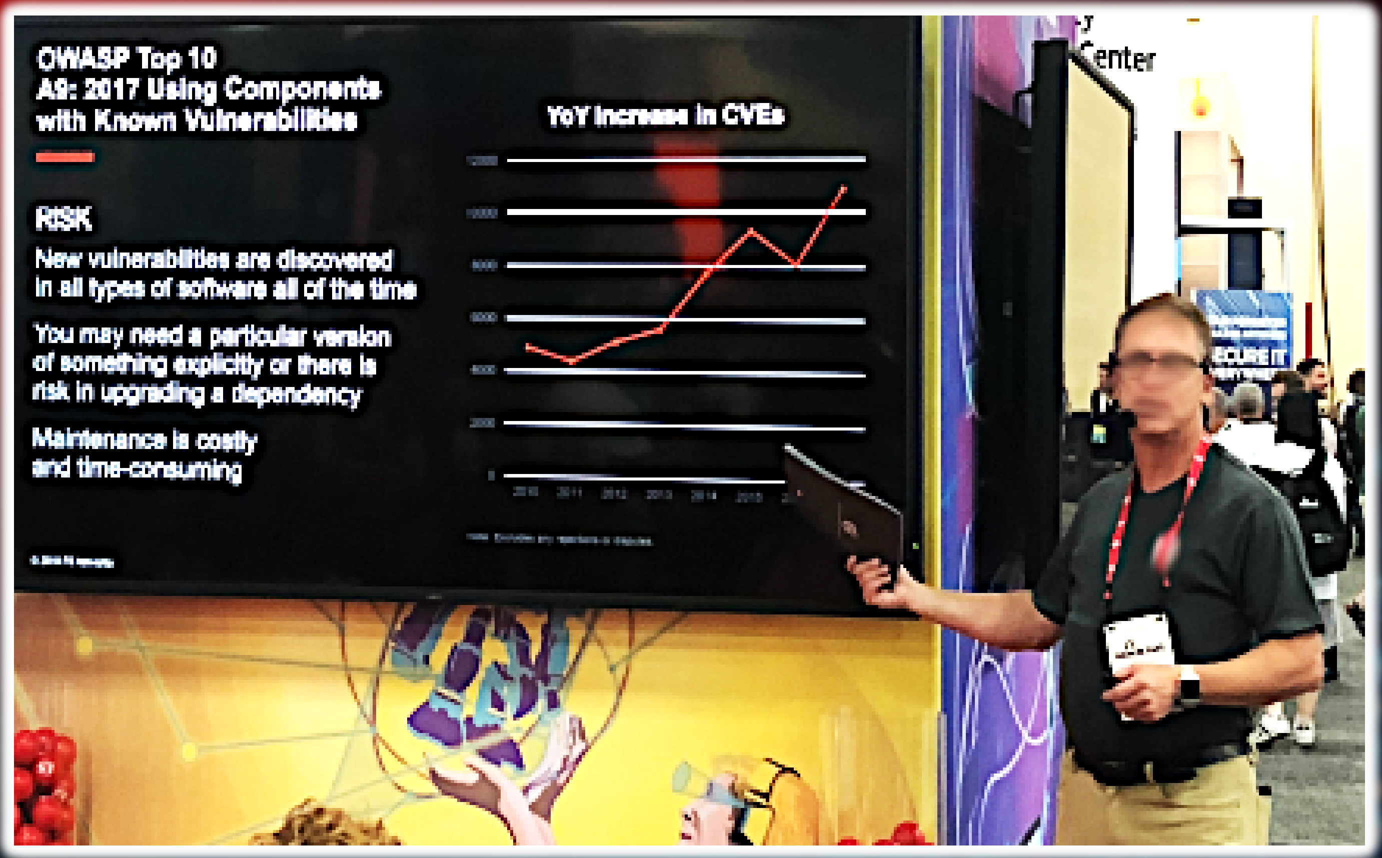
Software Bill of Materials (sBOMs)

(Removing Barriers to the application of tooling to C-SCRM and Software Assurance)

Robert A. Martin

Sr. Secure Software & Technology Prin. Eng.
Trust & Assurance Cyber Technologies Dept.
Cyber Solutions Technical Center



OWASP Top 10
A9: 2017 Using Components with Known Vulnerabilities

RISK

New vulnerabilities are discovered in all types of software all of the time

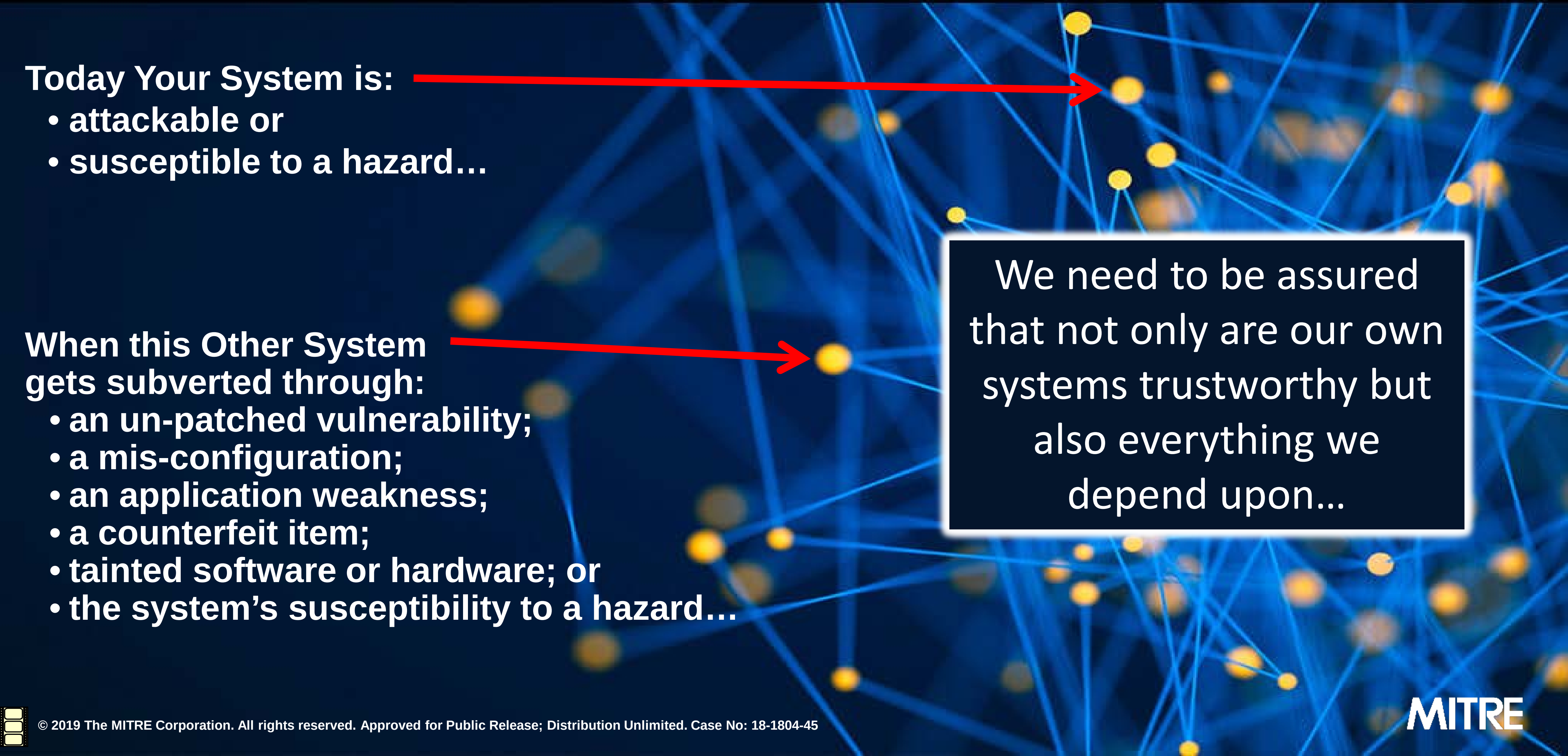
You may need a particular version of something explicitly or there is risk in upgrading a dependency

Maintenance is costly and time-consuming

Presented at the DoD, DHS, NIST, and GSA sponsored Software and Supply Chain Assurance Forum hosted at MITRE McLean, VA

MITRE

Everything is Becoming Software-Enabled and Connected, Either through Task Dependency, Supply Chain, or Information Flow



Today Your System is:

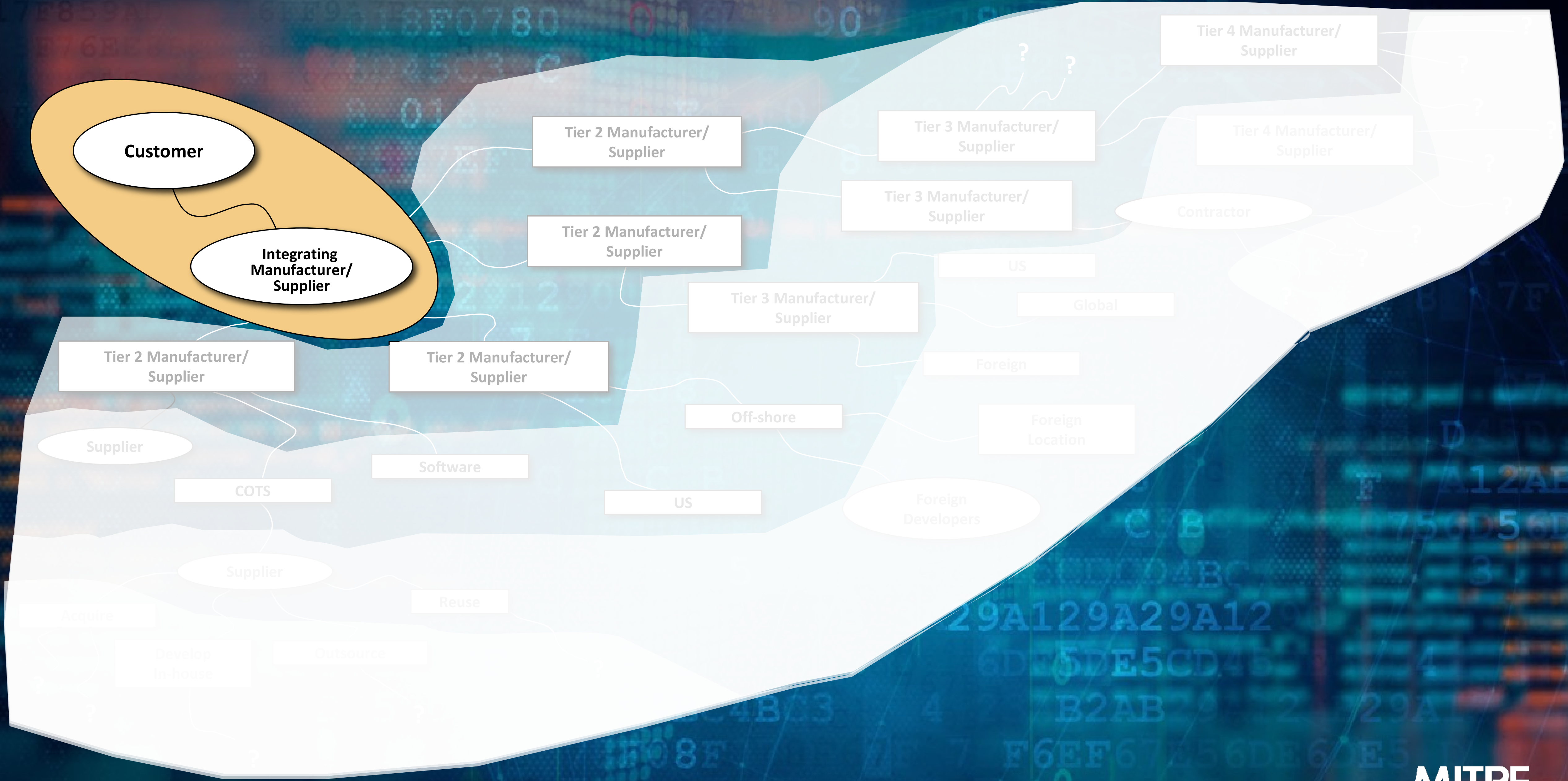
- **attackable or**
- **susceptible to a hazard…**

When this Other System gets subverted through:

- **an un-patched vulnerability;**
- **a mis-configuration;**
- **an application weakness;**
- **a counterfeit item;**
- **tainted software or hardware; or**
- **the system’s susceptibility to a hazard…**

We need to be assured that not only are our own systems trustworthy but also everything we depend upon…

The Supply Chain for Software-Enabled Capabilities is Opaque


Customer
Integrating Manufacturer/ Supplier
Tier 2 Manufacturer/ Supplier
Tier 2 Manufacturer/ Supplier
Tier 2 Manufacturer/ Supplier
Tier 2 Manufacturer/ Supplier
Tier 3 Manufacturer/ Supplier
Tier 3 Manufacturer/ Supplier
Tier 3 Manufacturer/ Supplier
Tier 4 Manufacturer/ Supplier
Tier 4 Manufacturer/ Supplier
Contractor
US
Global
Foreign
Off-shore
Foreign Location
Supplier
Software
COTS
US
Foreign Developers
Supplier
Reuse
Acquire
Develop In-house
Outsource
?

Market Transparency through “Software Bill of Materials”

- **Third party components are a known systemic risk.**
  - Transparency can drive tools and behavior to document risk, support mitigations, and drive better SW development practices.
- **NTIA at Commerce launched an open, community-driven, cross-sector “multistakeholder process” to promote software component transparency.**
  - Understand the problem and define basics of SBOM
  - Develop use cases across sectors on how such data can be used, today and in the future.
  - Guidance on how to use existing standards to implement SBOM
    - Software ID tags (SWID)
    - Software Package Data Exchange (SPDX)
- **Expected draft deliverables late spring 2019**
- **More info or to join: afriedman@ntia.doc.gov**

Use Cases for sBOM

Refer-Acquire-Transfer
(definition of what it is)

Pedigree
(history of how it was produced)

License Management
(conditions about its use)

Provenance
(chain of custody of it)

Integrity
(cryptographic basis of unalteredness)

Market Transparency
(public DBs of components)

Formulation
(how it was compiled/formed)

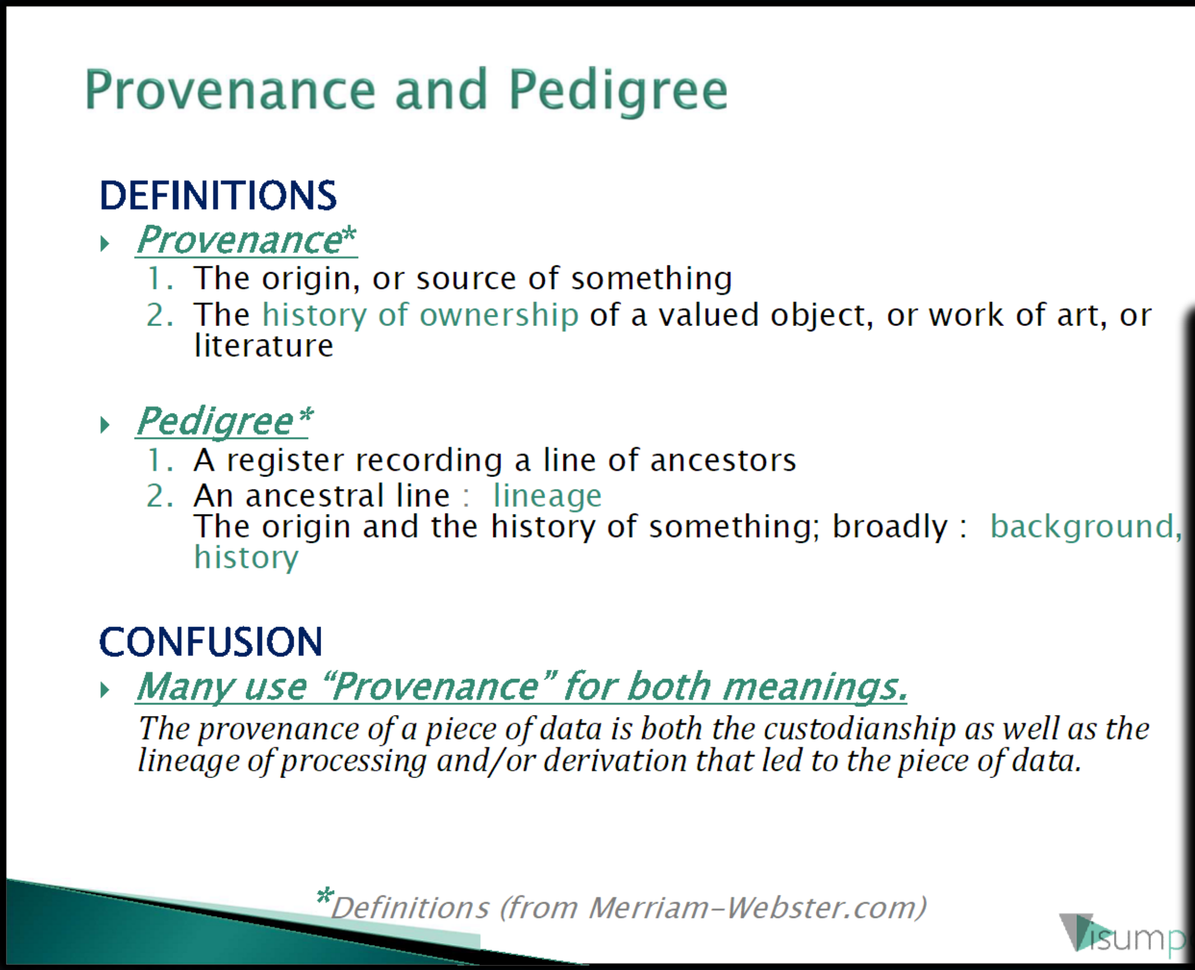
Assurance
(trustworthiness of it)

sBOM of a Service
(sBOM of sw delivering service)

Patch Currency
(known fixes are applied to it)

Automated Response
(sBOM parsing/action)

Assured Mket Transp
(public DBs of components)



Provenance and Pedigree

DEFINITIONS

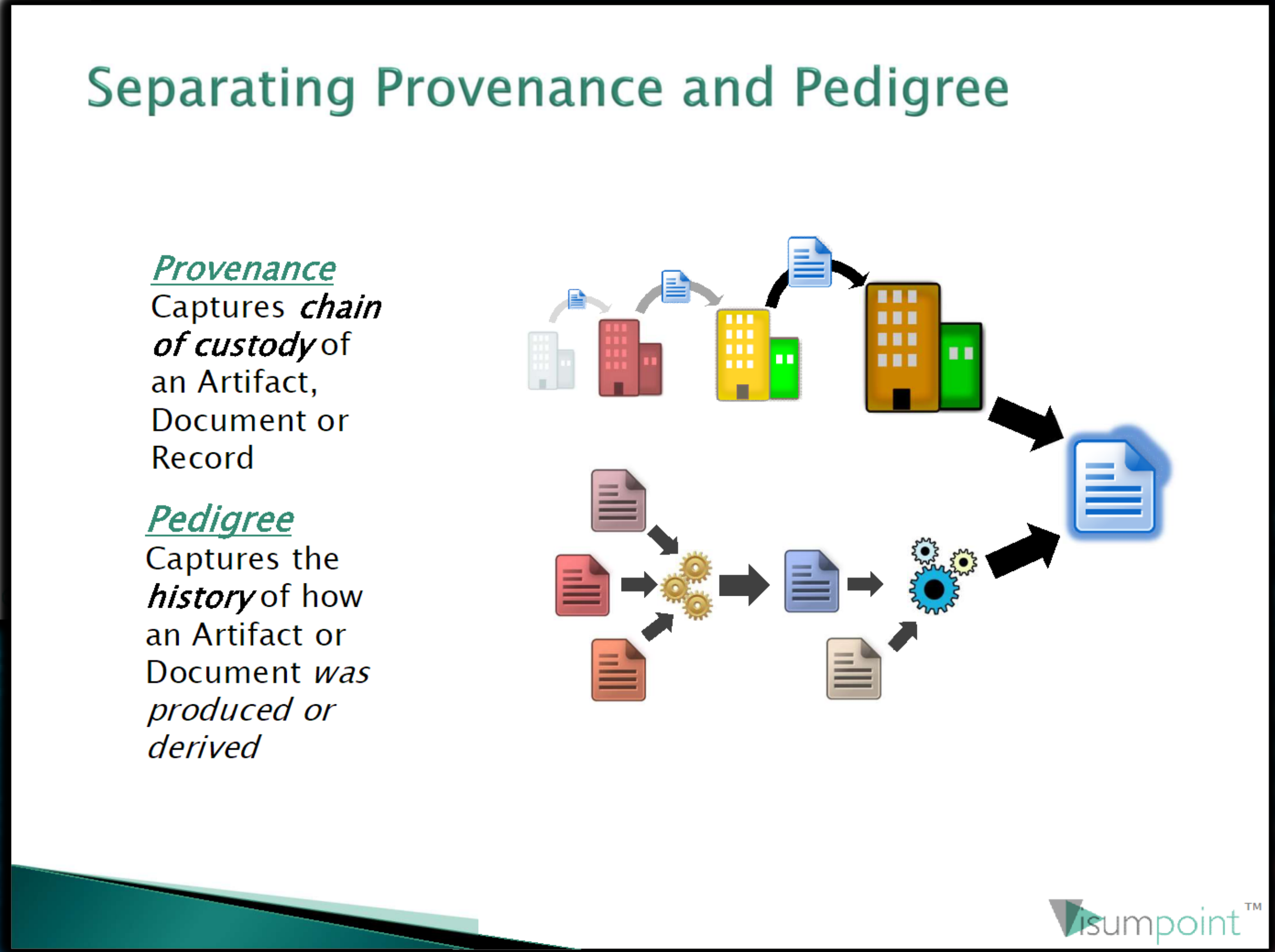
- *Provenance**
  1. The origin, or source of something
  2. The history of ownership of a valued object, or work of art, or literature
- *Pedigree**
  1. A register recording a line of ancestors
  2. An ancestral line : lineage
     The origin and the history of something; broadly : background, history

CONFUSION

- *Many use "Provenance" for both meanings.*

  The provenance of a piece of data is both the custodianship as well as the lineage of processing and/or derivation that led to the piece of data.

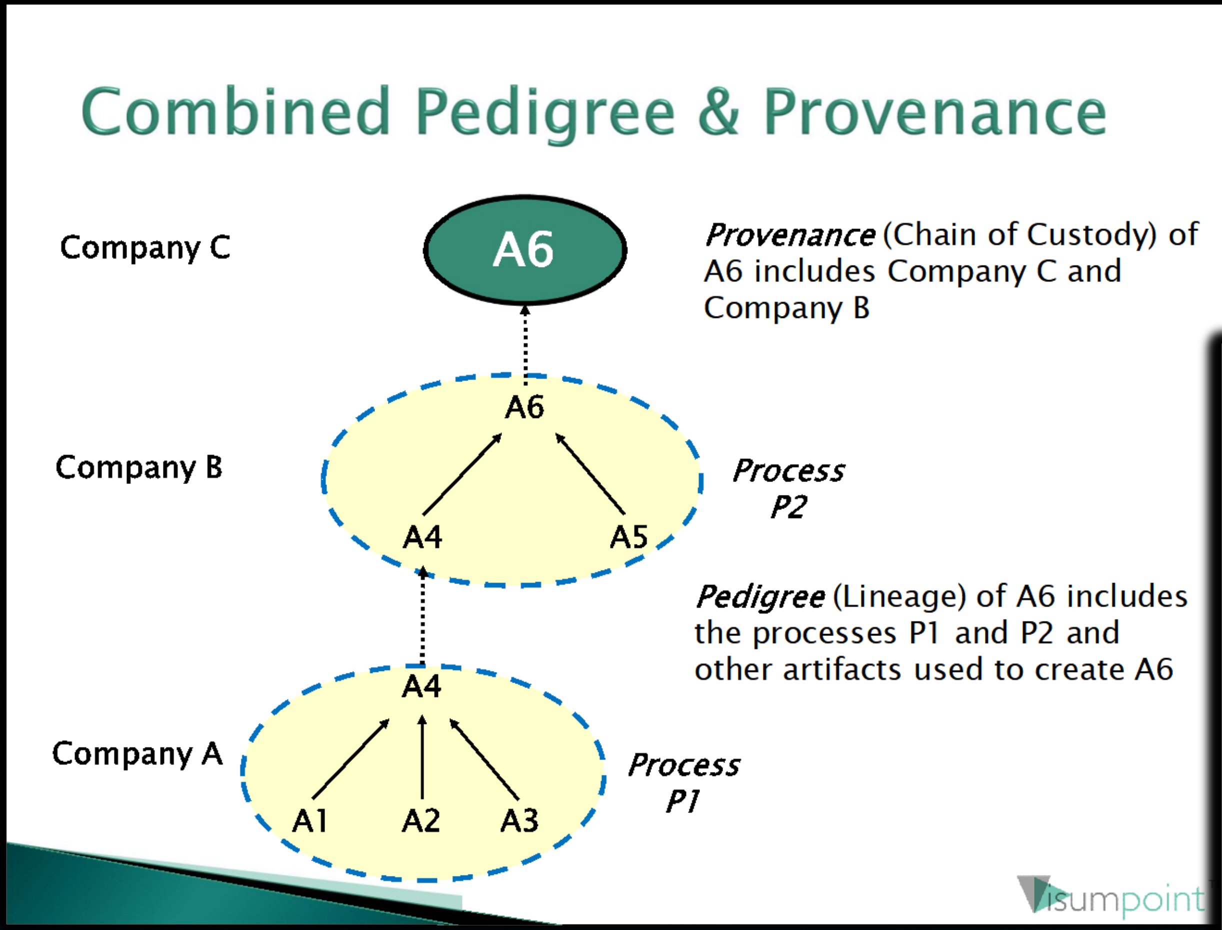
*Definitions (from Merriam-Webster.com)



Separating Provenance and Pedigree

Provenance
Captures ***chain of custody*** of an Artifact, Document or Record

Pedigree
Captures the ***history*** of how an Artifact or Document *was produced or derived*



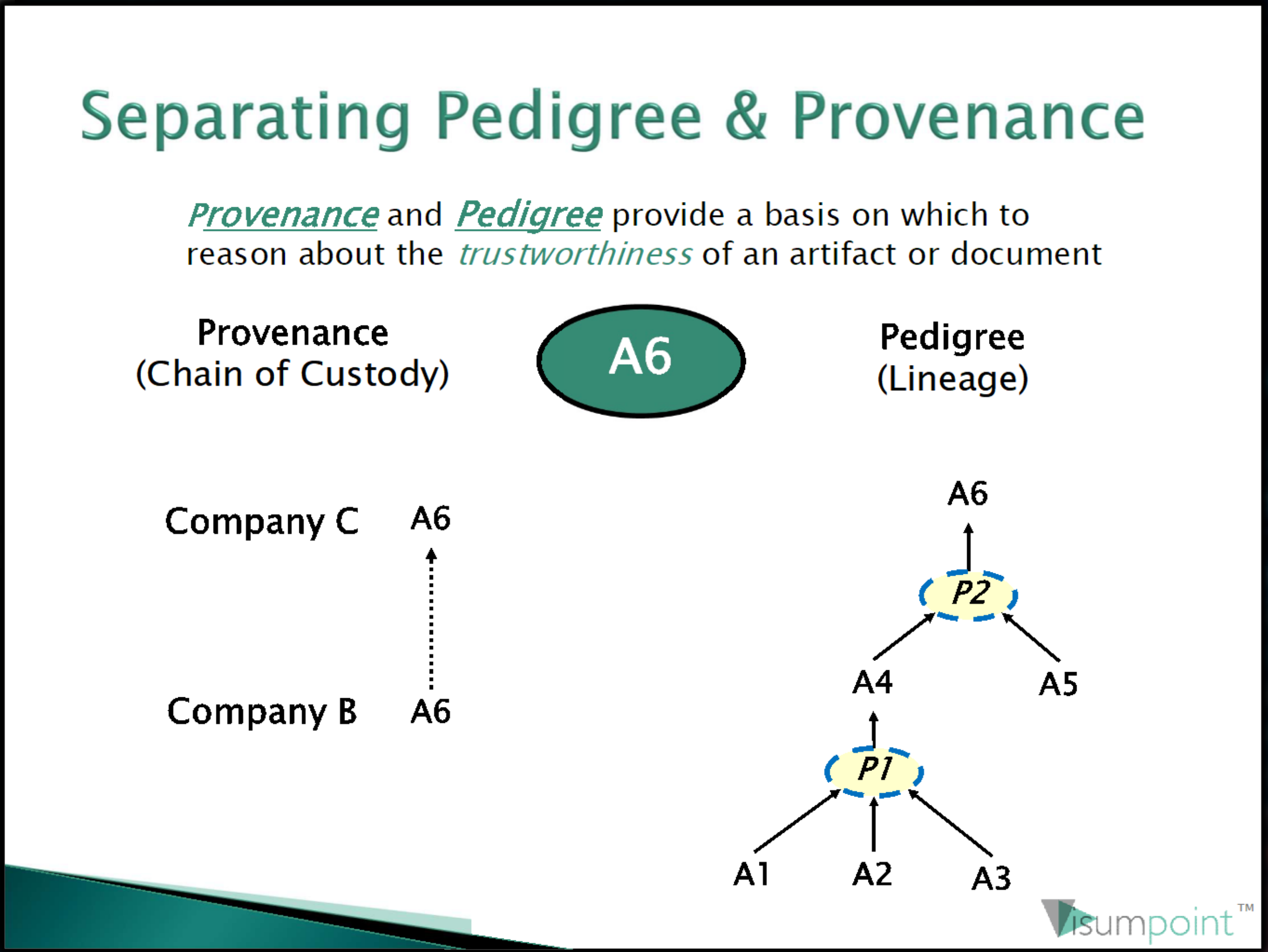
Combined Pedigree & Provenance

Provenance (Chain of Custody) of A6 includes Company C and Company B

Process P2

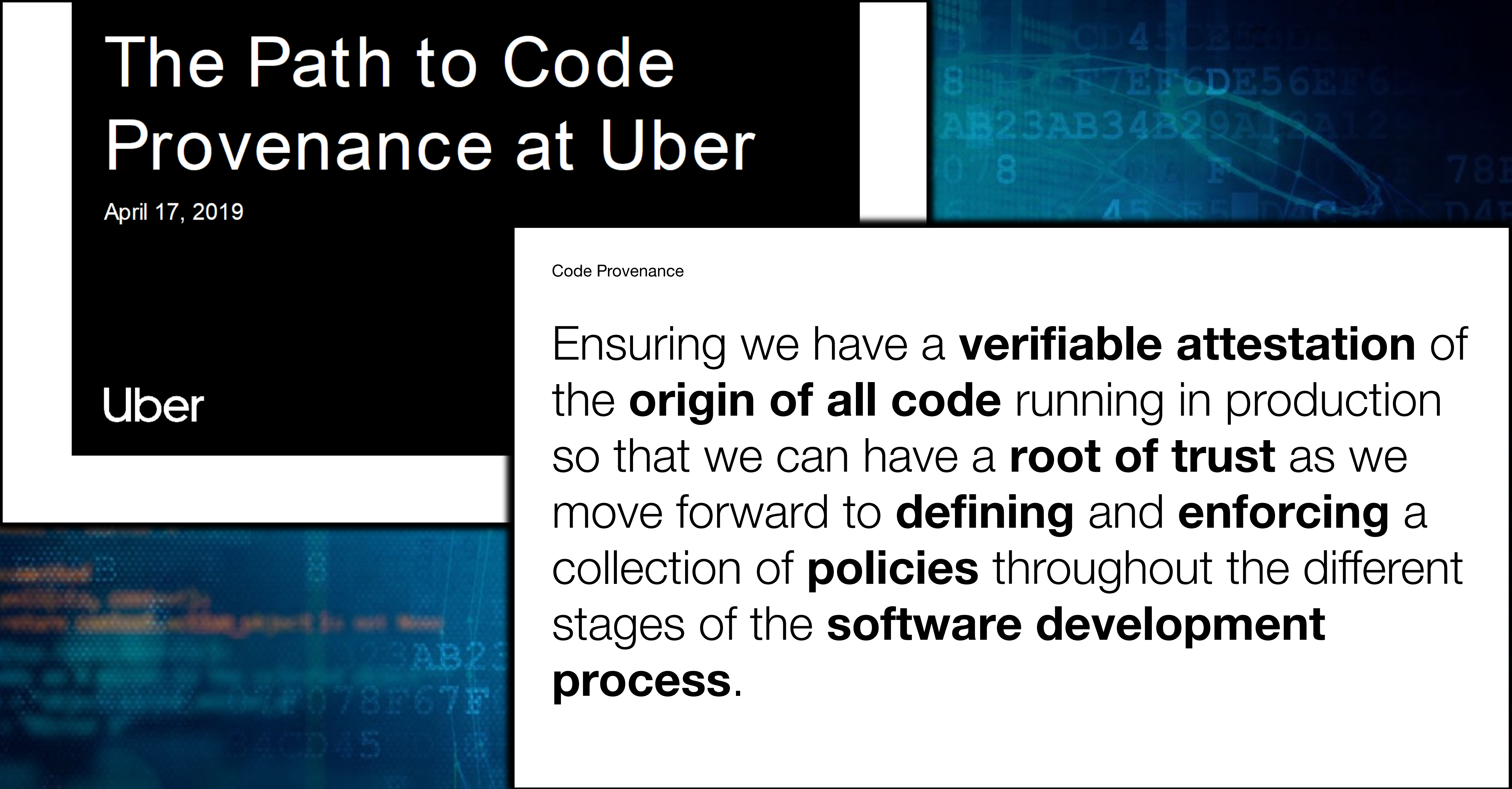
Pedigree (Lineage) of A6 includes the processes P1 and P2 and other artifacts used to create A6

Process P1



Separating Pedigree & Provenance

Provenance and *Pedigree* provide a basis on which to reason about the *trustworthiness* of an artifact or document



The Path to Code Provenance at Uber

April 17, 2019

Uber

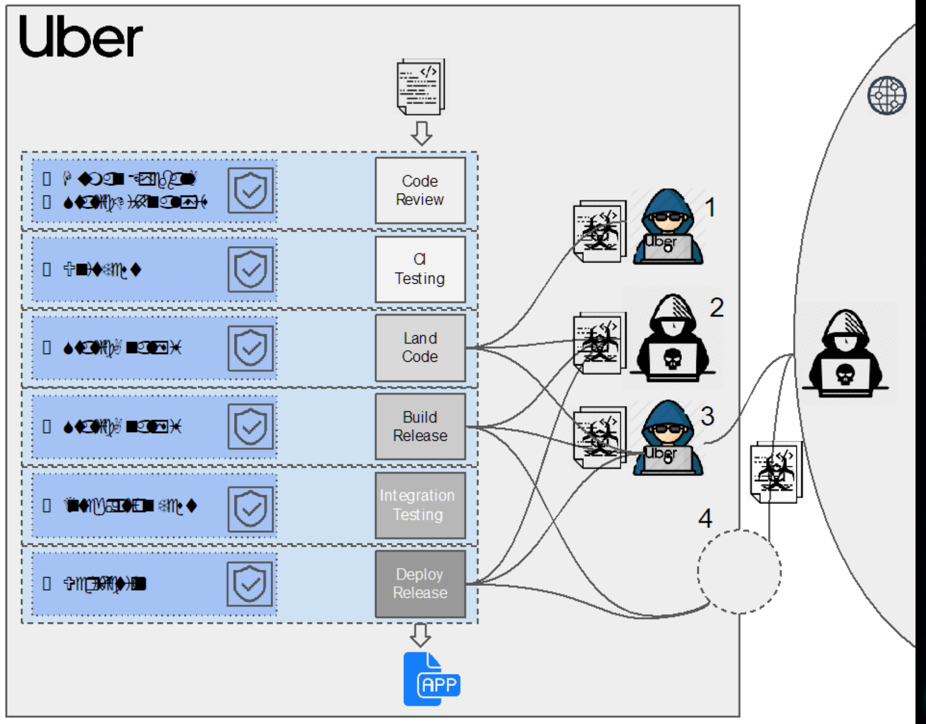
Code Provenance

Ensuring we have a **verifiable attestation** of the **origin of all code** running in production so that we can have a **root of trust** as we move forward to **defining** and **enforcing** a collection of **policies** throughout the different stages of the **software development process**.

Code Provenance

What are we protecting against?

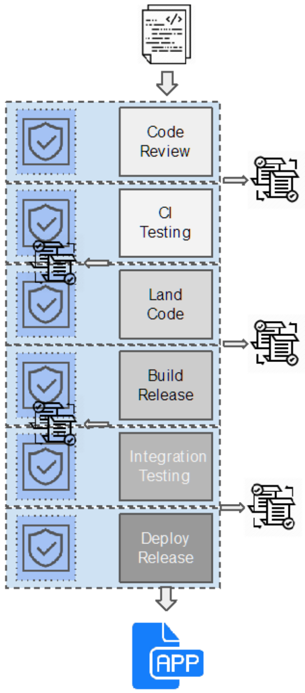
- [illegible]
- [illegible]
- [illegible]
- [illegible]



Code Provenance

What do we get out of all this?

- "Chain of custody" for all code landing in production releases
- Enabling response in the event that anything goes awry
- Flexible, enforced policies for what code is allowed to land in production releases



Use Cases for sBOM

Refer-Acquire-Transfer
(definition of what it is)

Pedigree
(history of how it was produced)

License Management
(conditions about its use)

Provenance
(chain of custody of it)

Integrity
(cryptographic basis of unalteredness)

Market Transparency
(public DBs of components)

Formulation
(how it was compiled/formed)

rance
thiness of it)

f a Service
elivering service)

Currency
re applied to it)

d Response
sing/action)

Mket Transp
f components)

Tools and environments for:

- Web apps
- Cloud apps
- Orchestration of co

Software Composition Analysis Tools

Bo

ers

Spanning Tiers

sBOM

Development Tools

Developer

Customer

Deployed S/W Tools

Contract/Agreement

MITRE

Potential sBOM elements

Refer-Acquire-Transfer
(definition of what it is)

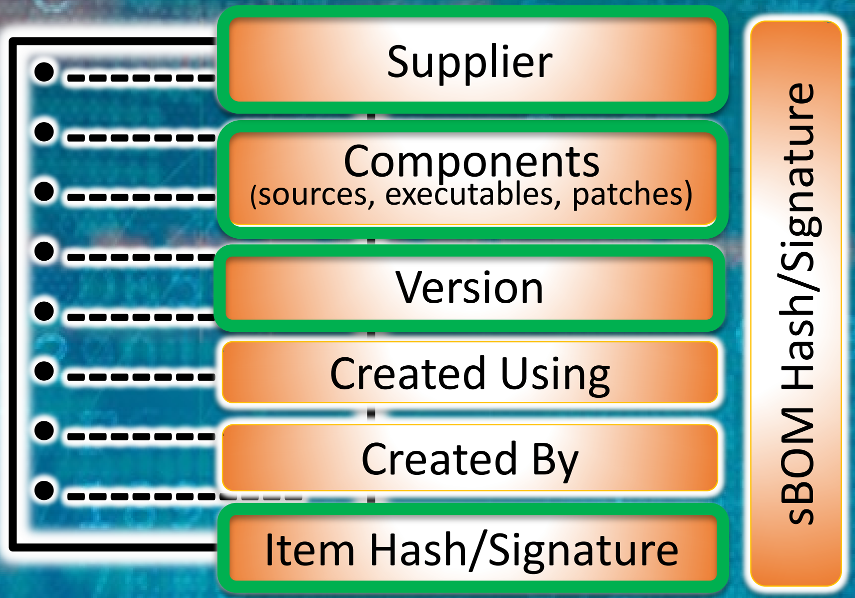
Pedigree
(history of how it was produced)

License Management
(conditions about its use)

Provenance
(chain of custody of it)

Integrity
(cryptographic basis of unalteredness)

Market Transparency
(public DBs of components)



Formulation
(how it was compiled/formed)

Assurance
(trustworthiness of it)

sBOM of a Service
(sBOM of sw delivering service)

Patch Currency
(known fixes are applied to it)

Automated Response
(sBOM parsing/action)

Assured Mket Transp
(public DBs of components)

Between Tiers

Spanning Tiers

sBOM

Development Tools

Developer → Customer ↔

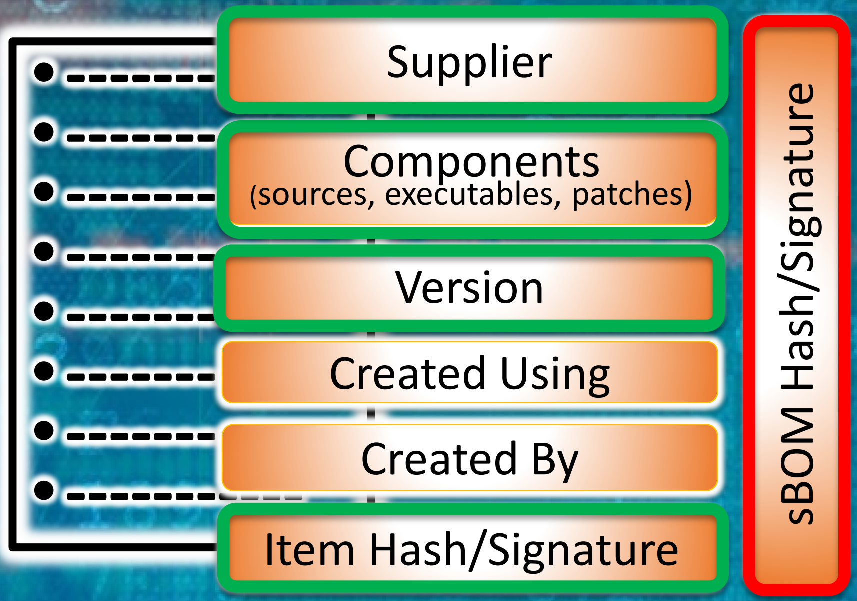
Deployed S/W Tools

Contract/Agreement

MITRE

Potential sBOM elements

- **Refer-Acquire-Transfer** (definition of what it is)
- **Pedigree** (history of how it was produced)
- **License Management** (conditions about its use)
- **Provenance** (chain of custody of it)
- **Integrity** (cryptographic basis of unalteredness)
- **Market Transparency** (public DBs of components)


Supplier

Components (sources, executables, patches)

Version

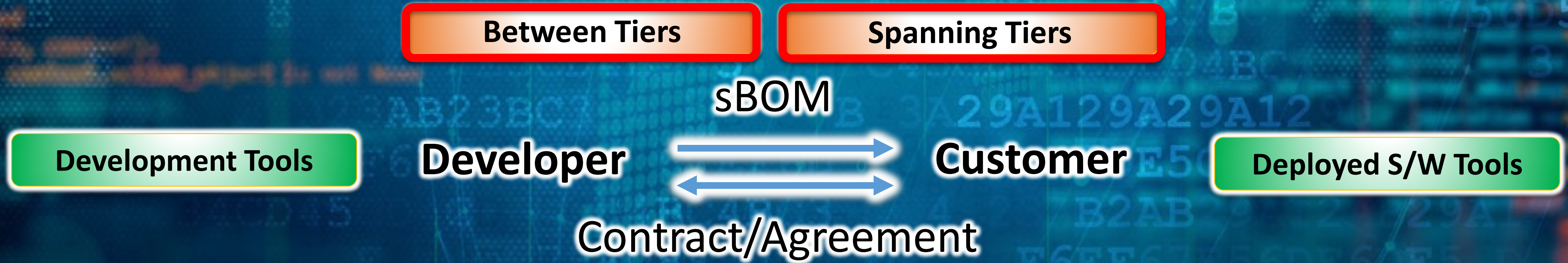
Created Using

Created By

Item Hash/Signature

sBOM Hash/Signature

- **Formulation** (how it was compiled/formed)
- **Assurance** (trustworthiness of it)
- **sBOM of a Service** (sBOM of sw delivering service)
- **Patch Currency** (known fixes are applied to it)
- **Automated Response** (sBOM parsing/action)
- **Assured Mket Transp** (public DBs of components)


Between Tiers

Spanning Tiers

Development Tools

sBOM

Developer

Customer

Contract/Agreement

Deployed S/W Tools

Potential sBOM elements

Refer-Acquire-Transfer (definition of what it is)

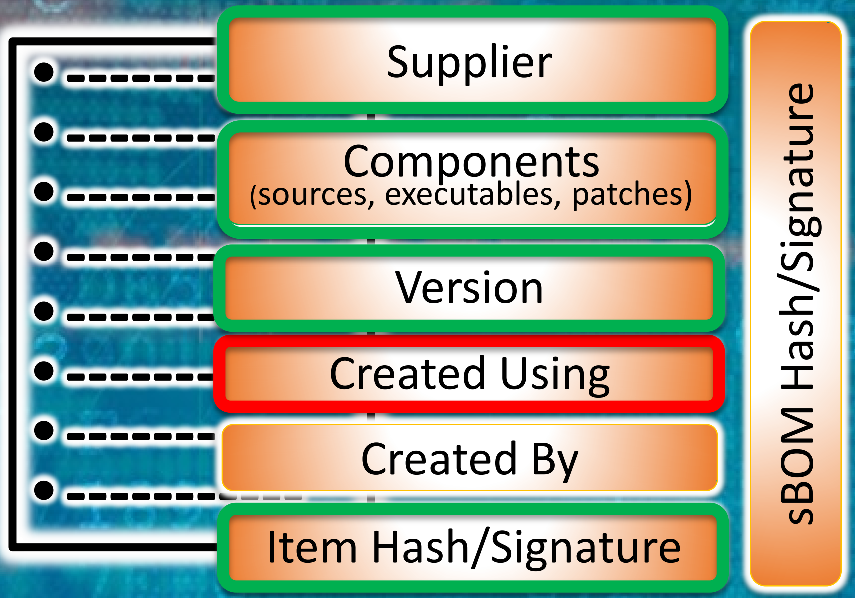
Pedigree (history of how it was produced)

License Management (conditions about its use)

Provenance (chain of custody of it)

Integrity (cryptographic basis of unalteredness)

Market Transparency (public DBs of components)



Supplier

Components (sources, executables, patches)

Version

Created Using

Created By

Item Hash/Signature

sBOM Hash/Signature

Formulation (how it was compiled/formed)

Assurance (trustworthiness of it)

sBOM of a Service (sBOM of sw delivering service)

Patch Currency (known fixes are applied to it)

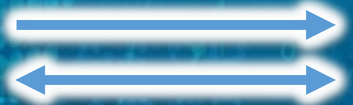
Automated Response (sBOM parsing/action)

Assured Mket Transp (public DBs of components)

Between Tiers

Spanning Tiers

Development Tools



sBOM

Developer

Customer

Contract/Agreement

Deployed S/W Tools

MITRE

Potential sBOM elements

Refer-Acquire-Transfer
(definition of what it is)

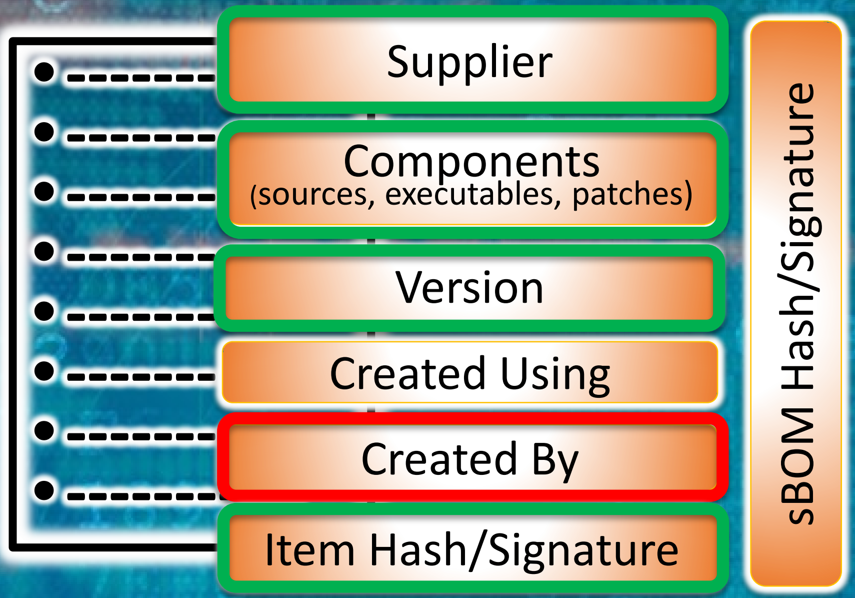
Pedigree
(history of how it was produced)

License Management
(conditions about its use)

Provenance
(chain of custody of it)

Integrity
(cryptographic basis of unalteredness)

Market Transparency
(public DBs of components)



Supplier

Components
(sources, executables, patches)

Version

Created Using

Created By

Item Hash/Signature

sBOM Hash/Signature

Formulation
(how it was compiled/formed)

Assurance
(trustworthiness of it)

sBOM of a Service
(sBOM of sw delivering service)

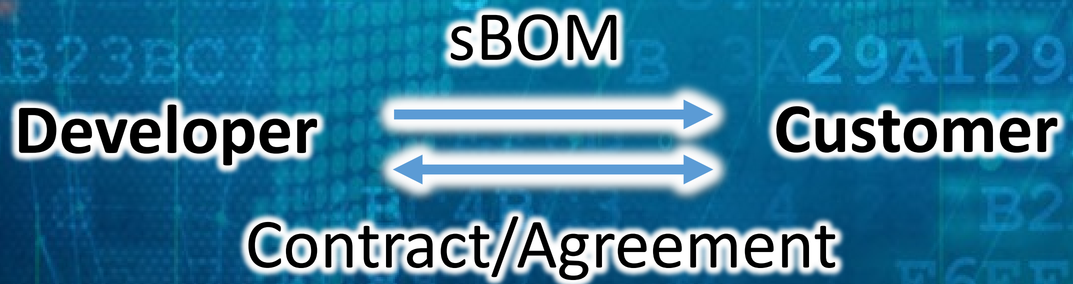
Patch Currency
(known fixes are applied to it)

Automated Response
(sBOM parsing/action)

Assured Mket Transp
(public DBs of components)

Between Tiers

Spanning Tiers



sBOM

Developer

Customer

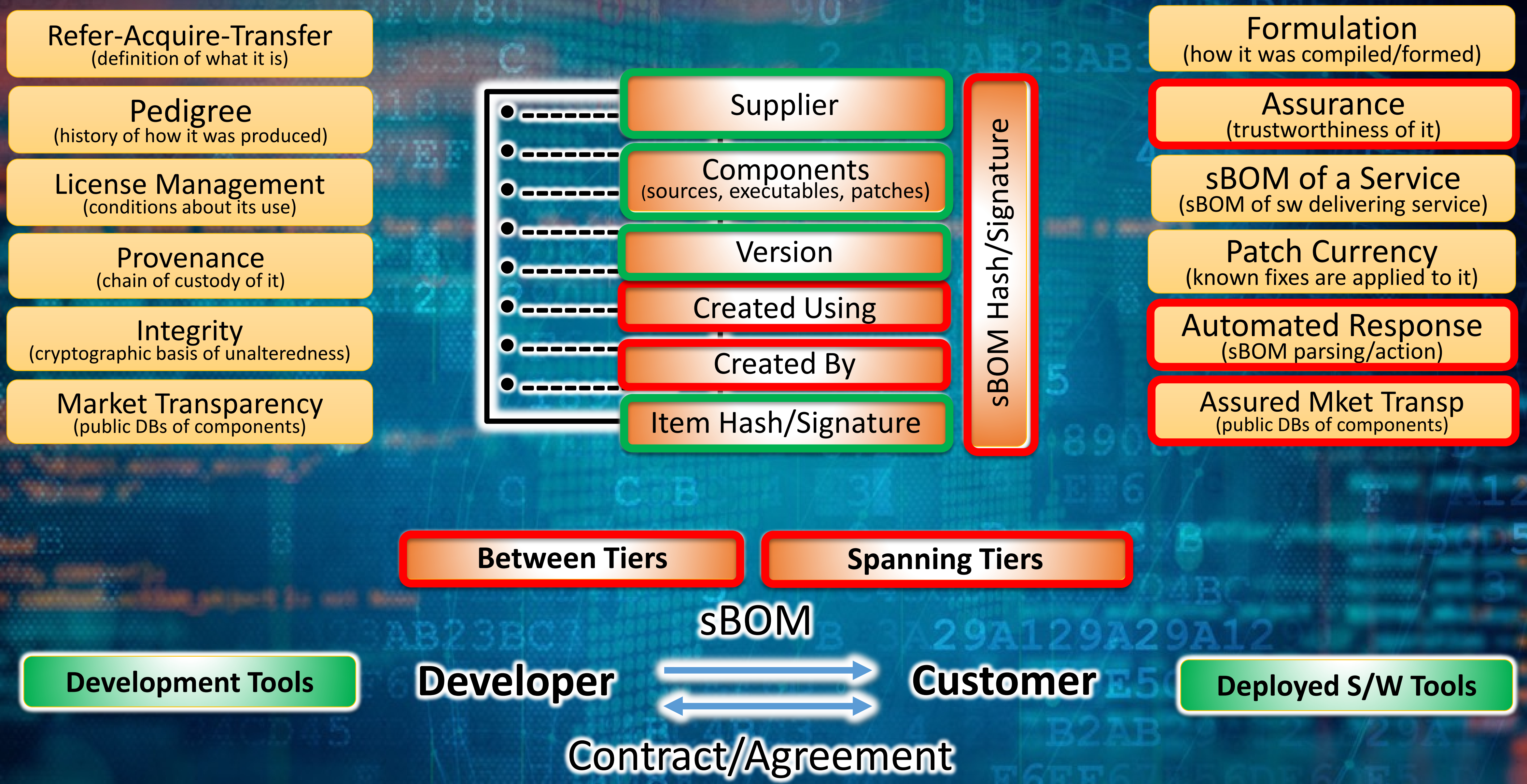
Contract/Agreement

Development Tools

Deployed S/W Tools

MITRE

Potential sBOM elements



Refer-Acquire-Transfer
(definition of what it is)

Pedigree
(history of how it was produced)

License Management
(conditions about its use)

Provenance
(chain of custody of it)

Integrity
(cryptographic basis of unalteredness)

Market Transparency
(public DBs of components)

Supplier

Components
(sources, executables, patches)

Version

Created Using

Created By

Item Hash/Signature

sBOM Hash/Signature

Formulation
(how it was compiled/formed)

Assurance
(trustworthiness of it)

sBOM of a Service
(sBOM of sw delivering service)

Patch Currency
(known fixes are applied to it)

Automated Response
(sBOM parsing/action)

Assured Mket Transp
(public DBs of components)

Between Tiers

Spanning Tiers

sBOM

Development Tools

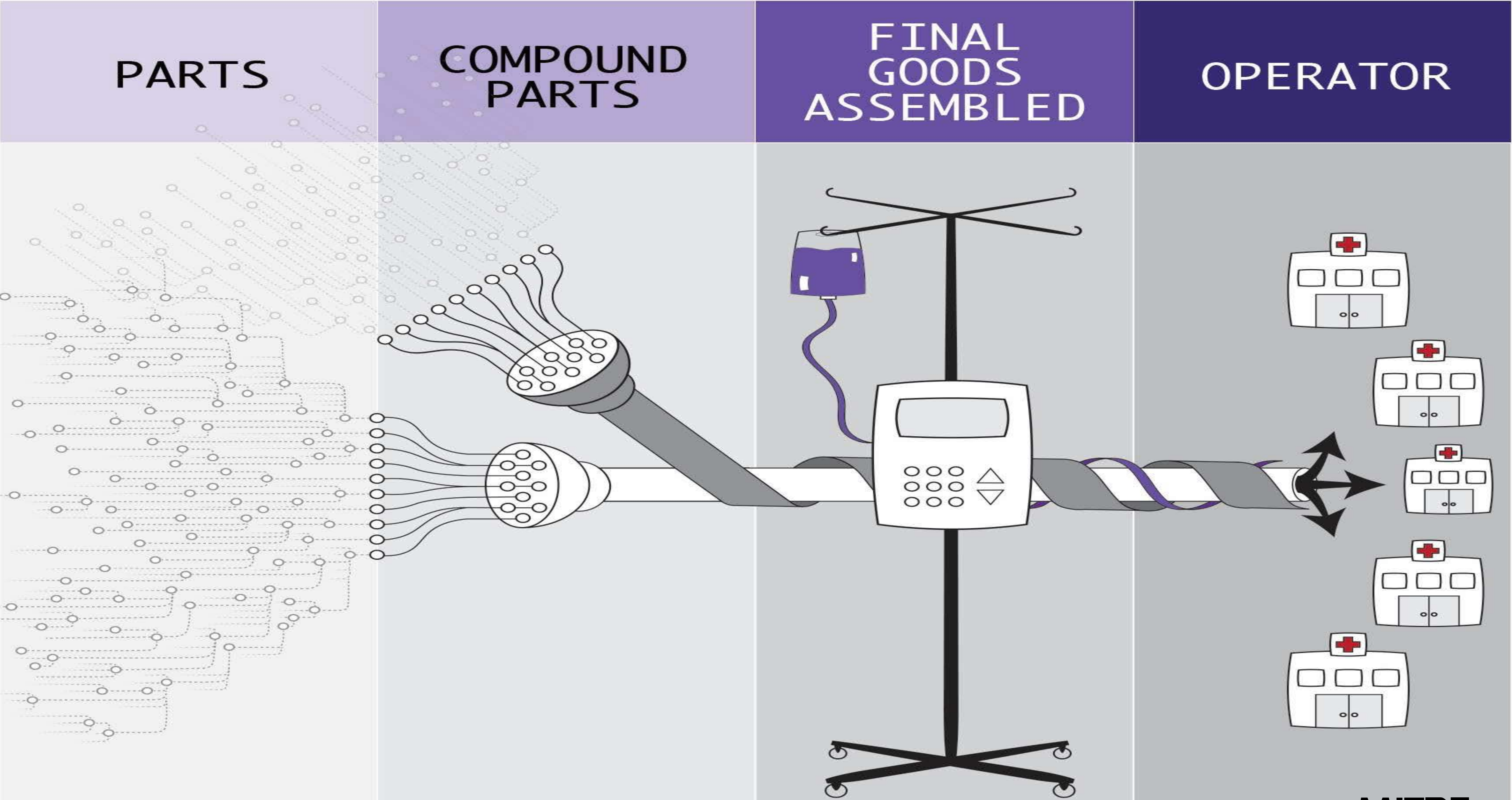
Developer

Customer

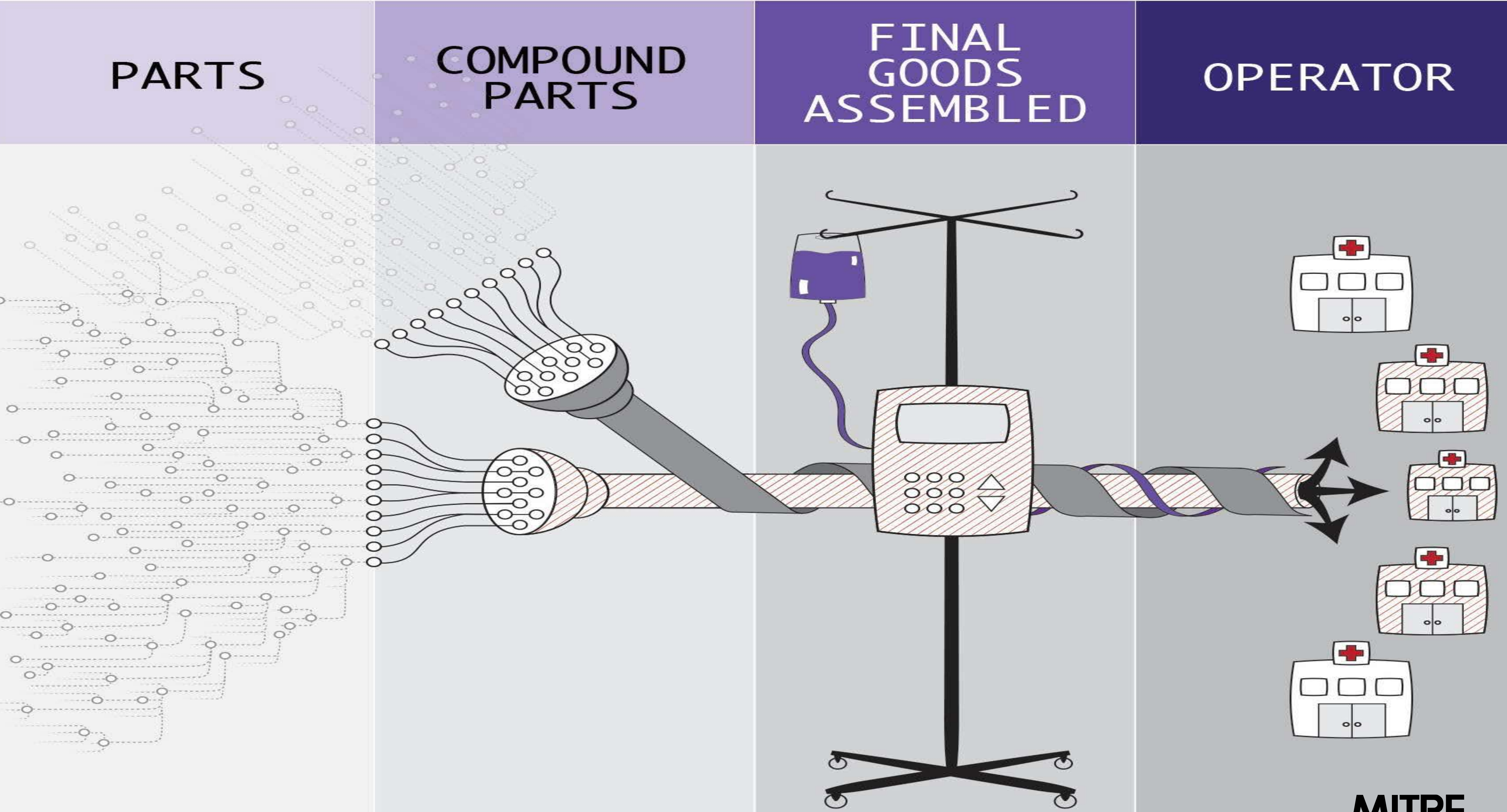
Deployed S/W Tools

Contract/Agreement

MITRE



MITRE



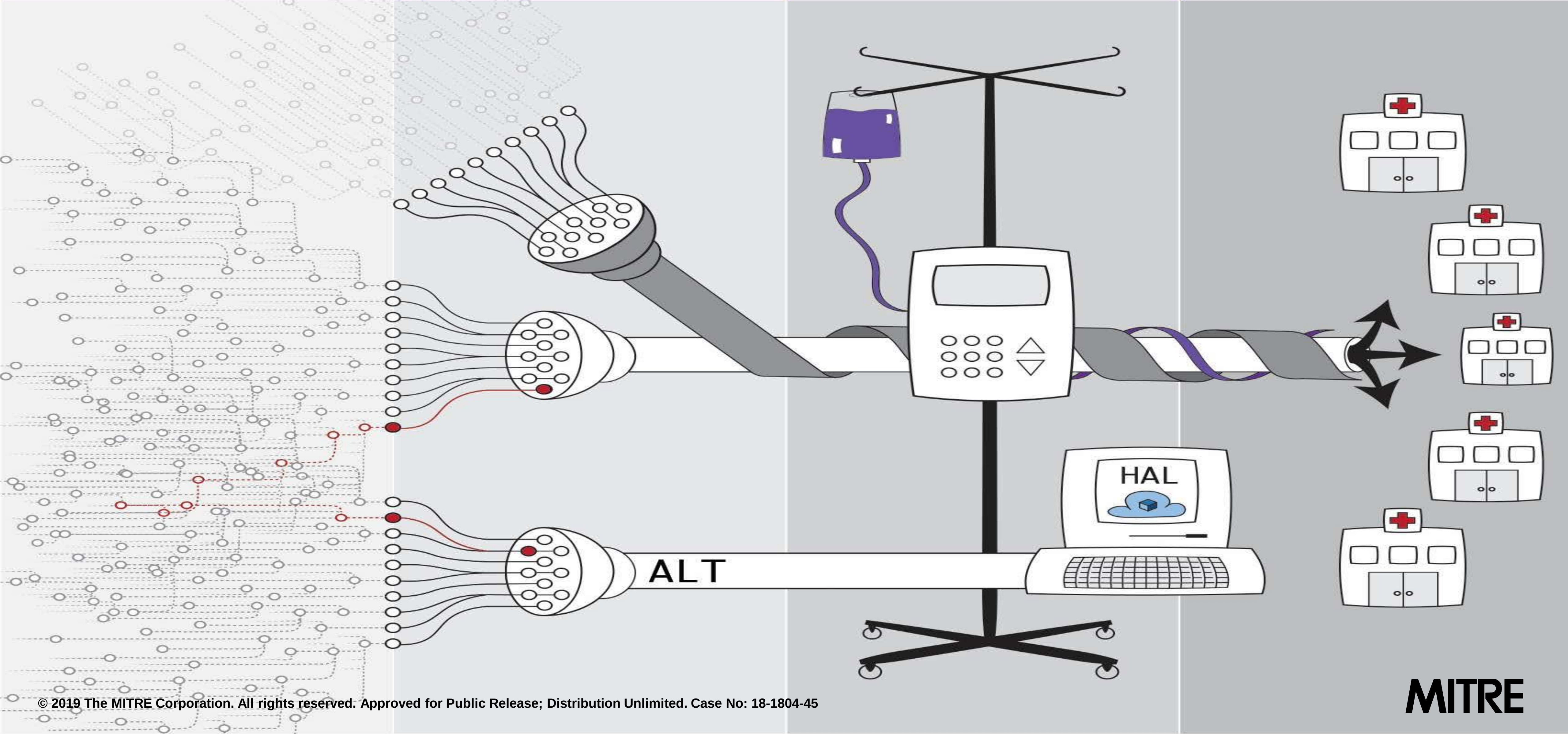
PARTS

COMPOUND PARTS

FINAL GOODS ASSEMBLED

OPERATOR

PARTS	COMPOUND PARTS	FINAL GOODS ASSEMBLED	OPERATOR



MITRE

PARTS

COMPOUND PARTS

FINAL GOODS ASSEMBLED

OPERATOR

ENTERPRISE

MEDICAL

FINANCIAL SERVICES

INDUSTRIAL

$OTHER

All types of Capabilities are becoming Software-Enabled…

Medical

Vehicles

Buildings

Temperature, Humidity,CO2

Motion Sensor

AC,Chiller

Electric power

Elevator

Entrance gate

Aeronautics

Manufacturing

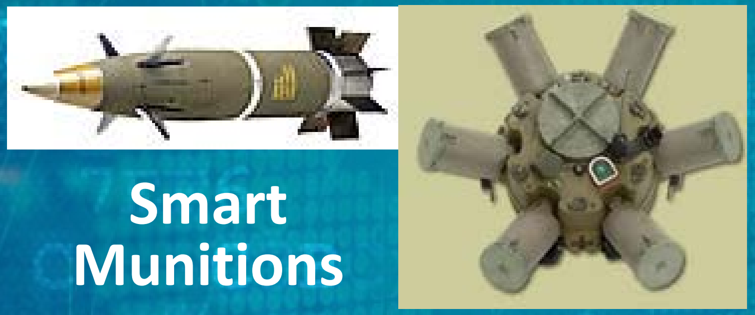
Energy

Shipping

These Changes Go Well beyond Traditional Information Technology…

Water Treatment

Status & Health Monitoring



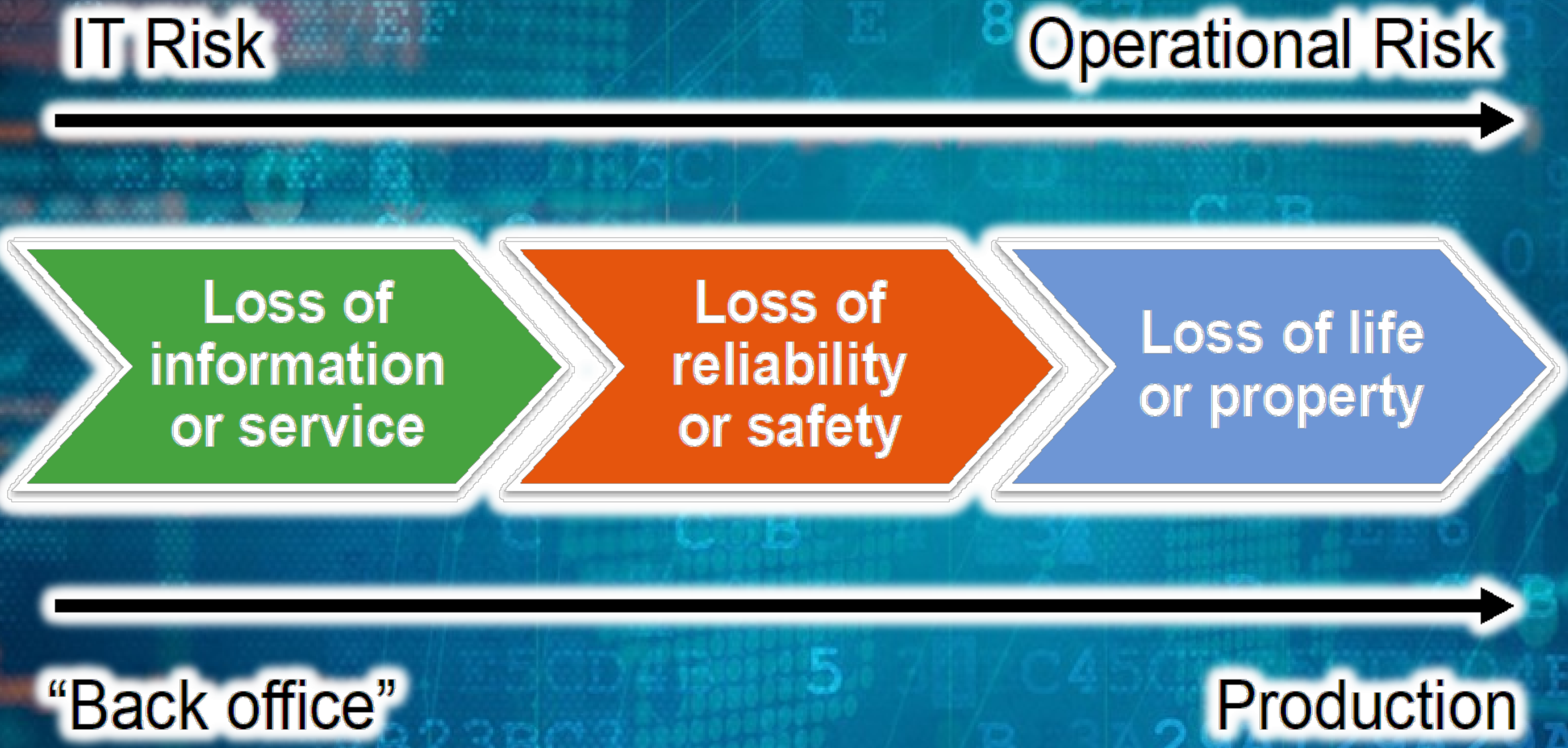
Smart Munitions

Remote Management

Oil & Gas

Hydro Power & Dam Mngt

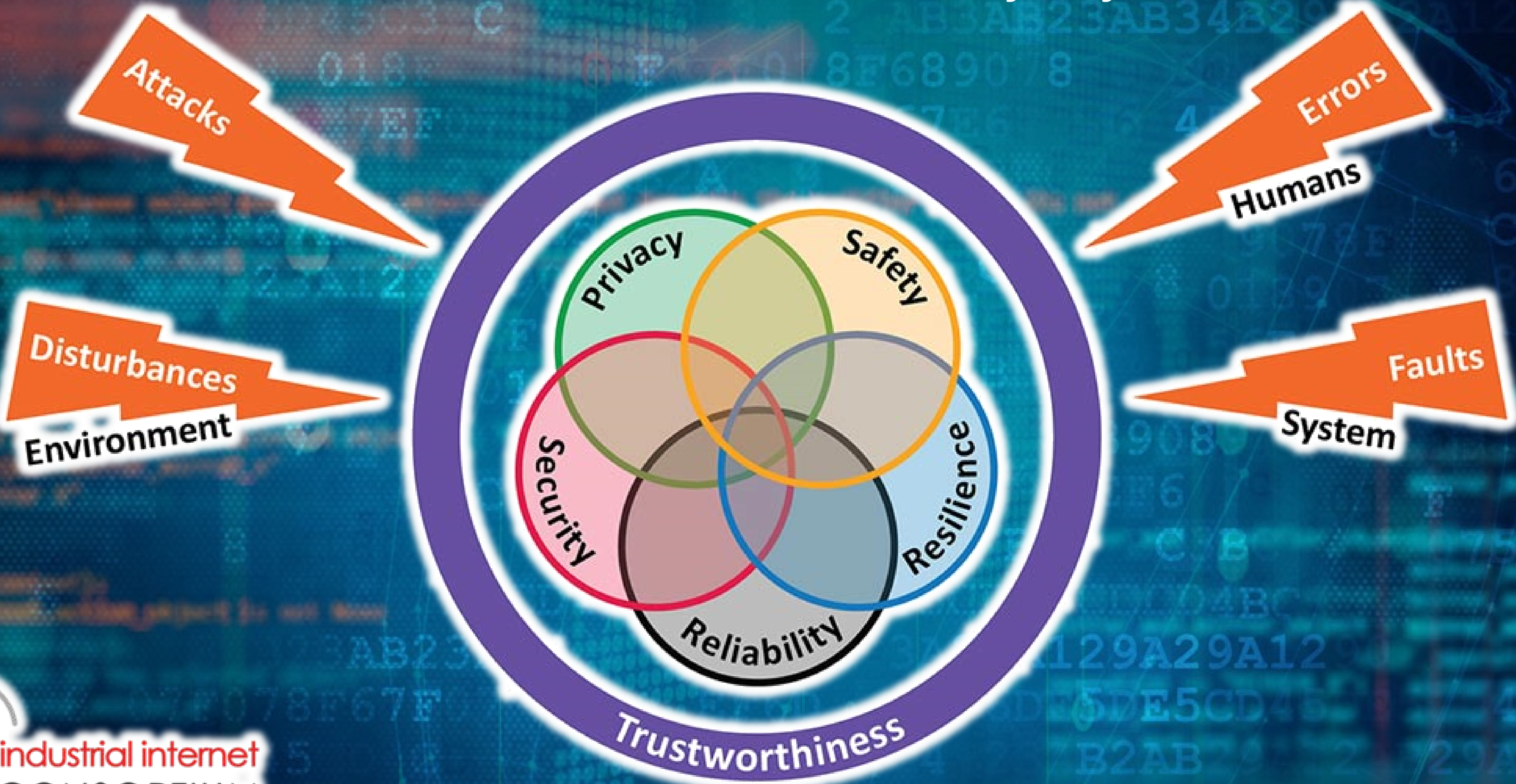
Need Secure, Safe, Reliable, and Resilient Behavior that Upholds Privacy Expectations



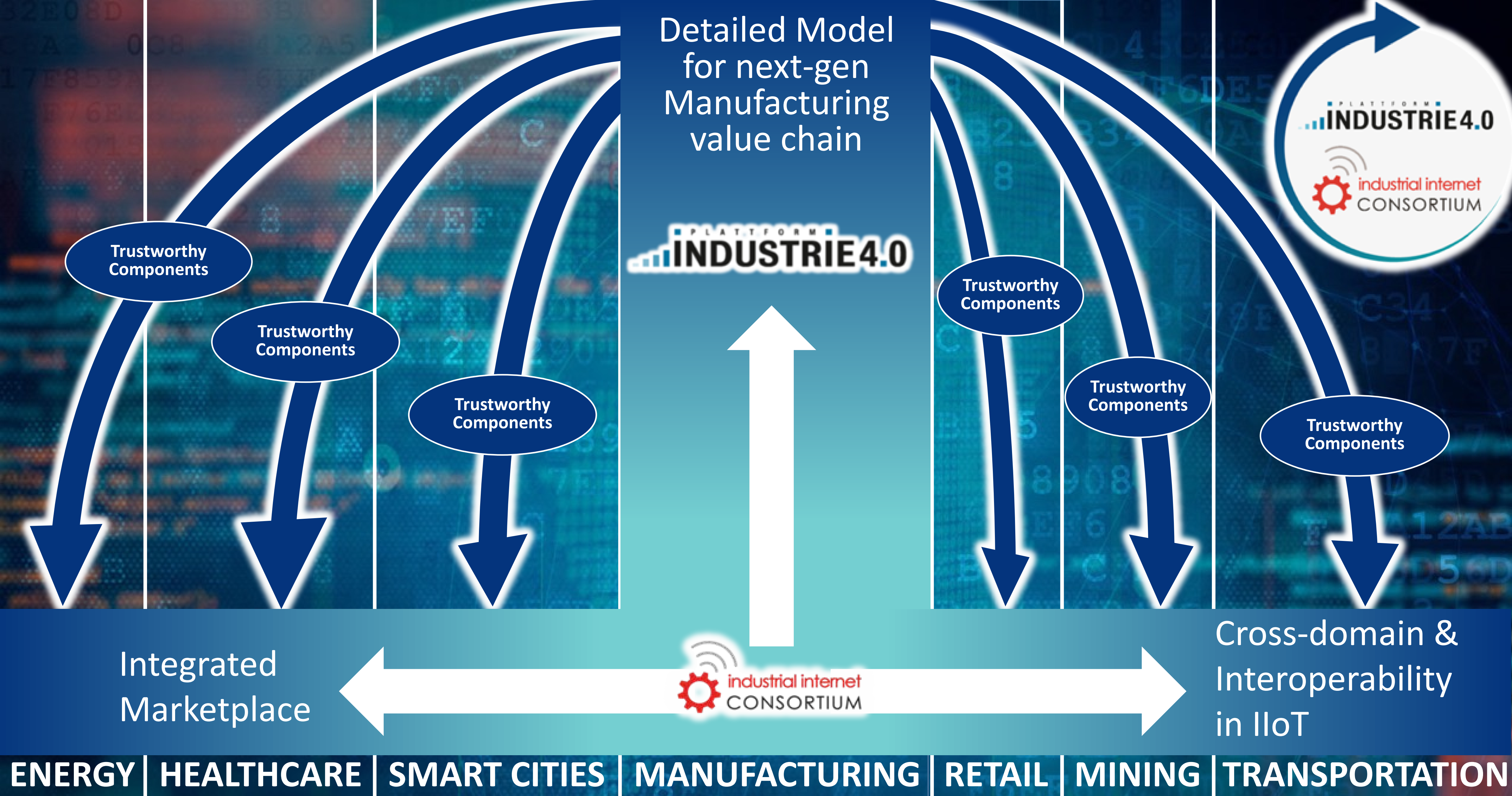
Gartner.

MITRE

– Need Assurance of More Than Security –
Need Assured Trustworthy Systems



industrial internet CONSORTIUM



Detailed Model for next-gen Manufacturing value chain

PLATTFORM INDUSTRIE4.0

industrial internet CONSORTIUM

Trustworthy Components

Trustworthy Components

Trustworthy Components

Trustworthy Components

Trustworthy Components

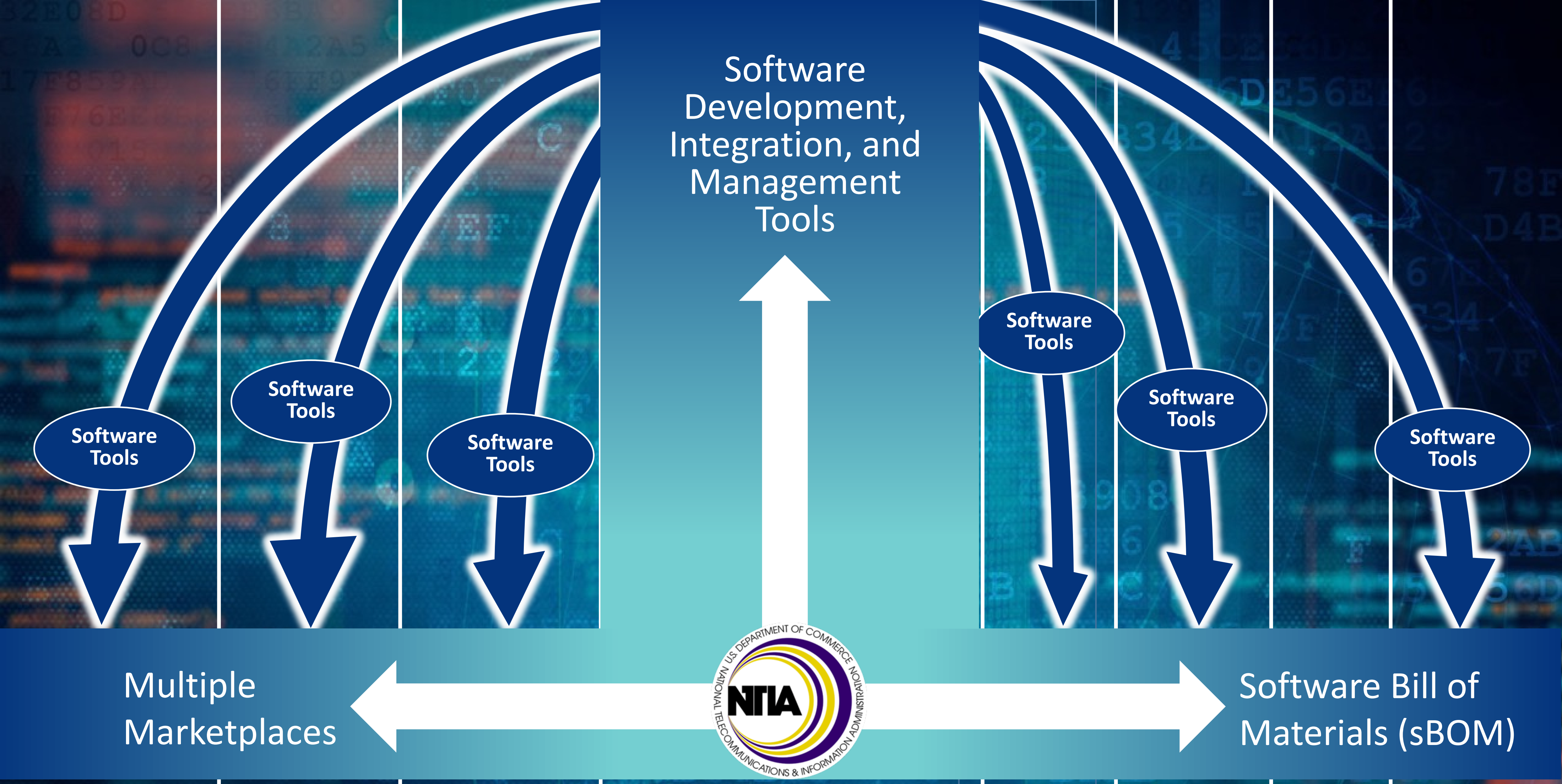
Trustworthy Components

Integrated Marketplace

Cross-domain & Interoperability in IIoT

ENERGY | HEALTHCARE | SMART CITIES | MANUFACTURING | RETAIL | MINING | TRANSPORTATION

MITRE



Software Development, Integration, and Management Tools

Software Tools

Software Tools

Software Tools

Software Tools

Software Tools

Software Tools

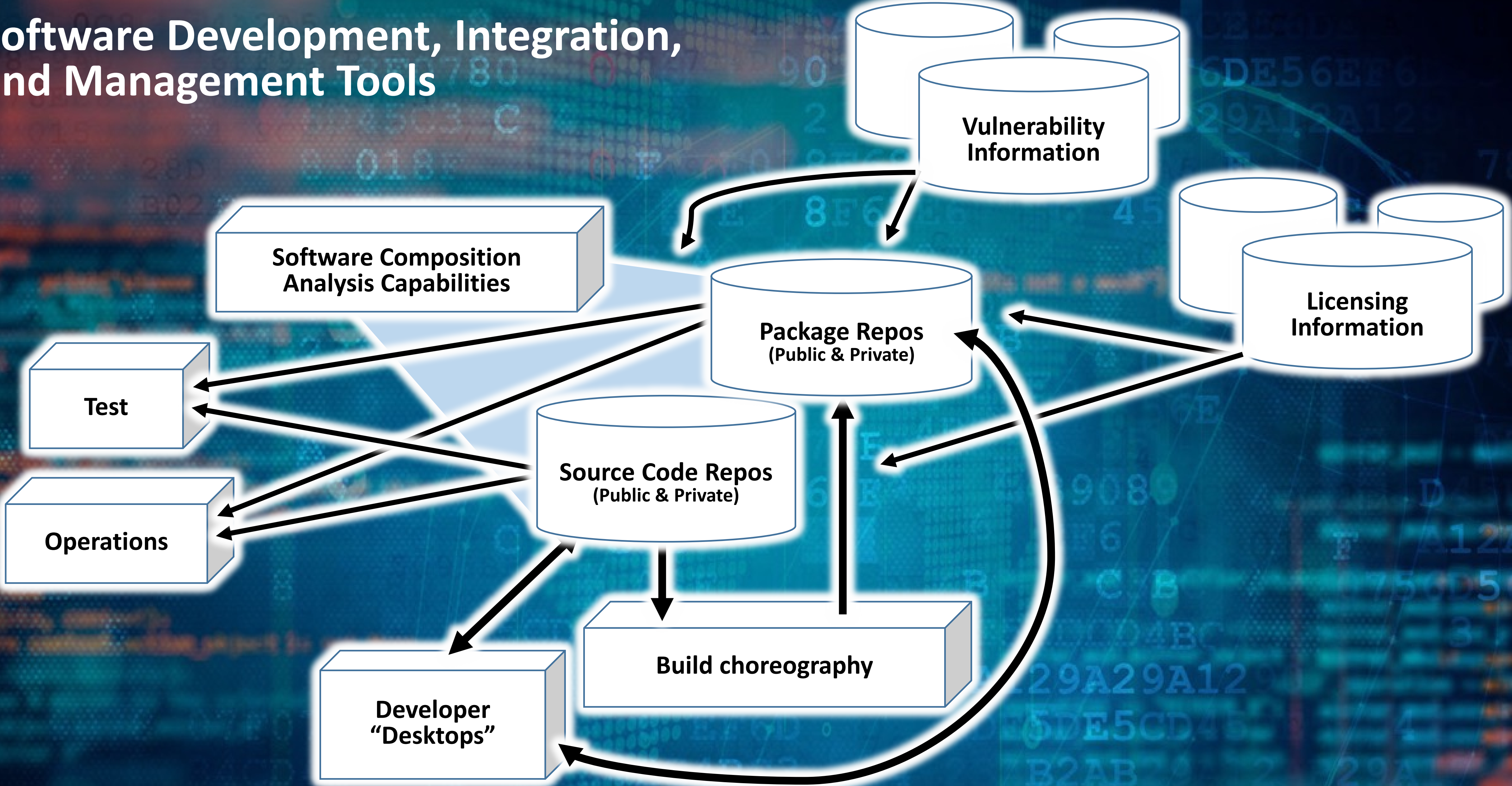
Multiple Marketplaces

Software Bill of Materials (sBOM)

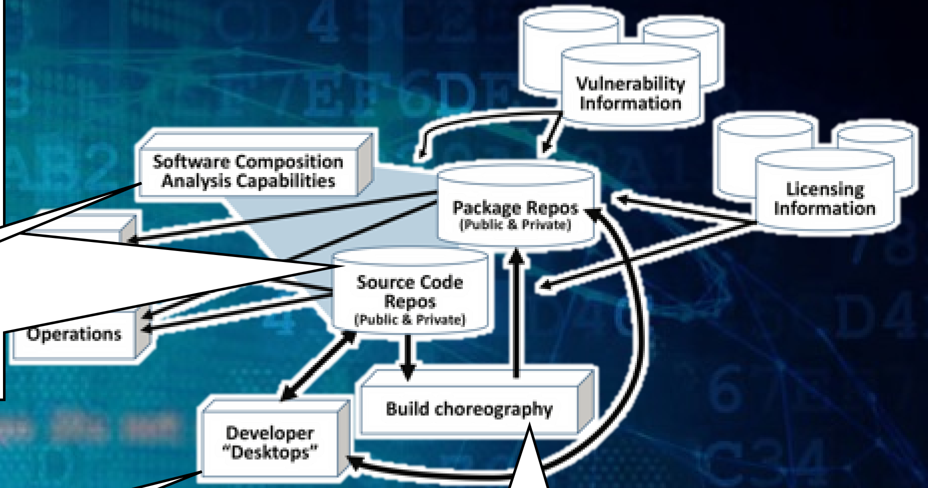
ENTERPRISE | MEDICAL | FINANCIAL | SOFTWARE INDUSTRY | RETAIL | MINING | … | ENERGY

MITRE

Software Development, Integration, and Management Tools



Vulnerability Information
Software Composition Analysis Capabilities
Package Repos (Public & Private)
Licensing Information
Test
Source Code Repos (Public & Private)
Operations
Build choreography
Developer "Desktops"


Vulnerability Information
Software Composition Analysis Capabilities
Package Repos (Public & Private)
Licensing Information
Source Code Repos (Public & Private)
Operations
Developer "Desktops"
Build choreography

Source and Package repos

docker, Unified Agent (File System Agent (FSA)), GitLab, kubernetes, SourceForge, Launchpad, CodePlex, Savannah, CCPForge
GitHub, JFrog Artifactory, JFrog Xray, inedo, Amazon ECR, Google Container Registry, Azure Container Registry, Bit Bucket, Subversion, ProjectLocker, CloudForge, Fog Creek Kiln, Codeplane, Assembla, Beanstalk, Codebase

Software Composition Analysis:

Sonatype
Black Duck (Synopsys)
WhiteSource (with plugins)
Protex, Palamida

Developer Desktops (Embedded, Web, Cloud, Desktops/Servers)

IDEs: LINX, NetBeans, Cloud9 IDE, Zend Studio, Atom, Spiralogics Application Architecture, CodeLobster, CodeCharge Studio, CodePen, Xcode, Eclipse, Android Studio, Code Blocks, BlueJ, MPLAB

Frameworks: Bootstrap, Expression Studio, HTML5 Buil
Online

Cloud Tools: Kwatee, Azure

Build Choreography

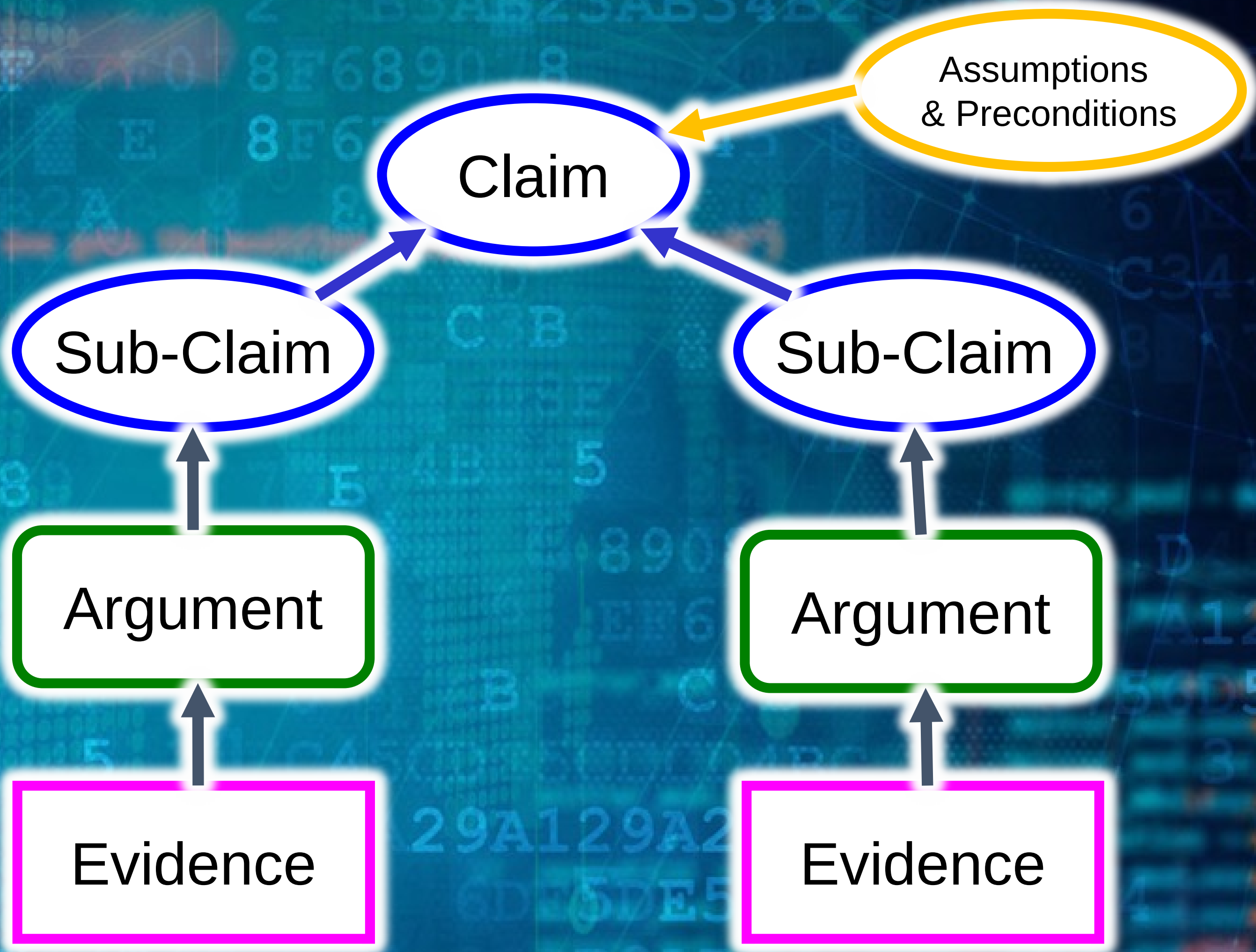
Jenkins, Travis CI, Final builder, CruiseControl, Integrity, GoCD, Urbancode, Autorabit, CircleCI, Buildkite, TeamCity, Wercker, Bitrise, Bamboo, Strider, Gitlab CI

The Basics of an Assurance Case

Claim =
assertion to be proven

Argument =
how evidence supports claim

Evidence =
required documentation

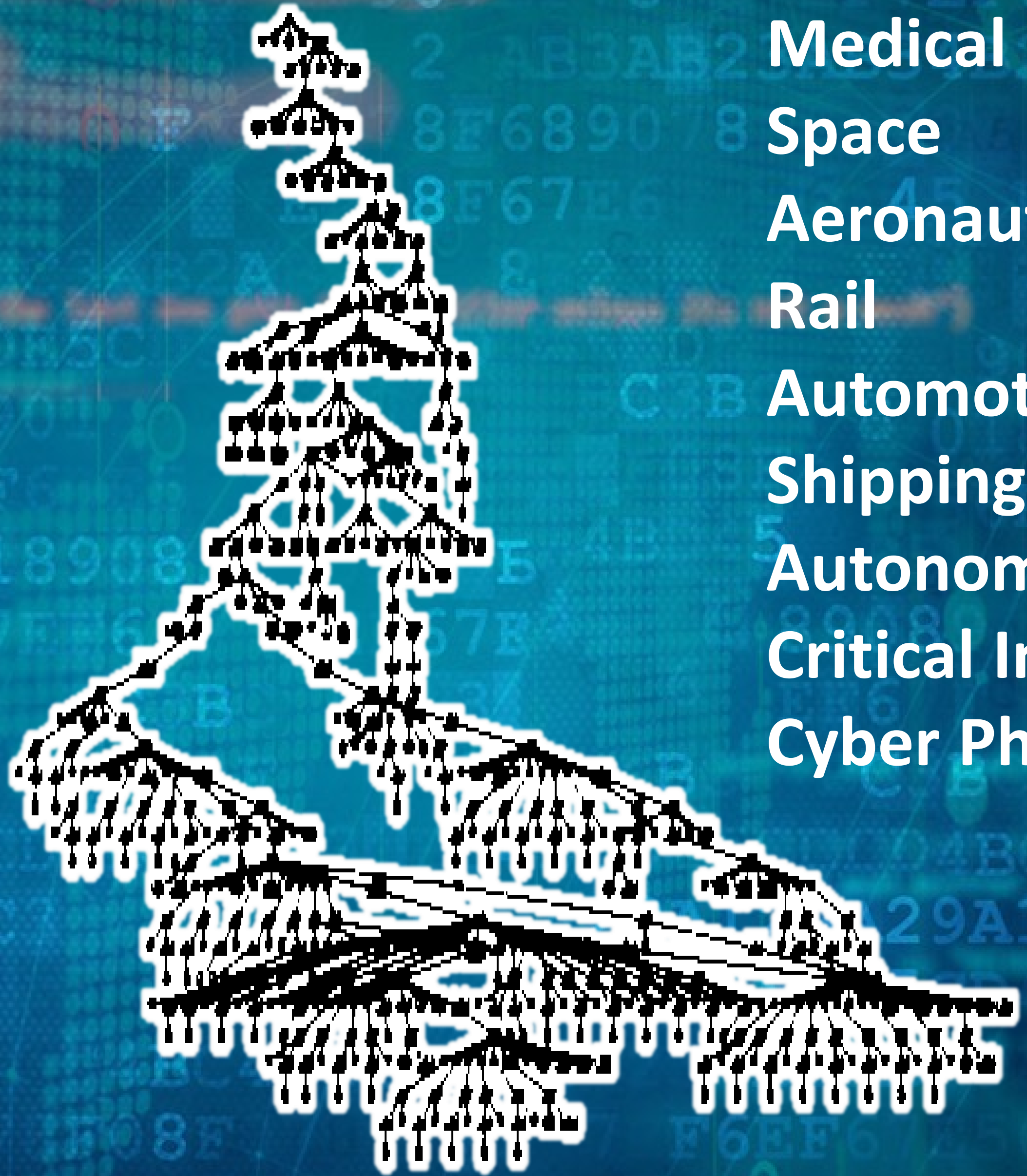


MITRE

The Assurance Case

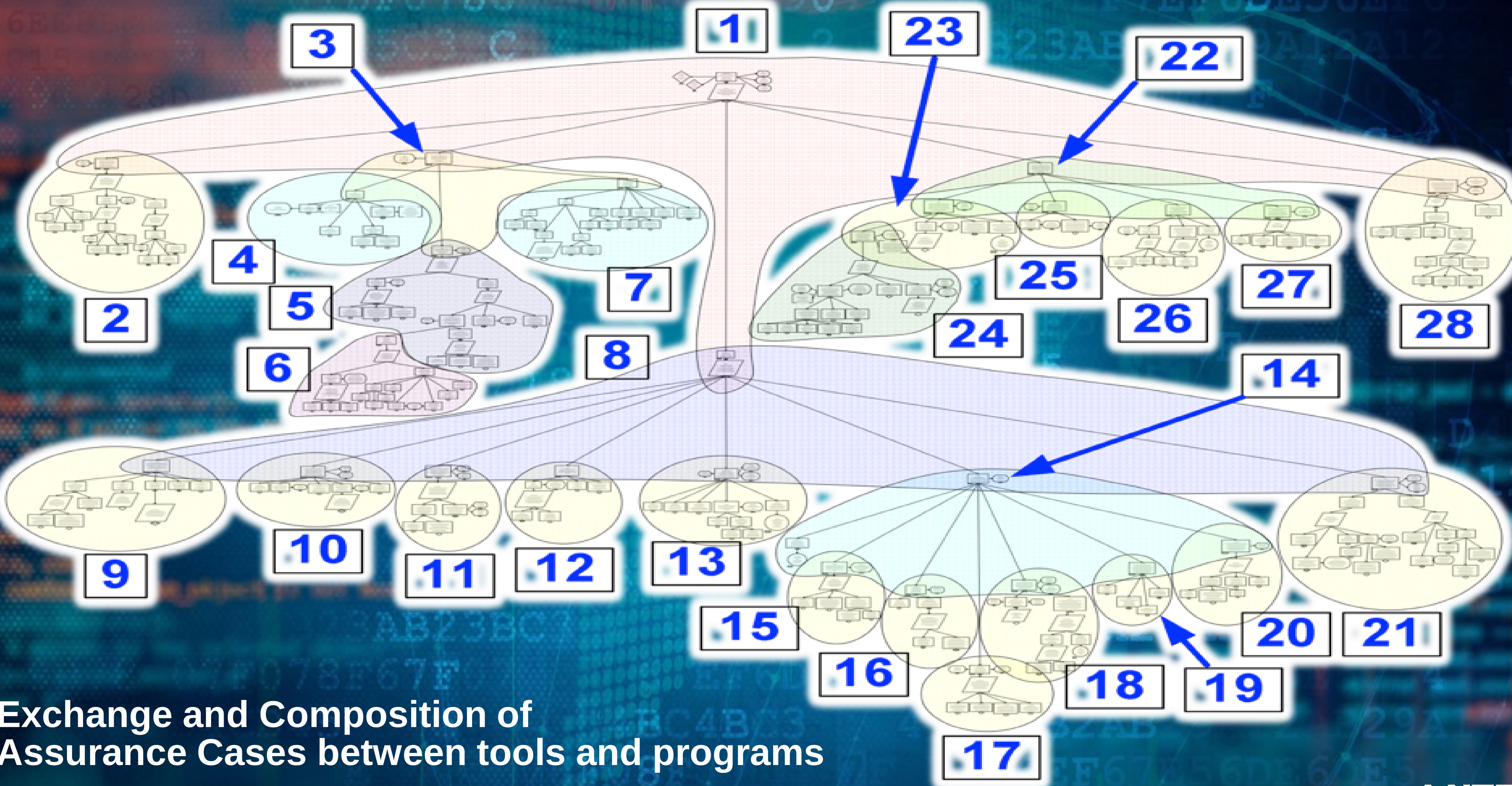
DEIS

Dependability
Engineering
Innovation for Cyber Physical
Systems



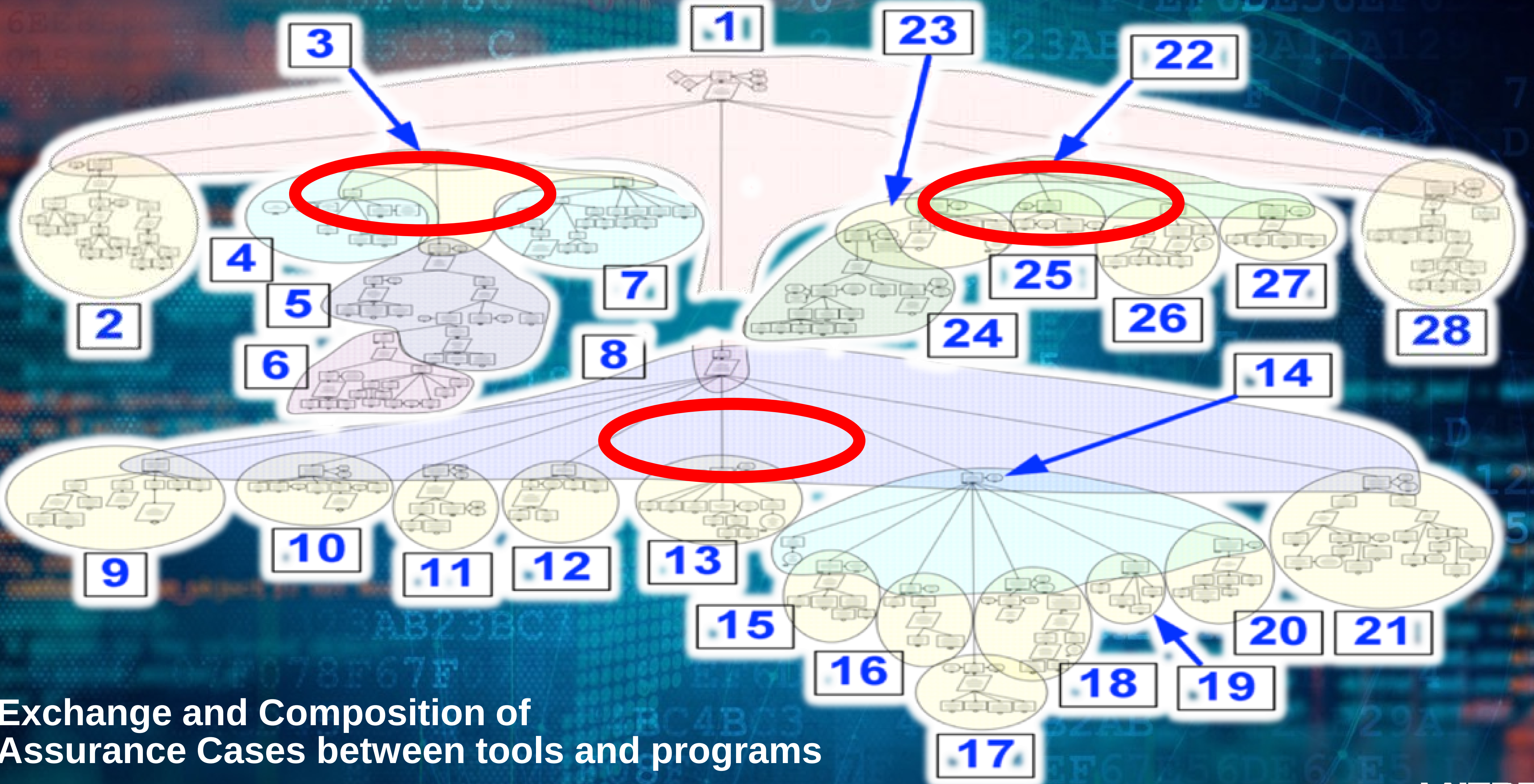
Medical
Space
Aeronautics
Rail
Automotive
Shipping
Autonomous
Critical Infrastructure
Cyber Physical Systems…

The Assurance Case for a System Builder using Assured Components



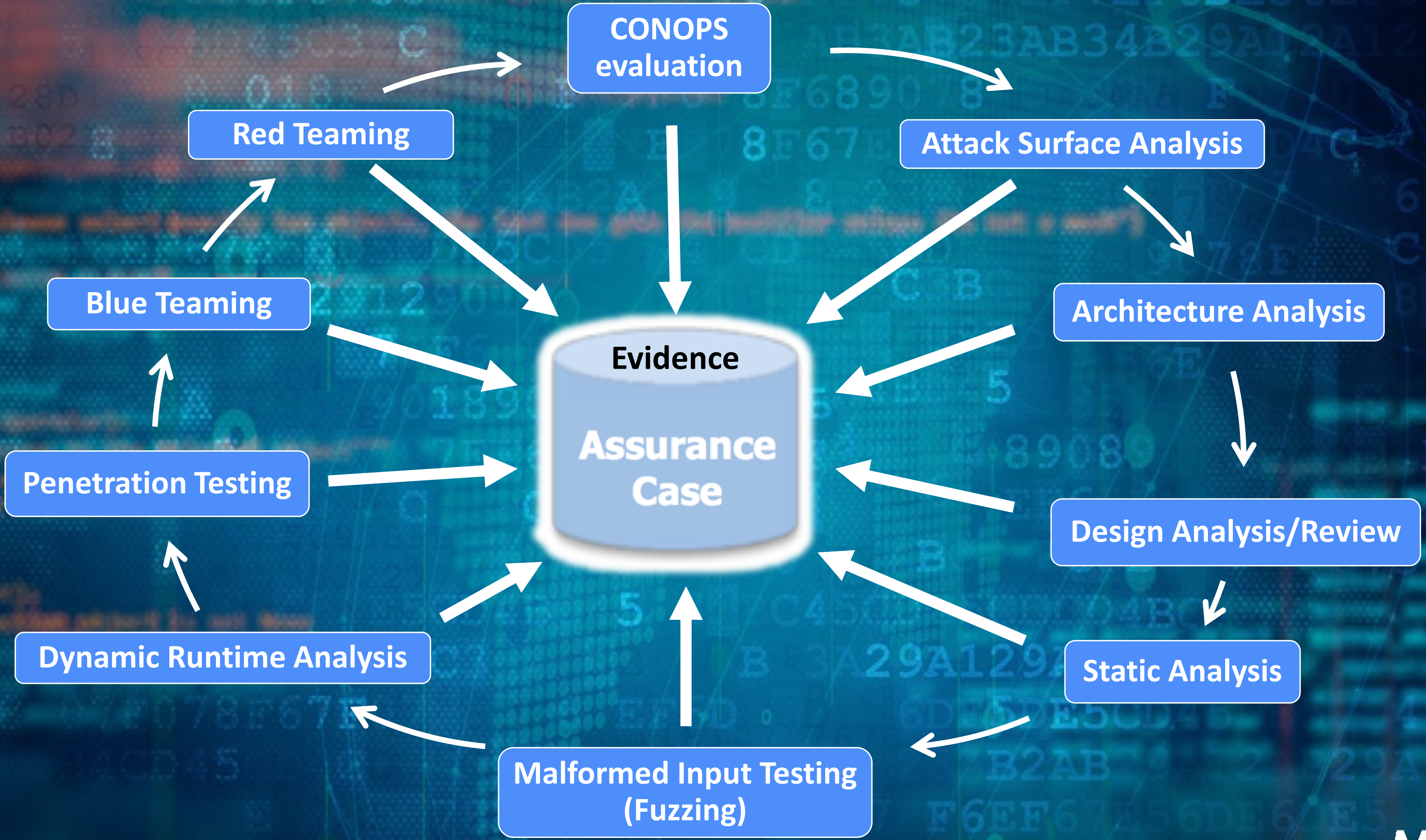
Exchange and Composition of Assurance Cases between tools and programs

The Assurance Case for a System Builder using Assured Components

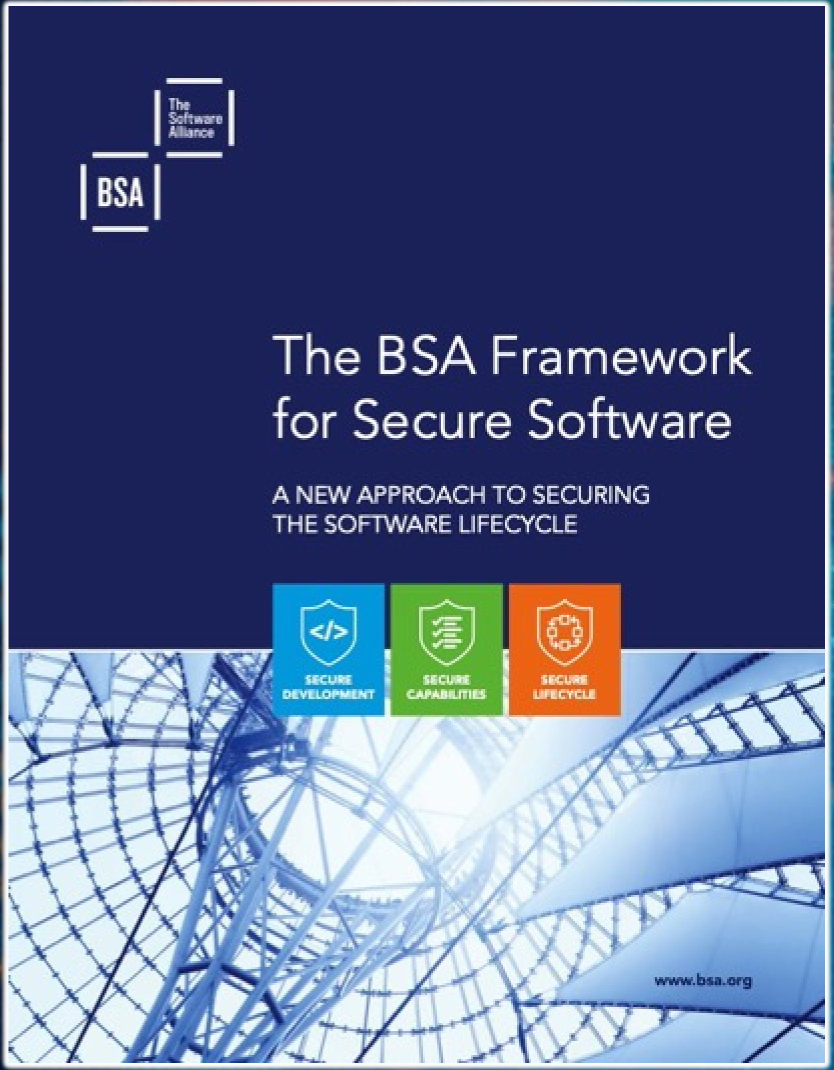


Exchange and Composition of Assurance Cases between tools and programs

Multiple Sources of Assurance Evidence from Across the Lifecycle of the item(s) needing Assurance.



MITRE



The Software Alliance

BSA

The BSA Framework for Secure Software

A NEW APPROACH TO SECURING THE SOFTWARE LIFECYCLE

SECURE DEVELOPMENT

SECURE CAPABILITIES

SECURE LIFECYCLE

www.bsa.org

Launched April 2019

Defining "Software Security"

Software security encompasses what a software development organization does to protect a software product and the associated critical data from vulnerabilities, internal and external threats, critical errors, or misconfigurations that can affect performance or expose data. It comprises both organizational processes and product capabilities.

> **Organizational processes** include governance structures, strategies, guidance, and clearly defined procedures that guide the development of software in a manner that identifies and incorporates security objectives throughout a product's lifecycle, protects the integrity of the development environment, applies resources to incident and vulnerability management, and manages the supply chain that supports the software development project.
>
> **Product security capabilities** are technical aspects of specific software products that are useful in enabling the products to address common security challenges, such as protecting data, preventing unauthorized access or use, tracking incidents and vulnerabilities, and managing unforeseen events.

Both organizational processes and product security capabilities are vital elements of software security.

Framework Basics

The Framework identifies best practices relating to both organizational processes and product capabilities across the entire software lifecycle. It is organized into six columns: Functions, Categories, Subcategories, Diagnostic Statements, Implementation Notes, and Informative References.

Functions organize fundamental software security activities at their highest level, consistent with the software lifecycle. The Functions are:

SECURE DEVELOPMENT

Secure development addresses security in the phase of software development when a software project is conceived, initiated, developed, and brought to market

SECURE CAPABILITIES

Secure capabilities identify key security characteristics recommended for a software product

SECURE LIFECYCLE

Secure lifecycle addresses considerations for maintaining security in a software product from its development through the end of its life

SECURE DEVELOPMENT

Category	Subcategory	Diagnostic Statement
Secure Coding (SC)	**SC.1.** Threat modeling and risk analysis are employed during software design to identify threats and potential mitigations.	**SC.1-1.** Software development organizations document likely threats.
		SC.1-2. Threats are rated and prioritized according to risk.
		SC.1-3. Software development organizations apply common threat modeling methodologies.
		SC.1-4. Compensating controls are identified and mapped to threats.
Secure Coding (SC) *(continued)*	**SC.2.** Software is developed according to recognized, enforceable coding standards.	**SC.2-1.** Standards are formally identified and documented.
		SC.2-2. Software uses canonical data formats.
	SC.3. The software is secure against known vulnerabilities, unsafe functions, and unsafe libraries.	**SC.3-1.** Software avoids, or includes documented mitigations for, known security vulnerabilities in included functions and libraries.
		SC.3-2. Software validates input and output to mitigate common vulnerabilities in software.
		SC.3-3. Software encodes data and/ or uses anti-cross site scripting (XSS) libraries.
	SC.4. Standard software assurance measures are employed in the software architecture and design.	**SC.4-1.** The software employs segmentation through sandboxing, containerization, or similar methodologies.
		SC.4-2. The software employs fault isolation mechanisms.

SECURE DEVELOPMENT

Category	Subcategory	Diagnostic Statement
Secure Coding (SC) *(continued)*	**SC.4.** Standard software assurance measures are employed in the software architecture and design.	**SC.4-3.** The software employs system element isolation mechanisms.
		SC.4-4. Software uses robust integer operations for dynamic memory allocations and array offsets.
Testing and Verification (TV)	**TV.1.** Analysis and validation of the software attack surface is conducted.	**TV.1-1.** Attack surface is identified and mapped.
		TV.1-2. Analysis is informed by threat model(s) and risk analysis.
	TV.2. Code review using manual and/ or automated tools is conducted.	**TV.2-1.** Code review release gates are established to guide software development.
	TV.3. A comprehensive test plan for testing the functionality and security of software is established.	**TV.3-1.** Test plan is based on threat model(s) and risk analysis.
		TV.3-2. The software is tested in a least privilege environment.
	TV.4. Software security controls are properly tested with appropriate techniques.	
	TV.5. Software is subjected to adversarial security testing techniques.	**TV.5-1.** Software development organizations establish security testing release gates.
		TV.5-2. Software is subjected to penetration testing.
Process and Documentation (PD)	**PD.1.** Secure development processes are documented throughout software development.	**PD.1-1.** Security requirements for the software are gathered from stakeholders and documented.
		PD.1-2. Security guidance for the development of the software is documented.
		PD.1-3. Security guidance for the development of software is updated to reflect the results of root cause analyses of new vulnerabilities.
		PD.1-4. Security documentation outlining best practices for software use by end-users and developers is made available electronically.
		PD.1-5. Testing and validation activities, including results, are documented.
		PD.1-6. Software development organizations maintain an up-to-date product history that documents changes to elements and configurations.

SECURE DEVELOPMENT

Category	Subcategory	Diagnostic Statement
Process and Documentation (PD)	**PD.2.** Software development personnel are accountable for software security.	**PD.2-1.** A security advisor is assigned to the software development team.
		PD.2-2. Software development personnel are trained on identified coding standards and role-specific best practices.
Supply Chain (SM)	**SM.1.** Software development is informed by supply chain risk management.	**SM.1-1.** An organizational supply chain management plan and processes for identification and reporting of supply chain incidents are established.
	SM.2. Approved acquisition measures are in place to ensure the visibility, traceability, and security of third-party components.	**SM.2-1.** Information about providers of third-party components is identified and collected.
		SM.2-2. Software development organization employs measures to document and, to the extent feasible, trace to their original source all third-party components directly acquired and incorporated into the software by the developer.
		SM.2-3. To the maximum feasible through the use of manual and automated technologies, subcomponents integrated in third-party components are documented, and their lineage and dependencies traced.
Supply Chain (SM) *(continued)*	**SM.2.** Approved acquisition measures are in place to ensure the visibility, traceability, and security of third-party components.	**SM.2-4.** Security requirements are incorporated into contracts, policies, and standards for vendors supplying software components.
	SM.3. Supply chain data — including information about software elements, design, testing, evaluation, threat assessments, delivery processes, and agreements language — is protected against unauthorized disclosure, access, modification, dissemination, destruction, and use.	**SM.3-1.** Supply chain data is protected at rest.
		SM.3-2. Supply chain data is protected in transit against unauthorized access.
	SM.4. Software incorporates measures to prevent counterfeiting and tampering.	**SM.4-1.** Software includes mechanisms to ensure the integrity of the software, such as code-signing, anti-reverse engineering, or anti-tamper mechanisms.
		SM.4-2. Software includes supplier source certification or authentication indicators and protects those indicators against tampering and counterfeiting.
		SM.4-3. Identification markers unique to the software's specific version are applied to each delivered product.

assigned task and only
for the necessary time
required to perform it.

ized
ions
ment
artifacts, and tools are
prevented and logged.

SECURE CAPABILITIES

Category	Subcategory	Diagnostic Statement
Support for Identity Management and Authentication (SI)	**SI.1.** The software avoids architectural weaknesses that create risk of authentication failure.	**SI.1-1.** The software avoids hard-coded passwords.
		SI.1-2. Software source code does not contain secrets.
		SI.1-3. Authentication mechanisms used by the software employ typical security techniques and avoid common security weaknesses.
		SI.1-4. The software does not store sensitive authentication information, which may include passwords or keys, in source code or publicly accessible infrastructure.
		SI.1-5. Any passwords or sensitive authentication information stored by the software is stored in accordance with current best practices.
	SI.2. The software supports strong identity management and authentication.	**SI.2-1.** The software implements features, configurations, and protocols that establish or support standard, tested authentication services.
		SI.2-2. The software is interoperable with applicable common industry standards for identity management and authentication.

SECURE CAPABILITIES

Category	Subcategory	Diagnostic Statement
Support for Identity Management and Authentication (SI) *(continued)*	**SI.2.** The software supports strong identity management and authentication.	**SI.2-3.** Authentication controls fail securely.
Patchability (PA)	**PA.1.** Software is capable of receiving secure updates and security patches.	**PA.1-1.** Software is capable of validating the integrity of a transmitted patch or update.
		PA.1-2. Software includes a mechanism to notify end users of patch or update installation.
		PA.1-3. Software reverts to a known-good state upon failed installation of updates or security patches.
Encryption (EN)	**EN.1.** Software is developed in accordance with an encryption strategy that defines what data should be encrypted and which encryption mechanisms should be used.	**EN.1-1.** Software enables the use of encryption to protect sensitive data from unauthorized disclosure.
		EN.1-2. Software enables the use of encryption to protect the software itself from tampering.
		EN.1-3. Software does not expose sensitive [illegible] encryption mechanisms.

SECURE CAPABILITIES

Category	Subcategory	Diagnostic Statement
Encryption (EN) *(continued)*	**EN.2.** Software avoids weak encryption.	**EN.2-1.** Software avoids custom encryption algorithms and implementations.
		EN.2-2. Software enables the use of authenticated encryption.
		EN.2-3. Encryption employed by the software enables strong algorithms.
		EN.2-4. Encryption employed by the software enables strong key lengths.
		EN.2-5. Encryption capabilities employed by the software are configured to select strong cipher modes and exclude weak ciphers by default.

SECURE CAPABILITIES

Category	Subcategory	Diagnostic Statement
Encryption (EN) *(continued)*	**EN.2.** Software avoids weak encryption.	**EN.2-6.** Software is configured to disable or prevent the use of weak encryption algorithms and key lengths.
	EN.3. Software protects and validates encryption keys.	**EN.3-1.** Software ensures that cryptographic keys can be securely stored and managed, separate from encrypted data.
		EN.3-2. Software includes a mechanism to manage key and certificate lifecycles.
		EN.3-3. Software includes a mechanism to validate certificates.
Authorization and Access Controls (AA)	**AA.1.** Software design reflects the principle of least privilege.	**AA.1-1.** The software operates using only those privileges or permissions necessary for software to run correctly.
		AA.1-2. Privileges are set in a configuration that is resistant to unauthorized changes.

SECURE CAPABILITIES

Category	Subcategory	Diagnostic Statement
Authorization and Access Controls (AA) *(continued)*	**AA.1.** Software design reflects the principle of least privilege.	**AA.1-3.** An authorization strategy that applies authorization policies, access controls, and design principles to classes of data is implemented in the software.
	AA.2. The software's design supports authorization and access controls.	**AA.2-1.** The software avoids functions that enable unauthorized privilege escalations.
		AA.2-2. In the case of failure, the software does not grant access to unauthorized or unauthenticated users.
Logging (LO)	**LO.1.** Software implements logging of all critical security incident and event information.	**LO.1-1.** Software differentiates between monitoring logs and auditing logs.
		LO.1-2. Software is capable of logging all security-relevant failures, errors, and exceptions.
		LO.1-3. Software is capable of logging timestamp and identifying information associated with security incidents and events.
	LO.2. Software security incident and event information logging mechanisms are implemented securely.	**LO.2-1.** Access to logs is restricted to authorized individuals.
		LO.2-2. Logging mechanisms include anti-tamper protections.

SECURE CAPABILITIES

Category	Subcategory	Diagnostic Statement
Logging (LO) *(continued)*	**LO.2.** Software security incident and event information logging mechanisms are implemented securely.	**LO.2-3.** Logs do not store sensitive information, such as unnecessary user information, system details, session identifiers, or passwords.
		LO.2-4. Software logging mechanisms employ input validation and output encoding.
Error and Exception Handling (EE)	**EE.1.** Software integrates error and exception handling capabilities.	**EE.1-1.** Software identifies predictable exceptions and errors that could occur during software execution and defines how the software will handle each instance.
		EE.1-2. Software defines how it will handle unpredicted exceptions and errors and safeguards against continued execution in an insecure state.
		EE.1-3. Notifications of errors and exceptions do not disclose sensitive technical or human information.
	EE.2. Software fails securely; if a program is forced to terminate unexpectedly, it shuts down in a safe and responsible manner.	**EE.2-1.** Software is designed to continue operating in a degraded manner until a threshold is reached that triggers orderly, secure termination.
		EE.2-2. In the case of failure, software [illegible] default [illegible]erve confidentiality and integrity.

MITRE

SECURE LIFECYCLE

Category	Subcategory	Diagnostic Statement
Vulnerability Management (VM)	**VM.1.** The vendor maintains an up-to-date vulnerability management plan.	**VM.1-1.** The vulnerability management plan outlines policies, responsibilities, and expectations for both internal and external stakeholders throughout the following phases of vulnerability management: (1) the vendor's identification or receipt of a vulnerability, (2) verification of the vulnerability, (3) remediation or mitigation of the vulnerability, (4) release of a solution, and (5) post-release.
		VM.1-2. The vulnerability management plan addresses security testing and vulnerability identification methodologies to be applied throughout a product's lifecycle.
		VM.1-3. The vulnerability management plan includes a process for gaining timely awareness of and managing vulnerabilities that are discovered in third-party components of the software.
	VM.2. Vulnerabilities are identified and resolved rapidly and comprehensively, according to risk-based prioritization.	**VM.2-1.** Upon identification, vulnerabilities are verified and subjected to root cause and risk analysis.
		VM.2-2. Vulnerabilities are assigned a unique identification number.

SECURE LIFECYCLE

Category	Subcategory	Diagnostic Statement
Vulnerability Management (VM) *(continued)*	**VM.2.** Vulnerabilities are identified and resolved rapidly and comprehensively, according to risk-based prioritization.	**VM.2-3.** Vulnerabilities are assigned a severity value based on risk, using a standardized scoring methodology.
		VM.2-4. Remediation and mitigation activities are informed by the severity of the vulnerability.
	VM.3. The vendor maintains a coordinated vulnerability disclosure program.	**VM.3-1.** The vendor establishes a clearly defined and easily accessible intake mechanism to accept vulnerability information (email, portal, etc.).
		VM.3-2. A vendor's intake mechanism provides for secure and confidential communication of sensitive vulnerability information.
		VM.3-3. The vendor publishes, in simple and clear language, its policies for interacting with vulnerability reporters, addressing, at minimum: (1) how the vendor would like to be contacted, (2) options for secure communication, (3) expectations for communication from the vendor regarding the status of a reported vulnerability, (4) desired information regarding a potential vulnerability, (5) issues that are out of scope of the vulnerability disclosure program, (6) how submitted vulnerability reports are tracked, and (7) expectations for whether and how a reporter will be credited.

SECURE LIFECYCLE

Category	Subcategory	Diagnostic Statement
Vulnerability Management (VM) *(continued)*	**VM.3.** The vendor maintains a coordinated vulnerability disclosure program.	**VM.3-4.** The vendor maintains a system to record and track all reports of potential vulnerabilities.
		VM.3-5. The vendor notifies vulnerability reporters of when reported vulnerabilities are remediated or mitigated.
Configuration (CF)	**CF.1.** The software is deployed with configurations and configuration guidance that facilitate secure installation and operation.	**CF.1-1.** The software documentation specifies configuration parameters that are as restrictive as feasible, to make sure the software is as resistant as possible to anticipated attacks and exploits.
		CF.1-2. The software documentation describes secure installation procedures for initial installation and installation for additional components, updates, and patches.
		CF.1-3. The software documentation describes configurations and procedures for secure configuration under normal operation.
		CF.1-4. The software prompts users to change any default passwords before the software becomes operational.
		CF.1-5. Configuration guidance statements and configuration controls are clearly communicated and automated wherever possible.

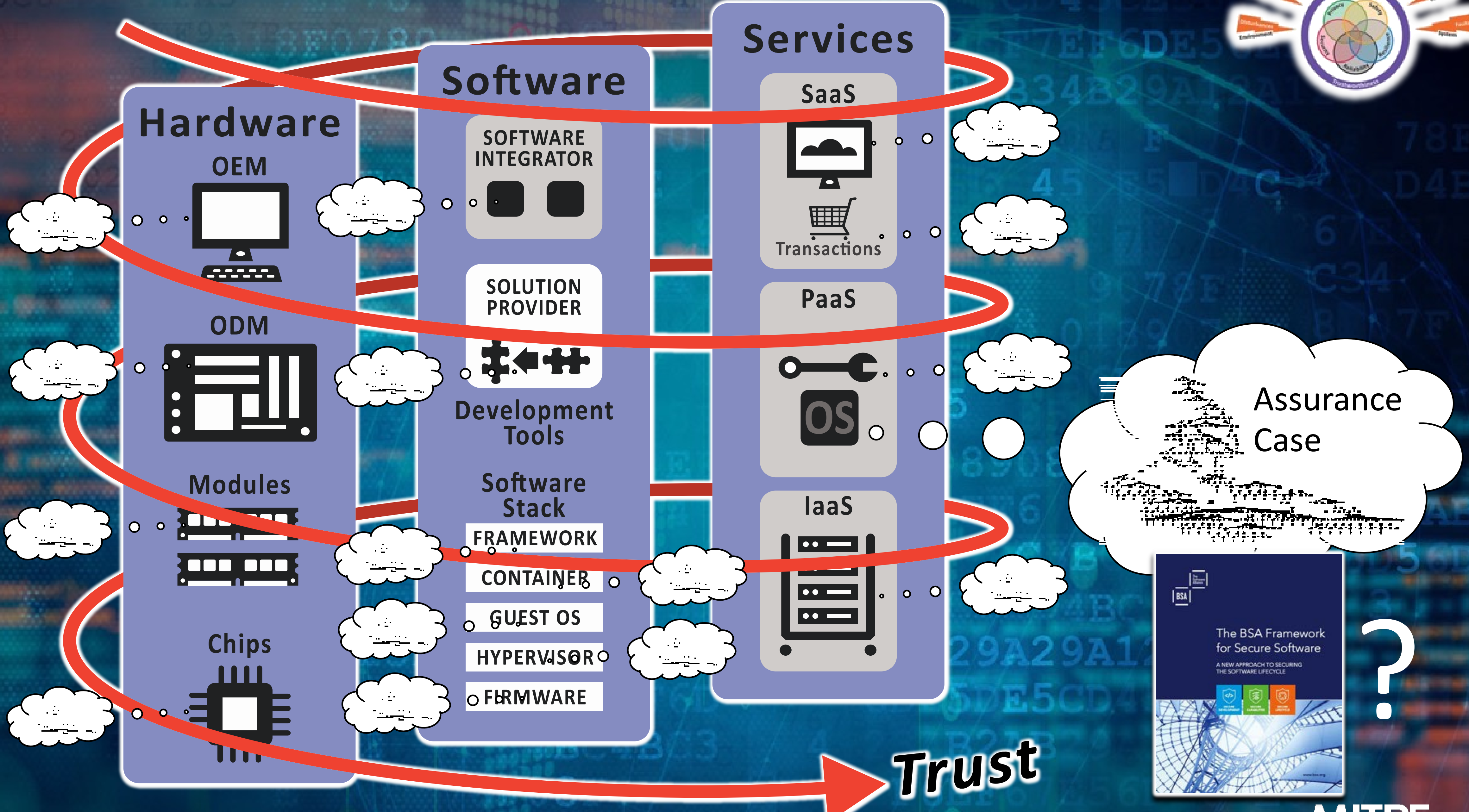
SECURE LIFECYCLE

Category	Subcategory	Diagnostic Statement
Configuration (CF) *(continued)*	**CF.1.** The software is deployed with configurations and configuration guidance that facilitate secure installation and operation.	**CF.1-6.** Software configuration settings can be altered to tailor security settings to the operating environment.
Vulnerability Notification and Patching (VN)	**VN.1.** Vendors disseminate timely patches or updates to address identified security issues.	**VN.1-1.** Patches or updates are developed and disseminated based on risk-informed prioritization, in accordance with the vendor's vulnerability management program.
		VN.1-2. Patches or updates are subjected to testing for functionality and security prior to release.
		VN.1-3. All patches and updates are documented.
		VN.1-4. Development and dissemination of patches or updates are coordinated with other vendors where appropriate to address multi-vendor security issues or supply chain security issues.
	VN.2. Patches or updates are disseminated securely.	**VN.2-1.** Patches or updates are transmitted in a manner that prevents exposure of the software image.
		VN.2-2. The patch or update deliverable is cryptographically signed to ensure its integrity and authenticity.

SECURE LIFECYCLE

Category	Subcategory	Diagnostic Statement
Vulnerability Notification and Patching (VN) *(continued)*	**VN.3.** Patches or updates for security issues are accompanied by advisory messages informing users of relevant information.	**VN.3-1.** Users are notified of a significant security issue when a remediation is in place for each supported version of the affected product.
		VN.3-2. Advisory messages notifying users of security issues include information on affected products, applicable versions, and platforms; a unique identification number; and a brief description of the vulnerability and its potential impact.
End-of-Life (EL)	**EL.1.** Vendor maintain consistent lifecycle guidance.	**EL.1-1.** Vendor communicates realistic assumptions and expectations regarding the nature and lifespan of product support in tandem with initial software delivery.
		EL.1-2. Vendor clearly communicates decisions to terminate support for a software product to customers and users, identifying the expected support termination date; the anticipated risk of continued product use beyond the termination of support; possible mitigation actions; and options for technical migration to replacement products.
		EL.1-3. Software is continually monitored to ensure that third-party components have [illegible]-of- [illegible] are [illegible]erwise remediated.

Transparent Assurance As a Basis for Trust - FUTURE



MITRE

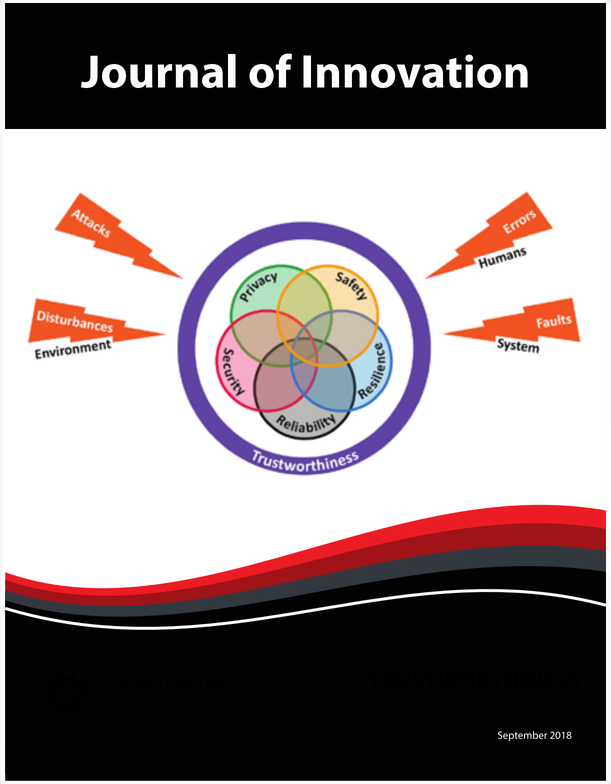
Questions?

IIC Journal of Innovation – September 2018 issue on Trustworthiness

https://www.iiconsortium.org/journal-of-innovation.htm

“Assuring Trustworthiness in an Open Global Market of IIoT Systems via Structured Assurance Cases”

https://www.iiconsortium.org/news/joi-articles/2018-Sept-JoI_Assuring_Trustworthiness-FINAL2.pdf



Journal of Innovation

September 2018

Assuring Trustworthiness in an Open Global Market of IIoT Systems via Structured Assurance Cases

Authors:
Robert A. Martin
Senior Principal Engineer
The MITRE Corporation
ramartin@mitre.org

Ken Modeste
Director
Connected Technologies, UL LLC
Ken.Modeste@ul.com

September 2018 - 44 -