IS研究所 信頼される安心を、社会へ。

IEICE総合大会

IETFにおけるサプライチェーンセキュリティ関連の標準化動向

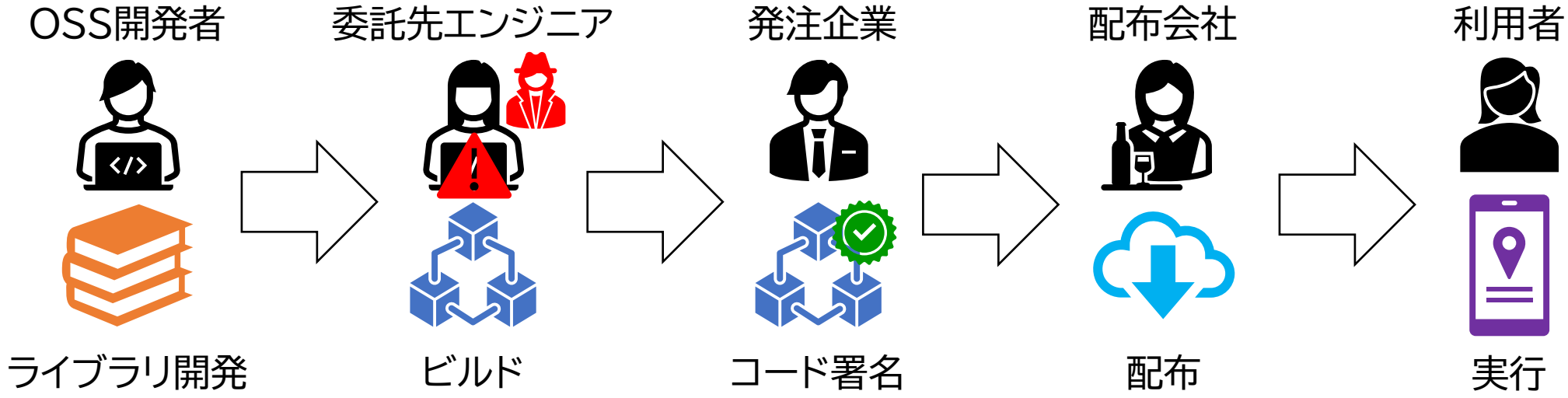
セコム株式会社　IS研究所
髙山献

本日お話しすること

- ソフトウェアサプライチェーンに対する攻撃
- IETFでの取り組み
  - IETFとは
  - SCITT WG(Working Group)での取り組み
  - TEEP WGとSUIT WGでの取り組み
- まとめ

背景:サプライチェーンに対する攻撃

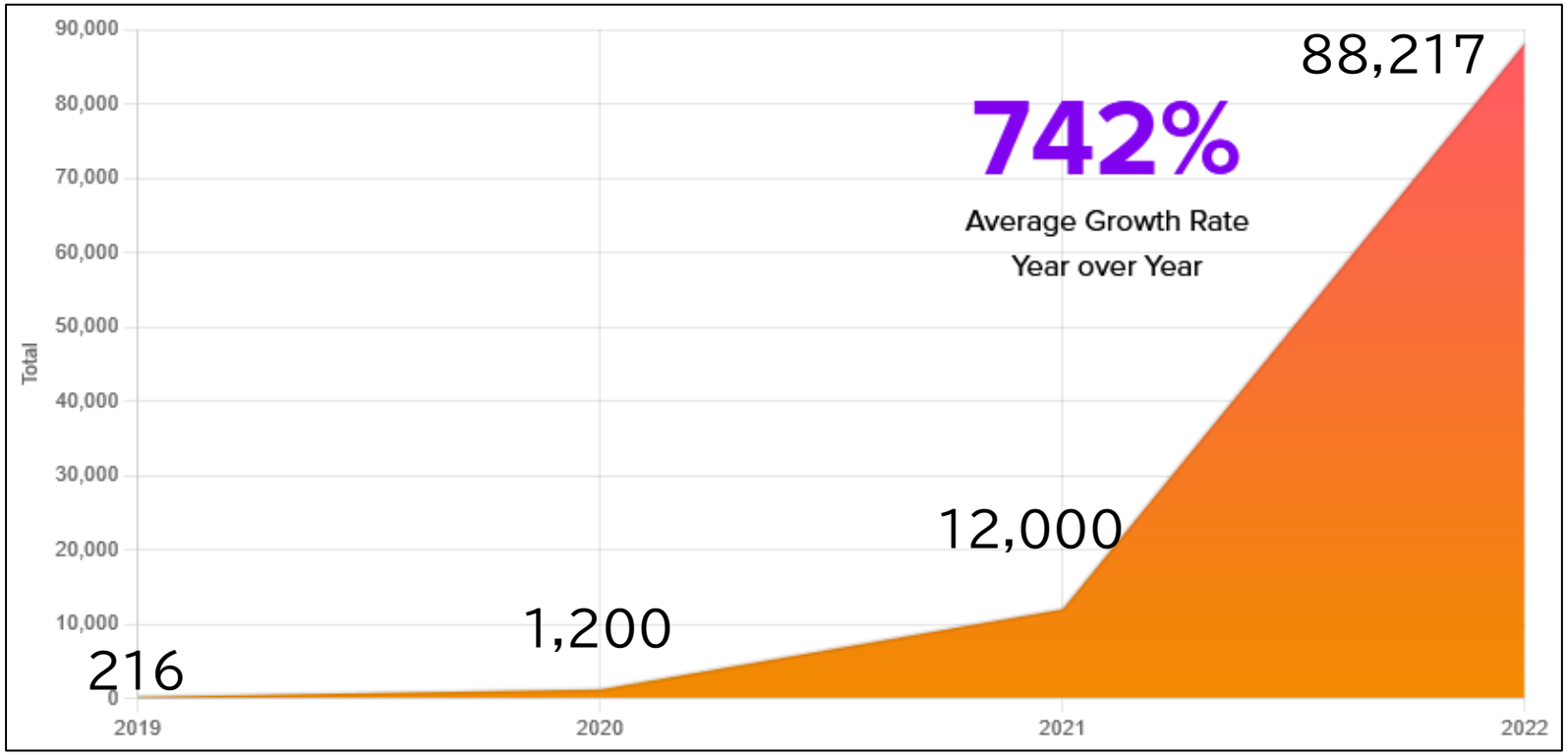
- 商品やサービスを提供するまでの取引先組織への攻撃
  - 例)子会社や業務委託先などがランサムウェアの攻撃を受けて機密を暴露される



- ソフトウェア開発で利用するOSS(オープンソースソフトウェア)などを介して攻撃されることもある
  - マルウェアが混入したものを利用してしまう
  - (既知の)脆弱性があるものを利用してしまう

ソフトウェアサプライチェーンへの攻撃

- OSSを介した攻撃は毎年数倍の勢いで増加中
  - マルウェア疑いが強いパッケージを88,217個/年発見(2022年)

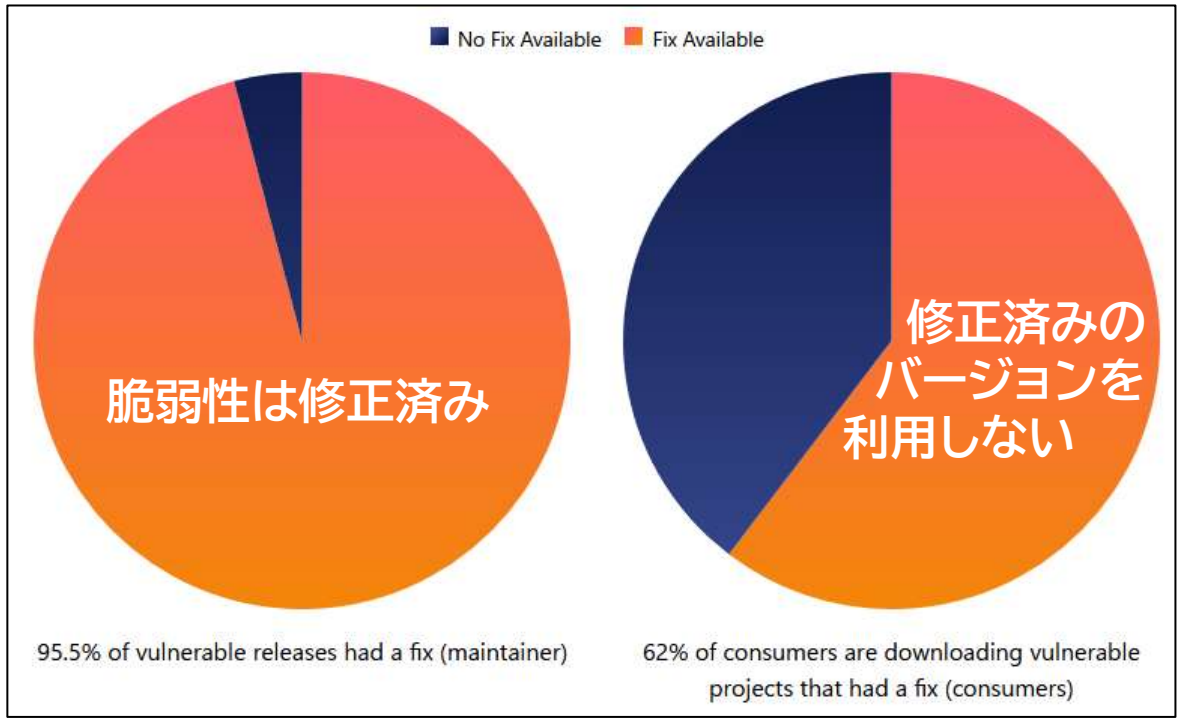


Sonatype社が自動的な
挙動解析などを通して検知し、
その後安全性が確認できた
ものを除いた数。

https://www.sonatype.com/state-of-the-software-supply-chain/open-source-supply-demand-security

サプライチェーンにおける脆弱性

- 対策の基本中の基本：バグが修正されたバージョンを利用する
  - 既知の脆弱性があるリリースのうち95.5%は脆弱性が修正済み
  - 開発者の62%は修正が利用できるにもかかわらず脆弱なものを利用している



Sonatype社がJavaのパッケージ管理システムMaven Central Repositoryを調査。
（左図）月間約86億ダウンロードのうち、12億は既知の脆弱性があり、5500万リリース（4.5%）は修正がないもの。
→月間11.8億ダウンロード（95.5%）は
　回避可能な脆弱性利用
（右図）872万人（62%）の開発者が修正バージョンが利用できるのに脆弱なバージョンを選び、572万人（38%）は修正版が存在せず仕方なく脆弱なバージョンを選んでいる

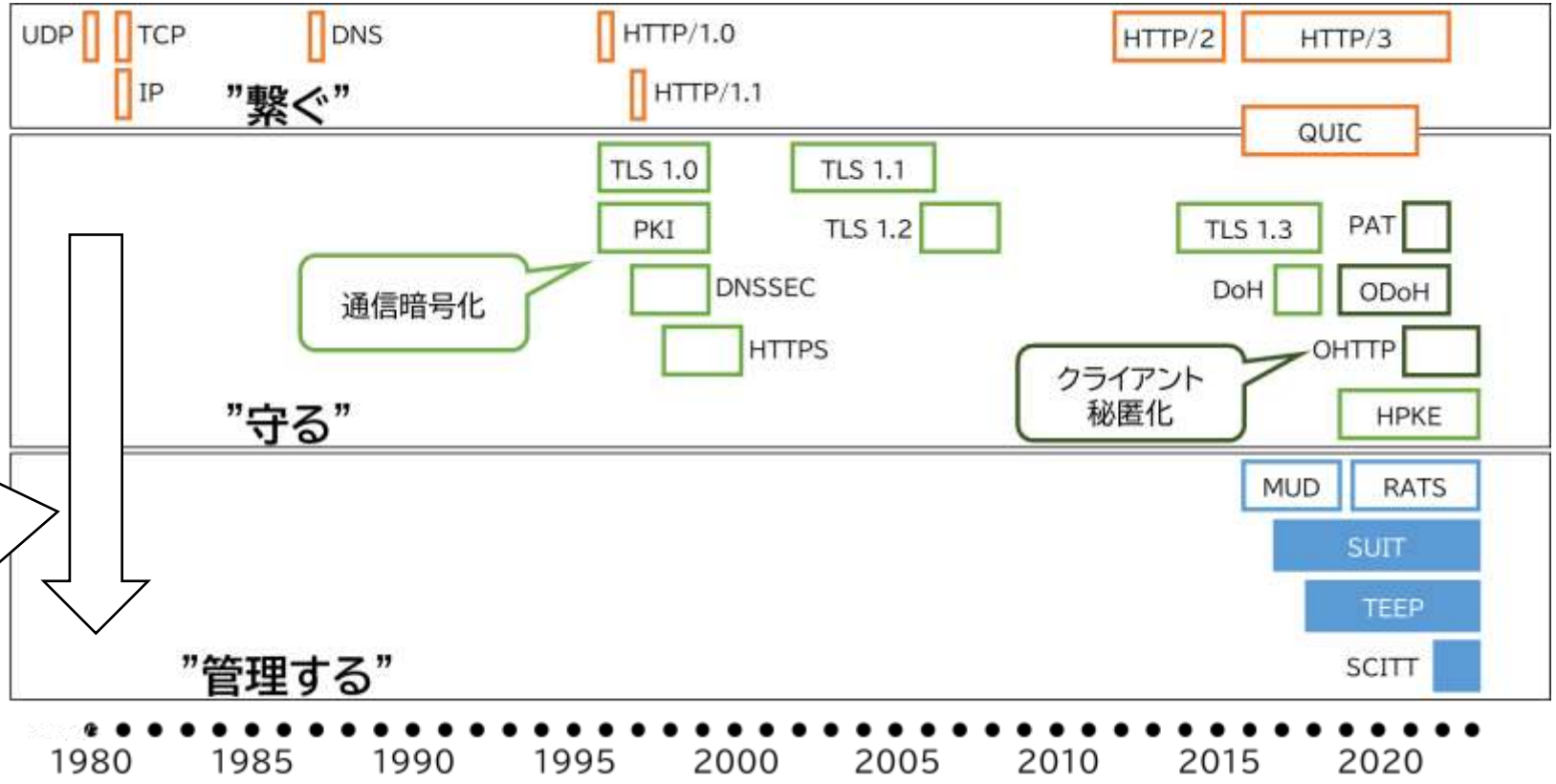
https://www.sonatype.com/state-of-the-software-supply-chain/open-source-dependency-management-trends-and-recommendations

IETFでの取り組み

IETFとは

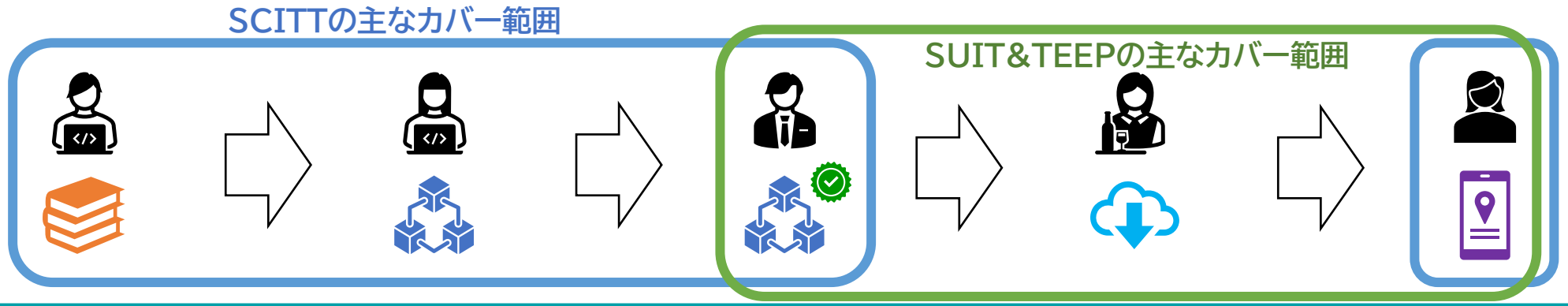
- Internet Engineering Task Force
  - インターネットに関連する技術標準をまとめる団体の一つ
  - 有名な標準(RFC)はTCP、IP、HTTP、HTTPS、QUICなど

近年は通信プロトコル
だけではなく、
暗号技術やその利用方法、
インターネットに接続する
機器の管理や連携
なども標準化対象に



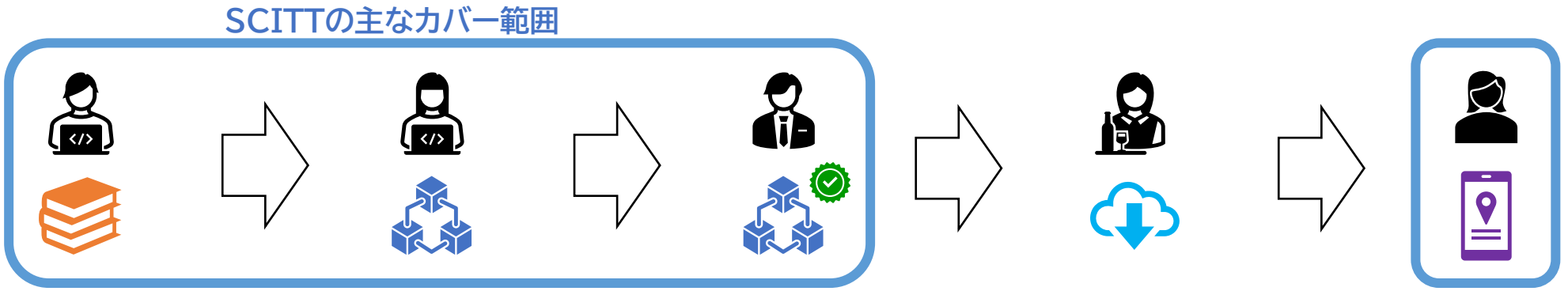
サプライチェーン攻撃を防ぐ取り組み

- 利用しているソフトウェアとそのバージョンを把握する
  - 誰が開発・ビルドしたどのソフトウェア、ライブラリなどを利用しているか
  - サプライチェーンの中でどのように取り扱われたか
  - ➢透明性、完全性、信頼性を確保する手段が求められている→SCITT
- 利用者の手元で意図した通りに動作しているか
  - ソフトウェアのインストール・実行前に改ざんがないことを検証する→SUIT
  - 改ざんされていない、意図した実行環境上で動作している→TEEP



サプライチェーン攻撃を防ぐ取り組み

- 利用しているソフトウェアとそのバージョンを把握する
  - 誰が開発・ビルドしたどのソフトウェア、ライブラリなどを利用しているか
  - サプライチェーンの中でどのように取り扱われたか
  - ➢透明性、完全性、信頼性を確保する手段が求められている→SCITT
- 利用者の手元で意図した通りに動作しているか
  - ソフトウェアのインストール・実行前に改ざんがないことを検証する→SUIT
  - 改ざんされていない、意図した実行環境上で動作している→TEEP



IETF SCITT Working Group

- Supply Chain Integrity, Transparency and Trust
  - 構成要素の出自を明確にする(Transparency＝透明性)
  - それを維持する(Integrity＝完全性)
  - それを検証できるようにする(Trust＝信頼性)
- まずはソフトウェアサプライチェーンを対象に取り組む
  - ユースケースの検討が進む https://datatracker.ietf.org/doc/html/draft-birkholz-scitt-software-use-cases-00

時期	出来事
2022年3月	IETF113 SecDispatchミーティングにてSCITT構想発表、グループ化
2022年6月	SCITT Interimミーティングにて BoF開催、問題設定と方向性の審議開始
2022年7月	IETF114にてSCITT BoF開催
2022年10月	SCITT WG Charterが承認され正式なWGへ
2022年11月	IETF115にてSCITTミーティング開催

背景:アメリカ政府が対策を指示

- アメリカ政府や重要インフラに大規模なサイバー攻撃を受け政権が対応
  - 米国内で一気にサプライチェーン攻撃に対する警戒が強まった

時期	出来事
2020年3-12月	米国SolarWinds社が狙われネットワーク監視製品にバックドアを挿入された アメリカ政府機関を含む100弱の組織内の情報が収集された
2021年4-5月	米国Colonial Pipeline社が狙われランサムウェアによるサイバー攻撃を受けた アメリカ東海岸の燃料供給パイプラインが1週間停止した
2021年5月	バイデン大統領がサイバーセキュリティ大統領令（E.O. 14028） 防衛・エネルギーなどで使用する重要製品の対策強化を各省庁に命令
2021年7月	NTIAがソフトウェアに関してSBOM（後述）の必要要件を発表
2022年2月	NISTがSecure Software Development Framework 1.1を発表（SP800-218）
2022年2月	商務省と国土安全保障省がソフトウェア開発でのOSS利用や外注に言及
2022年2月	国防省がOSSを優先採用することを定め、従業員がOSSに貢献することも認める

アメリカでは行政が積極的に動いて対策が続けられている

背景:日本でも進むOSS利用と対策

IS研究所 信頼される安心を、社会へ。

- 行政・民間企業のOSS依存が進む中、利活用と対策を模索している

時期	出来事
2020年3月	東京都が新型コロナウイルス感染症対策サイトのソースコードを公開
2020年9月	厚生労働省が接触確認アプリCOCOAのソースコードを公開
2021年4月	経済産業省がOSSの利活用及びそのセキュリティ確保に向けた管理手法に関する事例集を各企業からヒアリングして作成
2022年2月	公正取引委員会が競争政策上の検討事項として、オープンな仕様の設計や情報システムのオープンソース化を挙げる
2022年5月	経済安全保障推進法成立、サプライチェーン強靭化のための支援も盛り込まれる
2022年6月	デジタル庁情報システム調達改革検討会にて、ベンダーロックイン予防のためにオープンソースソフトウェアの活用やオープンソース化・官公庁内での共有を議論
2022年8月	OSS Security Summit Japan開催（主催：Linux Foundation、OpenSSF）
2023年1月	IPA情報処理推進機構が情報セキュリティ脅威の組織部門2位に「サプライチェーンの弱点を悪用した攻撃」に挙げる

日本でも各省庁がOSSの利用やOSS化などについて指針を示している

SCITTで目指していること

- ソフトウェアサプライチェーン関連の署名・検証フローの標準化
  - 必要に応じてIETFが標準化してきた技術を再利用する
    - COSEやRATSなど
  - 外部の標準化組織と連携する
    - OpenSSF、W3C、ISO、Trusted Computing Groupなど

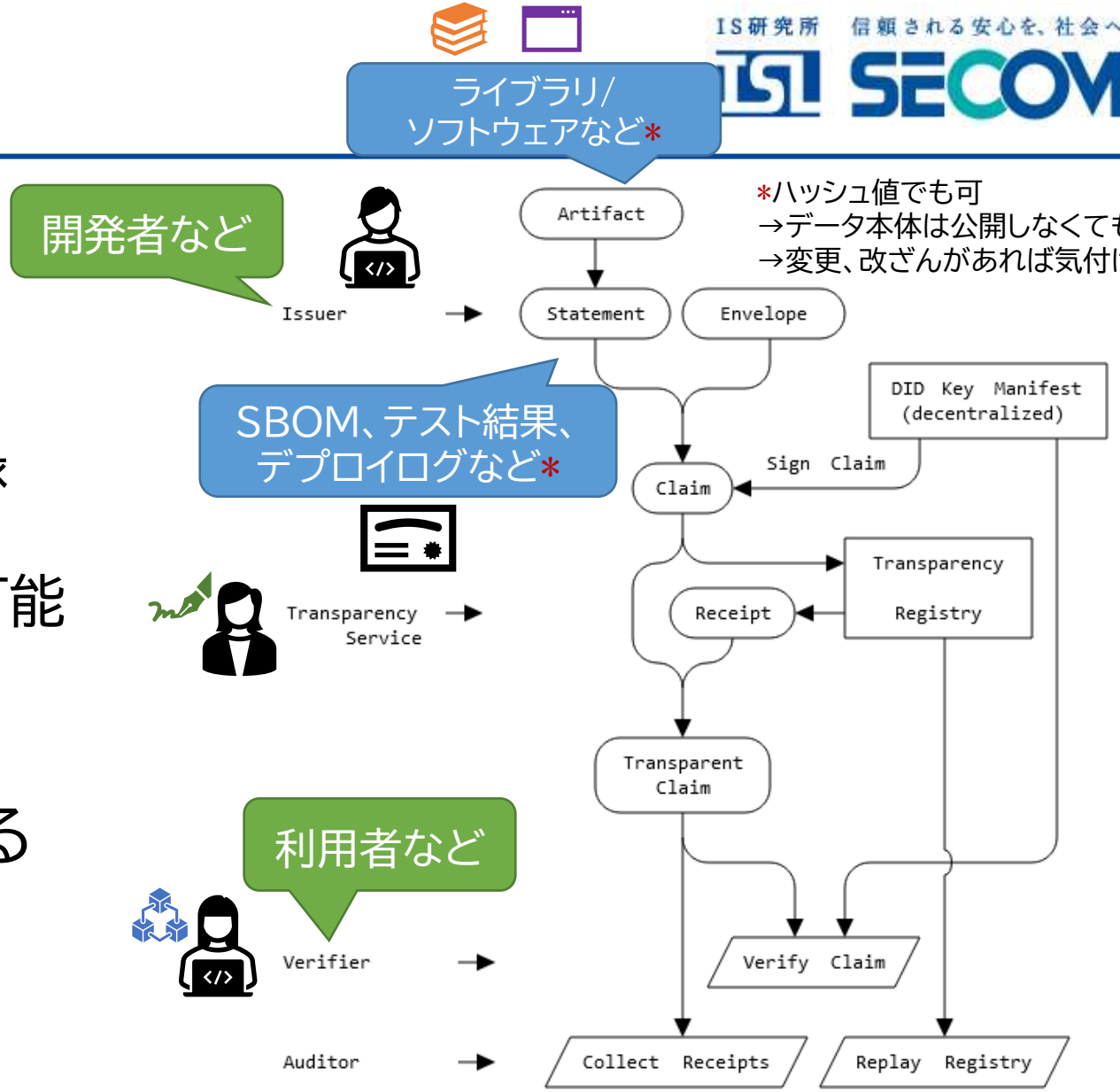
➢IoT機器のソフトウェア管理にも利用可能な仕組み

- 利用するソフトウェアがいつ、誰に作られたかの記録が残る
- 採用前、攻撃や脆弱性が発覚した時に、診断に用いることができる

SCITT Architecture

- 各者の役割とフローを定義
  - Issuerは作成したソフトウェアとそのSBOMなどを提出
  - Transparency Serviceが記録
    - Notary(公証人)とも呼ばれる
  - Verifierは証拠の正しさを検証可能
  - AuditorはTSの運用を監査

➢OSSの出自などを記録し検証を可能とする場として適する



https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture

既存のサプライチェーン攻撃対策例

- SBOM（Software Bill of Materials）
  - SCITTアーキテクチャの中で証拠として伝達されるもの
  - 目的：開発時に意図しないソフトウェアが混入することを防ぐ
  - 方法：依存したソフトウェアの一覧を表示し確認する

- Sigstoreプロジェクト
  - SCITTアーキテクチャに近い実装
  - 目的：開発時に、依存するソフトウェア等に改ざんがないことを検証可能にする
  - 方法：依存するOSSのハッシュ値と開発者の署名を公開ログに残す

SBOM

- ソフトウェアの構成表（食品の原材料表記に近い）
- セキュアなソフトウェアサプライチェーン管理の一要素に過ぎない
  - SBOMを導入したからといって脆弱性が無くなるわけではない
  - 必要十分な情報は何か、使いやすいツールは何か模索中
- 主要なフォーマットはSPDX（Software Package Data Exchange）
  - 依存しているソフトウェア名、バージョンなどの一覧をSPDX Viewerで表示可能
  - SPDX 2.2.1はISO/IEC 5962:2021として標準化された
  - もともとはOSSライセンス遵守を主目的に作られた

加工食品の品質表示基準に例えると

SBOMの例）Mbed TLS

- Armが開発する暗号ライブラリ

```
SPDXVersion: SPDX-2.1
DataLicense: CC0-1.0
SPDXID: SPDXRef-DOCUMENT
DocumentName: mbedtls-2.16.2
DocumentNamespace: http://spdx.org/spdxdocs/mbedtls-2.16.2-d92dab8c-
6849-42f7-b4ef-843092878b27
Creator: Person: Tomo Dote(fu7mu4@gmail.com)
Creator: Organization: OpenChainProject ()
Created: 2019-01-25T01:01:01Z

PackageName: mbedtls
SPDXID: SPDXRef-1
PackageVersion:        2.16.2
PackageFileName: mbedtls-2.16.2-apache.tgz
PackageDownloadLocation: https://tls.mbed.org/download
FilesAnalyzed: false
PackageHomePage: https://tls.mbed.org/
PackageLicenseConcluded: Apache-2.0
# PackageLicenseInfoFromFiles: Apache-2.0
PackageLicenseDeclared: (Apache-2.0 OR GPL-2.0-only)
```

Apacheライセンス版Mbed TLSの記述例(tag:value形式)

https://github.com/OpenChain-Project/OpenChain-JWG/blob/master/subgroups/sbom-sg/outcomes/SPDX-Lite/sample/mbedtls/mbedtls-apache.txt

SBOMの例）PowerShell

- MicrosoftはPowerShell 7.2以降SBOMを配布

ビルド時に依存したもの？

```
{
  "files": [
    {
      "fileName": "./Accessibility.dll",
      "SPDXID": "SPDXRef-File--Accessibility.dll-F2E4AB20D6B6...",
      "checksums": [
        {
          "algorithm": "SHA256",
          "checksumValue": "fe3ddbb74a3f2fbad2dcd65ace650971b..."
        },
        {
          "algorithm": "SHA1",
          "checksumValue": "f2e4ab20d6b6552a54b575e35d9c11e02..."
        }
      ],
      "licenseConcluded": "NOASSERTION",
      "licenseInfoInFiles": [
        "NOASSERTION"
      ],
      "copyrightText": "NOASSERTION"
    },
    ...
  ],
```

実行時に依存するもの

```
"packages": [
    ...
    {
      "name": "Microsoft.Win32.SystemEvents",
      "SPDXID": "SPDXRef-Package-B21155C90DFAD7E2D5BF088...",
      "downloadLocation": "NOASSERTION",
      "filesAnalyzed": false,
      "licenseConcluded": "NOASSERTION",
      "licenseInfoFromFiles": [
        "NOASSERTION"
      ],
      "licenseDeclared": "NOASSERTION",
      "copyrightText": "NOASSERTION",
      "versionInfo": "7.0.0",
      "externalRefs": [
        {
          "referenceCategory": "PACKAGE-MANAGER",
          "referenceType": "purl",
          "referenceLocator":
                  "pkg:nuget/Microsoft.Win32.SystemEvents@7.0.0"
        }
      ],
      "supplier": "NOASSERTION"
    },
    ...
  ],
```

ソフトウェアの概要

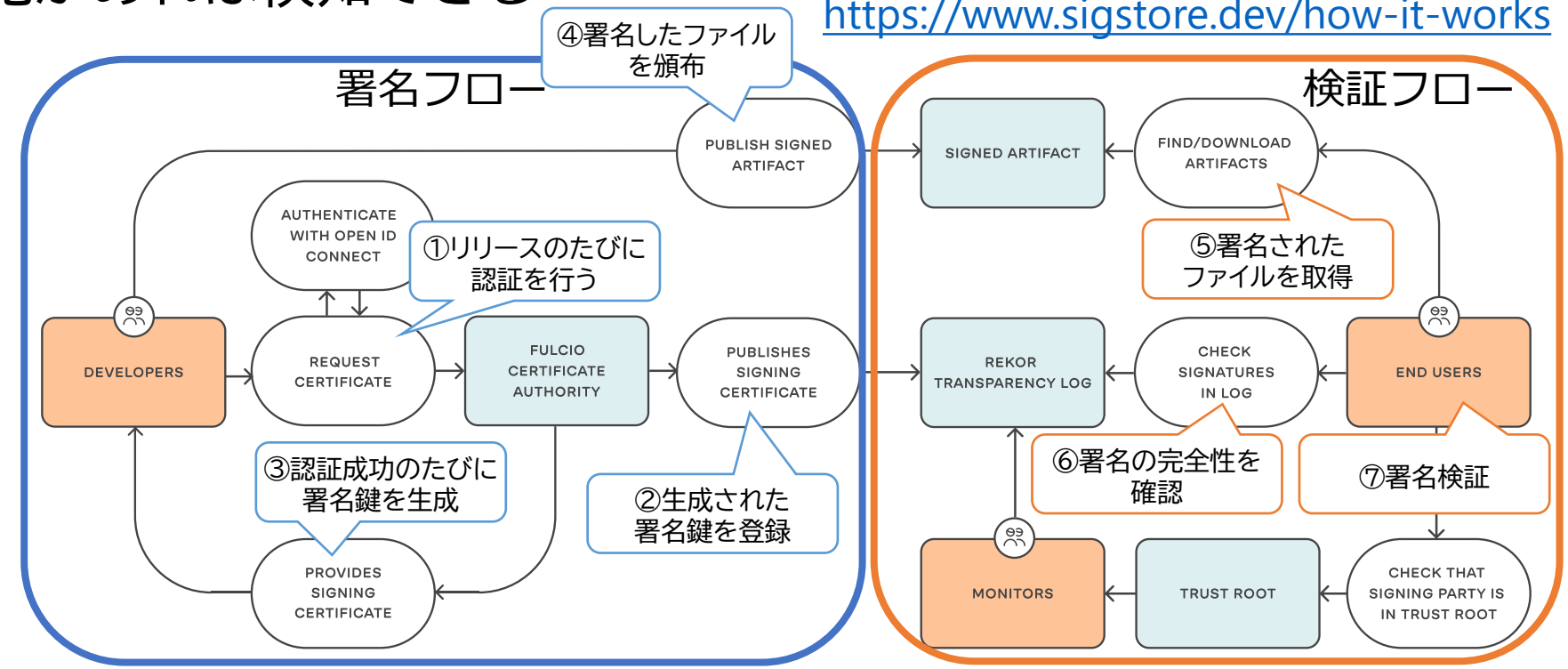
```
  "relationships": [
    ...
    {
      "relationshipType": "DEPENDS_ON",
      "relatedSpdxElement": "SPDXRef-Package-B21155C90DFAD7E2...",
      "spdxElementId": "SPDXRef-RootPackage"
    },
    ...
  ],
  "spdxVersion": "SPDX-2.2",
  "dataLicense": "CC0-1.0",
  "SPDXID": "SPDXRef-DOCUMENT",
  "name": "PowerShell Windows x64 release 7.3.2",
  "documentNamespace": "https://sbom.microsoft/1:2QSF7qZlb...",
  "creationInfo": {
    "created": "2023-01-18T23:27:43Z",
    "creators": [
      "Organization: Microsoft",
      "Tool: Microsoft.SBOMTool-0.3.1"
    ]
  },
  "documentDescribes": [
    "SPDXRef-RootPackage"
  ]
}
```

髙山の環境で `PS> Get-Content $PSHOME/_manifest/spdx_2.2/manifest.spdx.json` を実行(18,752行)

Sigstore

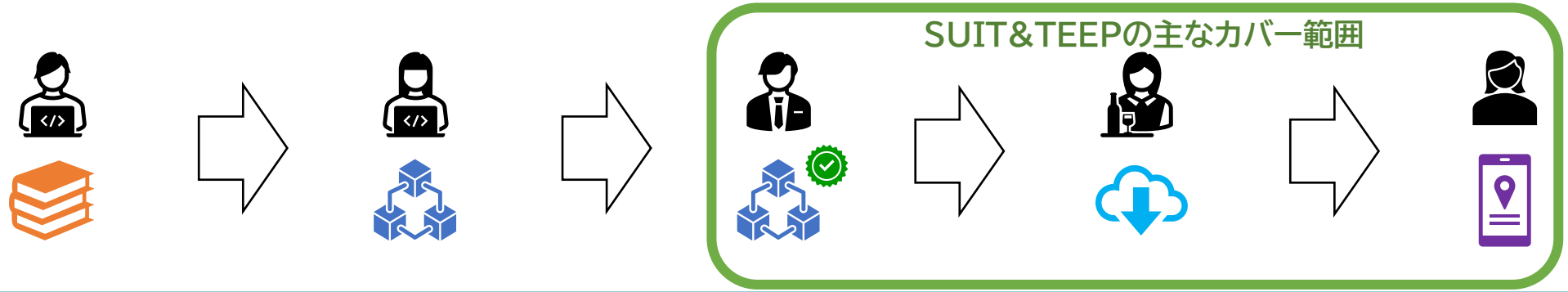
- OSSの頒布イメージに対する署名・検証プロジェクト
  - 誰が、いつ、どこで頒布したものかを誰でも検証できる
  - 自動化されたツールを使って登録できる
  - OSSの利用者は変化があれば検知できる

https://www.sigstore.dev/how-it-works



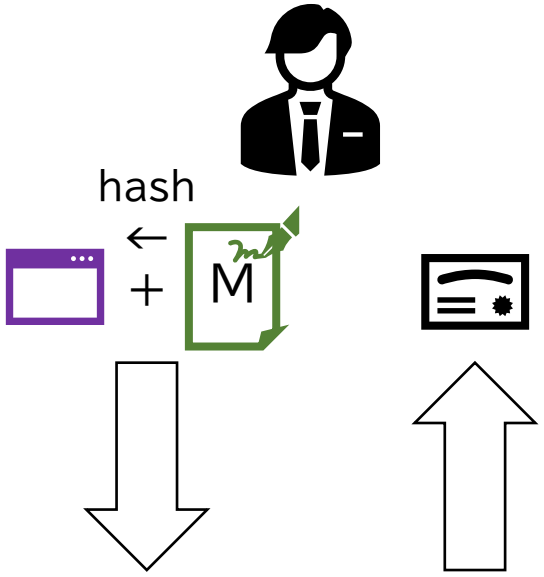
サプライチェーン攻撃を防ぐ取り組み

- 利用しているソフトウェアとそのバージョンを把握する
  - 誰が開発・ビルドしたどのソフトウェア、ライブラリなどを利用しているか
  - サプライチェーンの中でどのように取り扱われたか
  - ➢透明性、完全性、信頼性を確保する手段が求められている→SCITT
- 利用者の手元で意図した通りに動作しているか
  - ソフトウェアのインストール・実行前に改ざんがないことを検証する→SUIT
  - 改ざんされていない、意図した実行環境上で動作している→TEEP



SUITとは

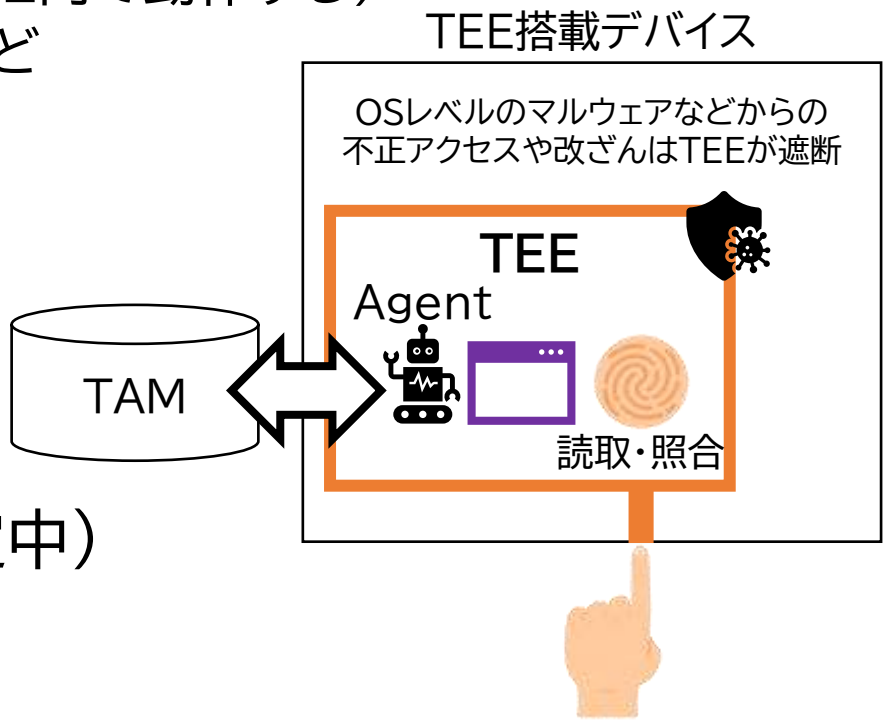
- Software Updates for Internet of Things
  - ソフトウェアアップデートを安全に行うための標準
  - IoT機器でも動作しやすいよう軽量＆パースしやすい
- **SUIT Manifest**
  - インストール＆実行手順を格納したメタデータ
  - 開発者の署名やハッシュ値によって実行前に検証可能
  - メタデータ部分は軽量なCBORを採用し数百バイト程度
- **SUIT Report**
  - インストール＆実行結果を格納するログ
  - デバイスの署名によって開発者が検証可能(策定中)



①SUIT Manifestを受領
②Manifestの署名を検証
③アップデート内容を受領
④アップデート内容の
ハッシュ値を検証
⑤インストール＆実行
⑥結果をReportに格納

TEEPとは

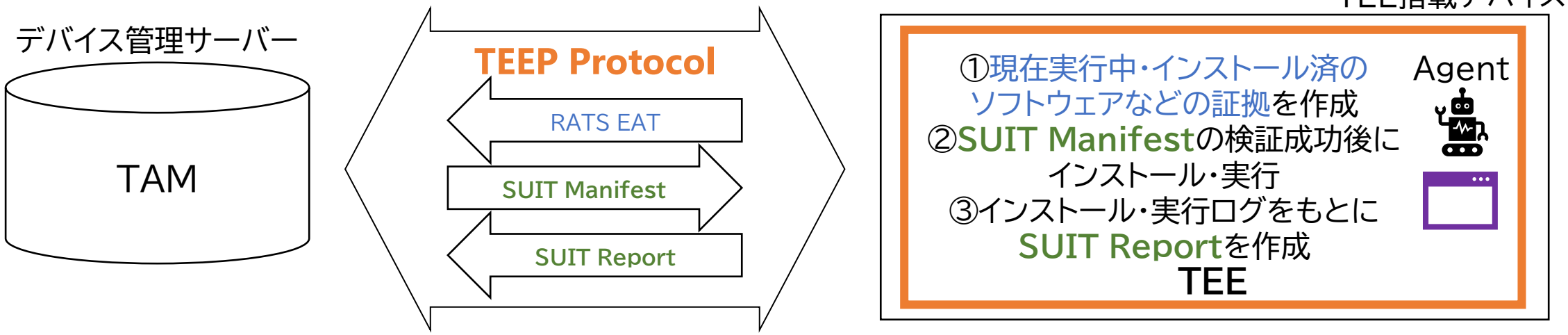
- Trusted Execution Environment Provisioning
  - TEE＝外部からの改ざんや読み込みから保護された隔離実行環境
    - TEE内のコードのみがTEE内のメモリを読み書きできる→完全性の確保
    - （デバイス開発者などによって認証されたコードのみがTEE内で動作する）
    - 実装例はIntel SGX、Arm TrustZone、AMD SEVなど
  - その初期化・パーソナライズ・アップデートを行う
- TEEP Protocol⇔
  - デバイス管理サーバー（TAM）と
    TEE内管理クライアント（Agent）の通信プロトコル
  - TEEを搭載したデバイスであること、
    Agentが改ざんされていないことなどを検証（策定中）
  - TEE用のソフトウェアなどをAgentに伝達する



TEE利用例：生体認証プログラム

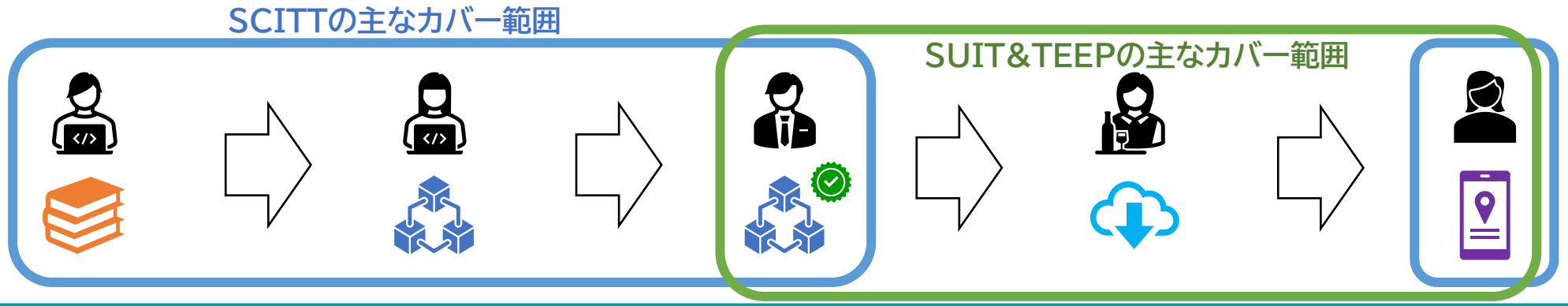
TEEP&SUIT

- TEEP Protocolを通して証拠とソフトウェアなどを伝達する
  - TEE側状態の証拠一覧はデバイスの秘密鍵を用いて署名(→信頼性)、送信
  - TAMは証拠をもとにインストールするソフトウェアを決定、SUIT Manifest送信
  - AgentはSUIT Manifestを検証し実行、ログをSUIT Reportにして返信
    - TEEを利用することで、改ざんされていないソフトウェアを実行したことが確認できる
- これらの**証拠**はデプロイ時の完全性・信頼性確保に利用可能
  - **SCITT**のTransparency Serviceに登録して透明性確保も可能



まとめ

- 利用しているソフトウェアとそのバージョンを把握する
  - 誰が開発・ビルドしたどのソフトウェア、ライブラリなどを利用しているか
  - サプライチェーンの中でどのように取り扱われたか
  - ➢透明性、完全性、信頼性を確保する手段が求められている→SCITT
- 利用者の手元で意図した通りに動作しているか
  - ソフトウェアのインストール・実行前に改ざんがないことを検証する→SUIT
  - 改ざんされていない、意図した実行環境上で動作している→TEEP



補足資料

OSSを介した攻撃

- 開発者になりすました攻撃者が作成したコードを取得・実行してしまう
  - 漏洩した開発者の秘密鍵を使用して、第三者が署名を作成する
  - 漏洩した開発者のtokenを使用して、第三者が頒布する
- 開発者が作成したコード自体に脆弱性・マルウェアが含まれる
  - 開発者が作成したコードに脆弱性が混入する
  - 開発者が意図的にマルウェアを仕込む
    - 有名プロジェクトの開発を引き継いだ攻撃者が仕込む場合もありうる
  - マルウェア入りのものが同名あるいは類似の名前で頒布される
    - Homograph attack(類似文字で読み間違い狙い←気付かずコピペしてしまう)
    - Typosquatting attack(タイプミス狙い)
    - Dependency confusion attack(意図しない場所からの依存解決狙い) など
- 開発者が開示したOSSの情報から情報を得る
  - ソースコードの利用＆頒布などで著作権表示などが必要なライセンスが多い
  - GPLなど一部のライセンスは依存側のソフトウェアもOSS化する義務がある

OSS関連の有名な事件例

時期	OSS名	分類	内容
(1984年)	(UNIX)	開発者の悪意	Cコンパイラに細工をし、一部のUNIXにバックドアを仕掛けたことがあると公表（Thompson Hack）
2014年3-4月	Struts 1&2	脆弱性混入	最新版Struts 2に脆弱性が発覚し修正、サポート切れのStruts 1にも同じ脆弱性があり使い続けていた利用者も多数被害に
2014年4月	OpenSSL	脆弱性混入	広く利用されているOpenSSLに、ソフトウェア内のメモリが読み込める脆弱性が存在した（Heartbleed）
2015年	QEMU	脆弱性混入	クラウドなどでも利用されるソフトウェアに深刻な脆弱性が発覚
2017年1月	WordPress	脆弱性混入	修正版へのアップグレードが遅れた10万超のWebサイトが被害に
2020年8月	electorn	名前	有名なelectronに似せた名前でマルウェアが頒布された
2021年10月	UAParser.js	なりすまし	開発者のnpmパッケージ配信システムのアカウントが乗っ取られ、マルウェア入りのバージョンが頒布された
2021年12月	Log4j	脆弱性混入	広く利用されていたJavaライブラリに、攻撃が容易で任意のコードを実行できる脆弱性があった
2022年1月	colors.js	開発者の悪意	開発者が意図的に無限ループを仕込んだ
2022年12月	PyTorch	名前	悪意のあるコードを含む同じパッケージ名のものが公開され、そちらが優先されてしまった (Dependency confusion)

SCITTに関連した組織の動向

時期	名前	出来事
2010年2月	SPDX	FOSSBazaarプロジェクト内のドキュメントがSPDXと呼ばれるようになる
2011年8月	SPDX	SPDX 1.0：利用するソフトウェア（主にOSS）のライセンスを列挙
2013年	OpenChain	発足、サプライチェーン内でOSSライセンスが遵守されることを目指す
2016年	OpenChain	OpenChain 1.0：OSSのLICENSEコンプライアンスのための手引き
2019-2020年	OpenChain	SPDX Lite：SPDXのサブセットを定義、軽量で可読性も高い
2020年8月	OpenSSF	発足、異業種連携でOSSのセキュリティ向上に取り組む
2020年12月	OpenChain	OpenChain 2.1をISO/IEC 5230:2020として標準化（SPDXなどを使っても良い）
2021年3月	Sigstore	プロジェクト発足、直後にLinux Foundationのプロジェクトになる
2021年5月		米国バイデン大統領がサイバーセキュリティ大統領令（E.O. 14028）
2021年7月		米国NTIAがSBOMの必要要件を発表
2021年8月	SPDX	SPDX 2.2.1がISO/IEC 5962:2021として標準化、SBOMの国際標準
2022年3月	IETF	IETF113 SecDispatchにてSCITT構想発表
2022年10月	IETF	SCITT WG正式発足、CharterをIETFが承認
2023年後半（予）	OpenChain	OpenChain Security Assurance SpecificationをISO/IECで標準化予定

SCITT用語の定義

- Artifact: a physical or non-physical item that is moving along the supply chain.
- Statement: any serializable information about an Artifact. … For example, a statement may represent a Software Bill Of Materials (SBOM) that lists the ingredients of a software Artifact, …
- Receipt: embeds cryptographic evidence that the Claim is recorded in the Registry. …
- Transparency Service: an entity that maintains and extends the Registry, and endorses its state. A Transparency Service is often referred to by its synonym Notary.
- Verifier: an entity that consumes Transparent Claims, verifying their proofs and inspecting their Statements, either before using their Artifacts, or later to audit their provenance on the supply chain.
- Auditor: an entity that checks the correctness and consistency of all Claim registered by a TS (a specialization of Claim Consumer).

https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture#section-3

Sigstore: ハイジャック対策

- 今までの問題
  - 開発者の秘密鍵が漏洩すると、第三者が署名できた←ここを解決
  - 開発者のTOKENが漏洩すると、第三者が頒布できた
  - 開発者に悪意があると、マルウェアを仕込むなどができた
- Keyless Signing:リリースのたびに署名鍵を作成する
  - Open ID Connectを使って認証を受けた後
  - そのOIDC tokenを使って署名鍵・検証鍵を生成する
  - 開発者はOIDC tokenと検証鍵の登録を依頼する→証明書を受け取る