INSTITUTE FOR DEFENSE ANALYSES

Cybersecurity Operational T&E Concept for Software Factories

William J. Robbins, Project Leader

Peter M. Mancini
Jason P. Sheldon
Erick D. McCroskey
Troy W. Lowry
Kathleen Falcon
Eliza M. Johannes
Brandon A. Shapiro

July 2022

Approved for Public Release.
Distribution Unlimited.

IDA Document D-21595

Log: H 2021-000083/2

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a nonprofit corporation that operates three Federally Funded Research and Development Centers. Its mission is to answer the most challenging U.S. security and science policy questions with objective analysis, leveraging extraordinary scientific, technical, and analytic expertise.

About This Publication
This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-19-D-0001, Task 229992, "Cybersecurity Operational T&E Concept for Software Factories," for the Office of the Director, Operational Test and Evaluation. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments
The IDA Technical Review Committee was chaired by Mr. Robert R. Soule, former Director, Operational Evaluation Division, and consisted of Dr. James C. Thome, Dr. Stacey L. Allison, Dr. Courtney Au-Yeng, Dr. Swati R. Varshney, Dr. Douglas A. Peek, and Dr. Dean Thomas from the Operational Evaluation Division.

For more information:
William J. Robbins, Project Leader
wrobbins@ida.org • (703) 845-2595

V. Bram Lillard, Director, Operational Evaluation Division
vlillard@ida.org • (703) 845-2230

Rigorous Analysis | Trusted Expertise | Service to the Nation

INSTITUTE FOR DEFENSE ANALYSES

IDA Document D-21595

Cybersecurity Operational T&E Concept for Software Factories

William J. Robbins, Project Leader

Peter M. Mancini
Jason P. Sheldon
Erick D. McCroskey
Troy W. Lowry
Kathleen Falcon
Eliza M. Johannes
Brandon A. Shapiro

Executive Summary

This cybersecurity test and evaluation concept provides a structured methodology by which to design a cybersecurity investigation of the software factory (SF) portion of an acquisition system's supply chain, in support of operational testing. DOT&E guidance requires operational test agencies (OTAs) to characterize the operational risks to oversight systems posed by a program's supply chain, and DOD-owned SFs are key components in the software supply chain. The test design provides the nomenclature and rationale required to communicate and negotiate data needs with the test community, and to track data delivery.

The concept includes a test design example using a notional SF that contains features of currently operating DOD SFs. This document describes the process and outlines the steps to create a test design for the notional SF. This design methodology is broadly applicable and tailorable to meet the testing needs of any SF.

System Description

SFs create, develop, and deploy large collections of interdependent software applications for DOD acquisition systems. Most active DOD software factories support Air Force and Space Force programs. For example, the Air Force's Kessel Run SF in Boston supports the F-35 and Air Operations Center oversight programs, and the Space Force's Kobayashi Maru SF in Los Angeles supports the Space Command and Control oversight program.

Most SFs familiar to us use some form of a continuous integration/continuous delivery (CI/CD) development strategy. In short, this is a "build, test, fix, repeat" method of software development that heavily leverages cloud services, employs software "pipelines" for in-house testing, and typically involves deploying software to operational environments prior to independent operational testing. Using the CI/CD process, SF operators regularly pull code from the code repository, push it through the pipeline, perform program-run developmental testing, and distribute the applications to the production environment. Thus, the SF is responsible for configuration changes, software

updates, rollbacks, and fixes based on user input. Faced with this non-traditional process, the operational test community has struggled with the question of how to assess the cybersecurity of the software supply chain.

Ultimately, SFs aim to deliver trusted software, free of cybersecurity compromise, that enables the supported oversight programs to complete their missions. In this concept, we apply our understanding of software development and deployment processes to outline a set of general mission-essential functions for SFs. This list forms the basis for an investigation of how our adversaries might affect the confidentiality, integrity, and availability of oversight system software.

Attack Surface

We present a generic SF based on known SFs and their roles in DOD software supply chains. The notional architecture includes core components of currently-employed DOD SFs. We enumerate the categories of interfaces to the SF that could be starting points for direct cyber aggressions. OTAs using the methodology described here should tailor this list to the specific SF under consideration, based on its physical architecture.

The attack surface is often geographically diverse. An SF may exist as a single physical building, or the "factory" may be spread across a physical headquarters, a remote cloud, at-home and on-premises developers, and other physical locations.

Software development is often done in unclassified environments, although the application's operational environment may span many classification levels. While classified development environments exist, they are not common. When developing an SF test design, OTAs should carefully consider both the physical infrastructure and processes that support information exchange between classification domains.

Test Design

In order to encapsulate the entire operational space (which will then be pared down when planning cyber assessments), this concept advocates a procedure to identify:

- Mission-essential functions of the SF
- Cyber defense organizations, plus materiel and software solutions
- Information exchange interfaces, both physical and logical

The proposed methodology requires identifying mission-critical SF subsystems and the physical and logical connections to the SF's interfaces, from which an adversary may aggress. Testers then systematically catalog the defenses (e.g., operators, defenders, firewalls, intrusion detection software). Figure 1 shows a sample worksheet for documenting the defenses for a single interface. A complete test design will include one such worksheet per SF interface.

What are the defensive capabilities against cyber aggression on the _____ interface?											
				Cyber Defensive Organizations						Non-Cyber	
				Developers	Administrators	SOC	Helpdesk	Service Provider	Higher Echelon	Physical	Policy
	Identify										
	Protect										
Defensive Action	Detect	Authenticated Access	Native Tool								
			Foreign Tool								
		Unauthenticated Access	Native Tool								
			Foreign Tool								
	Respond	Treat									
		Tolerate									
		Terminate									
		Transfer									
	Recover										

Figure 1: Defensive capabilities should be identified during test planning and defensive data and defender responses should be collected during test execution.

This concept defines the space of attacks required for the investigation by mapping all possible combinations of (1) cyber-attack categories (confidentiality, integrity, and availability), (2) system interfaces, and (3) attack postures (role and access) to each mission-essential function. Like a thief entering the back door of a vacant home and stealing both jewelry from the bedroom and electronics from the den, it is possible that multiple forms of attack could be employed by a cyber adversary (the thief) with access to a single SF interface (the back door). A test design should include one worksheet like the example in Figure 1 (see the attached test design) cataloguing the attacks against each SF interface. Figure 2 is a graphical representation of the combination of all such worksheets for the "Develop Software" mission-essential function; note that test cases not required for an adequate assessment are omitted.

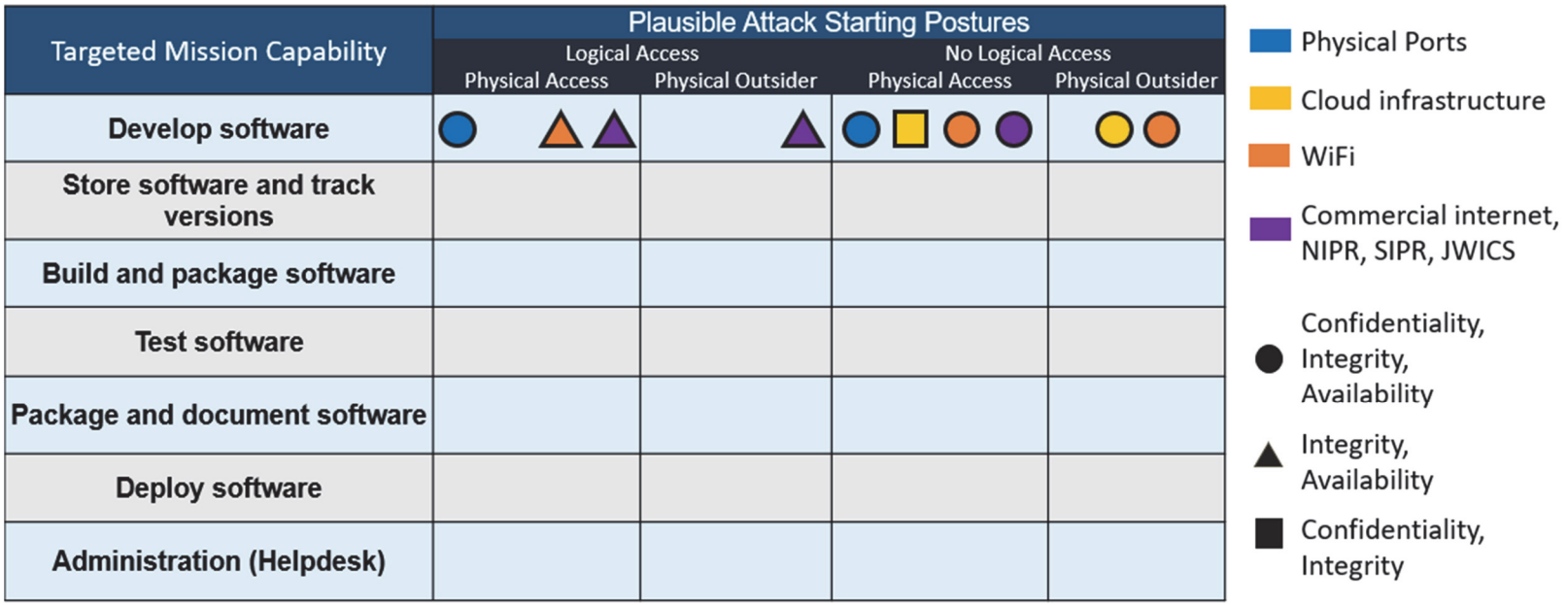


Figure 2: Test planning should identify all possible attack vectors and attack types that exist on the system. The specific capabilities and components targeted during cyber assessments will be a subset of identified potential attack vectors.

Test Planning and Execution

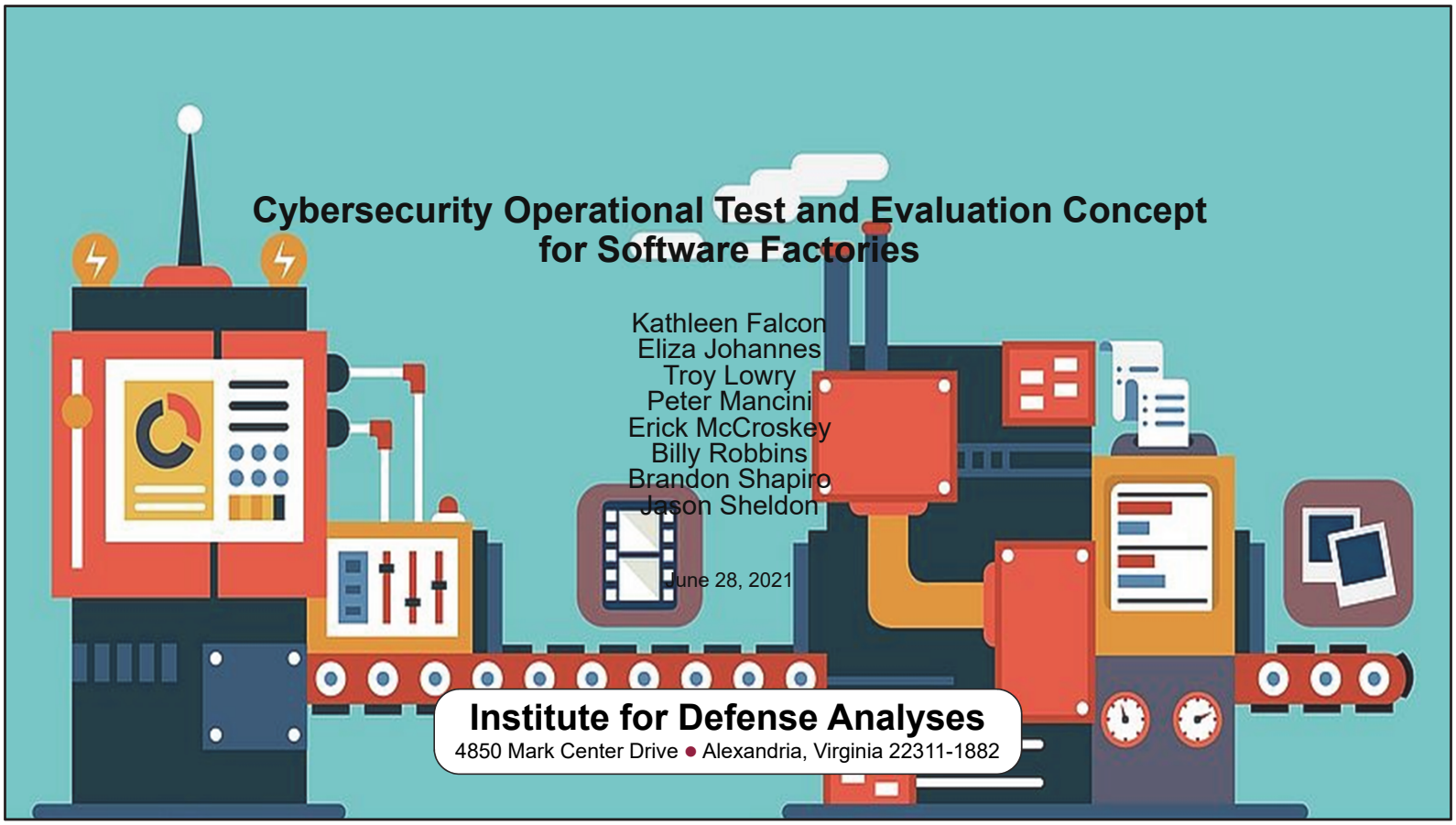
SF assessments pose unique challenges for operational testers, a common one being nonexistent documentation for the SF architecture. In addition, the use of cloud infrastructure and modern security controls require both advanced coordination by the OTA and specific cybersecurity team expertise. We discuss test execution generally, but highlight some SF-specific lessons learned.

We recommend that OTAs conduct cybersecurity assessments of SFs at regular intervals. Because operational SFs are complex and ever-evolving, it is unlikely that an OTA will be able to collect all of the information needed to inform operational testing from a single pair of cooperative and adversarial assessments. Each assessment of the SF is a snapshot in time, relevant to some applications but maybe not others. The testing cadence should account for the cadence of the acquisition program software updates, the duration over which to reasonably assume risk, and funding. If the SF has not changed since it was last assessed, the OTA should be able to leverage previous results, paired with updated threat intelligence to inform operational testing of the oversight program. However, using previous data successfully requires an understanding of all limitations and assumptions involved in the earlier testing.

Conclusion

This SF test and evaluation concept is intended to provide a roadmap for the investigation of a portion of the software supply chain, and includes a worked example on which to build. Our intent is to provide the operational test community with the tools to focus the scope of SF investigations to the areas that matter. We believe that the systematic process described here provides a backstop against the failure of imagination that can occur when test design is delegated to individual testers in the heat of a cyber assessment, or when it is conducted using simple document reviews. Finally, we document some challenges identified in prior SF investigations and provide suggestions for future assessments.

Contents



Cybersecurity Operational Test and Evaluation Concept for Software Factories

Kathleen Falcon
Eliza Johannes
Troy Lowry
Peter Mancini
Erick McCroskey
Billy Robbins
Brandon Shapiro
Jason Sheldon

June 28, 2021

Institute for Defense Analyses
4850 Mark Center Drive ● Alexandria, Virginia 22311-1882

This cybersecurity OT&E concept proposes a method for assessing the software factory component of a system's supply chain

Major OT&E considerations for test adequacy

- A program's cybersecurity posture inherently depends on its supply chain
- Scope of software factory assessments should account for all interfaces and relevant data sources
- OT conclusions should be relevant for ever-changing software factory architecture and processes
- The required resources could be substantial

This OT&E concept should

- Promote dialogue on
  - The need to assess software factory supply chain risk to inform the cybersecurity OT&E of oversight programs
  - Methods to accomplish the assessment
- Provide testers with a common nomenclature and tools for cybersecurity OT&E design
- Guide the development of test strategies and allocation of test resources

IDA | 2

This brief provides an overview of software factories, motivates their incorporation into operational evaluations, and discusses some of the challenges in doing so. We provide a framework by which to design operational cybersecurity tests for software factories and to facilitate discussion with the operational test community. Based on recent experience with planning and executing software factory operational assessments, we highlight some challenges to testing and offer some suggestions.

The resulting test design from this process is included as an attachment to this brief. This test design can be used to facilitate conversations with the OTA about the scope of the SF assessments and tools and resources needed for the investigation. Because the process is systematic, the tools and resources outlined can be connected to the specific data elements required from the investigation. Defining data elements and the instrumentation with which to collect the data allows for an upfront negotiation of what constitutes an adequate investigation and facilitates tracking of what data has and has not been provided.

Presentation Roadmap

Introduction to Software Factories

Establishing a conceptual software factory

Developing a cyber test design for software factories

Executing an operational cyber test for software factories

Incorporating software factory conclusions in oversight program OT&E

Software factories are a critical component of the software supply chain of the oversight systems they support. Using the common and bespoke features of the software factories familiar to us, we present a conceptual example of a software factory. We then design a cybersecurity test of that software factory using standard operational test events and terminology. This brief presents one perspective on what constitutes an adequate test of a software factory. The purpose of this brief is to provide a structured methodology by which to design a cybersecurity investigation of the software factory portion of an acquisition system's supply chain in support of operational testing. This brief should also help facilitate conversations between stakeholders during test planning and execution, and to provide data collectors and evaluators a means by which to determine what data has and has not been collected.

What is a software factory?

A software factory is the infrastructure and personnel used to develop, field, and sustain software for some DoD programs

Software factories (SF) have critical missions

Develop Software
- Build Software
- Document Software
- Store Software/Documentation and Track Versions
- Test Software

Deploy Software
- Package Software/Documentation
- Distribute Software/Documentation

Internal SF Administration
- Personnel Management
- Supplies/Procurement
- Facilities Management
- Policy Development and Enforcement

Manage configuration of fielded software and environments
- Make configuration changes
- Monitor/track configurations

Provide User Help Desk Capability
- User trouble ticket management
- Technical support to system admins
- Coordination with third-party developers/vendors

Cyber defense for fielded software and environments

IDA | 5

Software factories (SF) are the physical and logical infrastructure by which large collections of interdependent software applications are developed for DoD acquisition systems. These DoD applications are often web-based (e.g., https://warpcore.spaceforce.mil/) and hosted on a local or remote server. SFs may also be involved in the deployment and monitoring of these applications.

Each SF differs in its physical architecture and internal development processes, but generally they all share common "missions." (We use the term missions to align with the nomenclature used in our test design process.) Similar to acquisition systems, if a SF fails to protect the confidentiality, integrity, or availability of information during any of these missions, this failure could affect the overall objective to deliver secure, effective software to the warfighter. Not all of the missions listed above may be applicable to each SF. If a particular software factory does not have a helpdesk, for example, an evaluator can simply remove this mission from the test design.

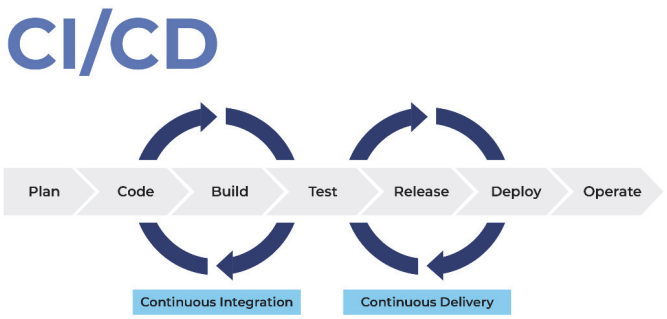
Additional software factory details: "DSB Task Force on Design and Acquisition of Software for Defense Systems" https://dsb.cto.mil/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

Description of DevSecOps: https://www.youtube.com/watch?v=5Eqz8hm3SVQ

According to Atlassian: "continuous integration is part of both continuous delivery and continuous deployment. And continuous deployment is like continuous delivery, except that releases happen automatically": https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

A software factory is the infrastructure and personnel used to develop, field, and sustain software for some DoD programs

DevSecOps software factories commonly employ a **continuous integration/continuous delivery (CI/CD)** strategy



IDA 6

Many software factories use DevSecOps style development and variations of the continuous integration/continuous delivery (CI/CD) processes. DevSecOps stands for Development, Security, and Operations; it is a set of practices and principles for software development that intrinsically link these three functions, rather implementing them sequentially, and is common in traditional software development.

Program offices and Service executives have argued that the continuously evolving software baseline and automatically implemented security checks and controls ensure that the software being fielded is secure, or that it can be refactored in response to identified or exploited problems. It is the role of operational testing to ensure that those security capabilities are being executed effectively.

Additional software factory details: "DSB Task Force on Design and Acquisition of Software for Defense Systems" https://dsb.cto.mil/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

Description of DevSecOps: https://www.youtube.com/watch?v=5Eqz8hm3SVQ

According to Atlassian: "continuous integration is part of both continuous delivery and continuous deployment. And continuous deployment is like continuous delivery, except that releases happen automatically": https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Our nation's weapons systems increasingly rely on software

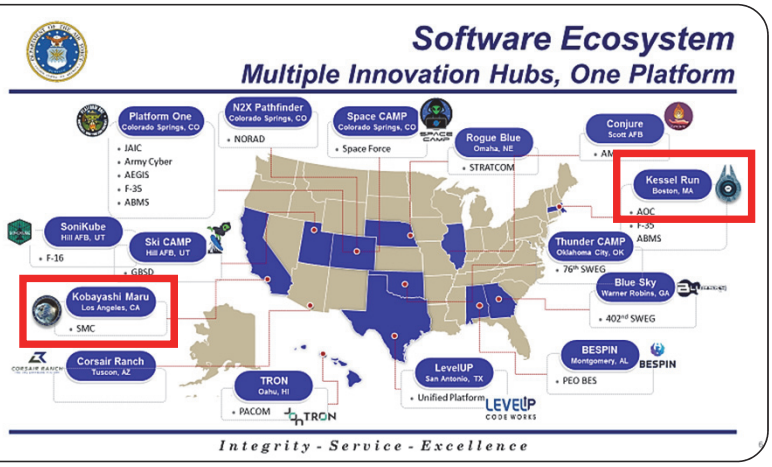
Increased dependence on software-reliant systems led to structural changes in software development

Waterfall development -> 1970s-1990s

Agile software development -> 2000s-present

Software factories typically include physically and logically distributed infrastructures, such as:

- Unclassified development environments
- Commercial cloud
- Remote developers over commercial internet



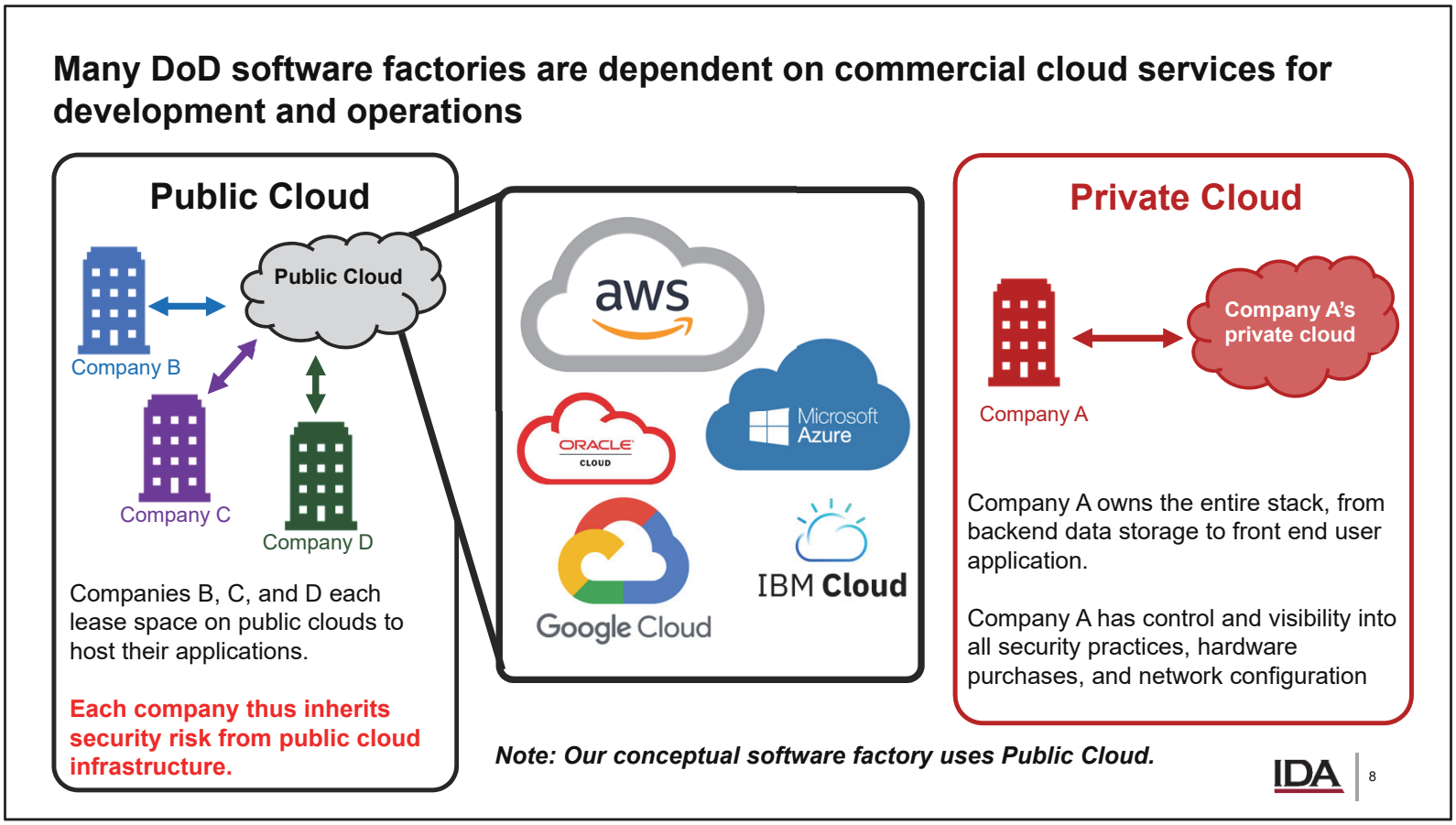
Above image and interactive list of Air Force software factories can be found at https://software.af.mil/software-factories/

IDA | 7

A software factory (SF) is part of the software supply chain for the acquisition systems they support. A SF can exist wholly within a building or it can combine physical facilities containing cloud-based infrastructure. Personnel may be constrained to work from a SF facility, or they may be able to work from remote locations, including their homes. This map shows the locations of the Air Force and Space Force's physical SF facilities, which may eventually rely on their PlatformOne and CloudOne initiatives – respectively.

There are some common features among the SFs familiar to us. They have physical workplaces in urban, mixed-tenant buildings, and leverage cloud services for authentication, network monitoring, code storage, code quality checking, compiling code, functional testing, and packaging applications for distribution. The cloud infrastructure includes software "pipelines," which are essentially cloud-based conveyor belts for the sequential steps between code quality checking and application packaging. These SFs use unclassified development environments, including allowing developers to work from any off-site location. The software produced in these facilities gets deployed to classified environments and may undergo additional development once in their classified environments. Developers, cyber defense personnel, and cleaning staff are often uncleared or not cleared to the highest classification level of the programs that they support. Classified development environments are notably absent from these examples.

While this map and the examples that we draw from are elements of the Air and Space Forces, the other Services appear to be moving toward SFs. The Navy is developing "The Forge" to support the Aegis combat system. That facility is being created with support from Raytheon, who were also involved in the creation of the Air Force's Kessel Run (https://www.raytheonmissilesanddefense.com/news/feature/updating-fleet-faster). Army Futures Command also intends to set up a SF in Austin, TX, though it is not clear to us which systems that facility will support (https://armyfuturescommand.com/software-factory/). Given that the guiding principles for these initiatives match Air and Space Force's, the arguments and machinery in this brief should apply.



Government organizations (specifically the Department of Defense) have become increasingly dependent on public cloud service providers (e.g. Amazon Web Services, Microsoft Azure) to provide the physical infrastructure and IT management necessary to host web applications and services. Commercial cloud providers allow customers to forego the management of computer networking, data storage, servers, and other IT tasks necessary to host applications and online services.

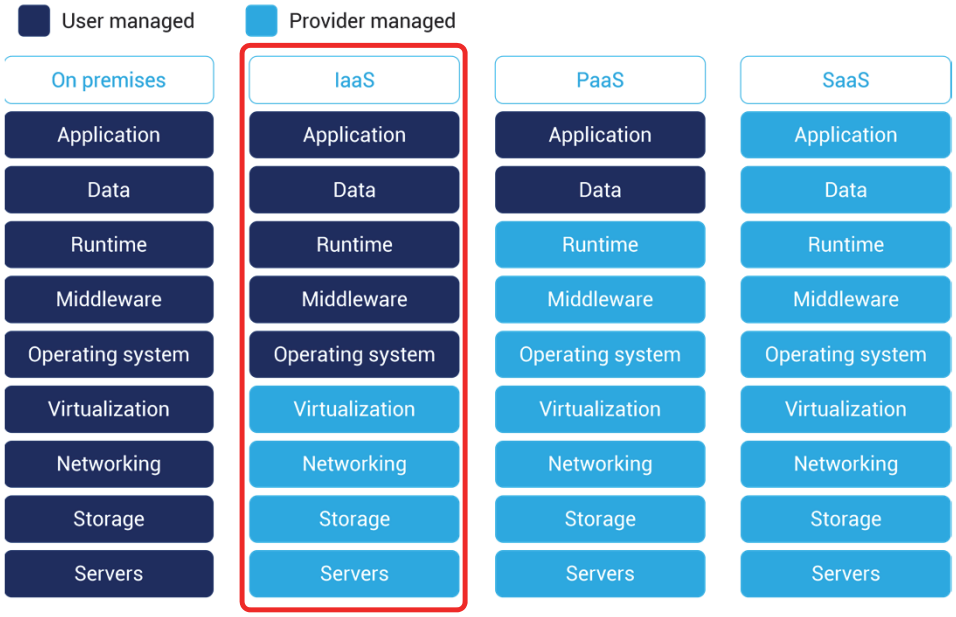
Prior to the existence of commercial cloud providers, enterprise organizations established and managed their own IT infrastructures. As the IT industry evolved, cloud services emerged as an alternative to on-premise management of IT infrastructure with three business models dominating the industry: Infrastructure as a Service, Platform as a Service, and Software as a Service.

Public cloud resources (e.g. servers and storage) are owned and operated by a third-party cloud service provider accessible via the internet. With a public cloud, all hardware, software, and other supporting infrastructure are owned and managed by the cloud provider.

A **private cloud** consists of cloud computing resources used exclusively by one business or organization. The private cloud can be physically located at the organization's on-site data center. But in a private cloud, the services and infrastructure are always maintained on a private network and the hardware and software are dedicated solely to your organization.

Some DoD organizations, such as the Missile Defense Agency, have set up private clouds to maintain full control and visibility of their cloud infrastructure.

Multiple levels of service exist for public cloud infrastructure



- Multiple organizations may be involved in monitoring and defense
- Development and operational environments can exist under different levels of service

Note: Our conceptual software factory uses Infrastructure as a Service (IaaS)

Provider managed services introduce test and evaluation limitations if the provider constrains ability to test end-to-end system, including those portions managed by the provider

IDA | 9

PaaS – Platform as a Service; SaaS – Software as a Service

Consumers can acquire cloud services using any one of several models. At one extreme, On-Premise clouds allow a software factory (SF) to run their own cloud services using hardware and software that they control, usually housed within their environment. At the other extreme, a SF could license software that achieves their objective; i.e. Software as a Service.

For the purposes of our conceptual SF, our cloud infrastructure is provided as Infrastructure as a Service (IaaS) by a commercial Cloud Service Provider (CSP), e.g. Amazon Web Services. The SF effectively rents a virtual environment in which they can create virtual machines, virtual networks, data stores et cetera. As consumers, the SF is responsible for maintenance, monitoring, and defense of roughly half the layers of the infrastructure that an adversary might attack. The CSP that provides the IaaS is responsible for the other half, namely on the server hardware and virtual machine networking. If IaaS is provided to the SF by another government entity, the government IaaS provider may present additional organizations with defensive responsibilities to consider. All defensive organizations should share alerting and indications of compromise.

In practice, SFs tend to employ IaaS for development environments but SaaS for things like project planning, project tracking, information sharing, and real-time threat detection; e.g., Atlassian, Crowdstrike. Each of these trust relationships can provide an avenue for an adversary to collect information or affect SF operations.

Commercial cloud services provide incredible value to the customer (i.e. our SF). However, the use of commercial entities can severely inhibit the government test community (e.g. DT&E, DOT&E) from conducting end-to-end tests on the entire system. Example: For an application on government oversight that is hosted on Amazon Web Services, Amazon may not allow government testers to evaluate the security of the hypervisor on which the application is hosted. Amazon might not have or provide documentation into the security practices, network architecture, red team reports, or other data elements that are necessary to provide a holistic assessment of the software supply chain

Moving to software factories changes the risk profile

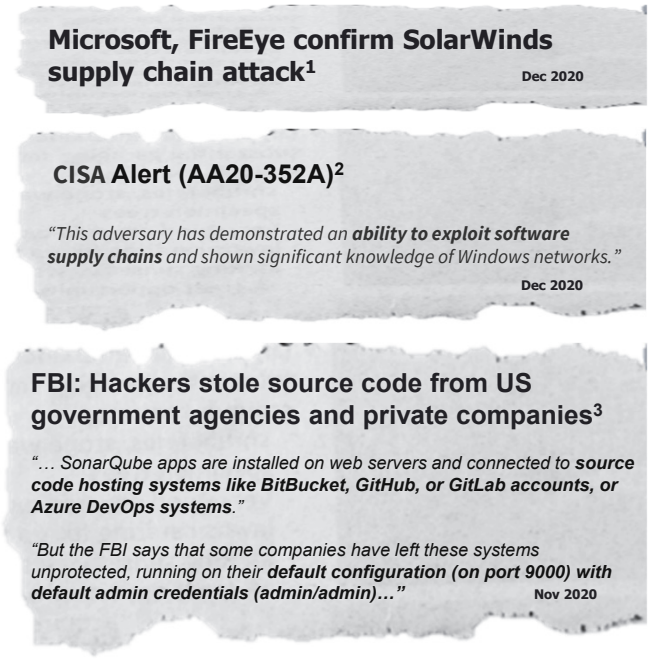
An operational evaluation of an acquisition system must consider the operational risk of a cybersecurity compromise to the supply chain.

Software factories are a vital element of the supply chain for the systems they support --- they provide much of a system's functionality.

Software factories may be interesting from a cyber targeting perspective:

- Mixed-tenant facilities in dense urban centers – places to hide
- Cloud-based infrastructure and offsite personnel at home – readily accessible
- Offsite personnel and distributed infrastructure – potentially less defender visibility, focus, or authority
- Geographically distributed personnel and some less well-vetted than others – opportunities for social engineering
- CI/CD fielding model – potentially limited additional scrutiny beyond development and employment

NB: Operational testing largely focuses on configuration flaws and feature misuse, rather than identifying fundamental problems in software



1. https://www.zdnet.com/article/microsoft-fireeye-confirm-solarwinds-supply-chain-attack/
2. https://us-cert.cisa.gov/ncas/alerts/aa20-352a
3. https://www.zdnet.com/article/fbi-hackers-stole-source-code-from-us-government-agencies-and-private-companies/

As we've seen in the fallout from the compromise of the SolarWinds software supply chain, the implications of an upstream compromise for the consumers of that software can be numerous and severe. Considering the implications of a similar attack, an attacker's intent may be to damage or undermine confidence in a critical defense system at a time of their choosing. That attack may begin with a supply chain compromise.

More than just identifying problems, one of the primary aims of operational testing with cybersecurity is to understand the operational risks of this kind of compromise. Thus, evaluators need to understand the potential for compromise of the relevant SF and the types of outcomes that an adversary might be able to achieve though SF compromise. A cybersecurity evaluation of the SF has direct implications for the system that consumes software from the SF.

As an example, a realistic threat actor can plausibly gain physical access to a SF and install a Raspberry Pi or MiFi device in that facility. They can plausibly rent a room across the street, enabling remote access to the SF. In order to make a determination of operational cybersecurity of the SF, an evaluator should consider the ability of that threat to affect the confidentiality, integrity, and availability of the information (and code) flowing through that SF. The evaluator may conclude that the threat can exfiltrate documentation and source code from their room across the street, or temporarily deny network access to the developers and security personnel to the network. These conclusions form the basis of operational questions for the acquisition system to be evaluated. Does the ability of an adversary to temporarily lock SF personnel out of their network have an operational effect on the performance of (e.g.) the deployed submarine that consumes the SFs software? What would be the operational impact of an adversary exfiltrating deep technical documentation and source code? These are questions that operational testing needs to answer.

The move to software factories enables Operational Test Agency investigations of the software supply chain

- Supply chain investigations have been hampered by the lack of visibility into contractor development facilities and processes.
- Software factories are government operated and managed, eliminating some legal restrictions.
- Existing DOT&E guidance is clear on the need, purpose, and objective:
  - "OTAs should also review risks introduced by the system supply chain to critical missions."
  - "Has the program provided all relevant documentation and developmental risk assessments to describe potential effects to critical missions from the system supply chain?"
  - "The purpose of testing cybersecurity during OT&E is to assess the ability of the system to enable operators to execute critical missions and tasks in the expected operational environment."
  - "The purpose of the AA is to characterize the operational effects to critical missions caused by threat-representative cyber activity against a unit trained and equipped with a system, as well as the effectiveness of defensive capabilities."
- Additional coordination with public cloud providers is needed if DOT&E wants to include provider-managed infrastructure in the supply chain investigation.

OTA – Operational Test Agency

Contract difficulties used to prevent substantive supply chain investigations in support of operational testing. Most software development was done by contractors in facilities not owned or managed by the government. The move from contractor facilities to DoD-controlled facilities alleviates some of these restrictions, since DOT&E has access to all data in the department.

Existing DOT&E guidance sufficiently motivates the need and objective of examining SFs: to inform operational testing with cybersecurity assessments of the system's supply chain. The critical missions of oversight systems depend on software produced by software factories. The OTAs have a duty to investigate the implications of a compromise of the software factory on critical system missions. If the OTA does not have sufficient documentation and technical knowledge of the software factory to fulfill this responsibility, they should conduct operational assessments to gather the requisite information.

Collecting data from software factories presents some unique challenges

A case study for SF OT&E: Kobayashi Maru (aka Space C2)

Space C2 consists of 30+ applications to support 6 mission areas and uses over 12 contractor developers.

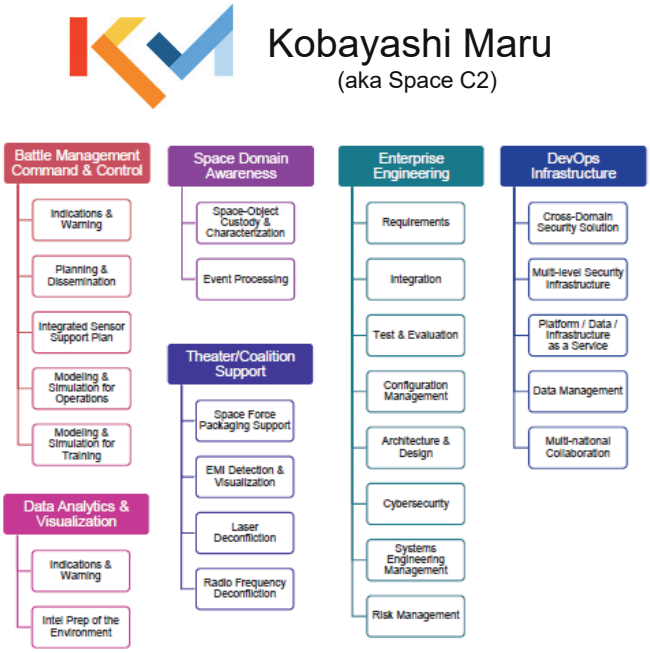
- Applications vary in operational importance (CAT A, B, C) and will require varying levels of DT/OT oversight.
- Applications will be deployed in Unclassified, Secret, and Top Secret domains.
- There is currently no plan for dedicated operational testing on the infrastructure and development pipeline.

Challenges:

Program unwilling or unable to provide detailed program documents (network architecture, application release dates, overarching test schedule).

Non-government contractors have pushed back on operational cyber team during cyber testing; e.g., team not given access to AWS cloud managed by Palantir.

Note: This T&E Concept is for the software factory infrastructure and pipeline, not the applications.



While the machinery of operational cybersecurity test design is applicable to any digital system, software factory architecture, processes, and tools introduce unique challenges for OTAs to collect software supply chain data. As mentioned earlier, the assets and personnel of any one software factory may be dispersed geographically and logically. To describe the vast geographic and logical footprint of a software factory, we will take Kobayashi Maru (aka Space C2) as an example.

The above chart outlines each of the capabilities that Space C2 aims to deliver by means of various software applications, each developed under the Space C2 program office. To develop these applications, Space C2 contracts developers from many industry partners (e.g. L3 Harris, L3-ADS, Lockheed, GDMS, MIT/LL, Ball, Palantir, ExoAnalytic Solutions, OrbitLogic, Leo Labs, Numerica, Centauri, BlueStaq, Solers, Omitron). Developers from these contractors may be embedded in the Space C2 facility or they may be stationed in their company's facility elsewhere in the country and communicate with Space C2 remotely. Although many of the delivered applications will be deployed in classified environments, a majority of the applications' architecture and functionality are developed at the unclassified level. The developers may or may not even be read in to the ultimate mission of the application.

The vastness and complexity of the software factory has presented challenges to operational testing. The program office has been unable to provide clear documentation detailing the network architecture of the software factory, the CI/CD pipeline, and interfaces with external partners. Similarly, they could not provide network architecture of the applications in development, despite entering iterative operational testing cycles.

Additional info:
Because the Space C2 program is developing many different applications, the US Space Force Command Enterprise Space Battle Management Command and Control (ESBMC2) Operational Acceptance Plan (OAP) has defined the

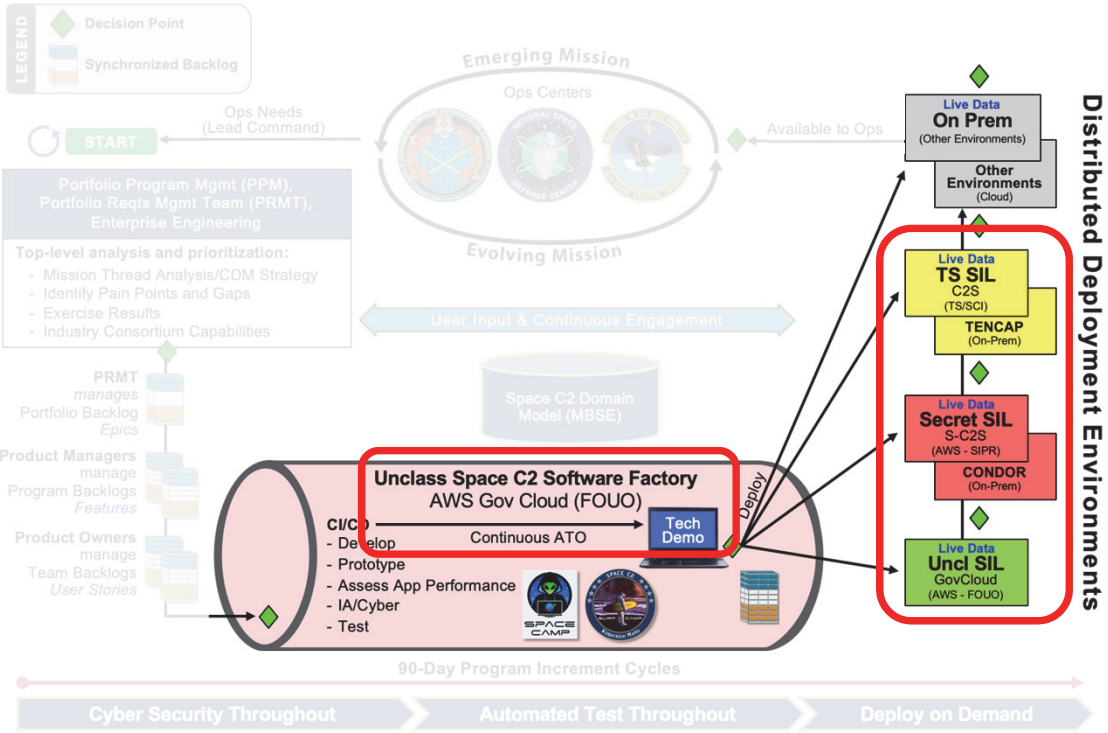
following categories and requirements for DT and OT testing.

Category A
Application provides key functionality for critical mission areas: SDA, BMC2, I&W, and Defense and Fires and will result in mission failure if unavailable
Testing: Complete DT/OT
Applications: ATLAS

Category B
Application provides key functionality for critical mission areas (i.e., SDA, BMC2, I&W, and Defense and Fires) and will result in reduced mission capability (work-arounds are available)
Testing: Limited DT/OT
Applications: C3PO, CPT

Category C
Application does not provide address a critical mission area or non essential support tool
Testing: Unit testing - No DT/OT required
Applications: Space Board, SDDCS

Unclassified software factories develop applications for deployment into all classification levels. KM factory and deployment environments reside on AWS.



AWS – Amazon Web Services; KM – Kobayashi Maru

The Space C2 development environment and CI/CD pipeline largely exist at the unclassified level, residing on AWS GovCloud. A majority of the applications' functionality and construction is conducted on unclassified operations floors or through a VPN connection from any remote location (home, coffee shop, etc.). Once the application reaches a level of maturity to deploy to operational users or to receive additional classified development, code is pushed to higher classification systems, presumably through a cross-domain solution or hand-carried via removable media (e.g. CD). The security of any cross-domain communication should be considered during operational testing.

Ideally, the classified systems receiving the newly-developed application would be simulated integration laboratories (SIL) or similar. That way, developers on the classified systems can verify, validate, and check the integrity of code coming from the unclassified environment before deploying it to operational users.

Presentation Roadmap

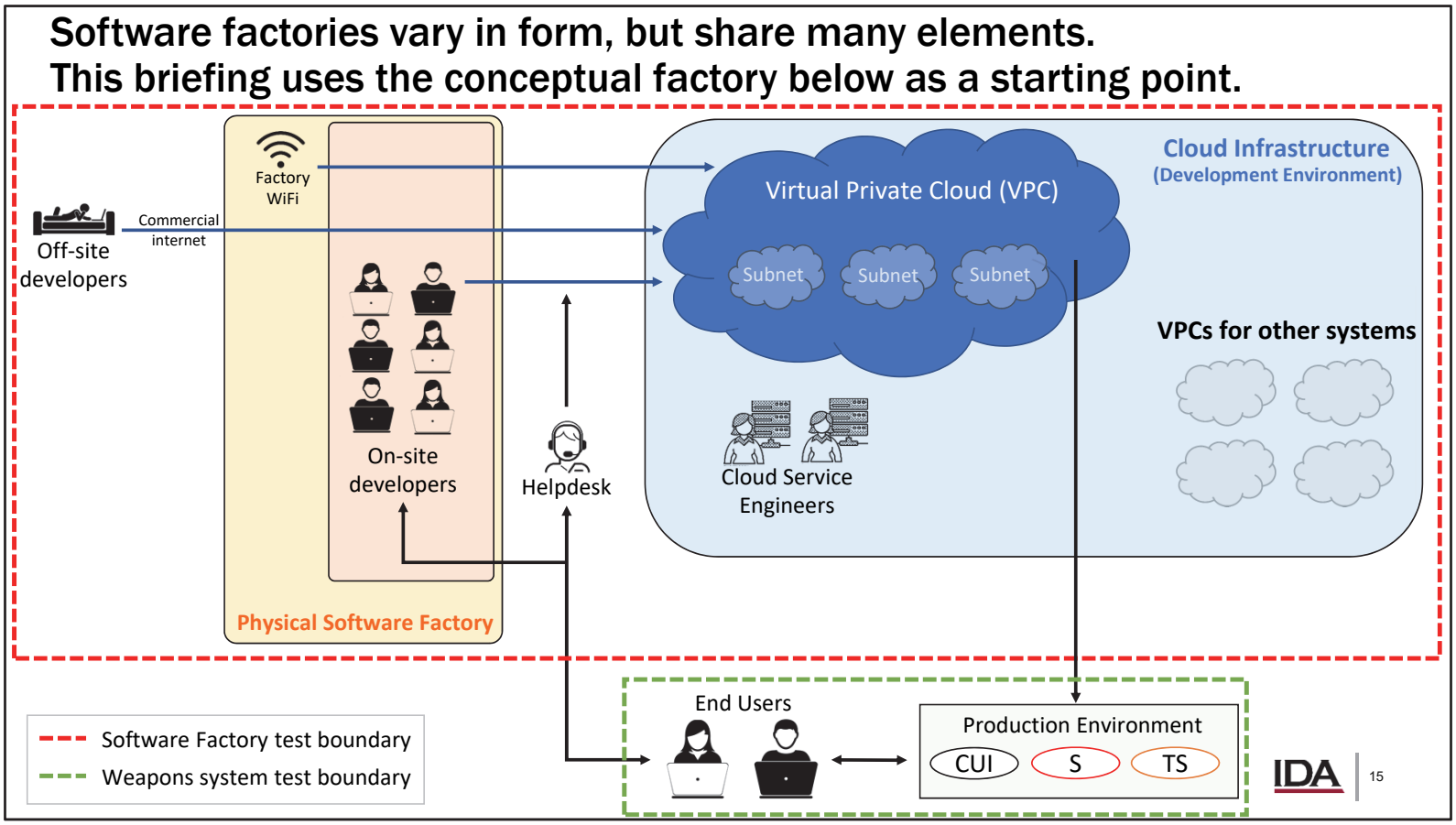
Introduction to Software Factories

Establishing a conceptual software factory

Developing a cyber test design for software factories

Executing an operational cyber test for software factories

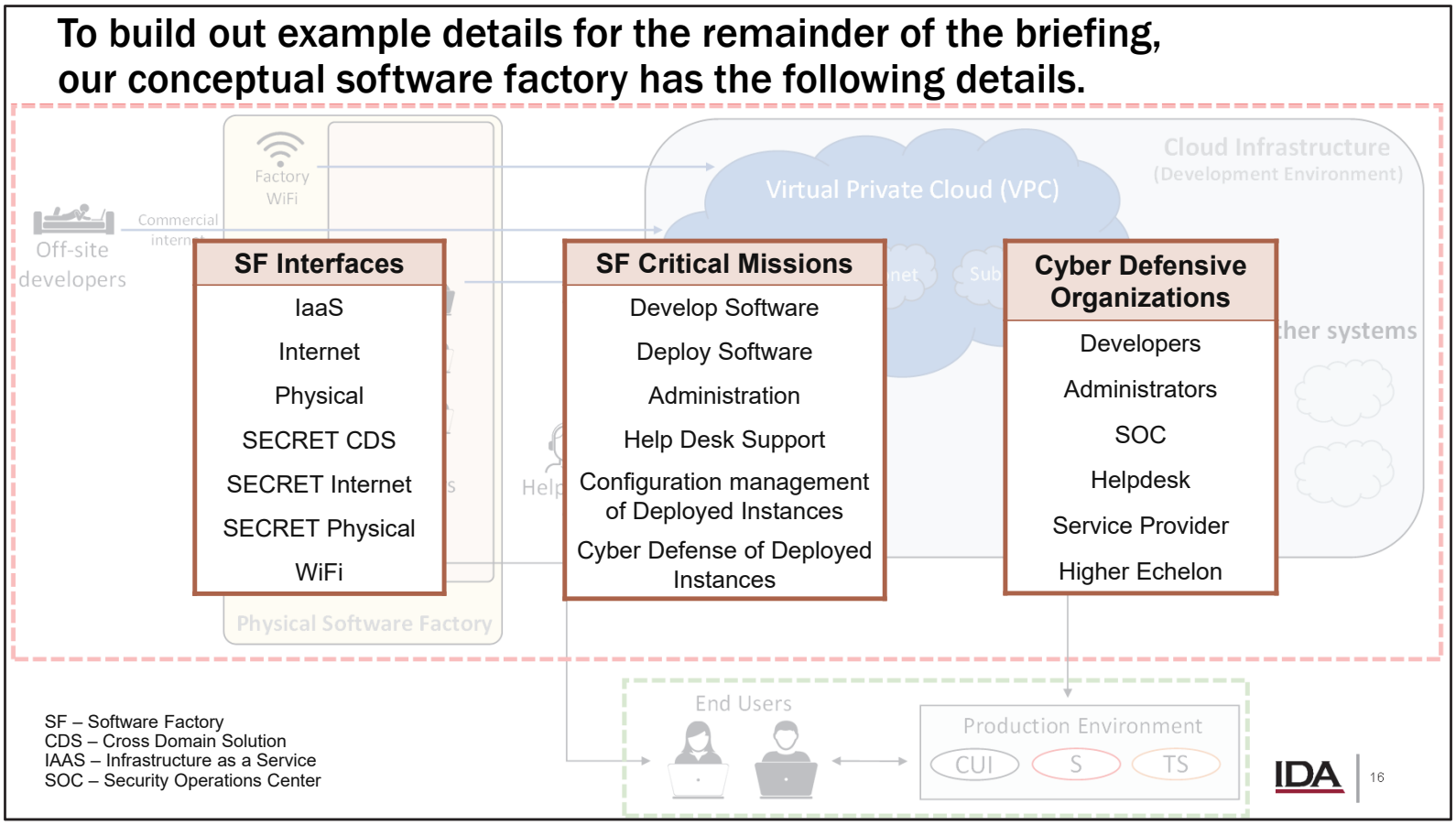
Incorporating software factory conclusions in oversight program OT&E



This slide introduces our conceptual software factory, which is representative of current DoD software factories. Notice the distinction between the software factory (red dashed box), where the application is developed, and the production environment (green dashed box), where the application will ultimately be used by operational users. This test concept only focuses on the software factory and its ability to support software production in the perpetually contested domain of cyberspace.

The Cloud Infrastructure – also referred to as Infrastructure-as-a-Service (IaaS) in this briefing – is a physically separate entity from the developers' locations and provides remote access to various cloud-based developer tools, including code repositories and the "pipeline" responsible for code quality checking, compiling source code, and packaging the applications for distribution. These are general, common categories of IaaS and could be hosted by any cloud service provider, like Microsoft Azure and Amazon Web Services (AWS). IaaS employees assist in managing the physical servers on which the Virtual Private Clouds (VPC) are hosted and may provide configuration and permission management support. However, IaaS personnel likely will not have permissions to access any of the code or files present on the VPCs themselves.

Using the continuous integration/continuous delivery (CI/CD) process, software factory operators regularly pull source code from the code repository, push it through the pipeline to compile the source code (if necessary) into a machine-readable format, perform developmental testing, and distribute the applications to the production environment. Thus, the software factory is responsible for configuration changes, software updates, rollbacks, and fixes based on user input.



Using our conceptual software factory, we will enumerate the categories of mission-critical functions that the SF performs, the interfaces by which an adversary might attempt to impede those functions, and a list of organizations with some role in the monitoring and defense of the environment.

The interfaces to our example SF are Infrastructure as a Service (IaaS; our public cloud provider), our connection to the internet, physical access to the building where on-site development occurs, and the on-site wireless internet access points. Our SF has a SECRET enclave, the interfaces to which include physical access, a connection to the SECRET internet, and the cross-domain solution that controls the flow of information between our unclassified and SECRET development enclaves. Additional or unusual interfaces, like radio frequency interfaces, can be treated like any other with the methodology described in this brief.

The software factory's critical missions can be binned into a few categories. We previously (on slide 5) defined the following categories for our conceptual software factory: develop software, deploy software, internal SF administration, manage configuration of fielded software and environments, user help desk capability, and cyber defense. In test design, accounting for mission-critical functionality is more important than the number of categories.

Our software factory has developers, administrators, a helpdesk, and a security operations center (SOC). Each of these types of personnel have the potential to identify cyber threat activity. External supporting organizations, including service providers and higher operational echelons have some visibility into the operations of our SF and therefore, also have a role to play in the defense of the system. Higher echelon could address the tiered cyber defenses for:

1. DoD (Base/local defenders, Service/Agency/CCMD CSSP, JFHQ-DODIN, USCYBERCOMMAND)
2. Commercial infrastructure (dependent on the commercial service provider)
3. U.S. Civilian infrastructure - Includes internet service provider defenses, Department of Homeland Security layers

Presentation Roadmap

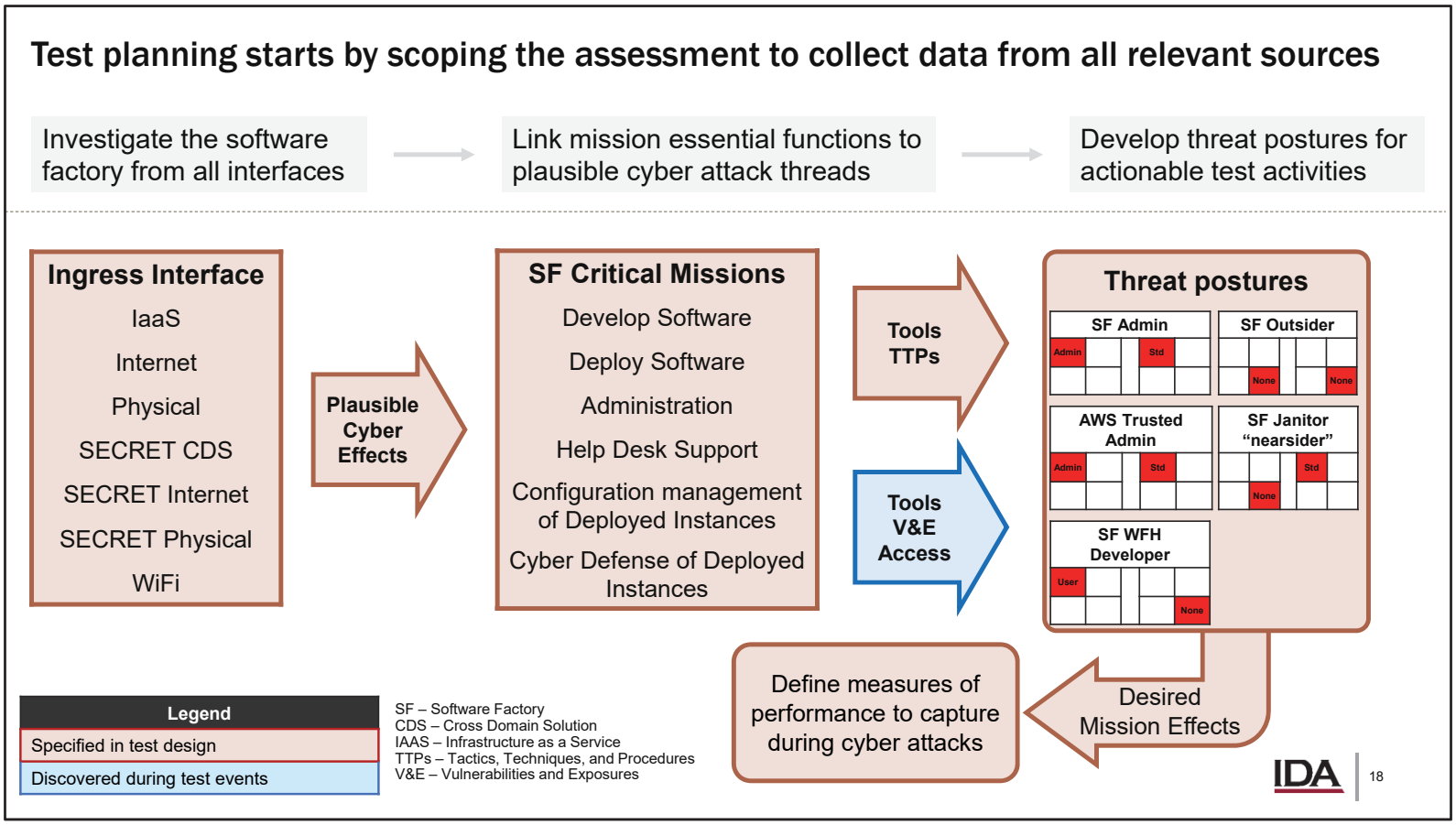
Introduction to Software Factories

Establishing a conceptual software factory

Developing a cyber test design for software factories

Executing an operational cyber test for software factories

Incorporating software factory conclusions in oversight program OT&E



Here we present our roadmap for the planning portion of the T&E process. In short, and from left to right, we look for connections between the external interfaces of the Software Factory (SF) and the devices and processes that it needs to operate. We consider the tooling necessary to investigate the SF from each interface. We consider how to enable (and resource for) a test team to identify vulnerabilities and exposures accessible from each interface, as an adversary would, during the assessments. We also incorporate relevant threat intelligence reporting to inform us of adversaries' capabilities and intents. Accurate threat intelligence motivates an investigation of the SF in the presence of multiple forms of relevant threats and threat postures. Finally, we consider how we might measure the change in performance of the SF's Critical Missions in the presence of those threats. We step through these in more detail in the following slides.

OTAs should identify all relevant defensive capabilities for each interface and map them to responsible cyber defensive organizations to plan data collection

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities

What are the defensive capabilities against cyber aggression on the _____ interface?

				Cyber Defensive Organizations						Non-Cyber	
				Developers	Administrators	SOC	Helpdesk	Service Provider	Higher Echelon	Physical	Policy
	Identify										
	Protect										
Defensive Action	Detect	Authenticated Access	Native Tool								
			Foreign Tool								
		Unauthenticated Access	Native Tool								
			Foreign Tool								
	Respond	Treat									
		Tolerate									
		Terminate									
		Transfer									
	Recover										

Complete this table for each interface to catalogue the cyber defensive posture for the software factory.

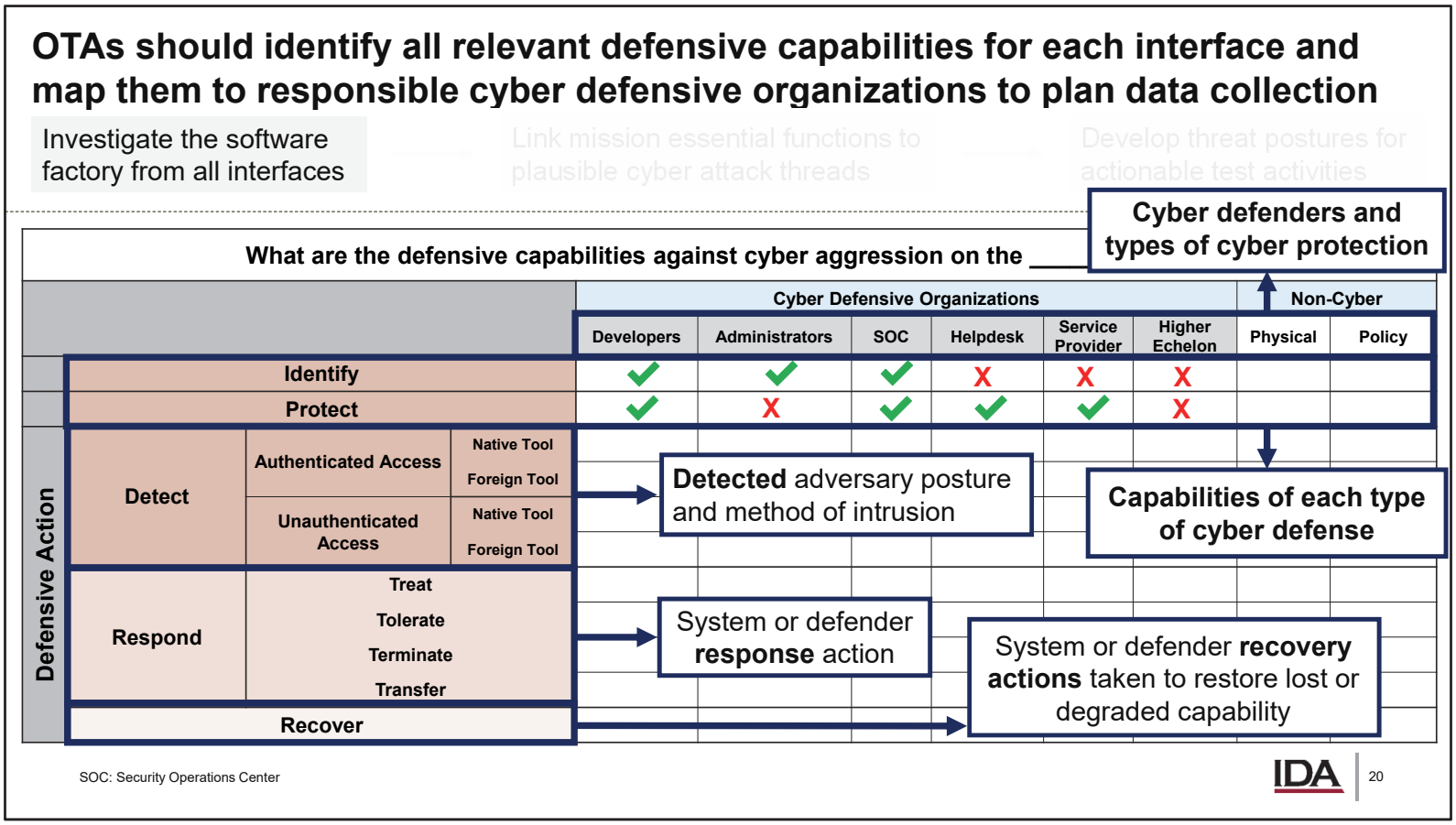
This mapping should influence where to place measurement instrumentation, data collectors, and observers in the operational test.

SOC – Security Operations Center

The first step in our test design process is identifying the defensive data needed for the evaluation. We present a worksheet to track defensive data collection on each interface to the software factory. The columns identify organizations with defensive responsibilities and authorities. The rows outline various responsibilities and capabilities that those defenders may have. As we collect documentation and data that refines our understanding of the software factory (SF), we will update the test design to reflect our understanding.

In order to evaluate the complete defensive posture of the SF, we will require information about each organization with a defensive role, responsibility, or authority. The factory's Security Operation Center (SOC) is perhaps the entity within the organization most responsible for, and with the most capability to implement, the security of the factory. However, there are likely Cybersecurity Service Provides (CSSPs) with greater capability and, in some cases greater authority, who share the responsibility for defense. For example, every DoD Service, most DoD Agencies, and even two Combatant Commands have their own CSSPs with enterprise-wide roles. Commercial Cloud Service Providers (CSPs) – the providers of our Infrastructure as a Service, have security organizations whose activities directly affect the security of software factories. Within DoD, "Higher Echelon" entities could include USCYBERCOMMAND, Joint Force Headquarters - DoD Information Networks (JFHQ-DODIN), and potentially other entities with authority over the organization that owns the software factory. Lastly, users, system administrators, and help desks often serve as the "first line of defense" by detecting and reporting anomalies in system functions to the responsible defenders.

Note that the defensive capabilities of a particular organization may vary within a single interface. It is important to be as granular as possible when collating defender responsibilities. For example, developers would certainly play a role in identifying malicious activity in the source code repositories, but might not have any visibility into a workplace training platform.



Every cell in the table represents a potential defensive activity for a defender to perform. In deciding which data are required for the evaluation, the operational test agency (OTA) should first determine which actions each defensive organization is tasked with and enabled to perform.

For example, the green check marks under "SOC" (security operations center) indicate that the organization has a role to 'identify', i.e. understanding and monitoring the internal architecture and standard processes used within the software factory (SF), and to 'protect' the SF against threat activity. In contrast, the red X under "Security Providers" indicates that higher-echelon defenders play no role in understanding the internal architecture or processes used in the SF.

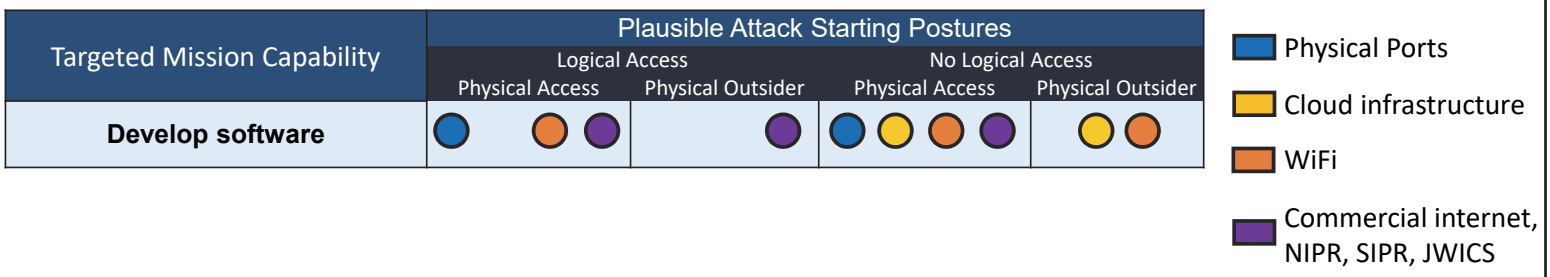
In practice, it can be challenging to collect information about the defensive organizations at the appropriate level of granularity during test planning. For example, an SOC might have multiple methods to identify malicious network traffic (e.g., endpoint detection, next generation firewalls, and automated log review). Interviewing technical leads in advance of test planning may be useful.

Cooperative assessments should capture information on the effectiveness of the defensive tools and processes to identify and eliminate cyber threats. The OTA should consider variations in the visibility of adversary activities; for example, a threat with system credentials, using tools installed on the network may require different tools to eliminate than an adversary attacking the firewall from the open internet. Similarly, the OTA should determine the utility of the various categories of actions that a defender might take to Respond to threat activity:

- "Treat" the activities with direct response actions
- "Tolerate" the activities because they don't substantially affect mission operations
- "Terminate" mission operations completely because the risk the adversary actions pose
- "Transfer" mission operations to another organization or enclave because the mission "must go on" but can't be performed without undue risk in the compromised environment.

OTAs should construct all plausible combinations of attack starting postures, interfaces, and the mission essential functions they could affect

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities



Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication)

IDA 21

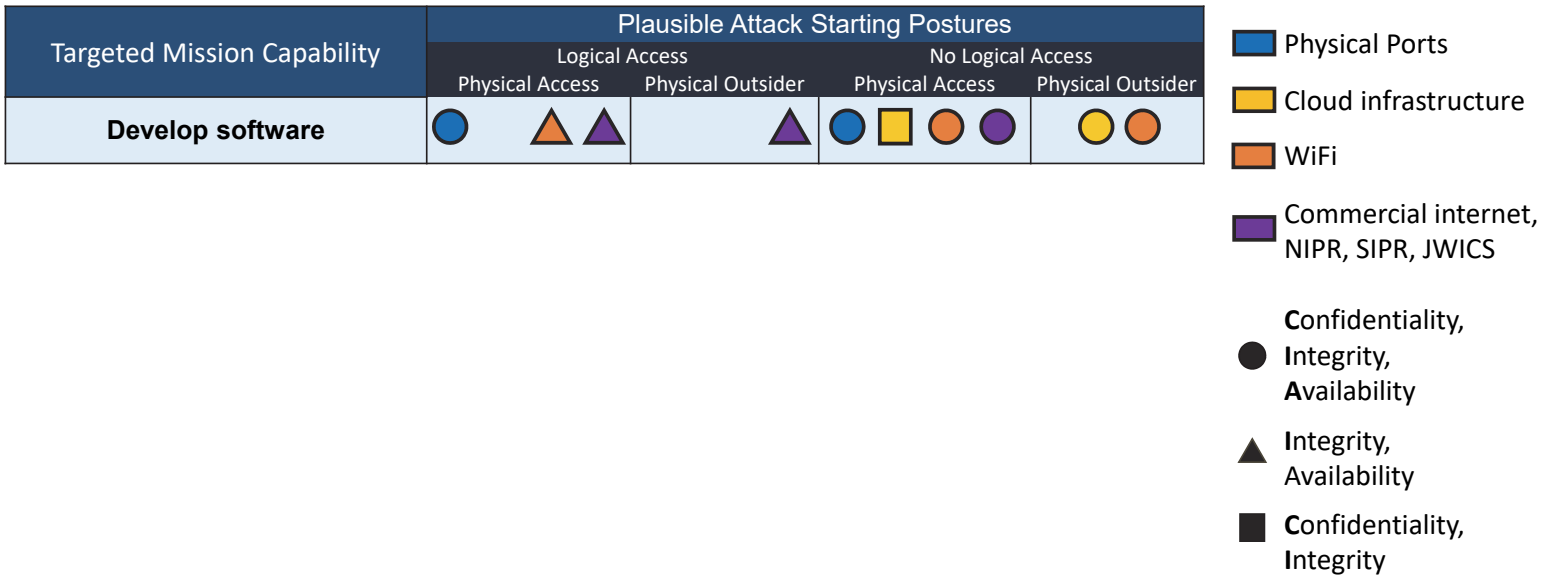
The next major step in our test design is looking at the ways an adversary might degrade the mission capability of the software factory (SF). This table will provide a visual representation of our postulated cyber attacks threads – combinations of the adversary's initial access, the interface they use to gain access to the SF, the critical function they want to degrade, and the forms of effects that they may want to use. As with the defensive data needs, we will update this chart as our understanding about the SF evolves; if testing demonstrates that a particular attack is not feasible, we remove that attack from our test design.

Note that our aim is to enumerate the set of feasible attacks, initially unconstrained by schedule, resources, tools, and the like. Waiting until we have a final, systematic test design to make scoping decisions allows us to make more granular and well-informed cost-benefit decisions, and ideally will help us prevent failure to imagine or fully consider plausible attacks. We use the following parameters to identify plausible attack categories:
1) The adversaries' initial access (columns); e.g., a bribed janitor may have physical access to the SF, but no network credentials
2) Ingress interface (colors); e.g., an adversary on the street may be able to access the wireless network access point
3) Mission of the SF (rows) plausibly targeted by adversaries. Here we show the ability to develop software, but we add others later
4) Type of attack (shapes); an adversary may see more benefit in affecting the Integrity of the software than in denying developers the ability to access it.

The mission capabilities – those critical duties of the SF that form the rows of this chart – are necessary inputs to the test design process. We use a general set of mission capabilities that should be applicable to most SFs, but these should necessarily be tailored as needed. Note that we will also need some information about the geographic distribution of assets and personnel, the physical and logical architecture of the SF, and the processes that it uses to create, test, package, and distribute software. This can come from documentation, physical inspection, interviews, or even early operational assessments.

OTAs should construct all plausible combinations of attack starting postures, interfaces, and the mission essential functions they could affect

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities



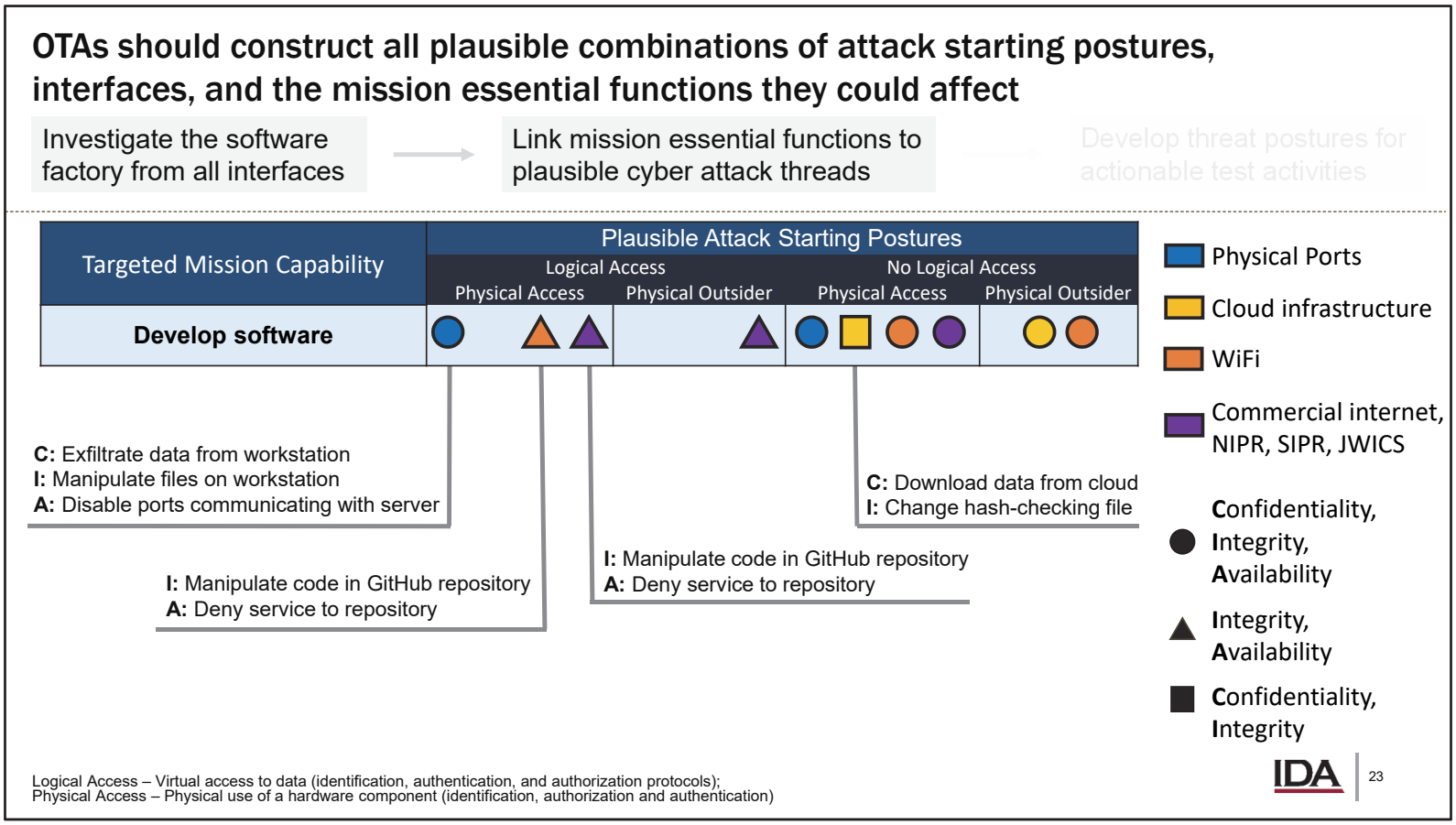
Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication)

IDA 22

The next major step in our test design is looking at the ways an adversary might degrade the mission capability of the software factory (SF). This table will provide a visual representation of our postulated cyber attacks threads – combinations of the adversary's initial access, the interface they use to gain access to the SF, the critical function they want to degrade, and the forms of effects that they may want to use. As with the defensive data needs, we will update this chart as our understanding about the SF evolves; if testing demonstrates that a particular attack is not feasible, we remove that attack from our test design.

Note that our aim is to enumerate the set of feasible attacks, initially unconstrained by schedule, resources, tools, and the like. Waiting until we have a final, systematic test design to make scoping decisions allows us to make more granular and well-informed cost-benefit decisions, and ideally will help us prevent failure to imagine or fully consider plausible attacks. We use the following parameters to identify plausible attack categories:
1) The adversaries' initial access (columns); e.g., a bribed janitor may have physical access to the SF, but no network credentials
2) Ingress interface (colors); e.g., an adversary on the street may be able to access the wireless network access point
3) Mission of the SF (rows) plausibly targeted by adversaries. Here we show the ability to develop software, but we add others later
4) Type of attack (shapes); an adversary may see more benefit in affecting the Integrity of the software than in denying developers the ability to access it.

The mission capabilities – those critical duties of the SF that form the rows of this chart – are necessary inputs to the test design process. We use a general set of mission capabilities that should be applicable to most SFs, but these should necessarily be tailored as needed. Note that we will also need some information about the geographic distribution of assets and personnel, the physical and logical architecture of the SF, and the processes that it uses to create, test, package, and distribute software. This can come from documentation, physical inspection, interviews, or even early operational assessments.



The next major step in our test design is looking at the ways an adversary might degrade the mission capability of the software factory (SF). This table will provide a visual representation of our postulated cyber attacks threads – combinations of the adversary's initial access, the interface they use to gain access to the SF, the critical function they want to degrade, and the forms of effects that they may want to use. As with the defensive data needs, we will update this chart as our understanding about the SF evolves; if testing demonstrates that a particular attack is not feasible, we remove that attack from our test design.

Note that our aim is to enumerate the set of feasible attacks, initially unconstrained by schedule, resources, tools, and the like. Waiting until we have a final, systematic test design to make scoping decisions allows us to make more granular and well-informed cost-benefit decisions, and ideally will help us prevent failure to imagine or fully consider plausible attacks. We use the following parameters to identify plausible attack categories:
1) The adversaries' initial access (columns); e.g., a bribed janitor may have physical access to the SF, but no network credentials
2) Ingress interface (colors); e.g., an adversary on the street may be able to access the wireless network access point
3) Mission of the SF (rows) plausibly targeted by adversaries. Here we show the ability to develop software, but we add others later
4) Type of attack (shapes); an adversary may see more benefit in affecting the Integrity of the software than in denying developers the ability to access it.

The mission capabilities – those critical duties of the SF that form the rows of this chart – are necessary inputs to the test design process. We use a general set of mission capabilities that should be applicable to most SFs, but these should necessarily be tailored as needed. Note that we will also need some information about the geographic distribution of assets and personnel, the physical and logical architecture of the SF, and the processes that it uses to create, test, package, and distribute software. This can come from documentation, physical inspection, interviews, or even early operational assessments.

OTAs should construct all plausible combinations of attack starting postures, interfaces, and the mission essential functions they could affect

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities

Targeted Mission Capability	Plausible Attack Starting Postures			
	Logical Access		No Logical Access	
	Physical Access	Physical Outsider	Physical Access	Physical Outsider
Develop software	● ▲ ▲	▲	● ■ ● ●	● ●
Store software and track versions				
Build and package software				
Test software				
Package and document software				
Deploy software				
Administration (Helpdesk)				

Physical Ports
Cloud infrastructure
WiFi
Commercial internet, NIPR, SIPR, JWICS
Confidentiality, **I**ntegrity, **A**vailability
Integrity, **A**vailability
Confidentiality, **I**ntegrity

Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication)

IDA 24

Based on our current state of knowledge, the shapes in this graphic represent the minimum set of cyber effects to investigate. This set of plausible attack categories is the starting point of a long process -- site visits, documentation reviews, cooperative reconnaissance. Our test design should evolve as we gain new information. For example, if we determine during cooperative testing that an adversary could not affect the ability of the software factory (SF) to build and package software, we should delete these objectives from our test design. To minimize resources, operational test agencies (OTAs) should use all existing sources of data (e.g., developmental assessments and Risk Mitigation Framework packages) and conduct integrated assessments.

In the lead up to adversarial assessments, our state of knowledge should allow us to identify a specific set of attacks needed for the evaluation. Based on our cooperative investigations, the remaining attacks should target all essential functions of the system that an adversary could plausibly degrade during a cyber compromise. These attack threads will exercise the defensive personnel responsible for monitoring and defending those critical functions, and enable the operational test agency to measure any degradation in performance. In turn, those degradations (e.g., the inability to prevent malicious code from being inserted into the source code) will inform future operational testing on the acquisition systems that acquire software from the SF.

Our selection criterion for the cyber effects in this chart is essentially, “what do we believe that an adversary could do?” Given what we learned during the various forms of reconnaissance, we consider all electrical connectivity and trust relationships that physically exist on the system. We do not consider what defensive tools and personnel “should” do, or what capabilities we believe our adversaries might have. This process differs from some other test design processes (e.g., the Air Force’s Mission-Based Risk Assessment Process for Cyber; MRAP-C) in that they often pare down the scope (e.g., based on resources, existing tools, or a perceived capability or intent of our adversaries) without fully considering all plausible scenarios. It is in these unstudied areas that attackers may find footholds to use to their advantage. Moreover, operational testing has demonstrated that there may exist creative ways to

emulate an attack for which there are not currently tools available.

For those already familiar with operational testing, note that the shapes here are not analogous to sampling points of a response surface that describes the SF's performance. The shapes represent spaces of data to be explored. Our expectations for attack threads to be emulated during Adversarial Assessments (AAs) should evolve with our understanding of the system. All of the shapes represent necessary investigations. We know of no statistical machinery (like Design of Experiments) that can be used to simplify the test design process or further minimize the required effort. On the other hand, judicious choices about the order or mechanisms for effects have demonstrated resource savings in operational testing.

OTAs should construct all plausible combinations of attack starting postures, interfaces, and the mission essential functions they could affect

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities

Targeted Mission Capability	Plausible Attack Starting Postures				Example performance measures
	Logical Access		No Logical Access		
	Physical Access	Physical Outsider	Physical Access	Physical Outsider	
Develop software					**Metric:** Successful Operator login **Threshold:** Mean time between failure
Store software and track versions					-
Build and package software					**Metric:** Successful code compilation **Threshold:** Mean time between failure
Test software					-
Package and document software					-
Deploy software					-
Administration (Helpdesk)					**Metric:** Failover, refactor, rollback software **Threshold:** Time to perform action

Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication)

IDA 25

Choosing performance measures for collection during adversarial assessments (AA) is a critical part of the test design process. Given that our investigation is of the supply chain for an acquisition system, we are interested in the extent to which the software factory (SF) can perform its critical functions with respect to delivering software that has not been compromised or been exfiltrated by an adversary. As part of the acquisition system's defensive capability, a SF might include the ability to refactor code following a cyber attack of the acquisition system. In this case, the measures would need to also include the ability to refactor code in a timely, and secure way.

In initial test designs, this could be a generic set of measurable aspects of the software factory's (SF's) critical mission capabilities. For example, the ability to build and package software might be partially quantified by the ability compile source code. To get a sense of the relative capability, we might measure the mean time between failures to compile source code. As we refine our attacks, we will necessarily need to refine our performance measures.

Measuring system performance under cyber aggression can give us an operationally relevant measure of the severity of cyber attacks. Oversight systems depend on the SF to deliver uncompromised software. Quantifying the inability of the SF to perform those tasks gives us a sense of the level of influence that attack might have on the oversight system that consumes software from our conceptual SF.

Applying the completed framework to develop test activities

To develop actionable test activities, map the framework's attack starting postures to the plausible threat postures made available by the SF's roles and privileges

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities

Threat posture identification framework

	Logical (network/system)		Physical (facility)		
	Authenticated	Unauthenticated	Authenticated	Unauthenticated	
Authorized	Trusted Insider	Insider - Unattributable Access	Trusted Insider	Insider - Unattributable Access	Authorized
Unauthorized	Captured Credentials	Outsider	Captured Credentials	Outsider	Unauthorized

Software Factory threat postures

SF Admin				
Admin			Std	

"Admin" and "Std" represent the network and facility level of privileges, respectively, that a SF Admin would have for our conceptual example.

These need to be defined for the real roles and privileges for the software factory of the system under test.

Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication);
SF – Software Factory; WFH – Work from Home;

The final step in test planning is determining the threat postures available to cyber threat actors relative to the software factory. Postures include all levels of access and privilege available to operational users. Continue to construct all plausible threat postures that an adversary could represent relative to the software factory. Any necessary account credentials should be provided to the test team prior to any cyber assessment.

The OTA should use the collection of all possible threat postures to develop actionable test activities and starting postures for the cyber test team. From each of the various starting postures, the cyber team will attempt to compromise select mission capabilities as directed by the OTA.

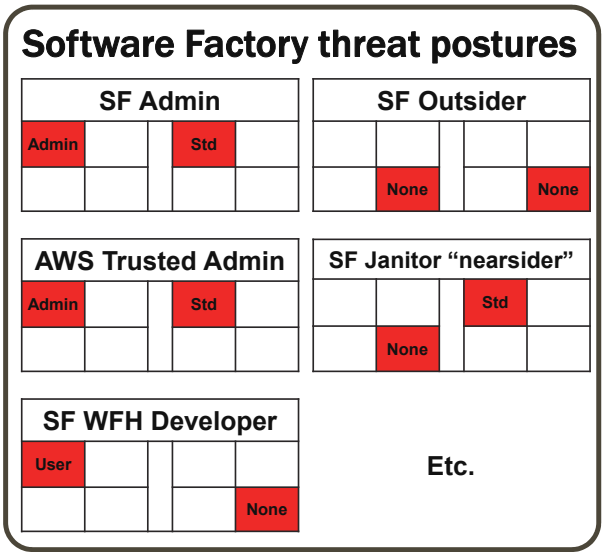
To develop actionable test activities, map the framework's attack starting postures to the plausible threat postures made available by the SF's roles and privileges

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities

Threat posture identification framework

	Logical (network/system)			Physical (facility)		
	Authenticated	Unauthenticated		Authenticated	Unauthenticated	
Authorized	Trusted Insider	Insider - Unattributable Access	Authorized	Trusted Insider	Insider - Unattributable Access	Authorized
Unauthorized	Captured Credentials	Outsider	Unauthorized	Captured Credentials	Outsider	Unauthorized

Software Factory threat postures



Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication);
SF – Software Factory; WFH – Work from Home;

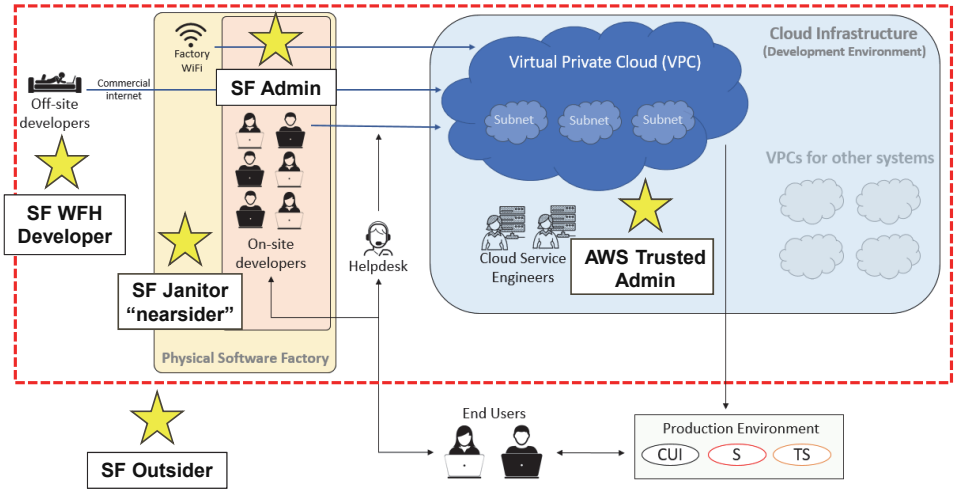
IDA 28

The final step in test planning is determining the threat postures available to cyber threat actors relative to the software factory. Postures include all levels of access and privilege available to operational users. Continue to construct all plausible threat postures that an adversary could represent relative to the software factory. Any necessary account credentials should be provided to the test team prior to any cyber assessment.

The OTA should use the collection of all possible threat postures to develop actionable test activities and starting postures for the cyber test team. From each of the various starting postures, the cyber team will attempt to compromise select mission capabilities as directed by the OTA.

To develop actionable test activities, map the framework's attack starting postures to the plausible threat postures made available by the SF's roles and privileges

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities



Software Factory threat postures

SF Admin				SF Outsider			
Admin		Std					
					None		None

AWS Trusted Admin				SF Janitor "nearsider"			
Admin		Std				Std	
					None		

SF WFH Developer			
User			
			None

Etc.

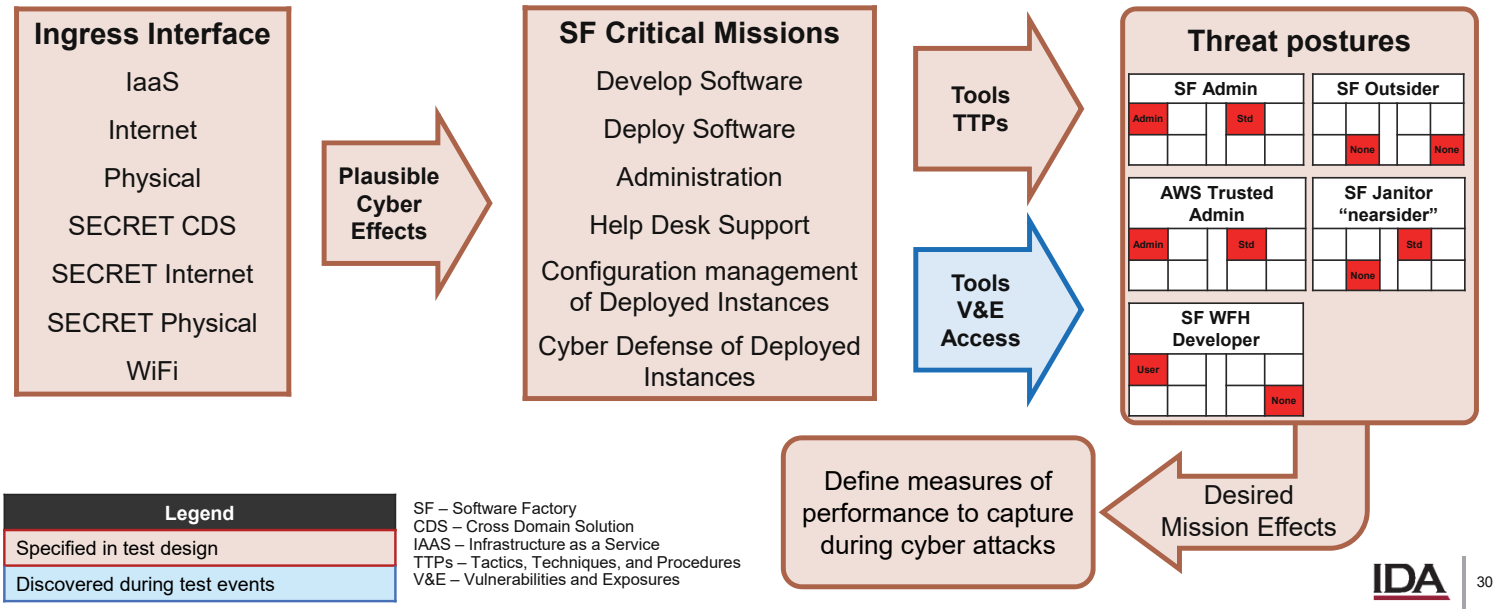
Logical Access – Virtual access to data (identification, authentication, and authorization protocols);
Physical Access – Physical use of a hardware component (identification, authorization and authentication);
SF – Software Factory; WFH – Work from Home;

IDA 29

The architecture diagram on the left, which we've already seen in a previous slide, now contains all of our potential adversarial cyber aggressors. Testing should be conducted such that an assessment can be made from each potential adversary posture and identify the categories of cyber affects that those actors could have on the mission functions of the software factory.

OTAs should ensure applicable attack postures are explored prior to test

Investigate the software factory from all interfaces → Link mission essential functions to plausible cyber attack threads → Develop threat postures for actionable test activities



Here is the planning roadmap we introduced earlier. All red boxes are items that should be understood and specified prior to test. Cyber team tools and all vulnerabilities and exposures will likely be discovered during cyber testing and cyber tabletop events.

Measure the software factory's performance in a cyber-contested environment

The OTA may consider collecting data on some of the following measures of performance, including confidentiality, integrity, and availability (C-I-A).

Measure	C-I-A	Description	Measure	Threshold for Effectiveness
Mean Time Between Critical Failure	A	A measure of system reliability that includes the effects of any fault tolerance that failure may exist. The average time between failures that cause a loss of a system function defined as "critical" by the customer.	Time (Hours)	Thresholds should be determined based upon user feedback and operational baselines.
Operational Availability	A	Basic measure for "real-world" availability	Uptime/(Uptime + Downtime)	
Mean Time to Restore Function	A	The time needed for maintenance actions to restore a system to fully operational condition including confirmation that no fault exists.	Time (Hours)	
Verification of signed code	I	This measure ensures that code is being verified when received by the user and that code contains the correct digital signature.	Valid signature/(total signatures checked)	
Time for notification of confidentiality anomalies (e.g. exfiltration)	C	The time elapsed before notification of certain operational anomalies, such as cyber-event indicators, configuration changes, slowed processing, or loss of functionality.	(Notifications within specified time) / (Notifications within specified time in non cyber-contested environment	

Some programs make the case that continuous integration and deployment of software means that software factories (SFs) can add features to acquisition systems or fix vulnerabilities quickly. In that case, the SF plays a role in the defense of the acquisition systems that use its software. Therefore, supply chain investigations should measure the effect of a cyber attack on the SF's ability to carry out its defensive actions. These Measures of Effectiveness are examples of measurables that could be used to measure the change in performance. Developing an appropriate list of measurables should be part of the development of a test plan.

Presentation Roadmap

Introduction to Software Factories

Establishing a conceptual software factory

Developing a cyber test design for software factories

Executing an operational cyber test for software factories

Incorporating software factory conclusions in oversight program OT&E

Planning: Gather information, develop a timeline, and allocate resources

Event	Frequency	Resources and Documentation	
Document collection and review	Prior to MBCRA	• Threat Assessment (e.g., STAR, VOLT) • Previous contractor testing • Message sets for each interface and protocol • Statement of Work • Agile process and sprint planning policies and procedures (including release cadence)	• Standard Operating Procedures • Maintainer handbook • Cross-domain solution documentation • Third-party hardware and software providers and versions
Mission-based cyber risk assessment (MBCRA)	Annual	• OTA cyber evaluator • Blue team representative(s) • Red team representative(s) • SF operator/SME representative(s)	• Up-to-date and detailed network architecture • Relevant, updated policy and procedure documentation (e.g., access control, version control, patch management)
Cooperative Vulnerability and Penetration Assessment (CVPA)	Based planned test cadence, informed by MBCRA (e.g. annually)	• OTA cyber evaluator • Blue team operators • Program office subject matter experts and constant on-site prime engineering support (including helpdesk personnel)	• Interface Control Documents for all interfaces and protocols • Detailed network architecture documents • Developer, administrative, and cloud credentials • Access to all hardware and software associated with mission-essential functions of the SF
Adversarial Assessment (AA)	Based planned test cadence, informed by MBCRA (e.g. annually)	• All above resources from CVPA • OTA data collector • Measurement instrumentation to collect system performance data.	• Red team operators • Blue team PMR data collectors • Trusted agents

OTA –Operational Test Agency; PMR–Protect, Mitigate, and Recover; SF–Software Factory; SME–Subject Matter Expert; STAR–System Threat Assessment Report; VOLT–Validated Online Lifecycle Threat

The evaluator should collect and review all available documentation for the software factory (SF). For SFs that are not well-documented, the OTA could combine site visits, observations, or operational assessments to collect the information required to begin building a test design.

The program office and OTA should resource for mission-based cyber risk assessments (MBCRA) and extensive research activities into the network architecture and data flows of the software factory. Such events include cyber tabletops, Mission-Based Risk Assessment – Cyber (MRAP-C), etc. Throughout these efforts, OTAs and program offices can identify all of the relevant data flows and properly scope the cyber assessments to satisfy DOT&E guidance and adequately assess system cyber survivability.

CVPAs an AAs are already familiar to the operational test community. We suggest continuing to use this construct to keep the objectives of the assessments clear.

Execution: Cooperative Vulnerability and Penetration Assessment & Adversarial Assessment

- This construct is familiar to OTAs and the test teams that support operational testing.
- This construct is just as applicable to supply chain investigations as it is to operational testing.
- Lessons learned in previous supply chain assessments:
  - Test team members should be "on-boarded" into active development teams and administrative roles for CVPAs, including provisioning of laptops and devices for multi-factor authentication. This access may also be useful for AAs.
  - To facilitate adversarial testing, existing SF members should be leveraged as Trusted Agents.
  - OTAs should have data collectors positioned with the defenders, in the SF.
  - A classified space should be made available for post-mortem dissection by attackers and defenders in AA.
  - The test plan or rules of engagement should clarify whether developer home networks are in-scope.
  - If the cloud service provider is a commercial entity, legal restrictions may limit the scope of the investigation.

CVPA – Cooperative Vulnerability and Penetration Assessment; AA – Adversarial Assessment OTA – Operational Test Agency; SF – software factory

The operational cyber test construct of cooperative vulnerability and penetration assessments (CVPA) and adversarial assessments (AA) should be familiar to OTAs and presents an adequate structure for cyber tests on SFs. We recommend OTAs follow the same resourcing procedures for personnel and equipment as they would for CVPAs and AAs on any oversight program.

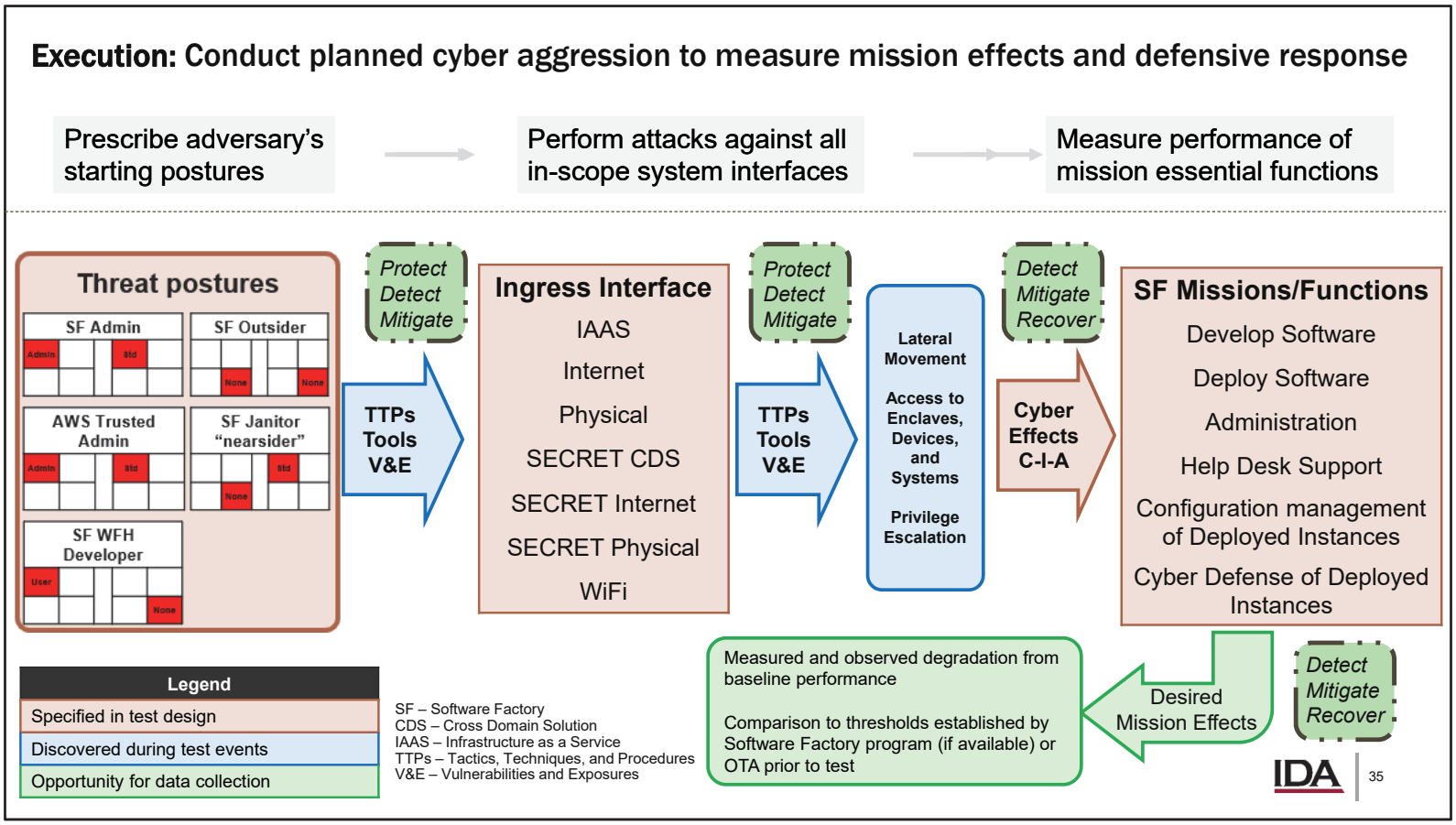
The program office should resource accounts for each role the testers intend to assume, e.g., developer, security team, and help desk. On-boarding the testers into the system gives testers the same perspective of the network and security controls as seen by the role they're assuming.

Because accounts provisioned for the testers will necessarily be known to defenders, OTAs should facilitate the use of non-tester accounts in the environment from which to launch attacks during AAs. This will test the ability of the materiel and personnel defenses to differentiate between normal and malicious activity, rather than simply the ability to identify activity on known-malicious accounts.

Remote test observation has proven to be very difficult to execute, especially in classified spaces, and often lacks the quality of in-person test observation and personnel interactions. We recommend that test personnel, subject matter experts, and other stakeholders be collocated during the assessment to maximize communication and collaboration, even though it may be possible to remotely access the software factory infrastructure (e.g. code repository) from disparate locations.

It is likely that some classified discussions will need to take place following testing. Some SF employees may not have appropriate clearance to discuss problems with the SF configuration, and the SF may not have an environment to discuss classified material. These logistics should be worked out in test plans.

Legal restrictions or liability considerations by the test team may inhibit the collection of data, or emulated aggression against SF employees working from home or a cafe. The test plan and ROE should be clear about whether these environments are in-scope, and offer mitigations to ensure that appropriate data are collected.



The **test execution** process contains most of the same components as the test planning process, but they are employed in a reversed order. At the onset of test, the cyber team (cooperative or adversarial) should position themselves in the prescribed attack postures. They will then explore and aggress the targeted system interfaces and perform any penetration testing or red team activity using their own set of tools and tactics, techniques, and procedures (TTPs). All of the cyber team activity should be collected by the OTA.

Performance metrics and system/operator response should be measured by the OTA, using data collection instrumentation to quantify mission effects. System/operator response (protect, detect, respond, recover) should be measured at the point of ingress, throughout lateral movement, and after deployment of each cyber effect.

Presentation Roadmap

Introduction to Software Factories

Establishing a conceptual software factory

Developing a cyber test design for software factories

Executing an operational cyber test for software factories

Incorporating software factory conclusions in oversight program OT&E

Software factories should undergo schedule-driven cybersecurity testing, which would:

- Allow for structured planning and resourcing of data collection activities in the lead up to OT&E of a DoD system
- Capture undocumented details on frequently evolving network architecture and procedures changes
- Allow test agencies to adjust to new data needs and mitigate data shortfalls
- Not preclude event-driven assessments around major changes to the software factory or its processes

We suggest that OTAs conduct cybersecurity assessments of software factories on a recurring basis at some regular interval. The testing cadence should be decided by DOT&E, the PM and the OTA by considering program increment cadence, ability to reasonably assume risk for the non-test period, and funding.

Software factories are large ever-changing systems supporting multiple programs, and each program generally requires multiple software applications to support operations. Each assessment of the software factory is a snapshot relevant to some applications and maybe not others. Since a supply chain assessment should necessarily consider the risks to all of the applications that eventually get integrated into an oversight program, multiple snapshots may be necessary. The OTA and PM should coordinate to mitigate risk as much as possible, including adding delta testing if the risk becomes high enough.

If software factory has not changed, the OTA should be able to leverage previous DT/OT results, threat intelligence, etc. to support their specific OT testing of the oversight program. Use of previous reporting is dependent on the extent to which the software factory was tested during that event. Using previous data requires an understanding of all limitations and assumptions involved in the earlier testing.

Developmental testing and cooperative operational assessments will provide the information needed to pare the scope of the software factory adversarial assessment down to a set of attack threads with a plausible impact to the operations of the oversight program.

Cyber effects against a software factory pose operational risks to warfighters through the systems they support:

Confidentiality: Gather data and intelligence into warfighter operations and procedures through compromised source code, applications, configuration files, and documentation

Integrity: Interfere with system operation through malicious manipulation of applications and configuration files

Availability: Prevent the system from receiving software updates for improved functionality or cyber defense

The software factory cybersecurity assessment results should inform the system OT&E by characterizing the intent, capability, and risk of cyber threats to the software factory and system.

We know from the experience of SolarWinds that a variety of options become available to adversaries that can access the software supply chain. The result of our software factory assessment is a better understanding of the parts of the software supply chain produced by the software factory that an adversary could access, along with the plausible outcomes for the posture of the applications produced.

Those applications get deployed to oversight systems. They reside on systems in classified enclaves, maybe running with administrative privilege on that system. They are intended to have access to some data, and they may have access to other data. The applications may have access to external network interfaces.

The software factory assessment helps the OT community narrow the scope of cybersecurity testing to the subset of applications and interfaces that can plausibly be affected by a supply chain attack. And perhaps more powerfully, if done properly, it provides a backstop against the failure of imagination of individual testers in the heat of an adversarial assessment against an oversight program. Our adversaries have the time, resources, and interest to think more deeply. And we should too.

Software factories are key components of oversight programs' supply chains, and assessing their cybersecurity is critical to adequate OT&E.

This test concept provides a framework and methodology for assessing the cybersecurity of software factories, which aligns with OT&E requirements from Title 10 and published DOT&E guidance.

Contract modifications or data sharing arrangements will be required to incorporate public cloud infrastructure in the supply chain investigation.

These products provide the ingredients for the OT&E community to reach consensus and to facilitate more objective, clear resourcing in OT&E strategies.

The software factory cybersecurity assessment results should inform the system OT&E by characterizing the intent, capability, and risk of cyber threats to the software factory and system.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
July 2022	Final	

4. TITLE AND SUBTITLE

Cybersecurity Operational T&E Concept for Software Factories

5a. CONTRACT NUMBER

HQ0034-19-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Peter M. Mancini (OED); William J. Robbins (OED); Jason P. Sheldon (OED); Erick D. McCroskey (OED); Brandon A. Shapiro (SFRD); Troy W. Lowry (OED); Kathleen Falcon (OED); Eliza M. Johannes (IAD)

5d. PROJECT NUMBER

BD-9-229992

5e. TASK NUMBER

229992

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

D-21595

H 2021-000083

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Director, Operational Test and Evaluation
1700 Defense Pentagon
Washington, DC 20301

10. SPONSOR/MONITOR'S ACRONYM(S)

DOT&E

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

Project Leader: Robbins, William J.

14. ABSTRACT

This test and evaluation concept highlights the critical role of Department of Defense-owned software factories as key components in the software supply chain of major acquisition systems and provides a structure by which to design operational cyber testing of a software factory to inform the supply chain evaluation of the program that consumes its software. This briefing describes a conceptual software factory. Actual factory architectures will likely be more complex, and will require open discussions from all stakeholders (program office, test community, oversight, operational users, etc.) during test planning and execution. This document can help facilitate those conversations and provide a framework for stakeholders to both plan operational tests and also keep an account of what data was collected during test execution.

15. SUBJECT TERMS

Software Factory; Cybersecurity Operational Tests

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			William J. Robbins
Unclassified	Unclassified	Unclassified	Unlimited	62	19b. TELEPHONE NUMBER *(Include area code)* (703) 845-2595