ソフトウェアサプライチェーンの現状 ２０２５

脅威の進化が、ソフトウェアの前提を覆す

はじめに

ソフトウェアサプライチェーン全体を適切に管理し、安全性を確保することは、信頼できるソフトウェアリリースを実現するための基盤です。しかし、実際にそれを実行するのは容易ではありません。

15年以上にわたり開発・セキュリティチームを支援してきたJFrogは、現在の組織が直面する脅威と課題を深く理解しています。AIの普及により、これらの課題はさらに加速し、多くのDevSecOps　チームが「この変化にどう対応すべきか」という問いに直面しています。

本レポートでは、数百万人のユーザーから得られたJFrog Platformの利用データ、JFrogセキュリティリサーチチームによるCVE分析、さらにセキュリティ・開発・運用の専門家1,400名を対象とした調査結果を組み合わせ、その問いに答えます。

分析を通じてソフトウェアサプライチェーンに潜む既存のリスクと新たなリスクを明らかにし、2025年に向けて必要となる対策を提示します。

本レポートが、皆様の取り組みに役立つことを願っています。フィードバックがありましたら、data_report@jfrog.com までお送りください。

概要

ソフトウェアサプライチェーンは、これまでにないスピードで進化しており、組織は新たな脅威にかつてない速さで直面しています。リスク対策において重要なのは、単にツールを増やすことではありません。ツールとプロセスをシンプルにし、効率的に運用することが求められます。迅速な開発や新技術の採用を実現するためには、「より多く」ではなく「より賢く」取り組むことが不可欠です。

「がむしゃらに働くのではなく、スマートに働く」という考え方が示すように、ツールチェーンとプロセスの簡素化は、迅速な意思決定や新しいテクノロジーの導入、そして競争優位性の確立を目指す組織にとって、最適なアプローチです。

ソフトウェアサプライチェーン より大きく、より高速に、より複雑に

オープンソースエコシステムの成長に、減速の兆しは見られません。イノベーションを追求する組織は、最新テクノロジを活用するために迅速に動いています。

- 64%の組織が7種類以上、44%が10種類以上のプログラミング言語を使用しています。これは前年比で、それぞれ53%および31%の増加です。
- パブリックリポジトリは引き続き成長しています。2024年には、Docker Hub に190万のイメージ、Hugging Face に100万のイメージが新たに追加されました。
- 一般的な組織は、年間458件の新規パッケージを導入しています。これは平均すると、月に38件の新規パッケージを導入している計算になります（開発者数により異なります）。

リスクの増加と透明性の減少

脆弱性の潜在的な影響を理解することは依然として複雑な　作業であり、これはリスクという氷山の一角に過ぎません。

- NVD（National Vulnerability Database）には膨大な　バックログが存在しますが、それでも新たな脆弱性の　公開は止まることなく続いています。2024年には、33,000件を超える新たなCVEが公開され、前年比で27%増加しました。
- JFrogセキュリティリサーチチームは、パブリックレジストリ上で25,229件の漏洩したシークレットトークンを検出しました。これは前年比64%の増加で.あり、そのうち6,790件がアクティブな状態でした。
- JFrogセキュリティリサーチチームが183件の重要な　CVEを詳細に分析した結果、63件のCVEはJFrog Cloudのお客様のスキャン対象アプリケーションでは悪用できないことが判明しました。

セキュリティ対策は基本から

多くの組織はさまざまなセキュリティフレームワークを導入し、数多くのセキュリティツールを活用しています。しかしその一方で、基本的なベストプラクティスが見落とされているケースも少なくありません。

- 71%の組織が、開発者によるインターネットからのパッケージ直接ダウンロードを許可しています。
- 73%の組織が7種類以上、49%が10種類以上のセキュリティソリューションを使用しています。
- これは前年の47%および33%から増加しています。
- コードレベルとバイナリレベルの両方でスキャンを実施している組織は、43%にとどまっています。
- 40%の回答者は、本番環境で実行されているソフトウェアの出所を完全に可視化できていません。

AIの導入は新たな段階へ

本番環境にサービスを導入するための選択肢はこれまでになく増えています。その結果、組織は新たなリスクや懸念事項への対応を迫られています。

- 2024年、Hugging Face には100万を超える新しいモデルとデータセットが追加されました。一方で、悪意のあるモデルも6.5倍に増加しています。
- 64%のチームがホスト型モデルへ移行しています。しかし、約半数の組織は独自モデルとオープンソースモデルの両方を、何らかの形でセルフホスティングしています。
- 37%の組織が、承認済みモデルのリストを手作業で管理しています。モデルアーティファクトの利用管理はいまだ手動プロセスに依存しています。

拡大し続ける
ソフトウェアサプライチェーン

現代ソフトウェアサプライチェーンは、グローバルに広がり、複数のテクノロジーやソースを組み合わせて構成されています。
主要なテクノロジーエコシステムには、毎年数百万もの新しいパッケージやライブラリが追加されています。

ソフトウェア開発組織は現在、これまでにない数のプログラミング言語と、それぞれに対応するパッケージエコシステムを活用しているのです。

従来のテクノロジーが引き続き広く使われる一方で、急速に成長するオープンソースエコシステムの採用は、大きな可能性と同時に新たなリスクももたらしています。

本レポートでは、これらの可能性とリスクについて詳しく分析します。

開発組織で使用しているプログラミング言語の数

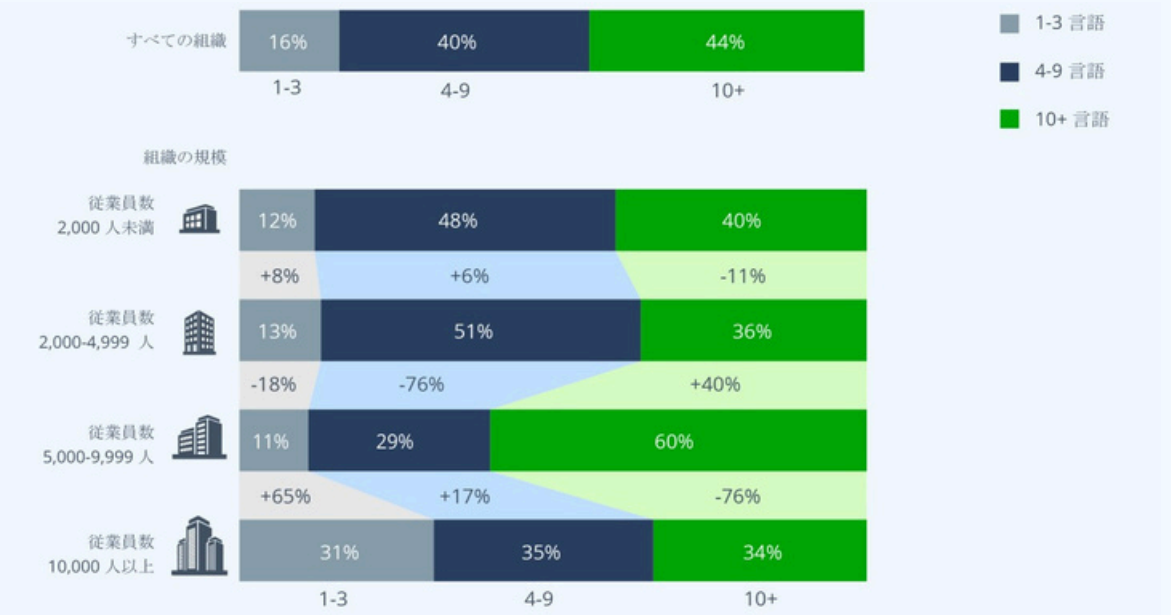


図 1.1. ソフトウェア開発組織で使用しているプログラミング言語の数は?
(委託調査、2024 年)

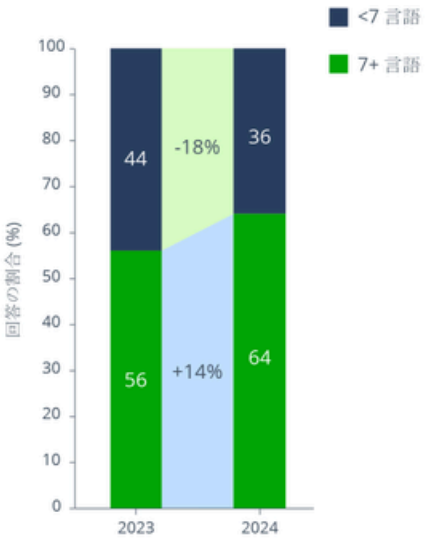


図 1.2

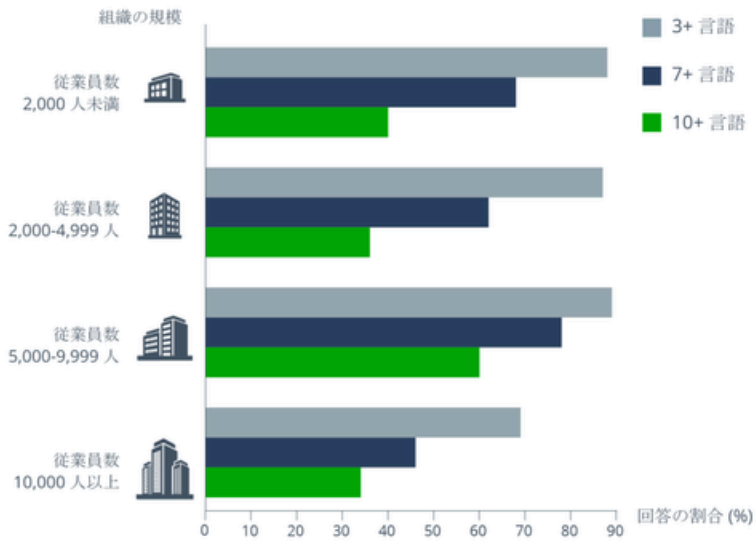


図 1.3

ソフトウェア専門家の約3分の（64%）が、所属組織で7種類以上のプログラミング言語を使用していると回答しています。昨年は回答者の半数強（56%）にとどまっていました。

この増加は、ソフトウェアサプライチェーン全体の複雑性が急速に高まっていることを示しています。

組織の規模が大きくなるほど、使用する言語の数も増加する傾向にあります。
一方で、従業員数が1万人を超える規模になると、使用言語数は逆に減少する傾向が見られます。

この変化は、組織が転換点を迎え、開発管理においてより積極的なアプローチを取り始めたことを示しています。

特定のテクノロジーへの標準化を進めることで、技術のスプロール（無秩序な拡散）を抑制する必要性に気づいたと考えられます。

また、大規模組織では実績のあるレガシーアプリケーションを継続利用するケースが多く、追加のテクノロジーエコシステムを必要とする新規プロジェクトが比較的少ないことも要因と考えられます。

パッケージ タイプ別の年間の新規パッケージ数

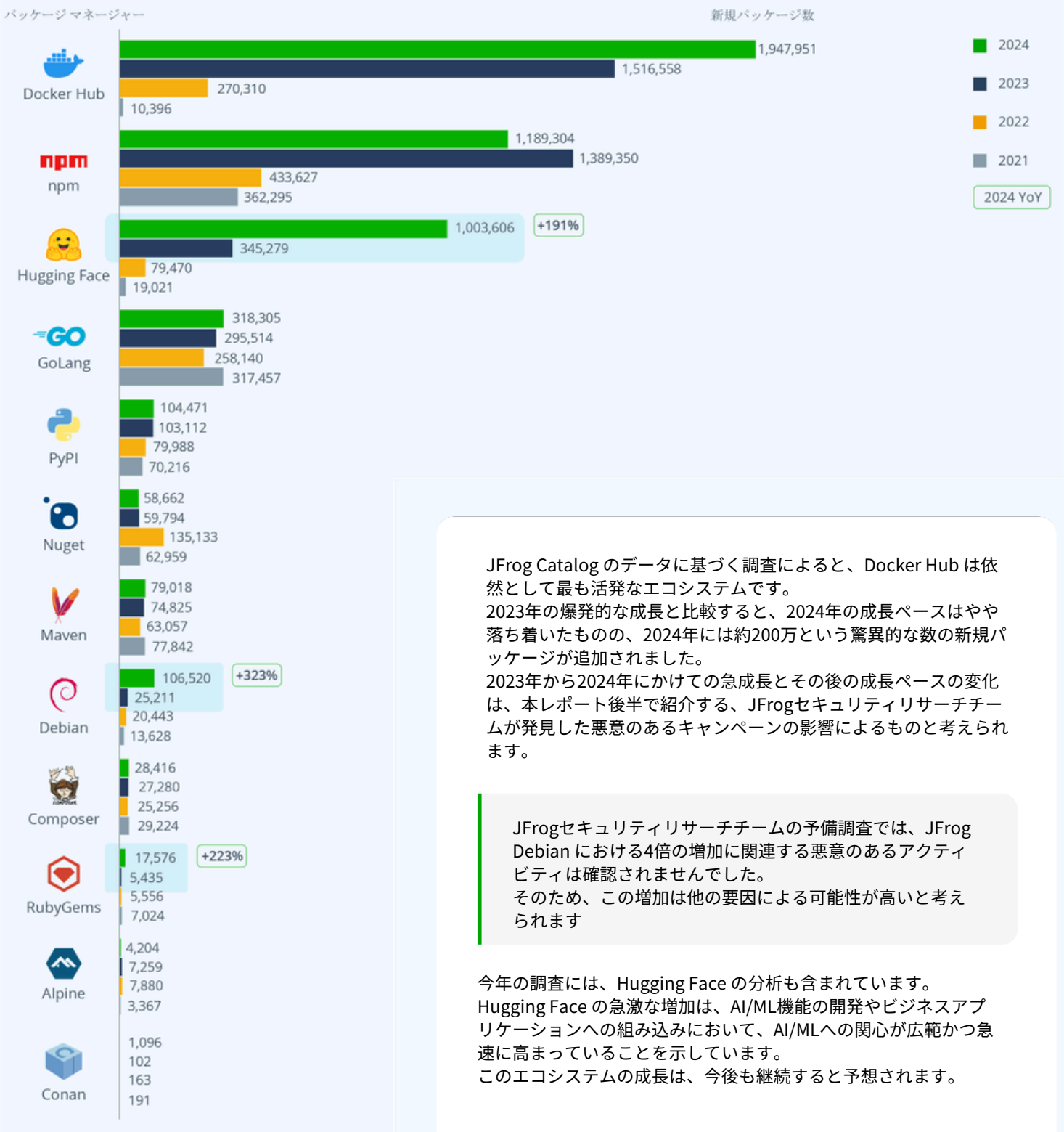


図 2 パッケージ タイプ別の年間の新規パッケージ数
(JFrog Catalog データベース、2024 年)

JFrog Catalog のデータに基づく調査によると、Docker Hub は依然として最も活発なエコシステムです。
2023年の爆発的な成長と比較すると、2024年の成長ペースはやや落ち着いたものの、2024年には約200万という驚異的な数の新規パッケージが追加されました。
2023年から2024年にかけての急成長とその後の成長ペースの変化は、本レポート後半で紹介する、JFrogセキュリティリサーチチームが発見した悪意のあるキャンペーンの影響によるものと考えられます。

> JFrogセキュリティリサーチチームの予備調査では、JFrog Debian における4倍の増加に関連する悪意のあるアクティビティは確認されませんでした。
> そのため、この増加は他の要因による可能性が高いと考えられます

今年の調査には、Hugging Face の分析も含まれています。
Hugging Face の急激な増加は、AI/ML機能の開発やビジネスアプリケーションへの組み込みにおいて、AI/MLへの関心が広範かつ急速に高まっていることを示しています。
このエコシステムの成長は、今後も継続すると予想されます。

組織で使用している主要パッケージ開発ツール

パッケージ タイプ	リクエスト*	リポジトリ数	アーティファクト
Maven	33.52%	104,955	2,567,881,564
npm	30.45%	48,549	674,010,130
Docker	15.45%	112,366	2,264,459,098
YUM	2.68%	14,669	20,785,724
PyPI	2.68%	22,352	66,838,230
Helm	1.61%	26,125	13,231,209
Nuget	1.45%	28,497	131,164,087
Debian	1.35%	8,184	8,066,185
Conan	1.33%	3,420	143,404,846
Gradle	0.99%	9,073	102,198,342
RubyGems	0.93%	3,736	46,728,889
Go	0.75%	9,034	16,511,299
OCI	0.47%	862	8,662,480
Cargo	0.13%	1,261	526,851
Sbt	0.12%	2,239	14,908,497
Helm OCI	0.07%	1,633	201,440
Ivy	0.06%	2,283	31,786,069
Composer	0.05%	2,413	614,957 675,684
Terraform	0.03%	3,566	33,812,836
Opkg	0.02%	529	1,538,832
Conda	0.02%	2,168	1,010,616
P2	0.02%	316	166,878
Pub	0.01%	363	1,345,299
Swift	0.01%	524	111,231
Alpine	0.01%	1,550	2,973,045
Cocoapods	<0.01%	1,400	816,170 1,692
Cran	<0.01%	2,403	150,462 7,326
VCS	<0.01%	273	395,004 44,161
Chef	<0.01%	1,530	4,470 17,758
Vagrant	<0.01%	680	12,638
Terraform Backend	<0.01%	2,307	
Bower	<0.01%	985	
Ansible	<0.01%	107	
Puppet	<0.01%	1,530	
Hugging Face	<0.01%	551	

*第 4 四半期の 570億のリクエストのうち各タイプのリクエストの割合

2024年末の第4四半期にスナップショットを作成し、JFrogがサポートする35以上のテクノロジータイプの中で、最も人気の高いテクノロジーをより正確に把握しました。
npm、Docker、Maven などの確立されたテクノロジーエコシステムは引き続き広く利用されています。
一方で、YUM と Cargo の人気も大幅に上昇しています。

近年、政府機関がメモリ安全な開発を強く推進していることもあり、Cargo の人気は着実に高まっています。
Rust が今後どこまで普及するのか、Java のように広く使われる存在になるのかは、まだ分かっていません。

OCI、Helm OCI、Terraform の使用量の増加も注目に値します。
JFrog は2024年に Terraform 専用リポジトリを導入し、多くのお客様に既にご活用いただいています。
これは、コンテナやその他のテクノロジーエコシステムにおいて、オープン標準への関心が高まっていることを示しています。

この流れを受け、リポジトリを拡張し、OpenTofu をネイティブにサポートしました。

一般的なテクノロジーの利用状況は、業界によって異なります。

- 自動車・IoT企業： Maven（バックエンド）、npm（フロントエンド）、Conan（組み込み開発）、Docker、PyPI（AI/ML）などを活用し、これらを汎用パッケージ（tar/zip）としてまとめて配布するケースが多く見られます。
- AI/ML・ロボティクス企業： Hugging Face やTensorFlow などのパブリックリポジトリから取得した PyPI パッケージやMLモデルを活用し、コンテナや汎用パッケージ（tar/zip）として管理しています。また、Hugging Face や JFrog の機械学習リポジトリなど、ネイティブなモデルリポジトリの採用も進んでいます。
- 保険・金融・小売業界
  - Maven、npm、Docker などを組み合わせて活用しています。近年はAI/MLの普及に伴い、競争力維持のため PyPI やMLモデルの活用も進んでいます。

*JFrogの機械学習リポジトリは ２０２５年１月に導入されたためこのレポートのデータには含まれていません。、

図 3 組織で使用されているテクノロジ別のアクション数、リポジトリ数、および保存アーティファクトの総容量

人気の高いライブラリ

ランク	Docker	Maven	PyPI	npm
1	library/alpine	org.slf4j:slf4j-api	urllib3	@types/node
2	library/node	commons-io:cozmons-io	requests	semver
3	library/python	commons-codec:commons-codec	certifi	minimatch
4	library/nginx	org.ow2.asm:asm	charset-normalizer	glob
5	library/redis	com.fasterxml.jackson.core:jackson-core	setuptools	electron-to-chromium
6	library/busybox	com.google.guava:guava	idna	lru-cache
7	library/postgres	com.fasterxml.jackson.core:jackson-databind	packaging	caniuse-lite
8	library/ubuntu	com.fasterxml.jackson. core:jackson-annotations	typing-extensions	acorn
9	library/openjdk	org.apache.commons:commons-compress	wheel	debug
10	library/debian	org.apache.commons:commons-lang3	PyYAML	@babel/parser
11	grafana/grafana	org.codehaus.plexus:plexus-utils	python-dateutil	strip-ansi
12	library/golang	junit:junit	numpy	browserslist
13	library/hello-world	org.apache.httpcomponents:httpcore	click	@babel/types
14	library/maven	org.apache.httpcomponents:httpclient	MarkupSafe	tslib
15	library/docker	com.google.code.findbugs:jsr305	pytz	resolve
16	library/eclipse-temurin	com.google.errorprone:error_prone_annotations	cryptography	commander
17	curlimages/curl	commons-logging:commons-logging	cffi	qs
18	library/mongo	net.bytebuddy:byte-buddy	importlib-metadata	@babel/code-frame
19	library/centos	org.objenesis:objenesis	zipp	@babel/generator
20	library/amazoncorretto	org.apache.maven:maven-artifact	attrs	chalk

図4. JFrog Cloud (SaaS) での Docker、Maven、PyPI、npm のダウンロード数上位 20 のパッケージ (JFrog データベース、2024年)

多くのパブリックレジストリでは、パッケージのダウンロード数などの指標が公開されています。

しかし、これらの数値はビルドのたびにパッケージが取得されることなどの影響を受けるため、実際の利用状況を正確に表しているとは限りません。

そこでJFrogでは、何千もの顧客アカウントにおけるSaaS環境へのリクエストを分析し、実際に使用されているライブラリを推定しました。

Dockerの上位20イメージに、主要なOSや開発言語が含まれているのは自然な結果です。これらは多くの場合、ベース（親）イメージとして利用されています。

また、1つを除いてすべてがDocker公式または検証済みの発行元によるイメージである点は安心材料です。

これらが継続的にメンテナンスされていることを示しています。

特に「Docker hello-world」が上位に入っていることは、コンテナが広く普及し、デモ・検証・学習用途として日常的に使われていることを示しています。

Maven、PyPI、npmの上位20パッケージにも大きな意外性はありませんでした。

ただし、それらが開発者によって直接選ばれたのか、依存関係として自動的に取り込まれたのかまでは分かりません。

たとえば、Apache Commons Compress は Maven の利用ランキングで9位に入っています。
このライブラリを詳しく見ると、Apache Commons IO、Apache Commons Codec、ASM、Apache Commons Lang など（2位・3位・4位・10位）に直接依存していることが分かります。

このことは、アプリケーションを構成する各コンポーネントを把握する重要性を示しています。

特定のコンポーネントに脆弱性が見つかったり、侵害されたり、ソフトウェアサプライチェーンから失われたりした場合、その影響範囲を正しく理解する必要があります。

そのため、アプリケーションに含まれるソフトウェアアーティファクトの最新のインベントリを維持することが不可欠です。

多くの場合、これは SBOM（Software Bill of Materials） の形式で管理されます。

組織における新規OSSパッケージ導入のペース

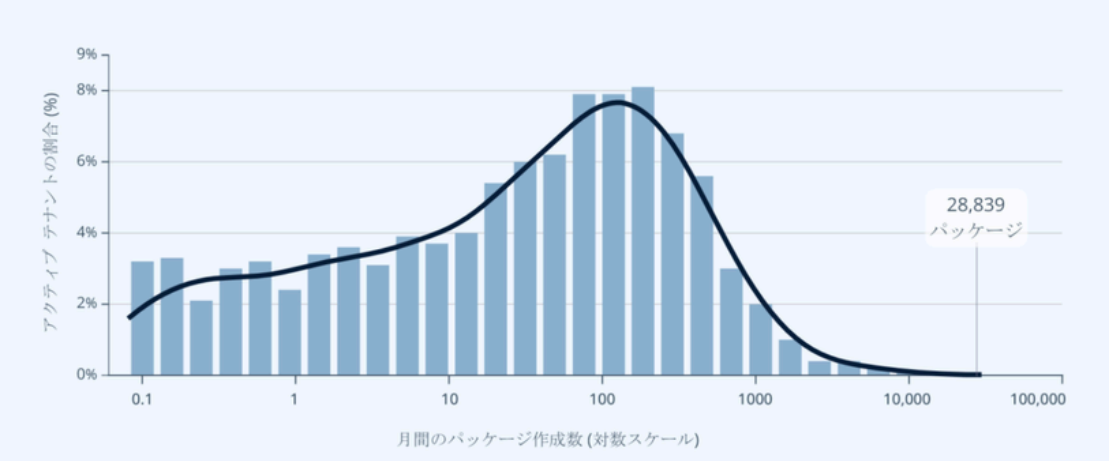


図5. 2024 年にアクティブ テナント向けに毎月作成された新規パッケージの分布
(JFrog データベース、2024 年)

2024年、JFrog Cloud などのクラウドネイティブサービスを利用する組織は、合計200万を超える新規パッケージをソフトウェアサプライチェーンに導入しました。

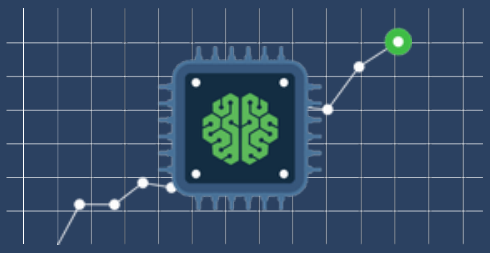
平均的な組織は年間約700件のパッケージを導入していますが、この数値は一部のヘビーユーザーによって押し上げられています。

最大規模の組織では年間346,000件の新規パッケージが導入されている一方、中央値の組織では231件と、より管理しやすい水準にとどまっています。

パッケージを導入していない組織を除外すると、新規パッケージ数の中央値は458件（毎月38件）に増加します。

データから、この数値が典型的な組織を最もよく表していると考えられます。
新規パッケージが1日1件以上のペースで追加される環境では、セキュリティ、運用リスク、ライセンスコンプライアンスをどのように管理するかが大きな課題となります。

重要なポイント



AIの急速な拡大

利用可能な AI/ML コンポーネントは急速に増加しており、コミュニティや企業がエコシステムへ積極的に貢献しています。

例えば NVIDIA は NIM や NVLM を公開しました。
また、組織が自社製品へのAIサービスの組み込みを進めていることは、JFrog ユーザーがすでに500以上の Hugging Face リポジトリを作成していることからも明らかです。

オープンソースのモデルやデータセットの活用方法と、そのセキュリティをどのように確保するかについて、明確なポリシーと戦略を定めることが重要です。

アプリケーションと開発の保護は最優先

米国政府をはじめとする各国の政策機関は、より安全な開発言語やフレームワークの利用を推進しています。

JFrog のデータでも Rust／Cargo の利用が増加しており、組織がアプリケーションの再設計を進めたり、セキュリティを重視した新規プロジェクトを開始したりしている可能性が示されています。

また、OCI の人気の高まりは、これまで利用してきたオープンソース技術が非公開化・商用化されたり、ライセンスが必要になったりすることへの懸念が背景にあると考えられます。

リスクの急増

組織の3分の2が7種類以上、ほぼ半数が10種類以上のプログラミング言語を使用しています。
その結果、複数の言語・チーム・脅威要因に対して一貫したパイプラインを維持する必要があり、組織のリスクは大きく増加しています。
本番環境へ安全にアプリケーションを提供するためには、開発段階から各エコシステム特有の脆弱性や攻撃リスク、構造の違いを考慮する必要があります。

迅速な対応で攻撃を防ぐ

変化の速い組織では、1日1件以上の新しいパッケージやバージョンが導入されています。
そのため、ソフトウェアサプライチェーンに取り込まれるコンポーネントの安全性を確保するには、自動化された効率的なプロセスが不可欠です。
組織がさらなるスピード向上を目指し、開発者とセキュリティチームが新しい課題に取り組み続ける中で、導入されるパッケージのペースは今後も増え続けると考えられます。

ソフトウェアサプライチェーンのリスク増大

組織は、悪意ある攻撃者との終わりのない競争に直面しています。しかも、その原因となる課題は増え続けています。

- CVE（既知の脆弱性）
- 悪意のあるパッケージ
- オープンソース ライセンスのリスク
- 運用リスク（パッケージ管理の不備、EoL など）
- シークレットの漏洩
- 設定ミス・人的ミス

分析によると、開発者やセキュリティ担当者が現在使用しているツールは、状況によっては有効に機能する一方で、逆にリスクを生む可能性もあります。

たとえばAIコード アシスタント は、設定や運用が不十分な場合、意図せず　リスクを高めてしまう可能性があります。

今年は、NVD（National Vulnerability Database：米国NISTが運営する脆弱性データベース）が、数か月にわたり新規CVEの分析と属性付けを実施できませんでした。

その結果、大量のバックログが発生しています。この影響により、本セクションで示している CVSS（脆弱性の深刻度スコア）や CWE（脆弱性の種類）の前年比比較には、一部不完全なデータが含まれています。

このバックログは、ライブラリやCVEの増加に対して、現在の仕組みが追いついていないという業界の課題を示しています。

組織は、増え続けるリスクを持続可能な方法で管理する必要があります。
さらに、現在の米国の政治状況を踏まえると、NVDおよびNISTの将来が不確実であることも懸念されています。

テクノロジー別・パッケージ別の脆弱性

2024年には、世界中のセキュリティ研究者によって約33,000件の新しいCVEが公開されました。

これは2023年と比べて27%の増加であり、公開されるCVEの数は年々増え続けています。新しいオープンソースパッケージが増加していることを考えれば驚くべきことではありません。

しかし、CVEの増加率（前年比27%）がパッケージの増加率（前年比24.5%）を上回っている点は、重要な警告と言えるでしょう。

図6,1. 2024年にパッケージタイプ別で公開されたCVE数

まず注目すべき点は、Debian におけるCVEの大幅な増加です。

2024年はエコシステムへの貢献パッケージ数が4倍に増加しており、CVEの増加自体は想定内と言えます。

幸いにも、重大・高リスクに分類されるCVEの割合は比較的低い水準にとどまりました。

2023年と同様に、Maven、npm、PyPI、そして今年新たに追加されたConanでは、CVE総数が減少したエコシステムもある一方で、重大CVEの割合は依然として高い状態が続いています。

ただし、年末時点のデータ全体を確認すると、継続的なリスクは存在するものの、npm、Maven、PyPI のリスク水準はやや低下していることが分かります。

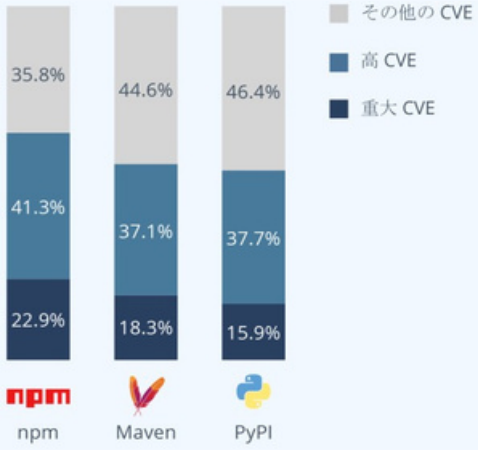


図 6.2. 2024年末時点における主要エコシステムの重大CVE・高CVEの割合

削除されたパッケージと非推奨になったパッケージの総数

* この年のデータは未収集

図 7. 削除されたパッケージと非推奨になったパッケージの総数
(JFrog データベース、2024 年)

パッパッケージはテクノロジーエコシステムに追加されるだけでなく、削除されることもあります。このデータで特に注目すべき点は、2023年と2024年に削除された Composer パッケージの数です。JFrogセキュリティリサーチチームの手動レビューにより、以下の可能性が考えられます。

削除されたパッケージの多くは、GitHub のソースコードリポジトリが「利用不可」となっていました。これは、作成者によって削除または非公開にされた可能性を示しています。

- 削除されたパッケージの中には、名称変更のみが行われたケースもあり、packagist がこれを「削除」として扱っている可能性があります。
- 一部のパッケージは削除され、「放棄」とマークされていますが、その基準は明確ではありません。
- packagist は2024年、GitHub リポジトリが無効になったパッケージを削除する新たな自動化を導入したと考えられます。

データソースによって、削除されたパッケージの報告方法は異なります。

この情報を提供するものもあれば、提供しないものもあります。

一部のエコシステムでは、利用可能なすべてのデータを分析し、時系列で比較しています。

ただしこの方法では、最初のデータ収集以前に削除されたパッケージは対象外となります。

また、定期的な更新の中で削除情報を報告するエコシステムもあります。

そのため、削除されたパッケージについて常に完全な情報を取得できるとは限りません。

最も一般的な脆弱性の種類

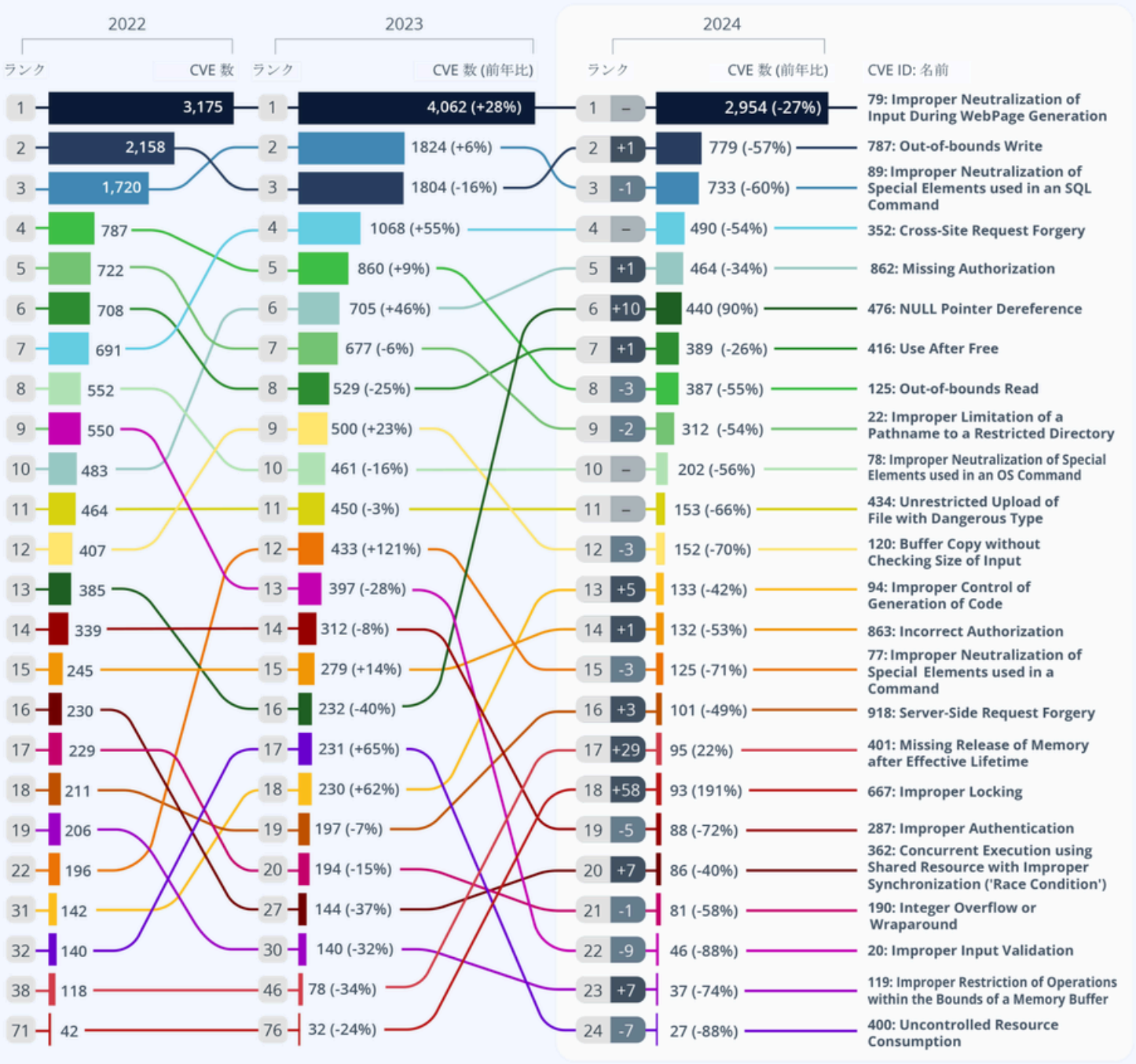


図8：2024年に公開された主な脆弱性（2023年・2022年・2021年との比較）

2024年には、243種類のCWEがCVEに割り当てられました。

上位3位は前年と変わらず、クロスサイトスクリプティング、境界外書き込み、SQLインジェクションでした。

一方で、上位20位には新たに3種類の脆弱性が加わり、いずれも急増しています。

CWE-401：有効期限後のメモリ解放漏れ

2024 17位
2023 46位

CWE-362：共有リソースの不適切な同期による同時実行（レースコンディション）

2024 20位
2023 28位

CWE-119：メモリバッファ境界内での操作制限の不備

2024 23位
2023 30位

SASTで検出される代表的なCWEである、クロスサイトスクリプティング、境界外書き込み、SQLインジェクションへの対策として、ソースコードを自動化されたSASTツールで継続的にスキャンすることが推奨されます。

特に境界外書き込みは、C/C++などのメモリセーフでない低水準言語に特有の問題です。

米国政府の提案 のとおり、高水準言語への移行はこれらのリスク低減に有効です。CWEの前年比トレンドは、一時的な要因や偶発的なイベントの影響を受けやすい点にも注意が必要です。
例えば、NVDのバックログは2024年のCWE分布に大きく影響している可能性があります。

すべてのCVEが整理・登録された後、数値は変化する可能性があります。

また、低水準言語と高水準言語の利用割合や、特定技術の人気の変化も、脆弱性の傾向に影響を与えます。

より長期的（例：20年規模）に分析することで、より意味のある傾向が見えてくるでしょう。

2024年に注目されたCVEの主な影響

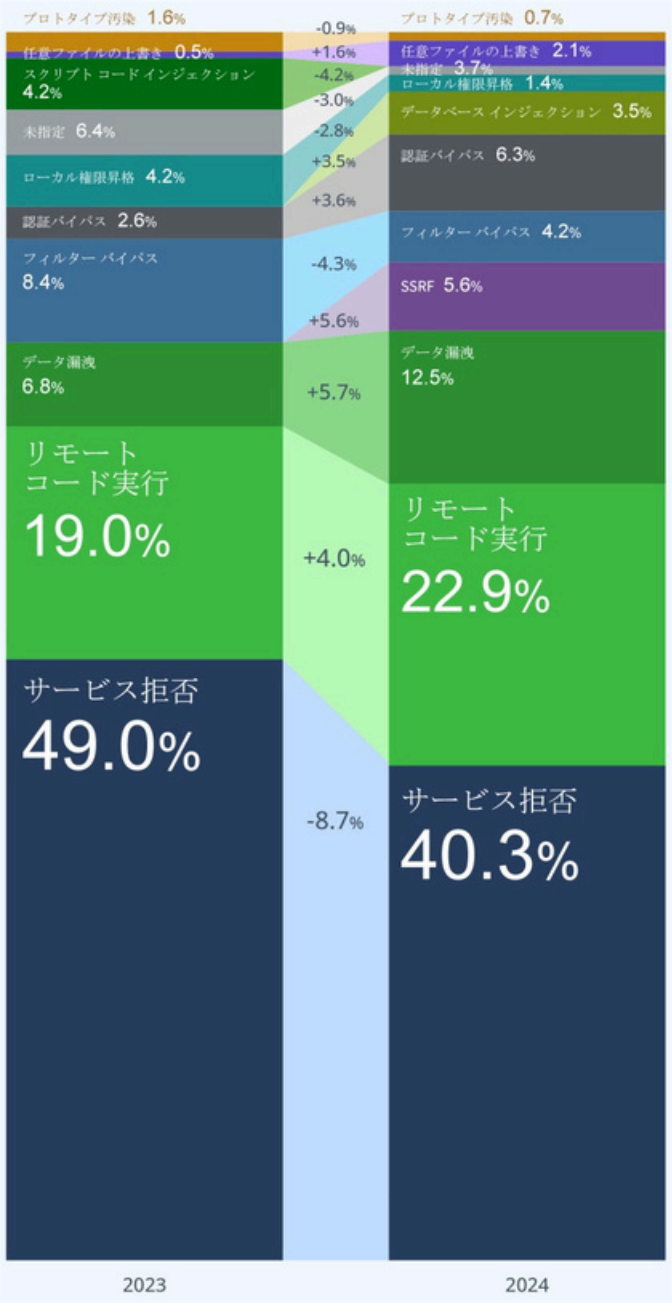


図9. 2023 年と 2024 年の注目度の高い CVE の一般的な脆弱性の影響

今年、JFrogセキュリティリサーチチームは、JFrog顧客への関連性と潜在的影響を基準に、140件以上の注目度の高いCVE（HPCVE）を分析しました。

最も多かった影響は、サービス拒否（DoS）で58件を占めています。2位はリモートコード実行（RCE）で、割合は18.9%から22.9%へと増加しました。

RCEは攻撃者に深刻な制御権を与える可能性があるため、その割合が増加していることは懸念されます。

3位は引き続きデータ漏えいで18件を占め、その割合は大きく増加しました。

また、認証バイパスや、今年新たに追加されたSSRF（Server-Side Request Forgery）も増加しています。
一方で、フィルターバイパスは減少しました。

JFrogセキュリティリサーチチームは、調査対象CVEの優先順位を決める際に複数の要素を考慮しています。

まず、JFrogの顧客に関連性の高いテクノロジーに重点を置き、CVSSスコア7.5以上の「高」または「重大」な脆弱性を優先します。

CVSSスコアが利用できない場合は、機械学習による重大度予測も活用します。

さらに、重大度が「中」または「低」であっても実際に悪用されている脆弱性やメディアで大きく
報じられた脆弱性は優先的に取り上げます。

ソフトウェアサプライチェーンにおける脆弱性の重大度

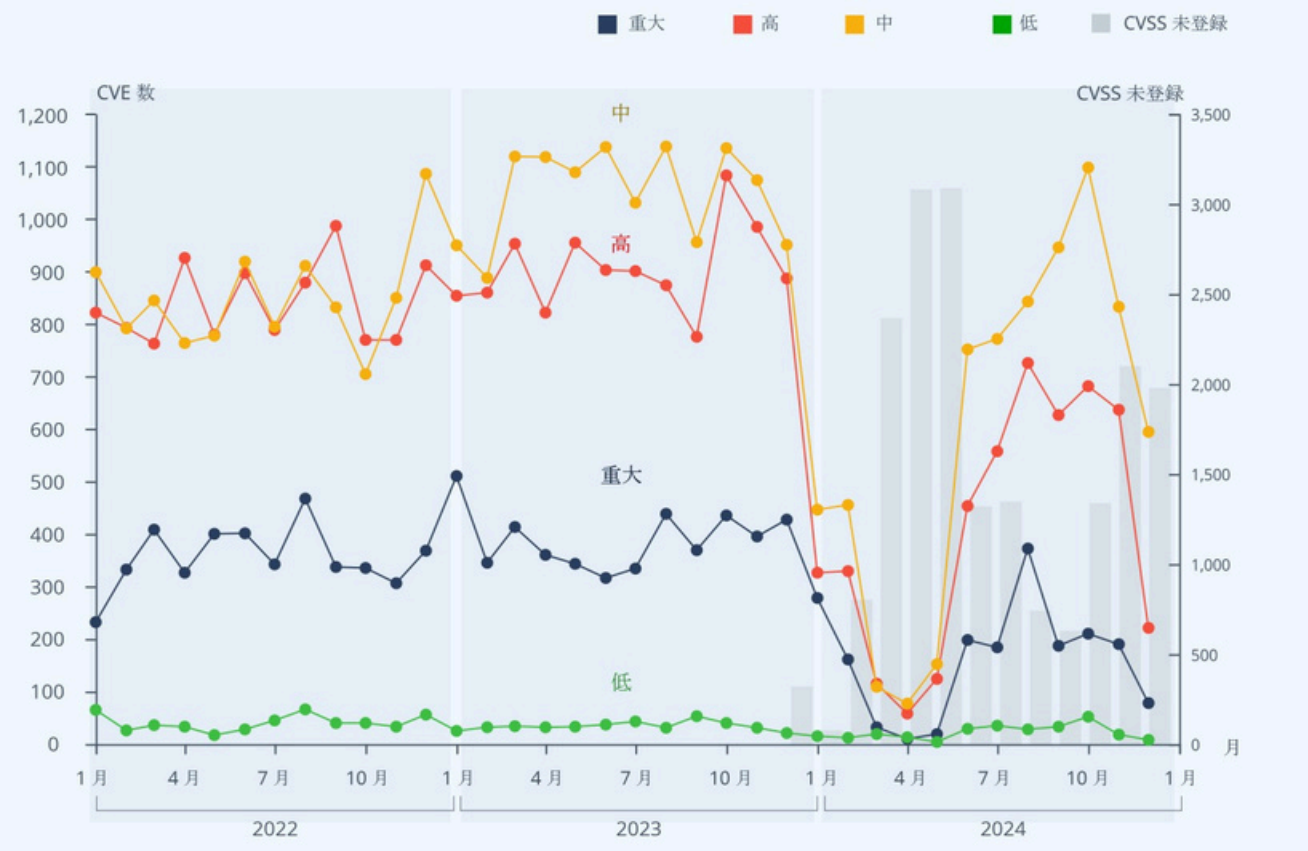


図10.1：過去3年間の月別・重大度別CVE数（NVD）

2024年は、CVSSの割り当て数が年半ばに大きく減少し、その後回復していますが、この変化は誤解を招く可能性があります。
原因は、NVDが2024年2月に人員削減を伴う組織再編を実施し、CVE調査を一時停止したことにあります。

この状況を受け、NVDは2024年6月にCISAと契約し、調査支援を開始しました。
この混乱がなければ、CVSSの推移はより安定していたと考えられます。

現在、NVDはリソース不足に対応するため、CVSSスコアリングの多くを「Authorized Data Publishers（ADP）」と呼ばれる外部機関に委託しています。
現時点での唯一のADPはCISAです。

JFrogセキュリティリサーチチームは、CISAによるスコアリング傾向を継続的に分析しており、初期分析ではNVDよりも.高めの重大度が付与される傾向が確認されています。

今後、他のADPが参加した場合、CVEスコアの一貫性が損なわれる可能性もあり、組織が対策の優先順位を決定する際の重要な考慮事項となります。

利用可能なNVDデータから見ると、全体傾向は依然として一貫しており、
「中」「高」重大度のCVEが最も多く、「重大」がそれに続き、「低」は比較的少数となっています。

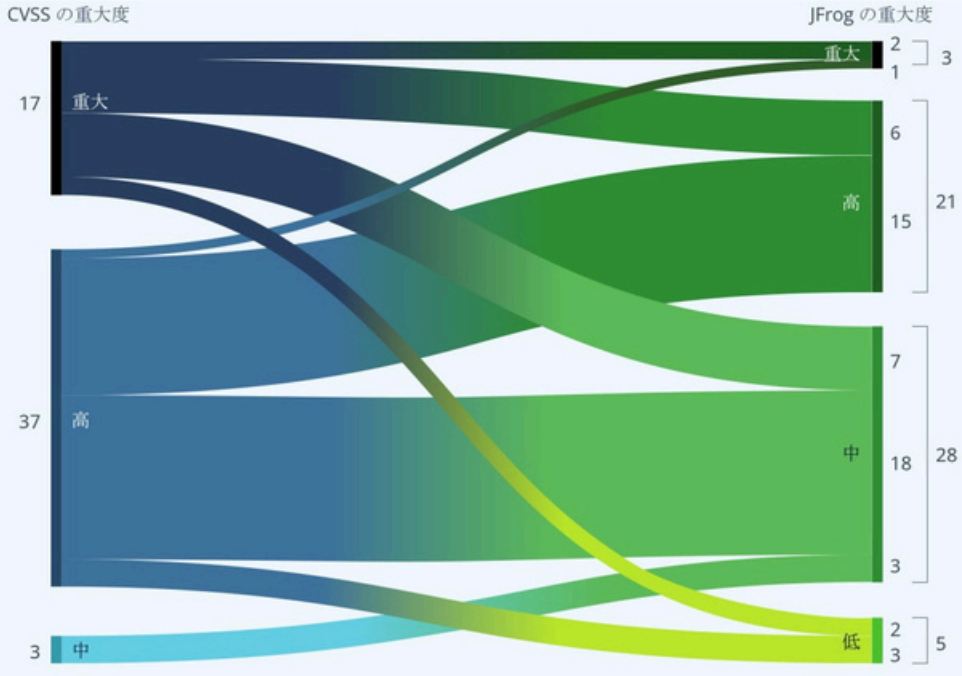


図10.2：CVE重大度スコアの比較（NVD評価とJFrog独自評価）

すべてのCVEが、スコア通りの影響を持つとは限りません。

JFrogセキュリティリサーチチームはCVEを定期的に評価し、実際の影響を踏まえた独自の重大度を付与しています。

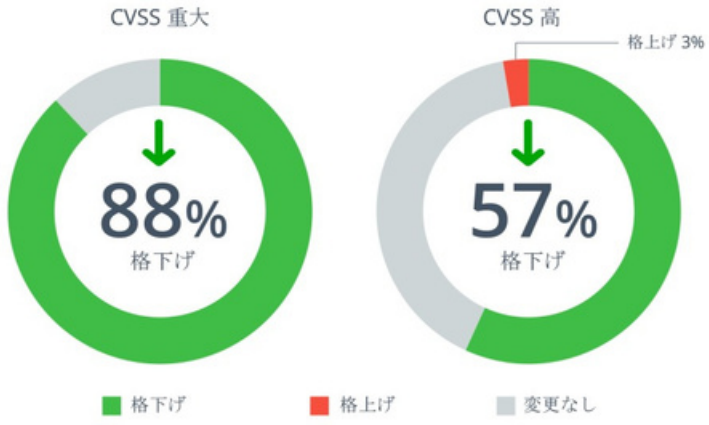
JFrogの評価では、脆弱性を悪用するために必要な構成条件も考慮されます。

一方、CVSSは「悪用された場合の影響」を評価するものであり、「実際に悪用　される可能性」は考慮されません。

そのため、標準設定では悪用が極めて困難なケースでも高スコアが付与されることがあります。

こうした「過大評価」の傾向は年々懸念されています。

スコアリング手法自体は変わっていないため、高スコア化が進むと誤検知が増えリスク判断を難しくする可能性があります。



140件の注目度の高いCVEを対象としたJFrogの分析では、
重大CVEの88%、高CVEの57%が、
CVSSスコアが示すほど深刻ではないことが判明しました。

図10.3

注目度の高い CVE の適用性評価

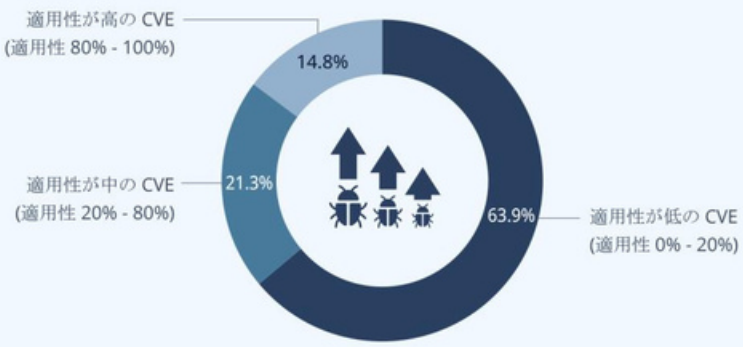


図 10.4 注目度の高い 183 件の CVE における適用性レーティング

CVE に重大度スコアを割り当てるだけでは、特定のソフトウェア製品に対する実際の影響を正確に評価することはできません。

JFrog のセキュリティリサーチチームは、単に独自の重大度スコアを付与するだけでなく、脆弱性が実際に悪用される可能性に影響を与える構成条件や実行要件も詳細に評価しています。

その結果、JFrog は、対象となるソフトウェア環境が実際に悪用条件を満たしているかどうかを判断するための「適用性（Applicability）スキャナー」を開発しました。

このアプローチにより、理論上のリスクではなく、実環境に基づいた現実的なリスク評価が可能になります。

JFrog セキュリティリサーチチームは、JFrog のお客様の間で特に利用頻度の高いコンポーネントおよびテクノロジに関連する高・重大レベルの CVE に焦点を当て、2024 年に公開された 183 件の CVE（CVE-2024-*）について適用性スキャナーを作成しました。

上記のグラフは、JFrog の顧客環境において「適用可能（＝実際に悪意のある攻撃者に悪用され得る）」と判断された CVE と、「適用不可（＝悪用条件を満たさない）」と判断された CVE の比率を示しています。

分析の結果、2024 年に JFrog Xray でスキャンされたアーティファクトのうち、適用性が 80% を超える ― すなわち実際に悪用される可能性が非常に高いと評価された CVE は、わずか 27 件（15%）にとどまりました。

一方で、適用性が 0%〜20% と評価され、悪用される可能性が低いと判断された CVE は 117 件（64%）を占めました。

この結果は、CVSS スコアのみでは実環境における真のリスクを正確に反映できないことを示しています。

CVE-2024-24792 は、適用性が非常に高い（99.6%）代表的な例の一つです。この脆弱性は、Go プログラミング言語の TIFF　解析パッケージに存在し、イメージのアップロードや処理機能を持つアプリケーションにおいて広く利用されている実装がトリガー条件となります。

特に、ユーザーが制御可能な TIFF 画像を処理するために当該ライブラリを使用している場合、細工された画像ファイルによって脆弱性が発動し、アプリケーションのパニック（クラッシュ）を引き起こす可能性があります。

そのため、この脆弱性は理論上のリスクにとどまらず、多くの実環境において現実的な影響を及ぼす可能性があるケースと評価されています。

J一方、CVE-2024-45490（C 言語で実装された XML パーサー「Expat」に関連）は、適用性が最も低い部類に分類される CVE の一つであり、実際に適用可能と判断されたケースは 10% 未満にとどまりました。

この脆弱性を悪用するには、攻撃者が Expat の API 関数 XML_ParseBuffer() に渡される「len」 パラメーターを意図的に操作する必要があります。

しかし実際の開発環境では、開発者は通常、stat() や XML_GetBuffer() といった関数を用いて XML 文書の長さを適切に取得・設定しています。そのため、「len」パラメーターが攻撃者によって任意に操作される状況は、現実的には極めて発生しにくいと考えられます。

このように、理論上は重大に分類され得る脆弱性であっても、実環境における悪用可能性は大きく異なる場合があります。

注目度の高いCVE の適用性分類

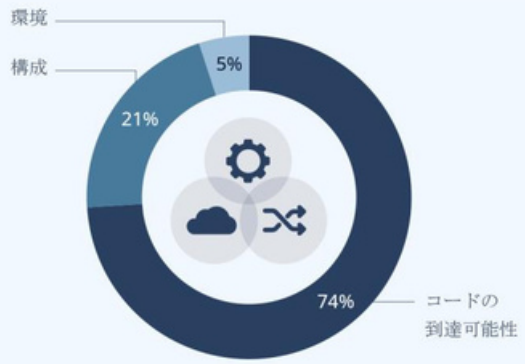


図 10.5. 2024 年の CVE の適用性タイプ
(CVE と JFrog データベースを使用した JFrog 独自のセキュリティ調査)

JFrog セキュリティリサーチチームは、これらの CVE がアプリケーション内で実際に到達可能か、さらに悪用可能かどうかについても詳細に分析しました。

脆弱性の適用性や悪用可能性を判断するには、従来の呼び出し到達可能性分析によって脆弱なコードへの到達可否を確認するだけでは不十分です。アプリケーションやライブラリの構成設定、さらには実行環境となるオペレーティングシステムの条件も併せて評価する必要があります。

このような包括的アプローチにより、潜在的なリスクをより正確に評価することが可能になります。

一方で、単に到達可能なコードを特定するだけでは、脆弱性の実際の悪用可能性に大きく影響する重要な要素を見落とす恐れがあります。

たとえば、有名な「Sudoedit バイパス」の脆弱性である CVE-2023-22809 の適用性を判断するには、Sudo の設定ファイル（sudoers）を確認し、特定の非デフォルト設定が有効になっているかどうかを調査する必要があります。

この脆弱性は、脆弱なコンポーネントである sudo がスタンドアロンのユーティリティとして動作する点に特徴があります。sudo はアプリケーションのコードから直接呼び出されるライブラリではないため、通常のコード到達可能性分析によって脆弱性の適用可否を判断することはできません。

そのため、このケースでは実行環境の構成や設定条件を評価することが不可欠となります。

一部の悪意あるパッケージは、特に高いリスクをもたらす

2024 年版レポートでは、npm エコシステムにおける悪意あるパッケージの蔓延について取り上げました。

2024 年末に実施した主要パッケージエコシステムの分析においても、悪意あるパッケージの検出件数という観点では、npm が依然として最も深刻な状況にあることが確認されました。

一方で、今年は Hugging Face エコシステムにアップロードされた悪意あるモデルの数が約 6.5 倍に増加しました。これは、同プラットフォームの急速な人気拡大を踏まえれば、ある意味で予測可能な動きとも言えます。

以下では、JFrog セキュリティリサーチチームが特に注目した 3 件の悪意ある攻撃事例を紹介します。

XZ Utils バックドア
3 月 29 日、主要な Linux ディストリビューションで広く利用されている圧縮ライブラリ「XZ Utils」に、不正なリモート SSH アクセスを可能にする悪意あるコードが含まれていたことが報告されました。
この高度に巧妙なバックドアは、バージョン 5.6.0 および 5.6.1 に仕込まれており、OpenSSH サーバーの認証処理に関連するルーチンを改変することで、特定の攻撃者が認証前に任意のペイロードを実行できる状態を作り出していました。
その結果、影響を受けるシステムでは、攻撃者がリモートから完全な制御権を取得できる可能性がありました。

出典 >

Docker Hub を標的とした大規模マルウェア攻撃
Docker Hub を標的とした最近のマルウェアキャンペーンでは、実際のコンテナーイメージを含まない「イメージレス」リポジトリが数百万件規模で作成されました。これらのリポジトリには、悪意あるメタデータのみが含まれていました。
特に深刻なのは、公開リポジトリ全体の約 20%（約 300 万件）が、自動化されたアカウントによってアップロードされた有害コンテンツをホストしていた点です。その内容は、海賊版ソフトウェアの宣伝スパムから、マルウェア配布やフィッシングサイトへの誘導に至るまで多岐にわたります。
この事例は、コンテナーエコシステムにおいても、信頼できると見なされてきたプラットフォームが攻撃の温床となり得ることを示しています。

出典 >

Hugging Face
AI モデルの監視活動により、Pickle ファイルのロード時に任意コードを実行する悪意あるモデルファミリーが確認されました。
これらのモデルは、ロード処理の過程でコードを実行し、攻撃者に対してコネクトバックシェルを確立します。その結果、バックドア経由で侵入したマシンの制御権を取得できる状態が作り出されます。
このような侵害はユーザーに気付かれにくく、影響を受けた環境において重大なリスクをもたらします。場合によっては、重要システムへの不正アクセスや、大規模なデータ侵害、さらには企業スパイ活動につながる可能性があります。

出典 >

コードに潜むその他のリスク要因

CISO やアプリケーションセキュリティチームは、オープンソースコミュニティから導入するコンポーネントを慎重に精査する必要があることを既に認識しています。

しかし、包括的なアプリケーションセキュリティを実現するうえで、注意すべきリスク要因はそれだけではありません。

設定ミスと人的エラー — ヒューマンファクターの影響

2024 年には、設定ミスやその他の人的エラーに起因するデータ漏洩インシデントが数多く発生しました。クラウド環境やアプリケーションの設定不備、アクセス制御の誤設定、公開範囲の誤認識などが原因となり、本来保護されるべき情報が外部からアクセス可能な状態に置かれる事例が相次いでいます。

2024年4月
Home Depot は、サードパーティの SaaS ベンダーにおけるデータ漏洩を起因として、約 1 万人の従業員の個人情報が流出するインシデントに見舞われました。
出典 >

2024年9月
Fortinet は、Azure 上の SharePoint サイトに保存されていた顧客データに対する不正アクセスにより、約 2,000 人分の情報がインターネット上に漏洩するインシデントを公表しました。
出典

2024年8月
数千の Oracle NetSuite 顧客が、SuiteCommerce や Site Builder で構築した外部向けストアから、認証されていないユーザーに機密データを漏洩していました。
出典 >

2024 年 9 月
ローコード SaaS プラットフォームである Microsoft Power Pages において、アクセス制御の設定ミスにより、数百万人に影響を及ぼす可能性のある重大なデータ漏洩が発覚しました。
出典 >

2024年9月
1,000 を超える ServiceNow のエンタープライズインスタンスにおける設定ミスにより、機密性の高い企業情報を含むナレッジベース（KB）記事が外部ユーザーや潜在的な脅威アクターに公開されていたことが判明しました。
出典 >

2024 年 12 月、フォルクスワーゲンの自動車ソフトウェア子会社 Cariad に関連するデータ漏洩が発覚しました。本件は、2024 年に発生した SaaS の設定ミス事例の中でも特に影響の大きいものの一つです。
このインシデントにより、約 80 万台の電気自動車から収集されたデータが漏洩し、その中には正確な車両位置情報や、運転者の氏名と関連付けられる可能性のある情報が含まれていました。
出典 >

バイナリアーティファクトに含まれるシークレットの漏洩状況

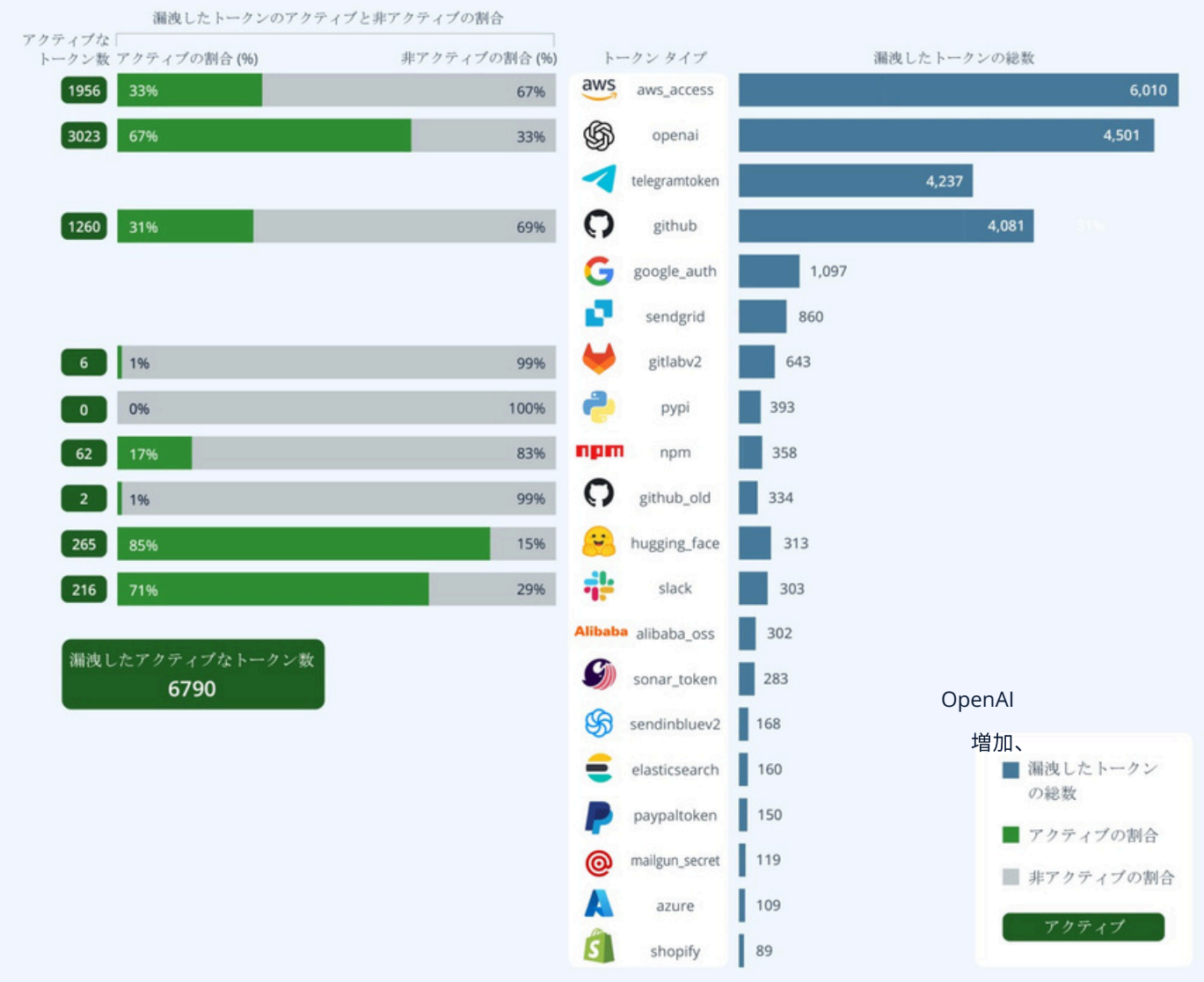


図 11.2　漏洩トークンの主な種類と前年比

JFrog セキュリティリサーチチームは、Docker Hub、npm、PyPI といった主要なオープンソースソフトウェアレジストリに保存された数百万件のアーティファクトをスキャンしました。今年はデータ収集時点で有効な「アクティブトークン」が発見された場所を記録しました。漏洩したシークレットの総数は前年比で 66% 増加しました。漏洩トークン数の上位カテゴリは前回レポートと同様でしたが、AWS（70% 増）、OpenAI（103% 増）、Telegram（62% 増）、GitHub（82% 増）と、いずれも大幅な増加が確認されました。

漏洩したシークレットの総数は前年比で 66% 増加しました。漏洩トークン数の上位カテゴリは前回レポートと同様でしたが、AWS（70% 増）、OpenAI（103% 増）、Telegram（62% 増）、GitHub（82% 増）と、いずれも大幅な増加が確認されました。さらに、GCP トークンも前年比 86% 増と顕著に増加しています。

発見されるトークンの種類はほぼすべてで増加傾向にあり、シークレット漏洩の拡大が継続していることが明らかになりました。

今年新たにスキャナーに追加された Hugging Face トークンは、オープンソースモデルやデータセットの急速な普及を反映するものです。特筆すべき点として、Hugging Face トークンは他のトークンと比較してアクティブ率が最も高く、約 85% が有効な状態で発見されました。

JFrog セキュリティリサーチチームは、データ収集時点で 6,790 件のアクティブなシークレットを特定しており、悪意のあるアクターが企業システムへアクセスするための潜在的な入口が依然として多数存在することを示しています。

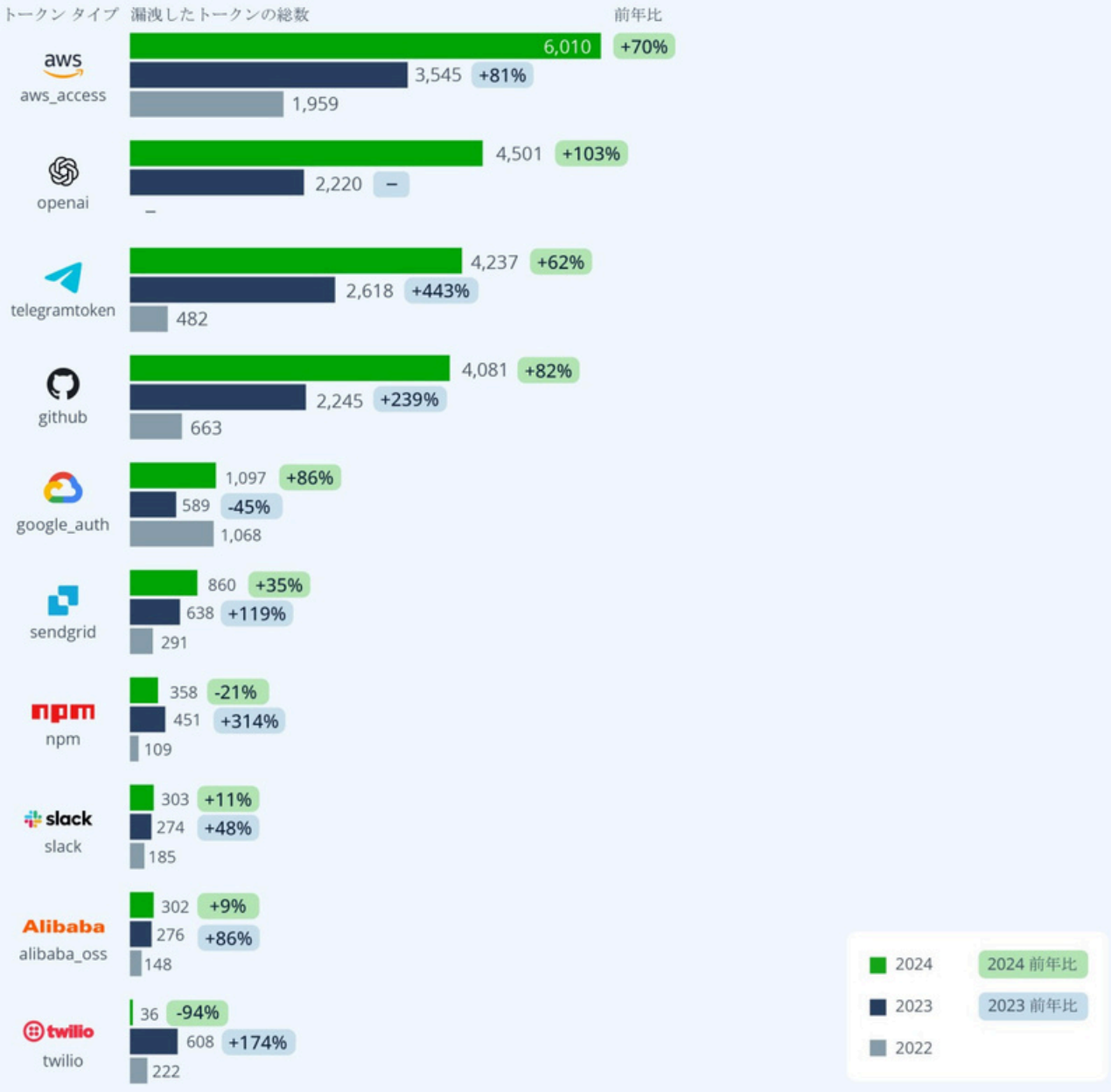


図 11.2　主要な漏洩トークンの件数および前年比

Q あなたの組織では、コードベースに残されたシークレットや漏洩トークンを検出するためのセキュリティ対策を講じていますか？（委託調査、2024 年）

多くの組織は、シークレットが悪意のあるアクターや第三者の手に渡らないよう、具体的な対策に投資しています。しかし、対策を講じていない、または実施状況を把握していないと回答した組織も 15% に上りました。

漏洩シークレットの増加傾向や、発見されたほぼすべてのトークン種別で件数が増加している事実を踏まえると、自らを脆弱な状態に置いている組織が一定数存在することは看過できません。

シークレット漏洩はどの程度深刻なリスクとなり得るか

2024 年 6 月、JFrog セキュリティリサーチチームは、Docker Hub 上で公開されていたコンテナーイメージ内に、Python、PyPI、および Python Software Foundation の GitHub リポジトリに対する管理者権限を持つアクセストークンが含まれていることを発見し、報告しました。

JFrog セキュリティリサーチチームはコミュニティへの貢献として、Docker Hub、npm、PyPI などの公開レジストリを継続的にスキャンし、悪意あるパッケージや漏洩したシークレットを特定しています。発見した情報は、攻撃者に悪用される前に、速やかに関連メンテナーへ報告されます。

今回のケースは特に重大でした。漏洩したトークンが悪意あるアクターの手に渡った場合、PyPI 上のパッケージだけでなく、Python 言語そのものに悪意あるコードが注入される可能性がありました。
チームは漏洩トークンを特定後、直ちに PyPI のセキュリティチームへ報告し、わずか 17 分でトークンは無効化されました。

今回は大惨事を回避できましたが、本件は「たった 1 つのシークレット漏洩」が壊滅的な影響をもたらし得ることを示しています。PyPI は世界最大級のソフトウェアレジストリの一つであり、もし悪用されていれば、その影響は無数のユーザーとプロジェクトに及んでいた可能性があります。
高度に管理され、広く利用されている Python / PyPI のような基盤であっても例外ではありません。本事例は、あらゆるプラットフォームや言語に潜在的なリスクが存在し、その脅威が常に現実的であることを示しています。

重要なポイント

データの冗長性の必要性

NVD の脆弱性データのみに依存している組織やセキュリティツールは、過去 1 年間に発生した大規模なバックログ遅延の影響により、重要な CVE 情報を見逃すリスクがあります。
これらの遅延により、新たに公開された脆弱性やその潜在的影響がデータベースへ迅速に反映されず、組織が認識しないまま新たな脅威にさらされる可能性があります。
このリスクを軽減するには、ベンダーアドバイザリや脅威インテリジェンスフィードなど、複数の情報源で NVD データを補完することが不可欠です。また、組織はセキュリティスキャンツールが参照するデータソースを評価し、十分なカバレッジと冗長性を確保する必要があります。

適用性、影響、優先順位付け

対処すべき CVE の数が増え続ける中、セキュリティチームや開発チームは、すべての脆弱性をトリアージする作業に追われ、本来注力すべき業務に支障が生じる可能性があります。
真に優先すべき脆弱性へ集中するためには、アプリケーションにおける CVE の適用性、攻撃ベクトル、そして潜在的な影響を正しく理解することが不可欠です。
JFrog セキュリティリサーチチームは、CVSS スコアに基づくリスク評価が実態よりも過大に示されているケースを継続的に確認しています。CISA が Authorized Data Publishers（認定データパブリッシャー）として CVE レコードの拡充に関与するようになって以降、この傾向はさらに顕著になっています。

シークレットの漏洩は誰にでも起こり得る

組織は、漏洩シークレットに対する保護を継続的に強化するとともに、個人プロジェクトやコミュニティ活動に取り組む開発者にも同様の保護を広げる必要があります。開発者の個人環境が侵害された場合でも、その影響が企業システムへ波及する可能性があるためです。
漏洩したシークレットやトークンが第三者に発見された場合、その影響は極めて深刻になり得ます。JFrog が発見した Python / PyPI の事例では、当該トークンの所有者が Python、PyPI、Python Software Foundation のリポジトリに対する管理者権限を有しており、大規模なソフトウェアサプライチェーン攻撃につながる可能性がありました。
Python / PyPI に起こり得ることは、あらゆる組織やプロジェクトにも起こり得ます。

悪意のあるアクターの高度化

悪意のあるアクターは、ますます創造的かつ巧妙な手法でソフトウェアサプライチェーンへの侵入を試みています。
XZ Utils バックドアの事例では、攻撃者は数年にわたり OSS 開発者としての信頼を築いたうえで、コードレビューによる検出を回避するため高度に難読化されたコードを挿入していました。
また、AI コードアシスタントが「存在しない」ライブラリを推奨する現象を悪用し、その名称に合わせて悪意あるライブラリを迅速に公開する攻撃も確認されています。これは、AI ツール自体が新たな攻撃ベクトルになり得ることを示しています。
組織は、評価の高いオープンソースプロジェクトであっても盲目的に信頼せず、「一夜漬け」で公開されたライブラリが誤ってサプライチェーンに組み込まれないよう、明確な運用リスクポリシーを策定する必要があります。

今日の組織における セキュリティ対策の 適用状況

今年は、セキュリティ、DevOps、エンジニアリング分野の専門家 1,402 人を対象にアンケートを実施しました。設問を拡充するとともに、JFrog がスポンサーを務めた他の調査レポートの結果も統合することで、ソフトウェア開発ライフサイクル（SDLC）全体を通じて、チームがどのようにアプリケーションリスクを管理しているかについて、より包括的な視点を得ることができました。

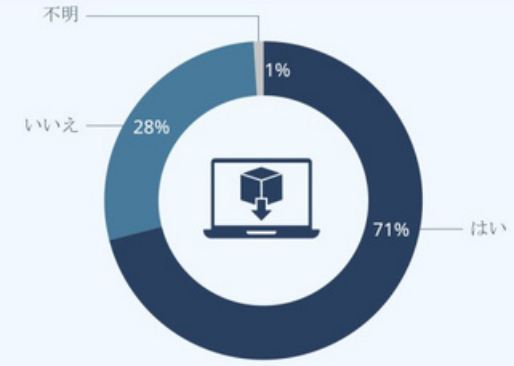
その結果、多くのチームがセキュリティフレームワークやツールを導入している一方で、サードパーティ製パッケージやライブラリをインターネットから直接ダウンロードするなど、依然としてリスクの高い行為が広く行われていることが明らかになりました。

調達制限

組織がソフトウェアサプライチェーンにおけるリスクへ対処するための最も効果的な方法の一つは、リスクの侵入そのものを未然に防ぐことです。
そのためには、開発フェーズにセキュリティを組み込む「シフトレフト」をさらに発展させ、リスクがサプライチェーンに入り込む前に遮断する「レフトオブレフト」のアプローチが求められます。

Q あなたの組織では、開発者がパブリックレジストリやその他のインターネット上のソースから、パッケージやソフトウェアコンポーネントを直接ダウンロードすることを許可していますか？（委託調査、2024 年）

71% の組織が開発者によるインターネットからのソフトウェアコンポーネントの直接ダウンロードを許可



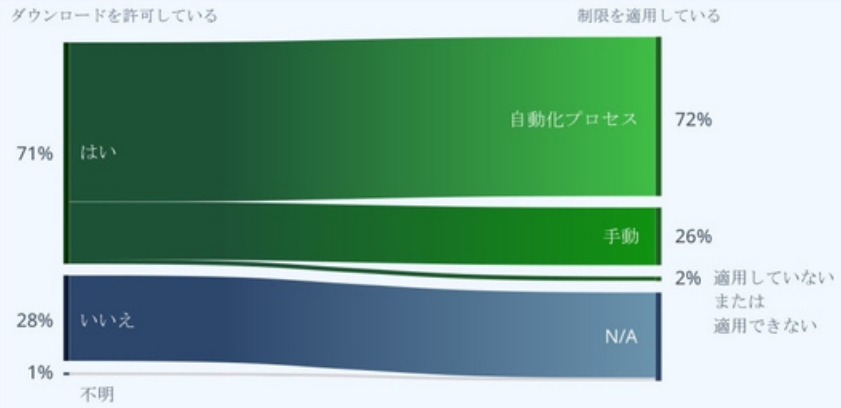
パッケージやライブラリを外部ソースから直接取得することは重大なリスクを伴います。単一の開発者端末が侵害されるだけで、組織全体が攻撃にさらされる可能性があるためです。

そのため、ベストプラクティスとしては、これらのダウンロードを制限し、信頼された内部レジストリを経由させることが推奨されます。また、直接ダウンロードを許可すると取得元の可視性が低下し、トレーサビリティの確保も困難になります。

しかし、多くの組織が直接ダウンロードを許可しているという事実は、現場にそのようなニーズが存在することを示しています。

アップストリームのパブリックレジストリをプロキシできるアーティファクト管理ソリューションは、この課題に対する有効な手段となります。プロキシ機能を備えたアーティファクトリポジトリは、サプライチェーンへ取り込まれるすべてのコンポーネントの中央制御ポイントとして機能し、可視性と統制を確保します。
このような仕組みにより、組織はコンポーネントの追跡と保護を一元化し、関連するリスクを抑制するとともに、大規模な潜在的損害を未然に防ぐことが可能になります。

Q あなたの組織では、パブリックレジストリやその他のインターネット上のソースからパッケージやソフトウェアコンポーネントを直接取得することに対する調達制限を、どのように追跡・適用していますか？（委託調査、2024 年）



制限を適用している回答者のうち、4 分の 1 以上（26%）が、パブリックレジストリやその他のインターネット上のソースからパッケージやソフトウェアコンポーネントを直接取得することに対する調達制限を、手動で追跡・適用していると回答しました。

制限を適用している回答者の 72% が自動化プロセスを導入しているという結果は、一見すると高い水準に見えます。しかし、この数字は、回答者が SDLC のどの段階を想定しているかによって解釈が変わる可能性があります。
たとえば、開発者がコードをチェックインする前にパッケージや依存関係を手動で確認し、その後 CI/CD パイプラインやリリースビルドの監査段階で自動化プロセスが実行されるケースも考えられます。

このアプローチの課題は、リスクの検出が後工程に偏ることで、開発者による手戻りが発生しやすくなる点にあります。また、本レポートで前述したとおり、SDLC の初期段階でリスクを排除できなければ、影響は拡大する可能性があります。

開発者によるインターネットからの直接ダウンロードを許可しつつ、調達制限の適用を手動で管理していると回答した組織は、制限を適用している回答者の 26%（全体の 19%）を占めました。

しかし、この方法は多大な手作業を伴い、リスクを効果的に遮断する手段とは言えません。

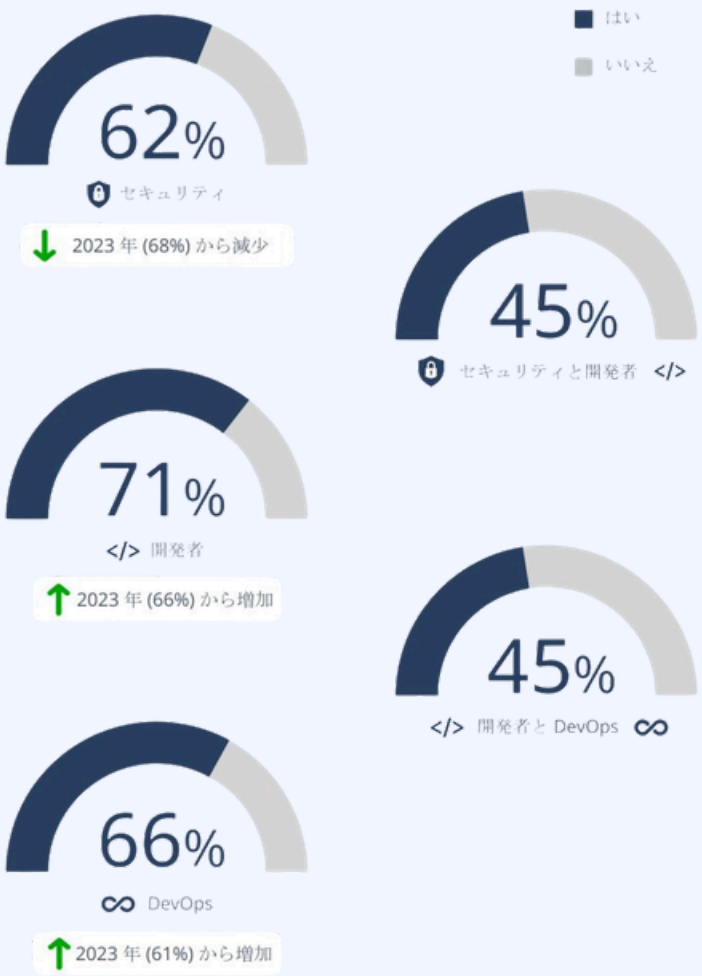
一方、直接ダウンロードを許可しながらも、調達制限の追跡・適用を自動化していると回答した組織は、制限を適用している回答者の 72%（全体の 52%）に上りました。

Q

ソフトウェアパッケージ、ライブラリ、フレームワークの最新バージョンを取得するプロセスは、セキュリティ部門と開発者のどちらが管理していますか？該当するものをすべて選択してください。
（委託調査、2024 年）

以前と比較すると、開発者が最新パッケージの管理においてより積極的な役割を担うようになっています。一方で、セキュリティ担当者は、特に使用パッケージのレビューや承認に関して引き続き重要な責任を負っています。
また、「セキュリティ担当者 + 開発者」や「開発者 + DevOps」といった複数チームで取得プロセスを管理する体制を採用している組織も見られます。

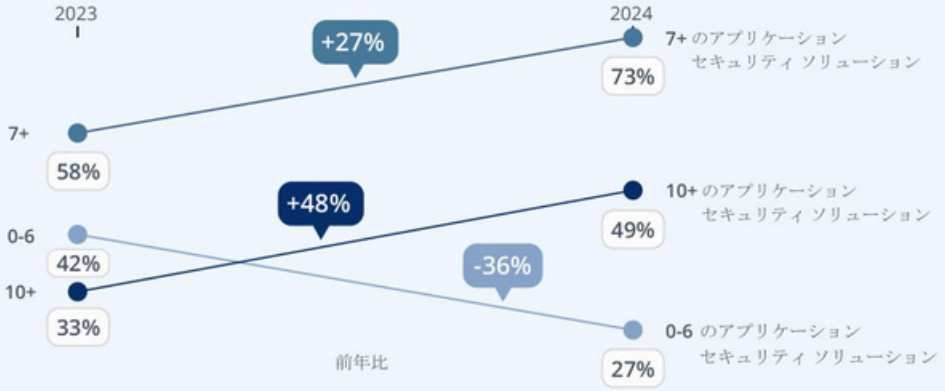
開発スピードを維持・向上させるためには、開発者が新しいバージョンのライブラリをセルフサービスで導入できる一方で、セキュリティポリシーに準拠した承認を自動化する仕組みが不可欠です。
さらに、アプリケーションセキュリティ担当者やセキュリティ部門が、例外（免除）管理を含む全体的なリスク管理戦略に統合できる体制を整えることも重要です。



スキャン、スキャン、スキャン

組織はこれまで以上に多くのセキュリティツールを導入していますが、依然としてカバレッジのギャップが存在します。
特に、コードとバイナリの両方に対するスキャン不足や、SDLC 全体および本番環境におけるスキャンの一貫性の欠如は、代表的な盲点です。

Q 使用しているアプリケーションセキュリティソリューションの数はいくつですか？（委託調査、2023 年および 2024 年）



回答者は、2023 年と比較して 2024 年には使用しているアプリケーションセキュリティソリューションの数が増加したと回答しています。

2024 年末時点で、7 つ以上のソリューションを使用していると回答した割合は 73% に達し、前年の 58% を大きく上回りました。

この結果は、市場でツール統合への関心が高まっていることや、安全なソフトウェア開発プロセスの合理化を求める JFrog の顧客の声から想定されていた傾向とは対照的です。
データによれば、組織は過剰なカバレッジによる見逃しリスクを回避するため、複数のスキャンソリューションを維持しつつ、重複する検出結果を統合・整理できる新たなカテゴリである ASPM（Application Security Posture Management）を導入しています。

しかし、ASPM はセキュリティツールのスプロールを管理するための暫定的な対応策に過ぎず、根本的な解決策ではありません。
組織が再びツール統合に注力し始めていることを踏まえると、セキュリティツール数の増加傾向が来年も継続するとは限らないと考えられます。

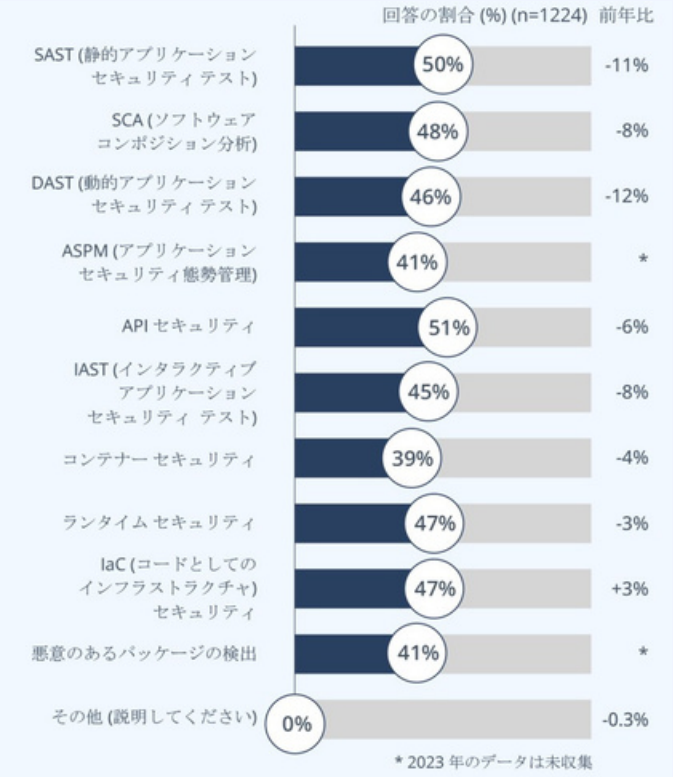
Q あなたの組織では、コードレベル、バイナリレベル（またはその両方）で
セキュリティスキャンを実施していますか？（委託調査、2024 年）

2024 年には、バイナリレベルのみでセキュリティスキャンを実施していると回答した割合が倍増しました。このレベルでのみスキャンを適用していると回答した割合は 25% に達し、2023 年の 12% から大きく増加しています。

一方、コードレベルとバイナリレベルの両方でスキャンを実施していると回答した割合は 43% で、2023 年の 56% からやや減少しました。
リスクを可能な限り早期に検出・防止するためには、コードとバイナリの両レベルでスキャンを行うことが理想的であり、この傾向は懸念されます。バイナリレベルでのみ現れる脆弱性も存在するためです。

たとえば、バイナリに埋め込まれたシークレットや、コンパイラによって挿入されたメモリ破損は、ソースコードには存在しない問題を引き起こす可能性があります。また、最終的に本番環境へデプロイされるビルドに、意図せずセキュリティ上の欠陥が残るリスクもあります。

Q 使用しているアプリケーションセキュリティソリューションのタイプは何ですか？
（委託調査、2024 年）



使用率が圧倒的に高いツールは存在せず、50% 以上の導入率を示したのは API セキュリティと SAST の 2 分野のみでした。

シフトレフトの取り組みが重視されていることを踏まえると、SAST の高い採用率は自然な結果といえます。また、マイクロサービスアーキテクチャの普及により API が攻撃対象となりやすくなっていることから、API セキュリティツールへの投資が進んでいる点も妥当です。

ツールタイプごとの使用傾向は前年と大きく変わっていませんが、各ツールを使用していると回答した割合は全体的に低下しています。セキュリティツールの総数が増加しているという前述の傾向と合わせて考えると、この結果は注目に値します。

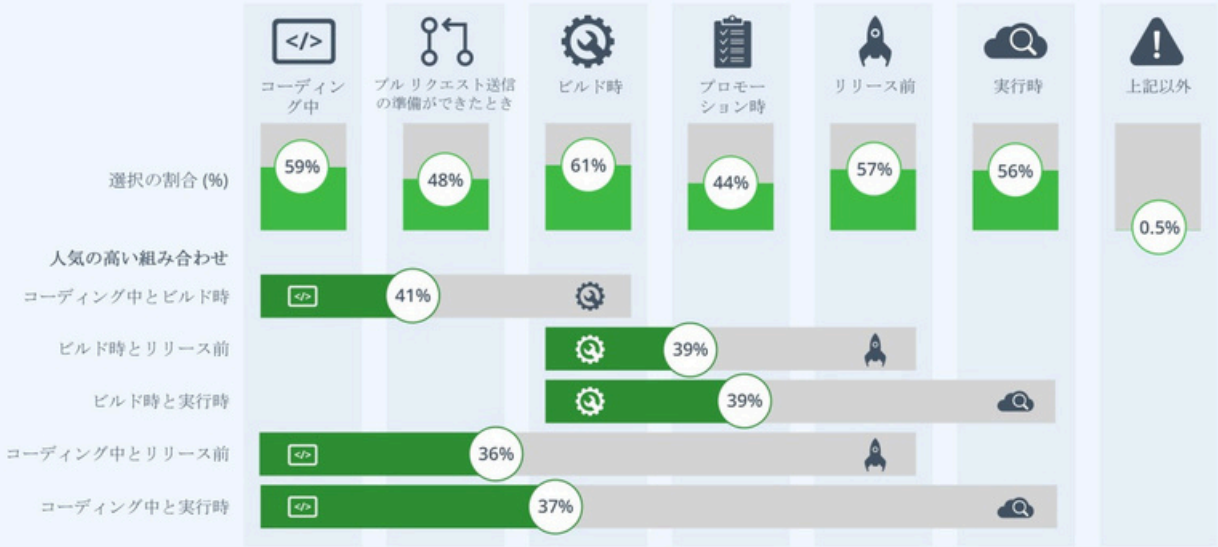
この背景には、機能が重複するツールの併用や、異なるチームが独自に選定したツールを並行して使用している可能性が考えられます。組織は、ツールの重複やカバレッジのギャップを特定するため、定期的なセキュリティツールの棚卸しや監査を検討すべきです。

Q あなたの組織では、SDLC のどの段階でセキュリティを適用するのが最適だと思いますか？（委託調査、2024 年）

ソフトウェア開発ライフサイクルのどの段階でセキュリティを適用するのが最適かという質問に対する上位 3 つの回答は、前年と同様でした。

コード作成時	ビルド時	オープン ソース ソフトウェア導入時
M=3.94	M=3.74	M=3.59

Q あなたの組織では通常、開発のどの段階でセキュリティスキャンを適用していますか？（委託調査、2024 年）



「コーディング中」は、SDLC において組織がセキュリティスキャンを実施する段階として最も多く挙げられました。

回答者の約 5 人に 3 人が、通常この段階でセキュリティスキャンを実行していると回答しています。

41%	コーディング中	+	ビルド時
39%	ビルド時	+	リリース前
30%	ビルド時	+	実行時
36%	コーディング中	+	リリース前
37%	コーディング中	+	実行時

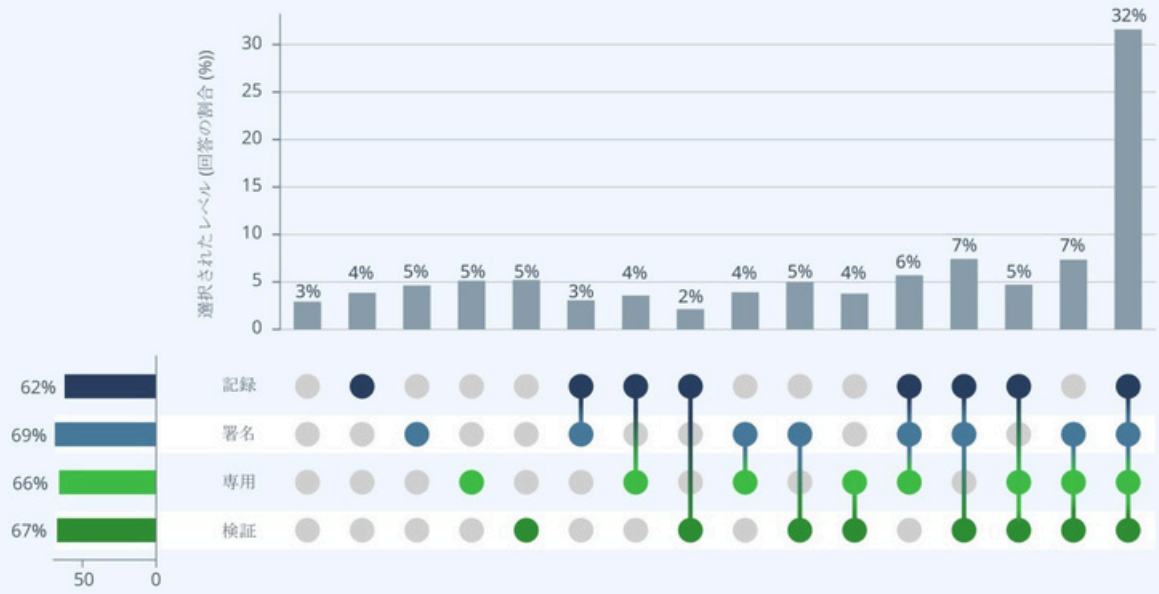
パイプライン全体を通じた可視性とガバナンスの強化

アプリケーションをビルドする際には、アプリケーションレベルで包括的にリスクを管理することが不可欠です。多くの組織は既に SDLC 全体を通じてアプリケーションを定義・追跡していますが、その制御とトレーサビリティの水準には大きなばらつきがあります。

リスクを確実に管理し、リリースするソフトウェアの信頼性を担保するためには、可視性だけでなく、統制とトレーサビリティの両方を強化する必要があります。
統制は、実装段階ではなく調達段階から始まります。すなわち、「アプリケーション」を構成する前段階において、どのコンポーネントやライブラリを取り込むかが、最終製品のセキュリティ態勢を根本的に左右します。

サードパーティのリソースを慎重に評価・選定することで、組織はリスクがアプリケーション内で顕在化する前に、その影響を抑制することが可能になります。

Q あなたの組織では、以下のどのレベルのセキュリティフレームワークを導入していますか？（委託調査、2024 年）



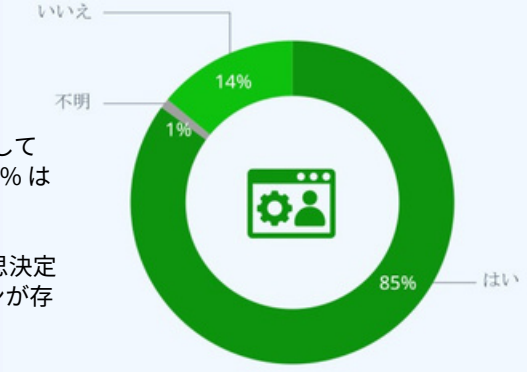
- 記録パッケージのビルドメタデータを記録
- 署名パッケージのビルドデータとメタデータに署名
- 専用パッケージを専用のホストでビルド
- 検証パッケージの署名を公開前に検証

調査対象となった組織の大多数は、ソフトウェアサプライチェーンのセキュリティと整合性を強化するために、SLSA（Supply-chain Levels for Software Artifacts）などのフレームワークを活用しています。

少なくとも 1 つの SLSA レベルは広く採用されており、回答者の約 3 分の 1 がすべての SLSA レベルを導入していると回答しました。

アプリケーション責任者の管理状況

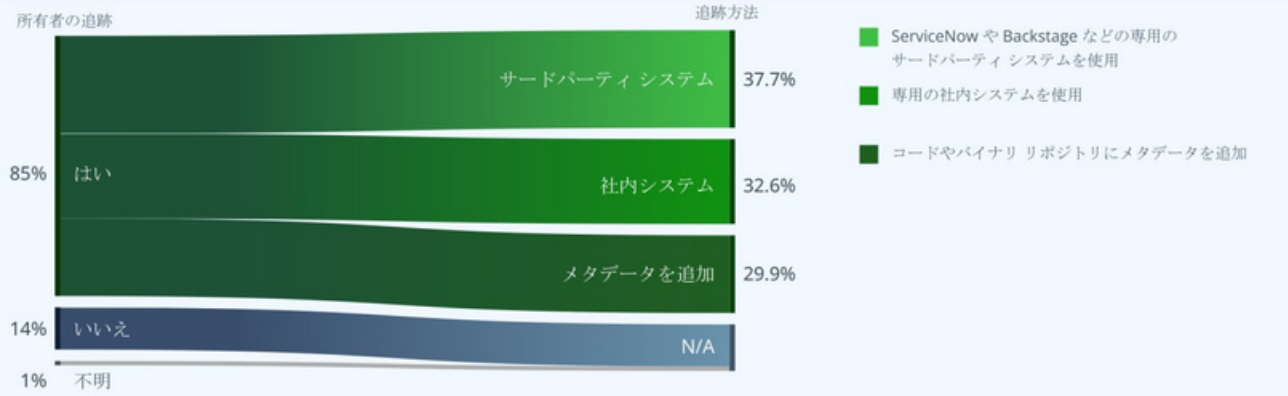
Q 組織でビルドする各アプリケーションについて、責任者（チームまたは個人）を明確にしていますか？（委託調査、2024 年）



調査の結果、85% の組織が各アプリケーションの責任者を明確にしていると回答しました。一方で、14% は責任者を定義しておらず、1% は不明と回答しています。

責任者の明確化は、脆弱性対応やインシデント発生時の迅速な意思決定に直結する重要なガバナンス要素です。未定義のアプリケーションが存在することは、リスク管理上の盲点となる可能性があります。

Q 組織でビルドする各アプリケーションについて、責任者をどのように管理していますか？（委託調査、2023 年および 2024 年）



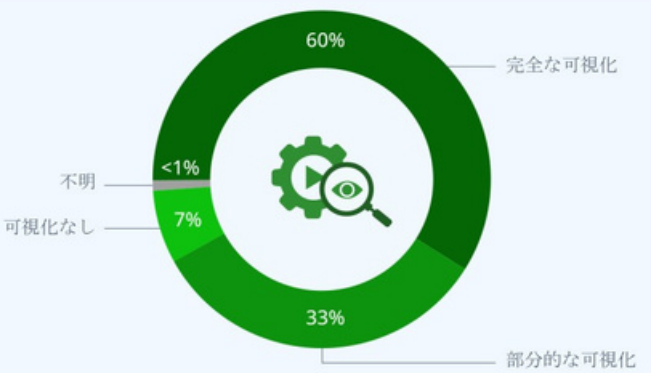
アプリケーションおよびそれを構成する各種マイクロサービスの責任者を明確に管理することは、迅速な問題解決、アプリケーション間の依存関係の把握、さらには適切なガバナンスや事業継続計画（BCP）の確立において不可欠です。

大多数の組織は、アプリケーションのビルド時に責任者情報を管理していると回答していますが、その手法にはばらつきがあります。
回答は、① 専用のサードパーティシステムの活用、② 社内専用システムの利用、③ コードまたはバイナリリポジトリへのメタデータ付与、の 3 つにほぼ均等に分かれました。

なお、これら 3 つ以外の方法を使用していると回答した組織はありませんでした。

Q 本番環境で実行しているソフトウェアの出所（コードのコミット者、実施されたテストや検証内容、依存関係の取得元など）を可視化できていますか？（委託調査、2023 年および 2024 年）

本番環境で実行しているソフトウェアの出所を「完全に可視化できている」と回答した組織は 60% にとどまりました。



「部分的に可視化できている」と回答した組織は約 3 分の 1（33%）であり、可視性がない、または出所が不明と回答した組織も 8% 弱存在します。

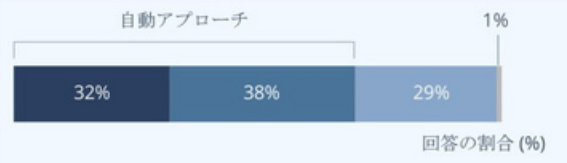
ソフトウェアの出所を把握することは、リリースする製品の品質とセキュリティを担保するうえで不可欠であり、近年では各国の規制やコンプライアンス要件においても重要性が高まっています。

可視性がない、または出所が不明と回答した約 8% の組織は、少なくともコードベースおよび外部パッケージのインベントリを整備し、ビルドのバージョン管理を自動化された CI/CD プロセスと連携させる必要があります。

多くの組織にとってソース管理は当然の前提と考えられていますが、この約 8% の中には、開発中のコード変更を十分に追跡できる成熟したソース管理体制をまだ整備していない組織が含まれている可能性があります。

Q コンプライアンスおよびガバナンスの観点から、ソフトウェアの作成・リリースプロセスにおいて、テストおよび品質基準が確保されていることをどのように確認していますか？ （委託調査、2023 年および 2024 年）

テストおよび品質基準を確保するための手法は、組織によって大きく異なります。大多数（70%）は自動化されたアプローチを採用していますが、約 3 分の 1（29%）は依然として SDLC の各段階において手動承認に依存しています。



- SDLC 全体を通じて、認証・承認の証跡を自動的に収集している
- 継続的インテグレーション（CI）プロセスに自動化された承認ゲートを組み込んでいる
- SDLC の次の段階へ進む前に、ソフトウェアを手動で承認している
- コンプライアンスおよびガバナンスに関する正式なプロセスは設けていない

組織がセキュリティ対策に費やしている時間とコスト

JFrog の委託により IDC が実施した調査によると開発チームまたはセキュリティチームがアプリケーションの脆弱性対応に毎月 4 日以上を費やしていると回答しています。

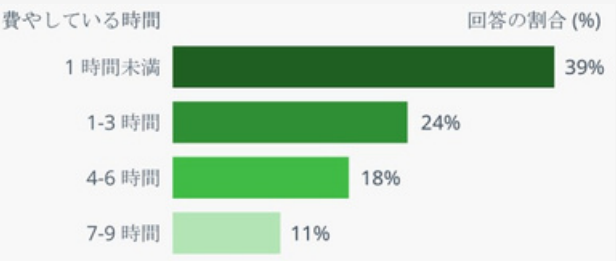
また、セキュリティ関連タスクに従事する開発者 1 人あたりの年間コストは、平均で約 28,000 ドルに上ると推計されています。

これは単なる財務的負担にとどまらず、開発者エクスペリエンス（DX）や生産性にも影響を及ぼす重要な課題です。

IDC: The Hidden Cost of DevSecOps
（DevSecOps の隠れたコスト）
2024 年 9 月発行 | IDC #US52537524

開発者がセキュリティ問題対応のために勤務時間外に費やす時間

調査によると、開発者はセキュリティ問題への対応のため、勤務時間外に週平均約 3.6 時間を費やしています。これは持続的な負担となり、開発者の疲弊やバーンアウトにつながるリスクがあります。



週平均 3.6 時間
開発者が勤務時間外にセキュリティ問題対応へ費やす時間

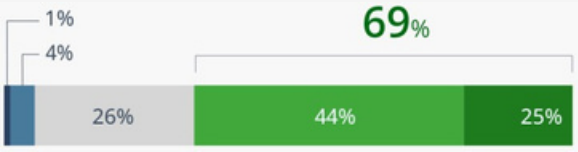
セキュリティ関連業務に費やされるコスト

平均的な組織では、開発者 1 人あたり年間約 28,000 ドルがセキュリティ関連業務に費やされています。

DevSecOps は現代のソフトウェア開発において不可欠な取り組みですが、非効率なツールや十分に最適化されていないプロセスは、開発者の生産性を低下させ、結果として追加のビジネスコストを生み出します。

年間 $28,000
開発者 1 人あたり年間 $28,000 がセキュリティ関連業務に費やされています

頻繁なコンテキストスイッチによる生産性の低下



- 強く同意する - ツールや環境を非常に頻繁に切り替えて います
- 同意する - ツールや環境を頻繁に切り替えています
- 未定
- 同意しない - ツールや環境をたまに切り替えています
- 強く同意しない - ツールや環境をほとんど、または全く切り替えていません

69% の開発者が、セキュリティ関連の責任を果たすために頻繁なコンテキストスイッチが必要であると回答しています。

組織が DevEx（開発者体験）の向上を目指すのであれば、ツール間の頻繁な切り替えが生産性を低下させるだけでなく、開発者がセキュリティ関連業務に積極的に取り組む意欲を損なう可能性がある点を考慮する必要があります。

69%
開発者がセキュリティ対応のために頻繁なコンテキストスイッチが必要と回答

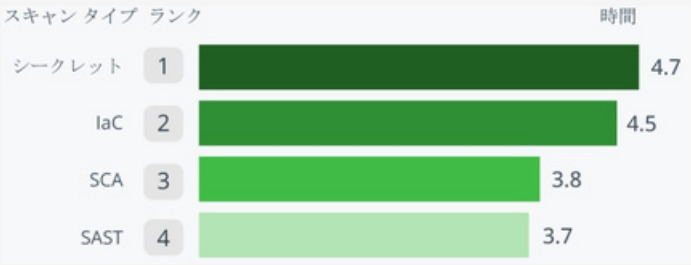
IDC: The Hidden Cost of DevSecOps
（DevSecOps の隠れたコスト）
2024 年 9 月発行 | IDC #US52537524

開発者が各スキャンタイプに費やす時間

開発者は、各種スキャンの中でもシークレットスキャンに最も多くの時間を費やしていることが分かりました。これは、トークンやシークレットを適切に取り扱うためのコーディングプラクティスに関する追加トレーニングの必要性、あるいはより効率的なシークレット管理ツールの導入余地を示唆しています。

コード内にシークレットが残存していないかを確認することは極めて重要です。たとえば、住所が書かれたキーホルダーに家の鍵を付けたまま落とすようなもので、攻撃者にとっては格好の侵入口となります。
コードにシークレットが残っている場合、重要なシステムや機密データへの不正アクセスを許すリスクがあります。

4.7
時間を開発者はシークレットスキャンに費やしている



IDC の調査は 2024 年 6 月にオンラインで実施され、DevSecOps を導入している米国および欧州の開発者、開発チームリーダー、マネージャー、プロダクトオーナー計 210 名から回答を得ました。

本調査では、DevSecOps における開発者の時間がビジネスに与える影響、時間を消費するツールやタスクの実態、開発者の時間の経済的価値、そしてセキュリティ関連業務が開発者のフローや満足度に与える影響について分析しています。

Q 新しいパッケージ／ライブラリの使用が承認されるまで、通常どのくらいの時間がかかりますか？（委託調査、2023 年および 2024 年）

開発者が新しいパッケージの承認を待つ時間は、これまで以上に長くなっています。
特に中規模組織（従業員数 5,000～10,000 人）では待機時間が長くなる傾向が顕著で、92% が 1 日以上、74% が 1 週間以上、49% が 2 週間以上待っていると回答しています。

開発者が承認プロセスに積極的に関与するようになった点は前向きな変化と言えます。

レビューや手作業中心のプロセスでは非効率が避けられません。新しいコンポーネントの導入を迅速かつ安全に進めるためには、セルフサービスを前提とした自動化の強化が求められます。

重要なポイント

71

ソフトウェア サプライ チェーン セキュリティの基本的なプラクティスの欠如

組織は、開発者やアプリケーションが利用する外部コンポーネントや依存関係を適切に管理し、少なくとも「何が取り込まれているのか」を常に把握できる状態を維持する必要があります。
しかし、71%以上の組織が開発者にインターネットからの直接ダウンロードを許可しているという現状は、大きなリスクをはらんでいます。これはソフトウェア サプライ チェーン セキュリティの基本的な考え方に反するものです。
パブリック レジストリをプロキシできるアーティファクト管理ソリューションを導入し、すべてのコンポーネントを中央で管理・追跡できる仕組みを整えることが不可欠です。

スキャナーが増えると、問題も増えるのか?

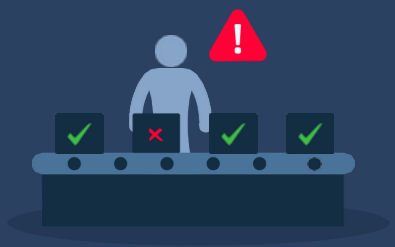
組織はこれまで以上に多くのセキュリティ ツールを導入しています。しかし、それが本当にセキュリティの向上につながっているのか、それとも運用の複雑化を招いているのかは明確ではありません。
ツールの数が増えても、カバレッジのギャップは依然として残っています。特に、コード レベルとバイナリ レベルの両方で一貫してスキャンを実施していない組織が多い点は懸念材料です。
スキャナーが増えれば可視性は高まりますが、その一方で、重複した検知結果やアラート疲れ、運用負荷の増大といった新たな課題も生まれます。重要なのはツールの数ではなく、統合された戦略と一貫した運用です

開発者体験を損なわない DevSecOps

セキュリティ対策は、開発者の時間を毎週何時間も消費しています。
しかし、アプリケーションの安全性を維持しながら、開発者への負担を抑えることは可能です。

そのためには、対処すべきリスクの適切な優先順位付け、文脈を踏まえた実用的な検知結果の提示、そしてプロセスの自動化が重要になります。

セキュリティを強化することと、開発効率を守ることは両立できます。求められているのは、開発者の作業フローに自然に組み込まれたセキュリティです。



アプリケーション管理のレベルアップ

組織の 85% は、社内でビルドしたアプリケーションの所有者を追跡しています。しかし、アプリケーションの標準をどのように維持しているかは組織によって大きく異なります。約 3 分の 1 の組織は、ソフトウェアを次の段階へ進める際に、依然として手作業による承認や確認を行っています。
手作業は柔軟性を持つ一方で、意図的であれ偶発的であれリスクを伴います。標準の一貫性やトレーサビリティを確保するには、より体系的で自動化されたアプローチへの移行が求められます。

リスクの新たな領域：AI と機械学習の開発

ほぼすべてのセキュリティツールや多くの開発ツールが、開発の高速化や脆弱性の検出・修復の高度化をうたってAI機能を前面に打ち出しています。

しかし、本セクションで焦点を当てるのは、AIツールの活用ではなく、AIそのものをビルドする際のリスクです。

AI/ML（人工知能／機械学習）を取り巻くソフトウェアサプライチェーンは、組織にとって新たなリスク領域であり、その成熟度は従来のソフトウェア開発よりもはるかに初期段階にあります。

実際、JFrogがInformationWeekに委託した調査によると、79%の企業が、セキュリティ上の懸念を理由にAI/ML機能の利用や統合が遅れていると回答しています。

AI の導入と DevSecOps の動向

InformationWeek の調査では、ソフトウェア開発者とサイバーセキュリティチームが、アプリケーション セキュリティをソフトウェア開発ライフサイクル（SDLC）に統合する重要性をどの程度認識しているかを分析しました。
さらに、悪意のあるコードから組織をどのように保護しているか、また AI テクノロジの不適切な利用をどのように防いでいるかについても調査しています。
以下に、本調査から得られた主な洞察を示します。

AI 導入と DevSecOps：安全性を確保しながら先を行く
2024 年 9 月発行｜InformationWeek & JFrog

企業全体での AI セキュリティに対する信頼の低さ

79% の企業が、セキュリティ上の懸念を理由に、AI/ML 機能のソフトウェアへの導入や統合が遅れていると回答しています。

企業が AI セキュリティに関して挙げた上位 3 つの懸念事項は、LLM の利用に伴うデータ漏洩、AI モデルに含まれる悪意のあるコード、そして AI バイアスです。

69% の組織が、ソフトウェアにおける AI 利用に関する新たな規制への準拠について、「まったく自信がない」または「やや自信がある程度」と回答しています。

AI サプライ チェーンの可視性は依然として不十分

49% の企業が、アプリケーションにおける ML モデルの利用を適切に管理するための信頼できる仕組みを持っていません。

AI モデルを含むすべてのソフトウェア コンポーネントについて、信頼できる単一の情報源（Single Source of Truth）を確立している組織は、4 分の 1 未満にとどまっています。

3 分の 2 以上の組織が、ML モデルに含まれる推移的依存関係（トランジティブ依存）まで含めて、ソフトウェア内のオープンソース パッケージを追跡するための信頼できる仕組みを持っていません。

AI ポリシーは依然として不十分

58% の企業が、開発者によるオープンソースの AI モデルやコンポーネントの利用に関する明確なポリシーを導入していない、または導入状況を把握していないと回答しています。

60% の企業が、開発者によるトレーニング データの調達やライセンスに関するポリシーを整備していません。

施行に一貫性がない

68% の回答者が、AI コンポーネントの利用を統制する仕組みを持っていない、または手動レビューに依存していると回答しています。

59% の回答者が、トレーニング データに関するポリシーを実効的に適用する仕組みを持っていない、または手動レビューに依存していると回答しています。

InformationWeek に委託して実施した本調査は、2024 年 5 月にオンラインで行われ、主に北米を拠点とする 210 人の IT およびサイバーセキュリティの専門家から回答を得ました。ます。

回答者は、あらゆる規模の企業に所属する幹部から一般社員まで幅広く、コンサルティング、銀行・金融サービス、教育、政府機関、テクノロジー、ヘルスケア、製造業など、21 以上の業種にわたります。

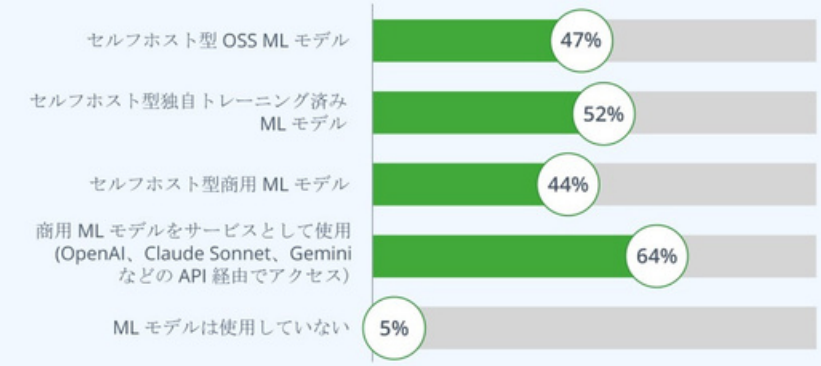
InformationWeek の調査から、いくつかの興味深い傾向が明らかになりました。本セクションでは、組織が AI サービスをアプリケーションにどのように導入し、その利用をどのように管理しているのかを、さらに詳しく見ていきます。

ML モデル アーティファクトの管理、ガバナンス、スキャン

Q 開発中のアプリケーションにおいて、ML モデルは主にどのような形で利用されていますか？ (委託調査、2024 年)

組織が AI サービスやアプリケーションを導入する方法は多岐にわたります。調査結果からは、多くの組織が複数のアプローチを組み合わせて活用していることが明らかになりました。

最も一般的なアプローチは、API 経由で利用する商用モデルの活用で、回答者の 64% がこの方法を採用しています。

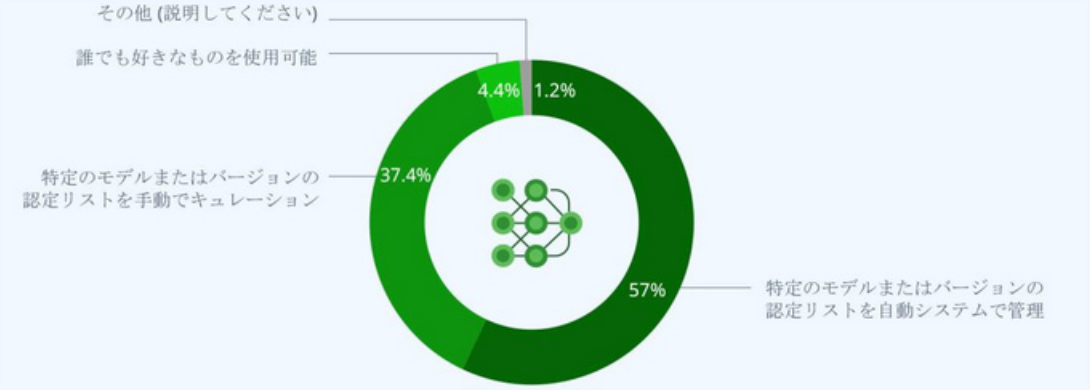


この方法では、初期開発やインフラへの大きな投資を行うことなく、高度で汎用的な AI 機能を迅速に活用できます。能を迅速に利用できます。

一方で、セルフホスティング型モデルへの投資も進んでいます。半数以上の組織が、特定のビジネス ニーズに合わせて構築した独自モデルを自社でホスティングしています。

また、回答者の約半数は、機械学習モデルの主な活用方法として、セルフホスト型のオープンソース（OSS）モデルを挙げています。

Q 開発組織内で、ML モデル アーティファクトの利用をどのように管理していますか?（委託調査、2024 年）



専門家の 3 人に 1 人以上（37%）が、手動でキュレーションした特定のモデルやバージョンのリストを用いて、ML モデル アーティファクトの利用を管理していると回答しています。

InformationWeek の調査によると、49% の企業が、アプリケーションにおける ML モデルの利用を管理するための信頼できる仕組みを持っていません。

この結果は、4% の回答者が、開発者による ML モデル アーティファクトの利用を（手動であっても）意図的に管理していないと回答している背景を示している可能性があります。

調査対象者全体の

16%

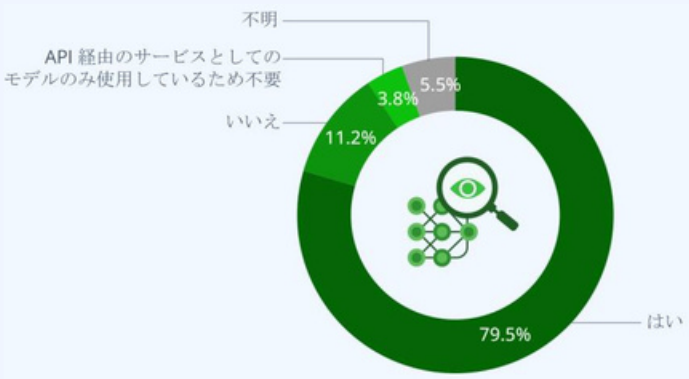
がセルフホスト型の OSS モデルを利用しており、そのモデル アーティファクトの利用を手動で管理しています。

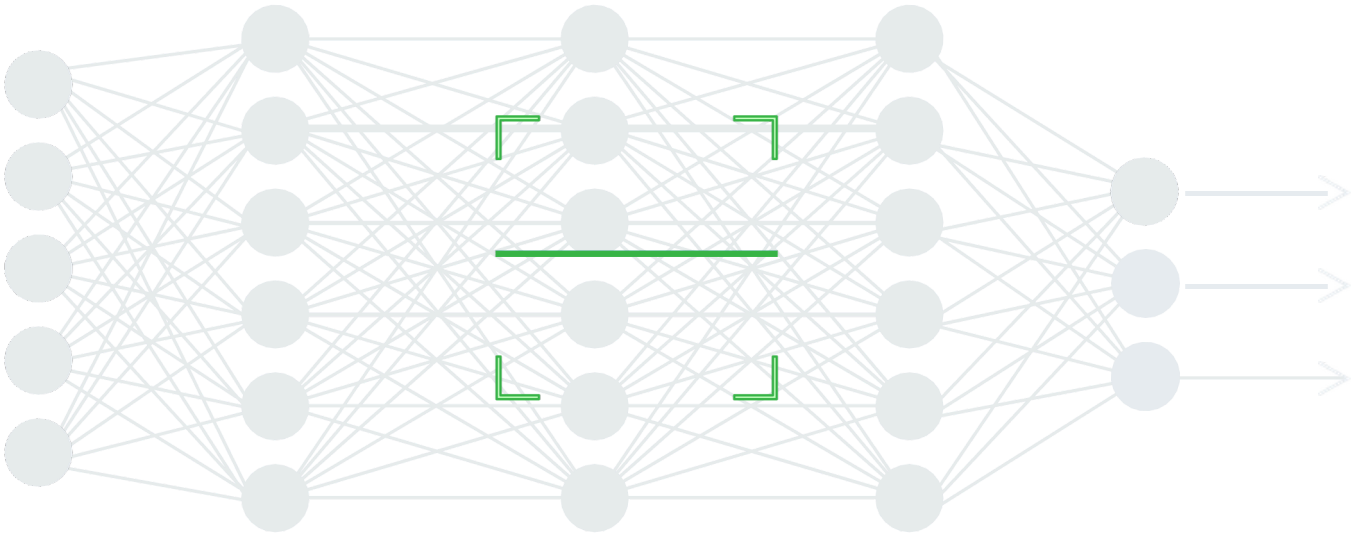
Q あなたの組織では、ML モデル アーティファクトにセキュリティの脆弱性や悪意のあるモデルが含まれていないかをスキャンする仕組みを導入していますか?（委託調査、2024 年）

大多数の組織（79%）が、ML モデル アーティファクトに脆弱性や悪意のあるコードが含まれていないかを確認するスキャンの仕組みを導入していると回答しています。

一方、11% は導入していないと回答しています。幸いなことに、セルフホスト型の OSS モデルを利用しながら、脆弱性や悪意のあるモデルを検出するスキャンを導入していないと回答したのは、全回答者のわずか 3% にとどまりました。

とはいえ、AI/ML 開発を安全に推進するためには、適切なツールとポリシーの整備を、組織レベルだけでなく業界全体でさらに進める必要があります。





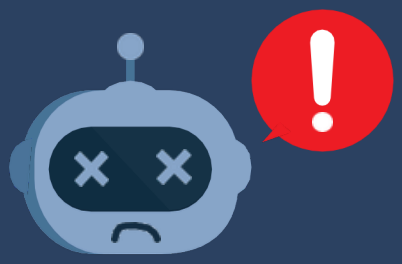
重要なポイント

商用モデルによる AI の導入

商用モデルの活用は、AI サービスをビジネス アプリケーションに導入するための一般的なアプローチです。API 経由で商用モデルにアクセスすることで、自社でモデルを構築・運用するために必要なツール、インフラ、専門知識の確保にかかる時間とコストを抑えることができます。
AI/ML の経験が限られている組織にとっては、専門性を備えたプロバイダーにモデルのセキュリティや運用を委ねることが、合理的な選択肢となる場合もあります。

AI サプライ チェーンの可視性は依然として不十分

多くの組織は、アプリケーションにおける機械学習モデルの利用を適切に管理するための信頼できる仕組みを確立できていません。ML モデルを含むすべてのソフトウェア コンポーネントについて、信頼できる単一の情報源（Single Source of Truth）を持たないケースも少なくありません。さらに、ML モデルに関連する推移的依存関係を含めたオープンソース パッケージの追跡には、大きな盲点が残っています。
こうしたギャップが存在すると、AI/ML サプライ チェーンの効率的な管理が難しくなるだけでなく、脆弱性や不正なコンポーネントの混入といったセキュリティ リスクも高まります。



AI セキュリティへの過信というリスク

79% の組織が何らかのレベルでモデル スキャンを実施していると回答している一方で、AI/ML モデル セキュリティのソリューションは依然として初期段階にあります。多くは単純な検出手法に依存しており、その精度や有効性には課題が残されています。

たとえば、現在のアプローチでは、Hugging Face 上の悪意のあるモデルを特定する際に 96% の誤検知率が発生し、さらに単純なスキャン手法では一部の脅威を見逃していることも確認されています。
多くのセキュリティ ツールがモデル アーティファクトへの対応を進めていますが、組織は各ベンダーのモデル セキュリティ ソリューションの実効性を慎重に評価する必要があります。

調査方法

本レポートは、JFrog の利用状況データ、JFrog セキュリティ リサーチ チームによる CVE 分析結果、ならびに第三者機関に委託した調査データから得られた　洞察を統合して作成しています。以下に、各情報源の詳細を示します。

JFrog Platform 利用データの分析

本レポートで取り上げているテクノロジ利用の傾向は、JFrog Platform for Cloud の匿名化された年末時点の利用状況データに基づいています。このデータは、数千の顧客、数十万のリポジトリ、そしてペタバイト規模のデータを対象としています。

パッケージの人気度は、特定のパッケージ タイプにおけるアクション数（アップロード／ダウンロード）、アーティファクト総数、リポジトリ総数、アーティファクトの総サイズを基に算出しています。

特にアクション数は、開発者が各パッケージ タイプをどの程度利用しているかを示す重要な指標であり、実際の開発現場での使用状況を反映しています。

一部の大規模ユーザーによってランキングが偏る可能性はありますが、アーティファクトのアクションデータもあわせて分析することで、開発プロセスで実際に活用されているパッケージ タイプをより正確に把握しています。

JFrog セキュリティ リサーチ チームによる調査・分析

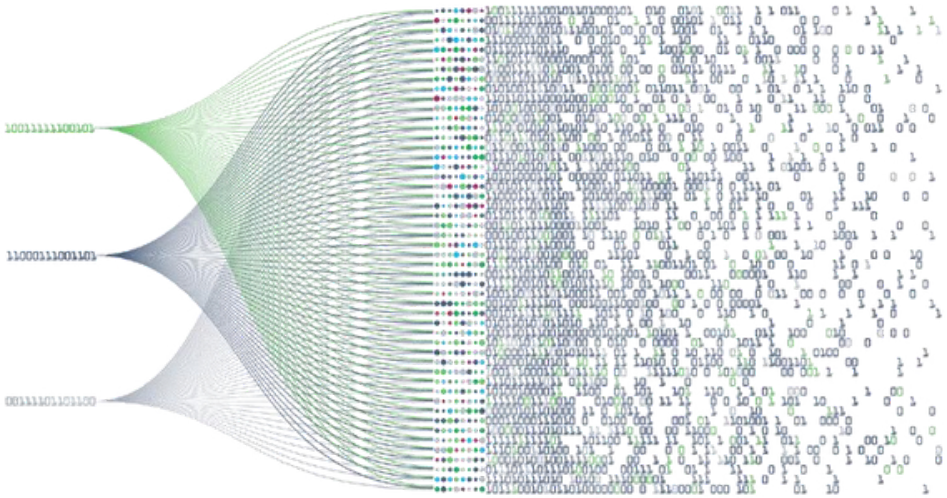
指定された CNA（CVE Numbering Authority）として、JFrog セキュリティ リサーチ チームは、新たに公開される脆弱性を継続的に監視・調査し、その実際の重大度を評価したうえで、コミュニティおよび JFrog のお客様に情報を公開しています。

本レポートには、JFrog Catalog サービスを通じてパブリック ソースから取得したデータ、NVD から取得した CVE 情報、そしてそれらのデータを基に JFrog セキュリティ リサーチ チームが実施した独自の分析結果が含まれています。

委託調査の結果

JFrog は Atomik Research に委託し、米国（n=375）、英国（n=205）、インド（n=206）、ドイツ（n=205）、フランス（n=205）、イスラエル（n=205）の特定業界[1] に勤務する計 1,402 名を対象に、国際的なオンライン調査を実施しました。
サンプルは、各組織の IT、情報システム、テクノロジ部門に所属し、特定の職務[2] を担う正社員で構成されています。すべての回答者は、総従業員数 1,000 人以上の企業に勤務しており、かつ組織内に少なくとも 50 名以上のメンバーを擁するソフトウェア開発チームが存在することを条件としています。

オンライン調査は、英語、フランス語、ドイツ語、ヘブライ語、ヒンディー語で実施されました。
サンプル全体の統計的誤差は ±3 パーセントポイント、信頼水準は 95% です。調査期間は 2024 年 11 月 22 日から 12 月 9 日までです。
Atomik Research は、グローバル市場を対象とした独立系の市場調査会社です。



[1] 業界条件
参加資格を得るため、回答者は以下のいずれかの業界にサービスを提供する組織に所属していることが条件とされました。
航空宇宙／建築・エンジニアリング／自動車／銀行・金融サービス・保険・フィンテック／エネルギー・石油・ガス／政府・公共部門／ヘルスケア・ライフサイエンス／ホスピタリティ／製造／小売／テクノロジ／運輸・物流／公益事業・通信・電力
あわせて、IT、情報システム、テクノロジ部門、または IT 製品開発部門に所属していることも参加条件としました。

[2] 職務条件
参加資格を得るため、回答者は以下のいずれか、または類似する職務に従事していることが条件とされました。
AI エンジニア／アプリケーション セキュリティ エンジニア／サイバーセキュリティ エンジニア／データ サイエンティスト／開発者／DevOps アーキテクト／DevOps エンジニア／エンジニアリング マネージャー／機械学習スペシャリスト（ML エンジニア）／プラットフォーム エンジニア／セキュリティ アーキテクト／セキュリティ研究者／サイト信頼性エンジニア（SRE）／ソフトウェア アーキテクト／ソフトウェア開発者／ソフトウェア エンジニア／ソリューション アーキテクト