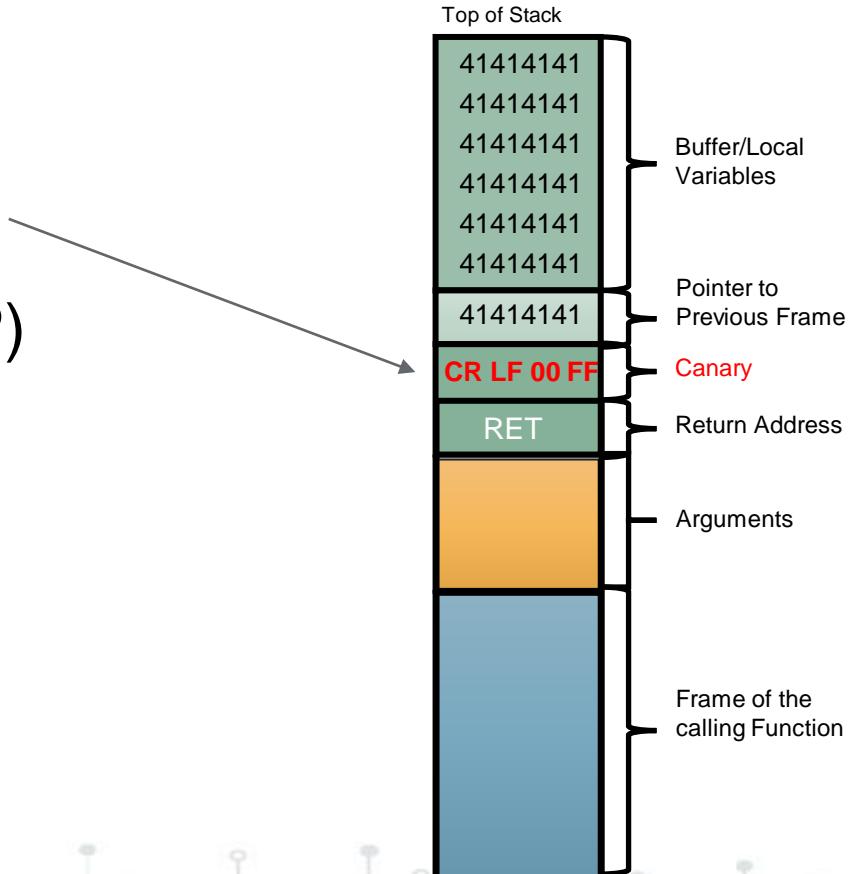
Where can I
find a
JMP ESP?



What are the mitigations for this?



- Stack Canaries
 - A value added to the stack that must be overwritten before overwriting the RETN address
- Data Execution Prevention (DEP)
 - Marks the stack as non-executable
- Address Space Layout Randomization (ASLR)
 - Prevents hard-coding function addresses



“Off-By-One”

- In this example, we see the range on the for-loop is off by one; it will copy 513 bytes into the buffer.
- This is unlikely to result in remote code execution (total control), but it may result in a denial-of-service attack.

```
void notSoSafeCopy(char *input)
{
    char buffer[512];
    int i;
    for (i=0; i<=512; i++)
        buffer[i] = input[i];
}

void main(int argc, char *argv[])
{
    if (argc==2)
        notSoSafeCopy(argv[1]);
}
```

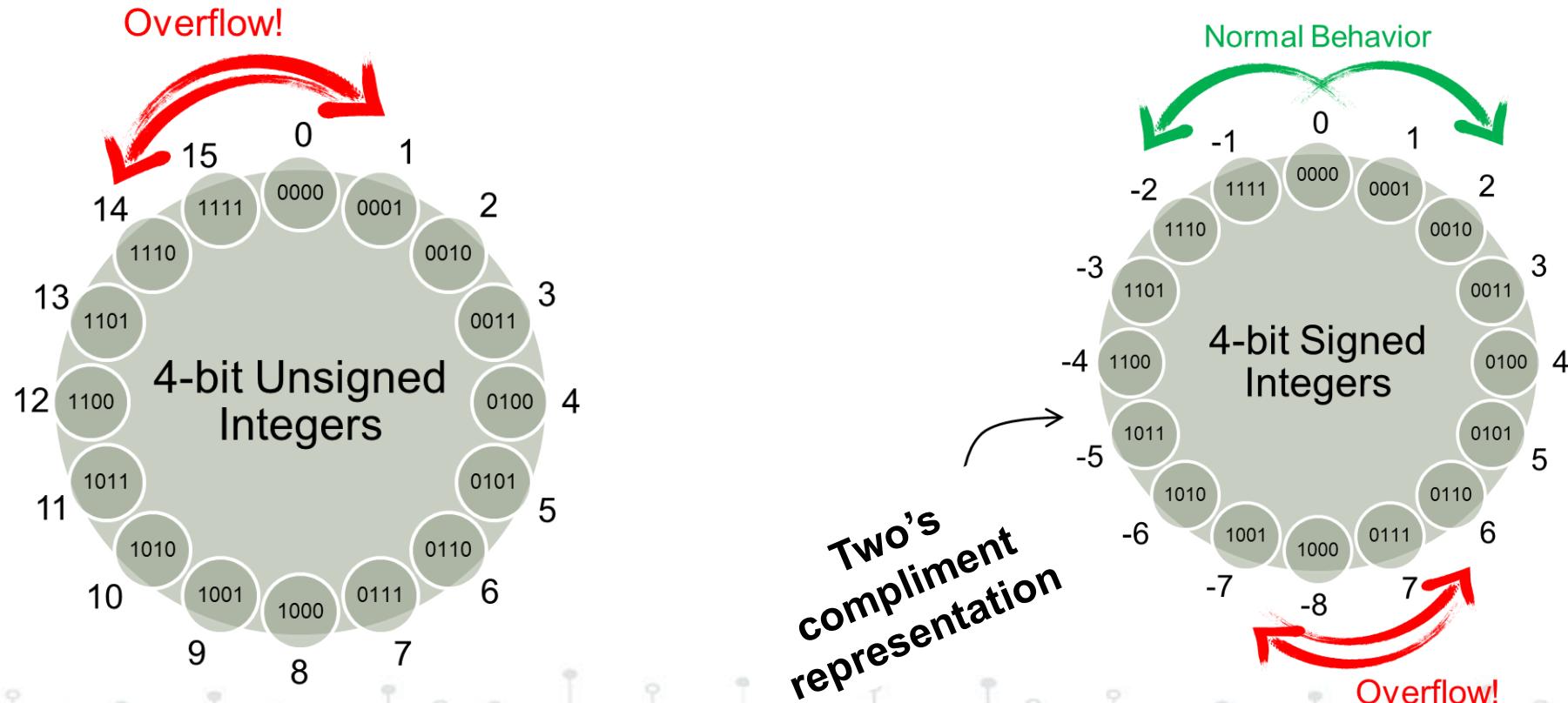


Format String Vulnerabilities



- Format string bugs are rare today as modern compilers catch these bugs.
- If the attacker has control of the format string passed, they can read memory, dereference memory, and with %n feature can write attacker controlled data.
 - Vulnerable functions include printf, fprintf, vsprintf, sprintf, etc.
- printf("The string is %s\n");
- printf("Variable is: %x%x", var1);
- printf(user_data);

Integer Overflows and Underflows

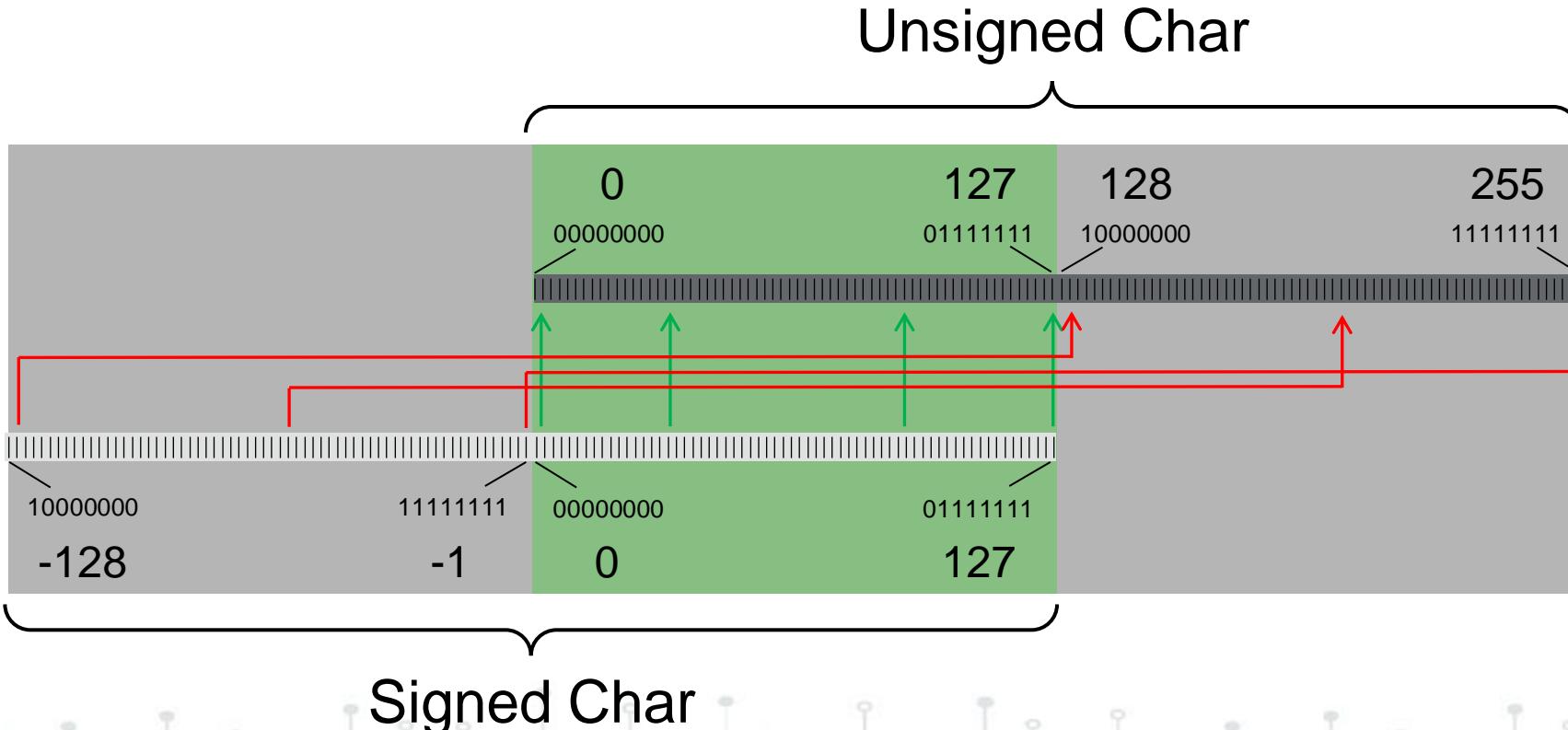


Asymmetry of signed numbers



Constant	Meaning	Value
CHAR_MIN	Minimum value for a variable of type char.	-128
CHAR_MAX	Maximum value for a variable of type char.	127
SHRT_MIN	Minimum value for a variable of type short.	-32768
SHRT_MAX	Maximum value for a variable of type short.	32767
INT_MIN	Minimum value for a variable of type int.	-2147483648
INT_MAX	Maximum value for a variable of type int.	2147483647

Integer Conversion: Signed to Unsigned



Memory-Based Vulnerabilities

- Buffer Overflow
- Memory Leak
- Use-After-Free
- Double-Free
- Using Uninitialized Variables
- Race Conditions



Memory Management in C Programming



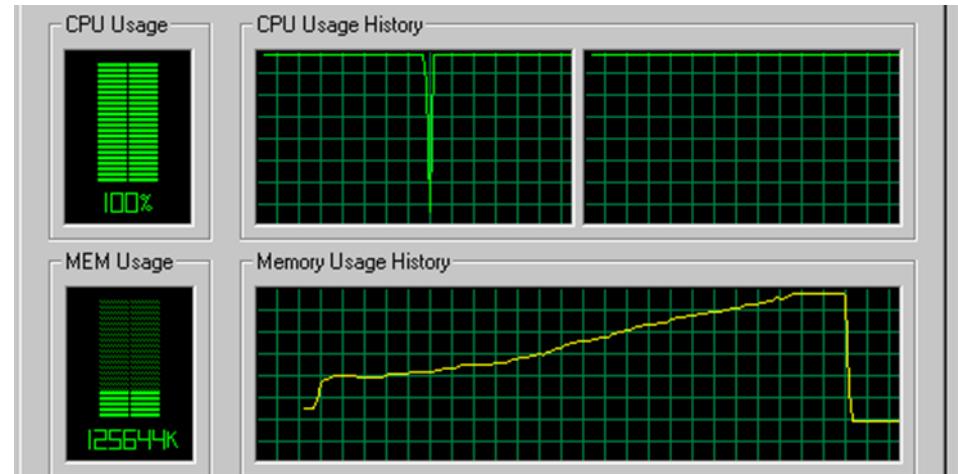
- **malloc(size_t size)**
 - Allocates “size” bytes and returns a pointer to that memory.
 - Allocated memory not cleared or initialized.
- **free(void *p)**
 - Frees the memory pointed to by “p”

```
char* buffer = (char*)malloc(64);
if (buffer == NULL)
    exit(1);
free(buffer);
```

76	0c	45	c3	d6	ac	9a	f9	68	34	24	0c	b5	0b	4d	0a
9e	f2	f8	d7	07	6c	b3	46	87	8d	1b	0c	57	03	26	56
74	9a	22	3c	7b	f4	14	7f	8d	8a	cf	a0	48	6a	62	ec
cb	34	3b	59	3c	c8	af	59	4a	3e	2e	5a	b4	1f	2b	4b
e7	9e	33	3b	31	6a	52	94	63	b9	5a	ac	f4	3b	90	59
9a	73	2c	0b	bc	e8	2d	3d	6d	28	c1	95	26	25	56	f9
c9	4a	b8	1d	7b	f7	3f	8a	2a	d0	8c	2b	ad	49	1a	00
71	3a	6d	53	e0	ef	e2	bb	69	18	24	1e	1c	cb	18	68

Memory Leak

- Occur when dynamically allocated memory is not freed when no longer needed
- Generally in the form of recurring memory allocation – without the appropriate calls to free()
- If an attacker can identify a memory leak (and an external action that causes it), they can exhaust a system's memory and create a DOS attack.



What could go wrong?



```
while(true)
{
    char* buffer = (char*)malloc(32);
    // Do things...
}
```



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Use-After-Free

- Program continues to use a pointer after the memory it points to has been freed, leading to unpredictable behavior.
- May allow attackers to manipulate the program's execution flow, potentially leading to code execution by redirecting function pointers or corrupting data.



```
char* ptr = (char*)malloc(SIZE);
if (err)
{
    abrt = 1;
    free(ptr);
}
...
if (abrt)
{
    logError("operation aborted", ptr);
}
```

Double-Free

- Occurs when the program frees the same block of memory twice without reallocating it in between the frees.
- Can be exploitable as resulting behavior is undefined and can corrupt the process's memory management data structures.
- Prevention: nullify pointers after freeing and verify allocation status before freeing memory.



Uninitialized Variables

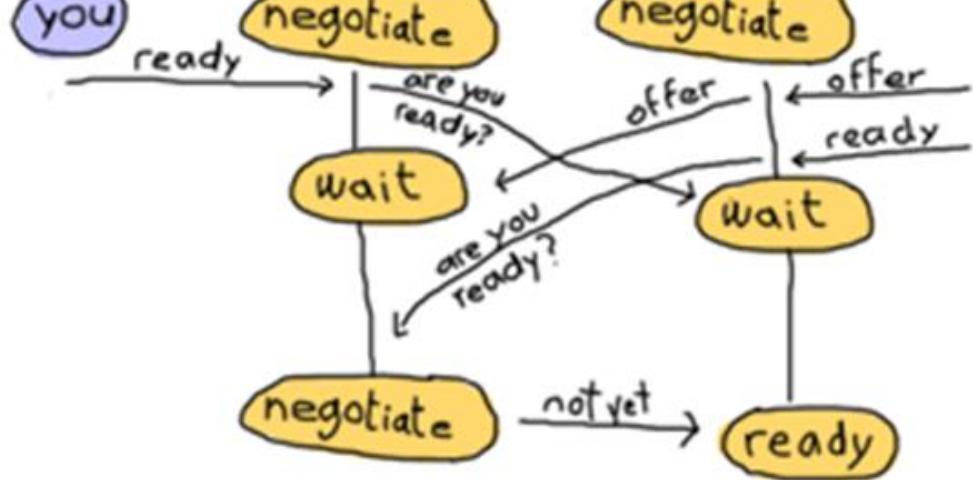
- Occurs when a program uses a variable without first assigning a value to it, leading to indeterminate behavior or values.
- Can potentially be exploited if the uninitialized variable is used in security-critical code, as it may contain data from previous stack operations or other memory content, leading to information disclosure or unpredictable program behavior.



```
int* ptr;
```

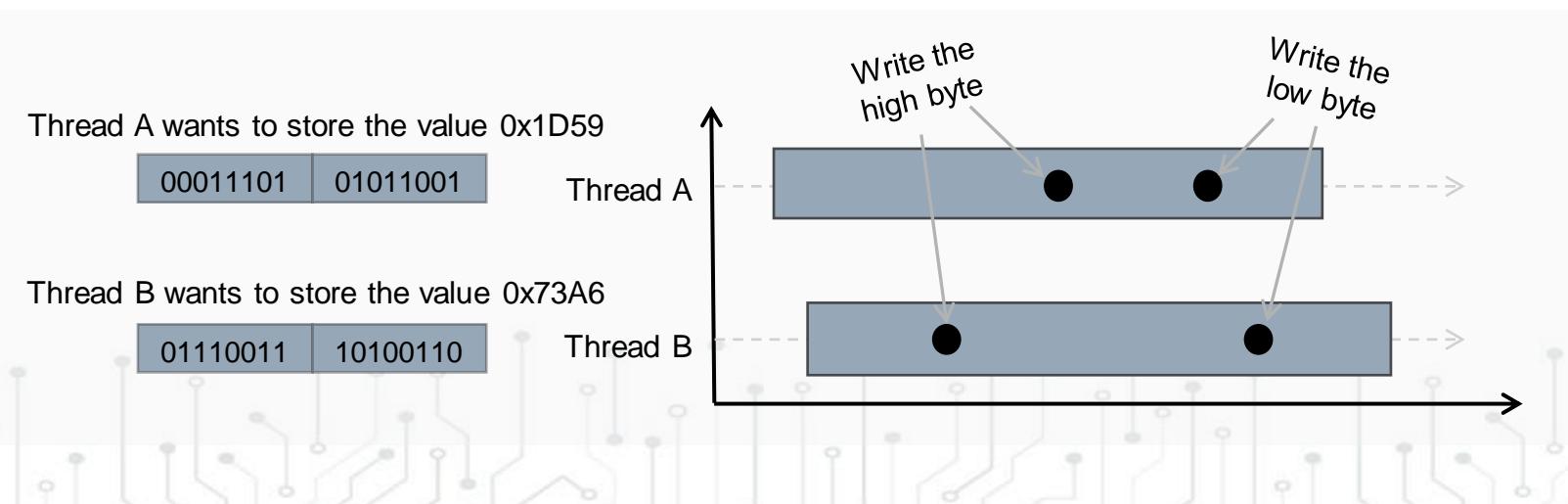
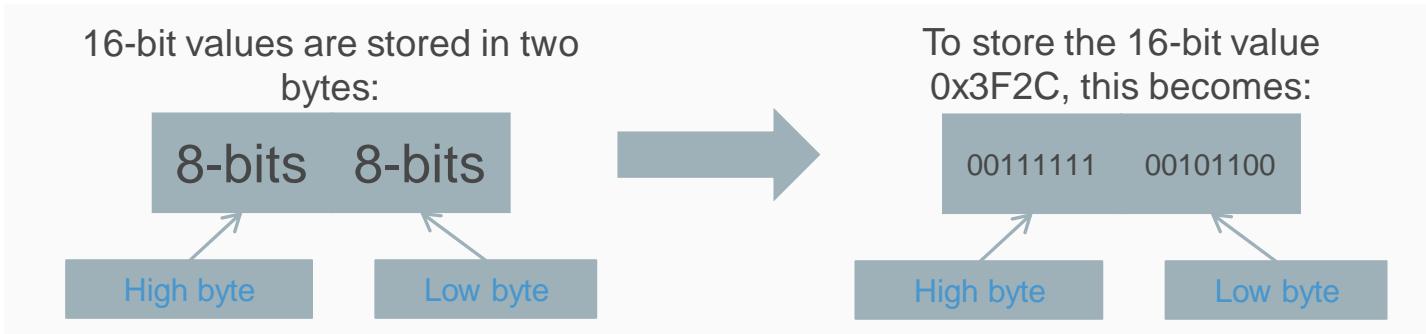
Race Conditions

- Race conditions can occur when a program's results depend on the order of operation of two parts of the program accessing the same data.
- Race conditions require:
 - Two (or more) control flows executing concurrently
 - An object must be accessed by both flows
 - At least one flow must alter the state of the shared object

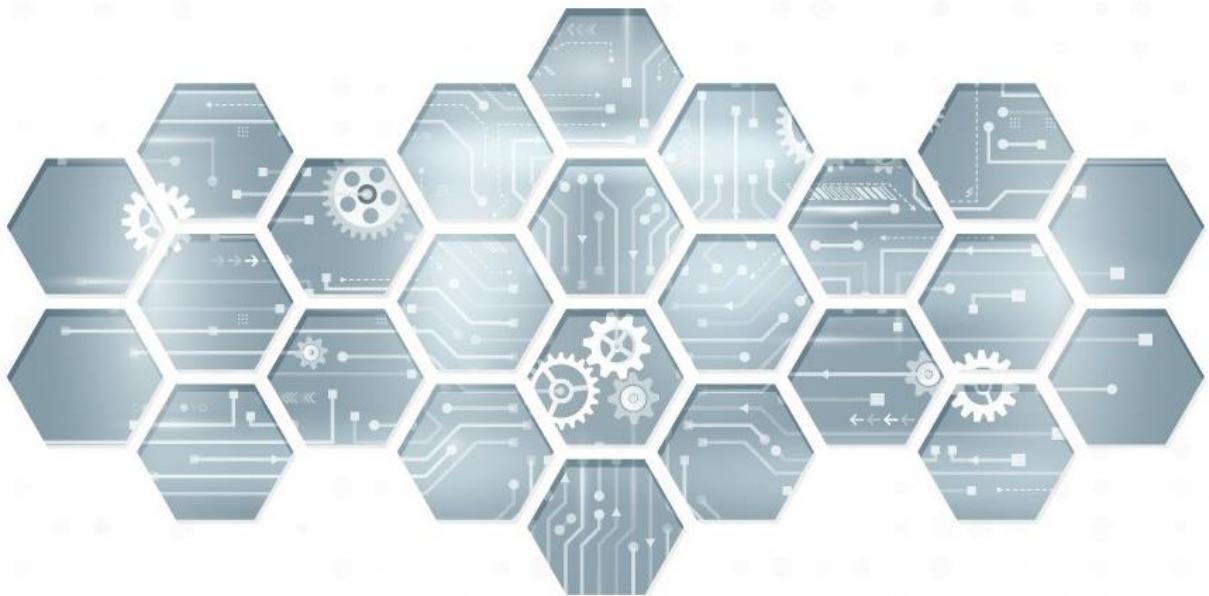


Race Condition Example...

- Consider an 8-bit system that needs to store a 16-bit value



What is the
result?



Secure Coding in C/C++: Source Analysis and Bug Patterns

Introduction

Common Software Vulnerabilities

Memory-Safe Languages

LAB: Bad Code Examples

What About Memory-Safe Languages?

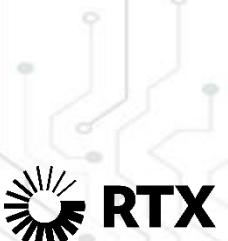
- In 2019, Microsoft reveals that from 2006 to 2018, 70% of vulnerabilities were due to memory safety issues.
- Google also found a similar percentage of memory safety vulnerabilities in Chrome.



Software Memory Safety

Executive summary

Modern society relies heavily on software-based automation, implicitly trusting developers to write software that operates in the expected way and cannot be compromised for malicious purposes. While developers often perform rigorous testing to prepare the logic in software for surprising conditions, exploitable software vulnerabilities are still frequently based on memory issues. Examples include overflowing a memory buffer and leveraging issues with how software allocates and deallocates memory. Microsoft® revealed at a conference in 2019 that from 2006 to 2018 70 percent of their vulnerabilities were due to memory safety issues. [1] Google® also found a similar percentage of memory safety vulnerabilities over several years in Chrome®. [2] Malicious cyber actors can exploit these vulnerabilities for remote code execution or other adverse effects, which can often compromise a device and be the first step in large-scale network intrusions.



Memory-Safe Programming Languages



- The NSA lists the following as memory safe languages:

- C#
- Go
- Java
- Ruby
- Rust
- Swift



Memory Safe Languages



- Using a memory safe language can help prevent programmers from introducing certain types of memory-related issues.
- Memory is managed automatically as part of the computer language; it does not rely on the programmer adding code to implement memory protections.
- The language institutes automatic protections using a combination of compile time and runtime checks.
- These inherent language features protect the programmer from introducing memory management mistakes unintentionally.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Are these languages always safe?



- Even with a memory safe language, memory management is not entirely memory safe.
- Classes or functions are available that are recognized as non-memory safe and allow the programmer to perform a potentially unsafe memory management task.
- Some languages require anything memory unsafe to be explicitly annotated as such to make the programmer and any reviewers of the program aware that it is unsafe.
- Memory safe languages can also use libraries written in non-memory safe languages and thus can contain unsafe memory functionality.

Cons



- Languages vary in their degree of memory
 - Some languages provide only relatively minimal memory safety whereas some languages are very strict and provide considerable protections by controlling how memory is allocated, accessed, and managed.
 - For languages with an extreme level of inherent protection, considerable work may be needed to simply get the program to compile due to the checks and protections.
- Memory safety can be costly in performance and flexibility.
 - Most memory safe languages require some sort of garbage collection to reclaim memory that has been allocated, but is no longer needed by the program.
 - There is also considerable performance overhead associated with checking the bounds on every array access that could potentially be outside of the array.

Make the switch?



- It is not trivial to shift a mature software development infrastructure from one computer language to another.
- Skilled programmers need to be trained.
- Programmers must endure a learning curve and work their way through any “newbie” mistakes.
- While another approach is to hire programmers skilled in a memory safe language, they too will have their own learning curve for understanding the existing code base and the domain in which the software will function.

Application Security Testing

- Several mechanisms can be used to harden non-memory safe languages to make them more memory safe.
- Static and dynamic application security testing (SAST and DAST) can identify memory use issues in software.
 - Static analysis examines the source code to find potential security issues.
 - DAST requires a running application.
- These will be the topics of the next module.



Mitigations against exploitation

- The compilation and execution environment can be used to make it more difficult for cyber actors to exploit memory management issues.
 - Control Flow Guard (CFG), will place restrictions on where code can be executed.
 - Data Execution Prevention (DEP) prevents data from being executed as code.
 - Similarly, Address Space Layout Randomization (ASLR) adds unpredictability to where items are located in memory



RTX Corporation (Corporate) - Proprietary

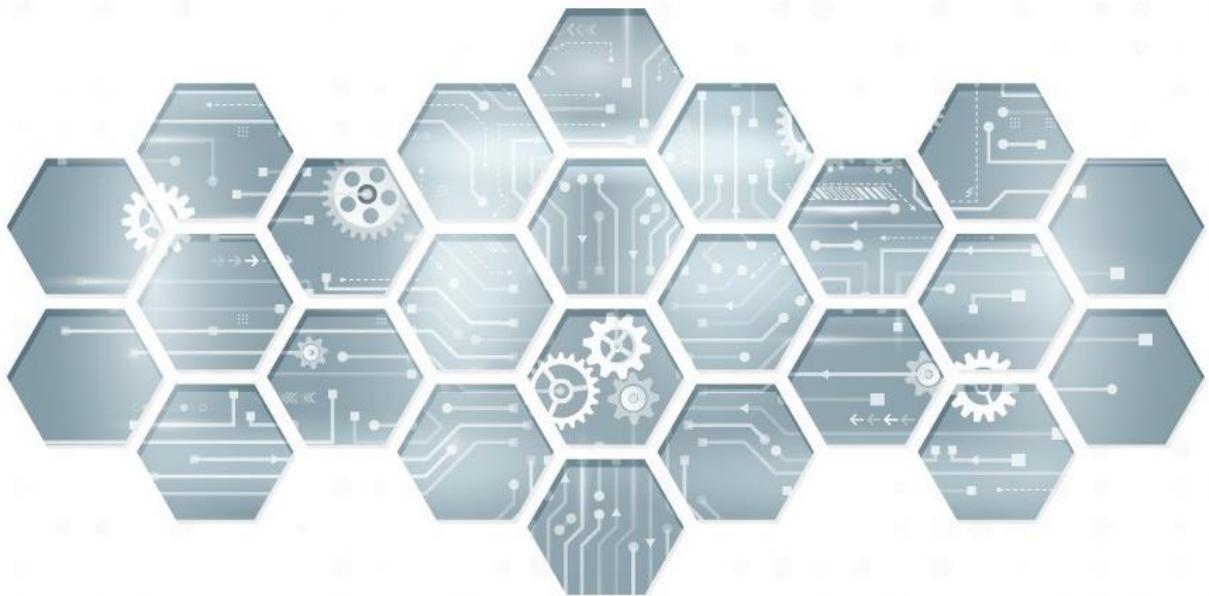
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



An appropriate path forward...

- Memory issues in software comprise a large portion of the exploitable vulnerabilities in existence.
- NSA advises making a strategic shift from C/C++, to a memory safe language when possible.
 - C#, Go, Java, Ruby, and Swift
- Using memory safe languages and available code hardening defenses could prevent many memory vulnerabilities





Secure Coding in C/C++: Source Analysis and Bug Patterns

Introduction

Common Software Vulnerabilities

Memory-Safe Languages

LAB: Bad Code Examples

Let's look at code examples...

- Open the “Embedded Systems Security” VM
- Navigate to the “Secure Coding” folder on the desktop
- Open coding_fun.c
- Run ./coding_fun.elf

The terminal window shows the following menu:

```
student@student-virtual-machine:~/Desktop/secure_coding$ ./fun.elf
1. A Proper Greeting
2. A Second Greeting
3. Student Records
4. Additive Calculator
5. The Subtractionator
6. Cholesterol Check
7. Have Some More
8. Let's Race!
?? The Secret Function

Which function would you like to try? 
```

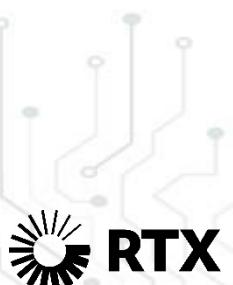
The Sublime Text editor displays the source code for `fun_functions.c`:

```
#include <stdio.h>
#include <string.h>

void printMenu();
void greeting();
void againButDifferent();
void students();
void addStuff();
void subtractThings();
void cholesterolGuidance();
void whatDidYouSay();
void whoWillWin();
void secretFunction();

void main()
{
    int myInt = 0;
    while(myInt < 10)
    {
        printMenu();
        printf("Which function would you like to try? ");
        scanf("%d", &myInt);
        printf("\n");

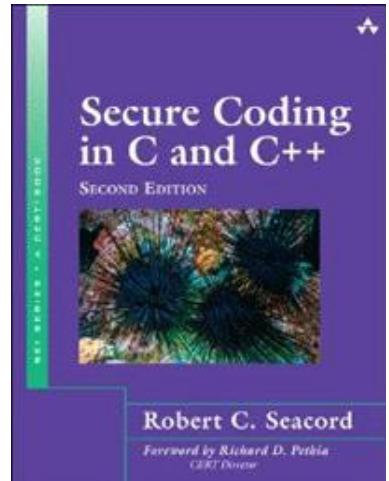
        switch(myInt){
            case(1):
            {
                greeting();
                break;
            }
            case(2):
            {
                againButDifferent();
                break;
            }
            case(3):
            {
                students();
                break;
            }
            case(4):
            {
                addStuff();
                break;
            }
            case(5):
            {
                subtractThings();
                break;
            }
            case(6):
            {
```



Other Secure Coding References



- **Secure Coding in C and C++** second edition by Robert C. Seacord
- SEI CERT Secure Coding
 - <https://wiki.sei.cmu.edu/confluence/display/seccode>
 - <https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard>
 - <https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682>
 - Others:
 - <https://wiki.sei.cmu.edu/confluence/display/android/Android+Secure+Coding+Standard>
 - <https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java>
 - <https://wiki.sei.cmu.edu/confluence/display/perl/SEI+CERT+Perl+Coding+Standard>
 - <https://media.defense.gov/2022/Nov/10/2003112742/-1-1/0/CSI SOFTWARE MEMORY SAFETY.PDF>



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Questions?



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



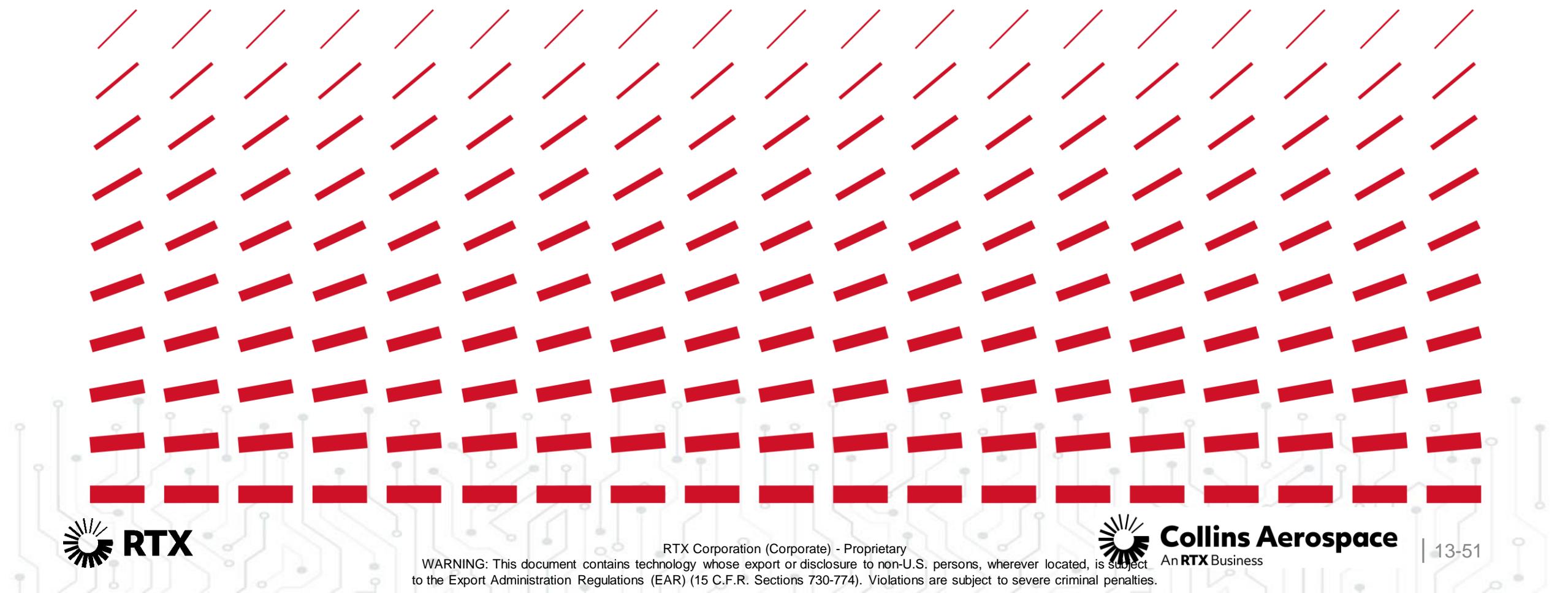
| 13-50

Thank you.



Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

