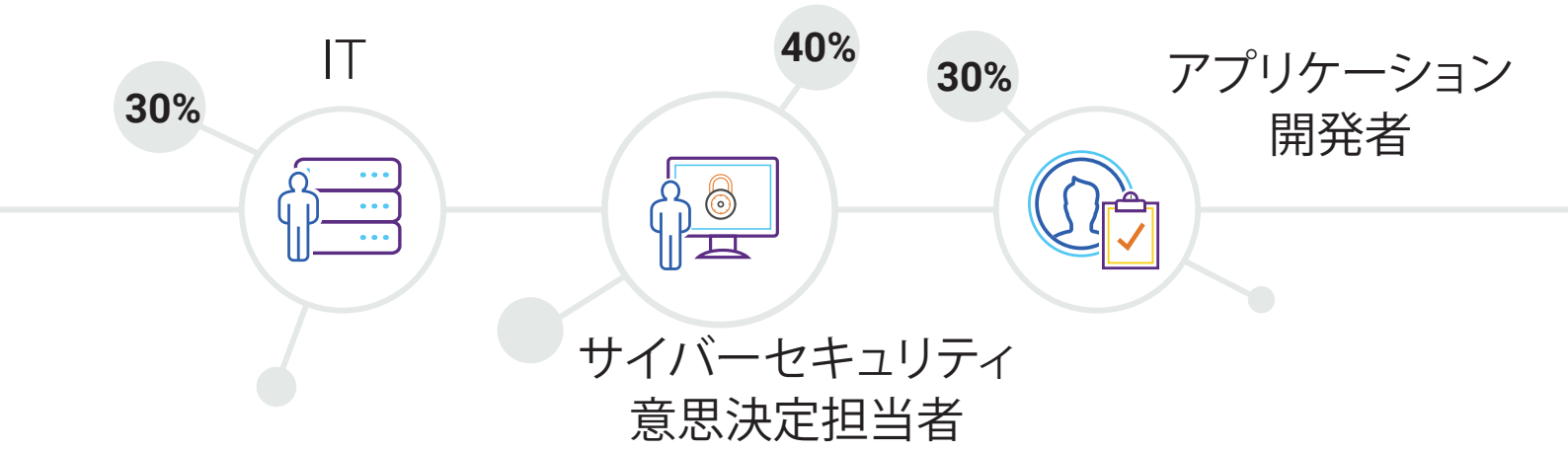
BLACKDUCK®

API セキュリティの現状

ソフトウェア・サプライチェーンは、アプリケーション・セキュリティに責任を負う、またはその影響を受けるすべての人にとって今なお最重要課題となっています。こうした状況をふまえ、ブラック・ダックが共同スポンサーとなって Enterprise Strategy Group（ESG）が作成したレポートが、「社会的な規範を守る：GitOps とシフト・レフト・セキュリティ」です。開発者中心のセキュリティに関するこのマルチクライアント調査レポートは、アプリケーション・セキュリティの現状を分析することにより、さまざまな業種で浮かび上がっている新しいトレンドを特定し、現在の状況において組織がどの部分で成果を上げ、どの部分で苦戦しているかをより深く理解することを目的としています。

ESG が 350 人のプロフェッショナルを対象に調査



北米地域の中堅企業（従業員数 100 ～ 999 人）と
大企業（同 1,000 人以上）が対象

この調査結果を精査した結果、ブラック・ダックは回答者が実施しているサプライチェーン・セキュリティ対策において API が明らかな懸念事項となっていることを特定しました。この eBook では、ESG のレポートの中から特に API に関する知見を取り上げ、それに対するブラック・ダックの見解、および API のセキュリティ・ストラテジー改善に向けた提言を示していきます。

知見 1 API のセキュリティがあらゆる業種で大きな懸念材料に

クラウドネイティブ・アプリケーション・スタックのうち、どの要素が最も攻撃を受けやすく、組織にとって最もリスクが大きいかを尋ねたところ、最大の懸念材料として挙げられたのが API でした。

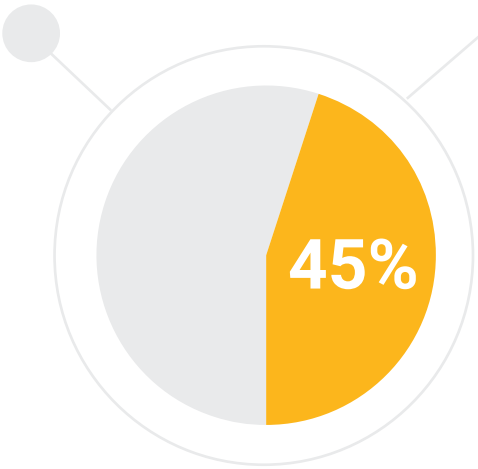
ブラック・ダックの見解

この調査で最も大きなセキュリティ上の懸念として挙げられたのが API（45%）で、それに僅差で続いたのがデータ・ストレージ・リポジトリ（42%）と自社開発のアプリケーション・ソース・コード（38%）でした。**簡単に言えば、あらゆるアプリケーション・セキュリティ対策の中で回答者が最も大きな懸念を抱いていたのが API のセキュリティだということです。**この調査結果に対し、ESG は「クラウドネイティブを取り巻くサイバーセキュリティ上の脅威が激化」との見解を示しています。これは、今後も API がアプリケーション・セキュリティ責任者を悩ませる最大の懸念材料であり続けるというブラック・ダックの見解とも一致しています。

今後 12 ～ 18 カ月間でクラウドネイティブ・アプリケーション・セキュリティに対する最優先の投資課題を尋ねたところ、多くの回答者が API のセキュリティ対策を強化する取り組みを直ちに開始したいと考えていることが明らかになりました。「ソース・コードに含まれる API を検出・検査する」を最優先課題に挙げた回答者は 30% で、「実行時の API セキュリティ対策を適用する」を挙げた回答者も 31% ありました。このように、セキュリティ・イニシアティブにおける最優先課題のうち、API に関連したものが 60% 以上を占めています。

API のセキュリティが懸念されるのには、十分な理由があります。API は、組織が重要なサービスを web 上に公開する際の中心的な手段となるものです。そのこと自体は必要なことですが、同時に多くのセキュリティ攻撃を寄せ付ける要因にもなっています。

- 注意を欠いたコーディング / 開発プラクティスにより、アプリケーション開発時にバグが作り込まれる。
- オープンソースおよびその他のサードパーティ・ライブラリの使用により、意図せずコア・システムに脆弱性が混入する。
- 本番環境で稼働中のアプリケーションにおいて、適切に保護されていない API が攻撃者の格好の標的となり得る。



調査では API が
最も大きなセキュリティ上の
懸念に

提言

web ファイアウォールや監視ツールだけでは API を保護できないことに注意することが重要です。効果的な API セキュリティには、それ以上のものが必要です。

API のセキュリティは、API 自体の開発ライフサイクルとして管理し、扱う必要があります。このプロセスの最初のステップとなるのが、効果的な計画で、堅牢な API ライフサイクル管理のロードマップの設計が含まれます。また設計フェーズでは、組織全体のビジネスリスク・プログラムに組み込まれる、適切な API ポリシーを含める必要があります。

組織内のすべての API ベースのアプリに関する完全かつ最新のインベントリを整備しておくことは、品質を管理し、API のリスク分類を決定し、評価活動の効果を高めるのに役立ちます。このような分類を行うことで、最もリスクの高い API に注意を向けることが可能となり、時間と労力の無駄を最小限に抑えることができます。

おそらく最も重要なのは（そして現在よく見落とされているのは）、リアルタイムの API テストと検証を継続的に実施することです。API セキュリティ・プログラムは、継続的に動的なテスト、検証、トリアージを実行できるものが理想です。API のセキュリティを API 自体のライフサイクルとして管理する方法について、詳しくはこちらのブログ記事をご参照ください。

知見 2 多くの組織が今も API のセキュリティ侵害を受けている

過去 12 カ月間に、組織内部で開発したクラウドネイティブ・アプリケーションに関連するサイバーセキュリティ・インシデントを経験したことがあるかを尋ねたところ、38% が API の安全でない使用が原因で攻撃を受け、データ漏えいを招いたと答えています。

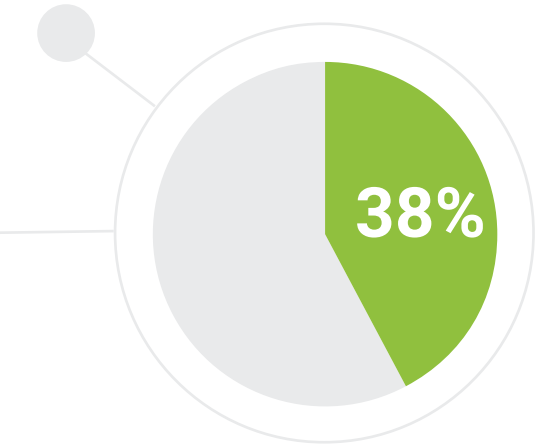
ブラック・ダックの見解

回答者の大半が、組織内部で開発したクラウドネイティブ・アプリケーションに関連して過去 1 年間にさまざまな種類のセキュリティ・インシデントに直面したと答えています。その中で最も多かったのが API の安全でない使用に関するもので、回答者の 38% が API の不適切な使用によってデータ漏洩を招いています。これは、すべての種類のサイバーセキュリティ・インシデントの中で最も高い割合です。

この調査結果は、モダン開発モデルを考えれば驚くべきことではありません。マイクロサービス開発フレームワーク、サーバーレス・テクノロジー、コンテナなどはいずれも、さまざまな種類の言語やフレームワーク、API を使用して開発された小規模な機能を多数組み合わせることによって構成されています。このように、小さなピースをパッチワーク的に組み合わせて全体が構成されているため、API のセキュリティ対策に唯一の万能なソリューションは存在しないことになります。セキュリティ・プログラムは多面的なもの、そして組織固有の環境や依存関係に合わせてカスタマイズできるものが求められます。このように必然的に複雑にならざるを得ない環境では、API がセキュリティ・チームにとっての大きな懸念材料であり、優先課題でもあるという状況が今後も続きそうです。

提言

リスク管理の観点から言えば、API セキュリティ・プログラムが効果を上げるには組織全体のセキュリティ文化が前提条件としてあり、ソフトウェア開発ライフサイクル（SDLC）全体でさまざまな活動を多角的に実施する必要があります。API のセキュリティ問題を一挙に解決してくれるシンプルなソリューションというものは存在しませんが、API のセキュリティ対策をとる上で検討すべき事項はいくつか存在します。



回答者の 38% が
API の不適切な使用によって
データ漏洩を招いています

- **API の検出とインベントリの作成。**ここで重要なのは、すべてのエンドポイントを特定できるかどうかです。適切なツールを使用すれば、アプリケーションで使用している API の検出は容易になります。API を特定できたら、次はそのインベントリを作成します。API を網羅したインベントリがあれば、セキュリティ・スキャンの結果を評価してリスク・フレームワークとポリシー・セットを確立するのが容易になります。

  ただし、API のドキュメントだけを頼りに API の検出とインベントリの作成を行うことはできません。ドキュメントは不完全であったり、内容に誤りが含まれることもよくあります。ドキュメントに記載されていなくても、アプリケーションによって外部に公開されるエンドポイントがあれば、それを特定する必要があります。このようなエンドポイントは過去に評価の対象から漏れている可能性があり、その存在が忘れられ、セキュリティ対策がとられていないことも十分に考えられます。

- **API の評価カバレッジ。**API および API に含まれるエンドポイントのインベントリの作成から始めることに異論はありませんが、API の評価活動を追跡できなければ、いくらインベントリを作成しても成果は上がりません。どの API をテストしたか、そしてこれら API 内のエンドポイントをどれだけ評価したかを特定できていれば回避できたはずの不必要なリスクを抱え込むことになるためです。評価活動は追跡と検証が可能であることが望まれます。つまり、API とそのエンドポイントをすべてスキャンしたことを検証できる必要があります。

- **API の評価。**API の評価は、以下の 3 つの質問から始めることを推奨します。
  - 業務上最も重要なのはどのアプリケーションまたは資産か？この点を重視してデータのセキュリティを高め、不正アクセスから保護します。
  - 呼び出し可能な API を追跡する手段はあるか？それらの API をテストしているか？
  - 使用している API は組織内部で開発したものか？内部開発したものであれば、API へのアクセスに対して厳格な制御を適用できます。サードパーティの API の場合は、制御のしようがありません。

- **API の統合。**API に関する独自のポリシーを作成し、分析します。「アイデンティティ / アクセス管理ツールや暗号化を使用しているか？」「DevOps ツールチェーンに API セキュリティを効果的に実装しているか？」この 2 つの質問に対する答えが両方とも「はい」となるように、あらゆる対策をとる必要があります。
- **API のテストと修正。**デプロイ段階から本番環境にかけて、脆弱な API をテスト、検出、および防止する手段を用意できているでしょうか。

知見 3

API のセキュリティには開発者が責任を負う

ソース・コードに含まれる API の検出と検査に対して誰が第一義的責任を負うかを尋ねたところ、41% が開発者と答え、40% がセキュリティ・チームと答えています。実行時の API セキュリティ対策の適用に対して誰が第一義的責任を負うかを尋ねたところ、44% が開発者の責任と答えています。

ブラック・ダックの見解

シフト・レフトとは開発ライフサイクルのより早い段階からより頻繁にチームがセキュリティ対策を実施することを意味します。この動きに後押しされる形で、開発者がより多くのセキュリティ責任を負うようになっています。しかしそこには課題もあります。回答者の 68% が開発者への権限付与を組織の高い優先事項に挙げている一方、開発チームがセキュリティ・テストの責任を負うことに対して実際に満足しているセキュリティ担当者は 36% にとどまっています。

セキュリティ・アクティビティおよびプロセスに対する開発者の関与を増やすことには明らかな利点があるものの、克服すべき課題もいくつか存在します。開発者がより多くのセキュリティ・タスクを引き受けることについての課題として最も多く挙げられたのが、開発者の負担が増えすぎる（44%）、開発者にはセキュリティ・タスクを実行する十分な資格がない（42%）という意見でした。また、こうした取り組みは最終的にサイバーセキュリティ・チームの仕事を増やしてしまう（43%）という声もありました。

これまでは開発者がセキュリティの責任を負うことがなかったため、開発者には新しいスキルセットの習得が求められます。開発者のほとんどは事前対処的にセキュアなコードを作成するというアプローチをとっておらず、セキュリティを後付けで考えたり、開発完了後の面倒なチェック項目の 1 つと考えたりしています。

API について言えば、その仕様や機能、呼び出しを開発する責任は開発者にあるため、そのセキュリティ責任も開発者が負う必要があります。また、API の構成を最もよく知っているのが開発者である以上、開発者はその修正の最適任者でもあります。開発が完了した後に API の問題を修正しようとしても、それはほとんど不可能であるという意見も一部には存在します。これについて、ブラック・ダックは不可能とまでは考えていないものの、開発完了後に設計上の弱点を見つけて修正することは理想的でないと考えています。したがって、設計の早い段階からセキュリティ専門家と共同で入念に検討と計画を進めることが何よりも重要となります。

提言

開発者が API のセキュリティに取り組むようになると、2 つの明らかな課題が浮かび上がってきます。1 つは開発者の現在のスキルセット、そしてもう 1 つは開発者の遂行能力に対してセキュリティ・チームが抱く不安です。SDLC の設計フェーズでのセキュリティ対策を支援してくれる適切なツールがなければ、こうした不安が生じるのも無理のないことです。

従来、設計フェーズでは静的アプリケーション・セキュリティ・テスト（SAST）が使われていましたが、SAST を使用して API のセキュリティ対策をとろうとしても、ほとんどの組織ではスピードと規模の面で課題が残ります。また、SAST ツールからはリアルタイムの修正ガイダンスやレポートといった、必要かつ重要なフィードバックが得られません。

API の脆弱性を減らすにはシフト・レフトとシフト・ライトのどちらのアプローチが最適なのか、すなわちテストをコーディングの早期段階で実施するのと、最終段階の本番環境で実施するのとどちらがより効果的なのかという点については、議論があります。ブラック・ダックは、あらゆるフェーズで継続的にテストを実施するというよりも、適切なテストを適切なタイミングで実施することが重要であると考えています。具体的には、コーディング中に IDE や CI パイプラインから直接脆弱性に対処できるツールだけでなく、開発者がコーディングを始める前の段階でのツール導入を推奨しています。これにより、脅威モデリングなどのセキュア設計プラクティスを確実に実施できるようになります。脅威モデルのない API は、真の意味でセキュアとは言えないとブラック・ダックは考えています。

最終的な目標は、プロセスのスピードを損なうような遅いツールを使用しないことです。セキュア SDLC の各ステージでさまざまなテストを実行すると、さまざまな情報が得られます。これらすべてのテストを実行するのは結構なことですが、だからと言ってすべてのテストを常時実行できるわけでもなく、そうすることが望ましいわけでもありません。

知見 4 ほとんどの組織が内部開発した API セキュリティ・ソリューションに頼っている

ソース・コードに含まれる API の検出と検査のためにどのような対策を採用しているかを尋ねたところ、38% が組織内部で開発したソリューションを使用していると答えています。

ブラック・ダックの見解

組織内部で開発したソリューションを使用して API のセキュリティに対処するのは、セキュリティ・プログラムの成熟度が低いことを示す指標です。組織に固有の複雑な要件に対処しようとして、より大きなセキュリティ問題に対して応急的な修正策を独自に組み立ててしまうこともあります。知見 1 と 2 からも明らかなように、多くの回答者にとって API のセキュリティが深刻な懸念となっているにもかかわらず、その対策はうまくいっていません。だとすると、これらの組織内部で開発されたソリューションの出来が良くないか、API のセキュリティ対策に必要なすべての機能を実行できていないと考えるのが自然です。特に大企業では、ビジネスの成長とスピードに合わせて拡張可能なエンタープライズ・アプリケーション・セキュリティ・ツールが望まれます。

提言

知見 3 で見てきたように、開発者が API のセキュリティに責任を負うようになっているため、組織は修正作業の迅速化を図るために監視ソリューションを開発者中心型のセキュリティ・ツールに統合しているようです。こうすれば、多くのチームの手を煩わさずにセキュリティの問題を修正できるようになるため、効率の向上には役立つかもしれませんが、こうしたツールにはおそらく十分なスケーラビリティがありません。回答者の 45% が API のセキュリティを最大の懸念に挙げている一方、過去 1 年間で実際に API が原因で攻撃を受けたことのある回答者が 38% にも達していることを考えると、その対策に何らかの問題があるように思われます。

しかし、適切なソリューションを使用すれば、必要なカバレッジを得ることができます。

- **インタラクティブ・アプリケーション・セキュリティ・テスト（IAST）**。使用している言語（Go、Python、Java など）に基づいて顧客の環境にカスタマイズし、バックグラウンドでエージェントを動作させて実施するもので、誤検知が少ないのが特長です。優れた IAST ツールならドキュメントに記載されていない API エンドポイントも特定できるほか、アタック・サーフェスのテスト・カバレッジも測定できます。
- **ファジング・テスト**。Wi-Fi、Bluetooth、IoT（Internet of Things）などの通信テクノロジーにおける未知の脆弱性を検出するもので、プロトコル・テストとも呼ばれます。

ブラック・ダックのソリューション

ブラック・ダック Seeker® インタラクティブ・アプリケーション・セキュリティ・テスト（IAST）は、REST や GraphQL など多くの API をテストします。通常の開発および QA テスト時に API のすべての経路とエンドポイントを自動で検出して表示するほか、DevOps の CI/CD パイプラインでも効果を発揮します。Seeker にはリアルタイムのアラート機能や、インバウンドとアウトバウンド双方向のすべての呼び出しを視覚的に表現するデータフロー・マップ機能のほか、どのコード行に脆弱性があるかを指摘して詳細なガイダンスを提示する機能もあるため、開発者は即座に修正が可能です。Seeker が提供する継続的なテストと検証により、DevSecOps チームはワークフローとの摩擦を最小限に抑えながら迅速に対応できるようになります。

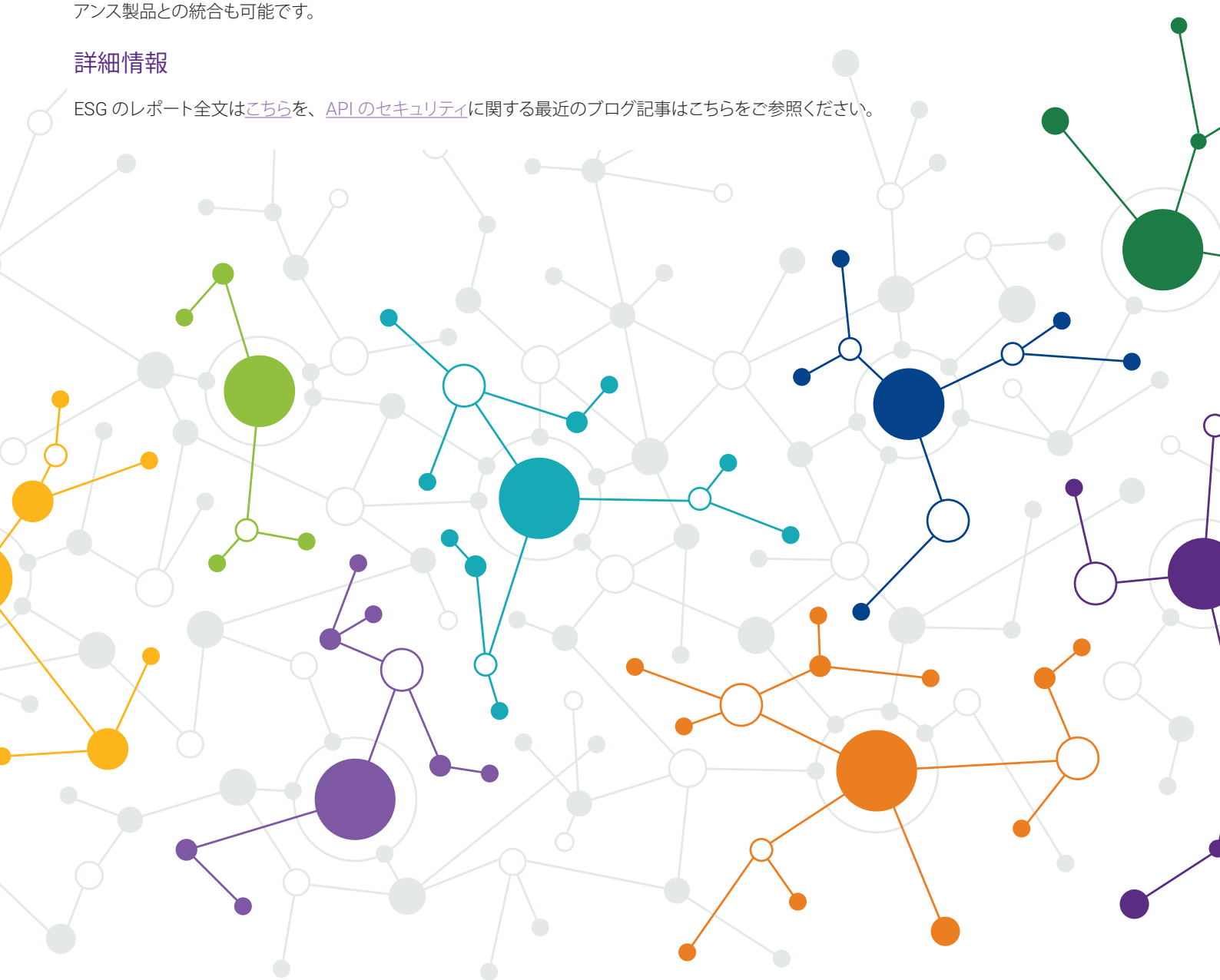
Seeker 以外にも、ブラック・ダックはクラウドネイティブ・アプリケーションのセキュリティ保護に役立つ各種スキャニング・テクノロジーを包括的に提供しています。Code Sight™ を使用すると、開発者は IDE から直接軽量の SAST を実行し、コードに潜む脆弱性を即座に検出して修正できます。Coverity® 静的解析および Black Duck® ソフトウェア・コンポジション解析は、IaC（Infrastructure-as-Code）やコンテナ化したアプリ、イメージのセキュリティ保護に役立ちます。

ブラック・ダック Continuous Dynamic は、SaaS（Software-as-a-Service）形式の動的アプリケーション・セキュリティ・テスト（DAST）ソリューションで、スケーラブルな web セキュリティ・プログラムの導入が可能です。web サイトの数や、その更新頻度にかかわらず、WhiteHat Dynamic はあらゆる要求にスケーラブルに応えます。セキュリティ・チームと開発チームは、QA および本番環境において迅速、正確、そして継続的な脆弱性診断が可能となります。WhiteHat Dynamic はハッカーと同じ手法を用いて弱点を見つけるため、ハッカーに悪用される前に先回りして弱点を修正できます。

また、Continuous Dynamic は一般的なバグ追跡システム、セキュリティ情報 / イベント管理ソリューション、ガバナンス / リスク / コンプライアンス製品との統合も可能です。

詳細情報

ESG のレポート全文はこちらを、API のセキュリティに関する最近のブログ記事はこちらをご参照ください。



ブラック・ダックについて

ブラック・ダックは、業界で最も包括的かつ強力で信頼できるアプリケーション・セキュリティ・ソリューション・ポートフォリオを提供します。ブラック・ダックには、世界中の組織がソフトウェアを迅速に保護し、開発環境にセキュリティを効率的に統合し、新しいテクノロジーで安全に革新できるよう支援してきた比類なき実績があります。ソフトウェア・セキュリティのリーダー、専門家、イノベーターとして認められているブラック・ダックは、ソフトウェアの信頼を築くために必要な要素をすべて備えています。
詳しくは www.blackduck.com/jp をご覧ください。

ブラック・ダック・ソフトウェア合同会社
www.blackduck.com/jp