CYBER GRID JOURNAL

サイバー・グリッド・ジャーナル

VOL. 8

「創る」と「衛る」の
合わせ技で防ぐ、
見えない脅威
「サプライチェーン・リスク」

CYBER GRID JOURNAL VOL.8

TABLE OF CONTENTS

巻頭言

サプライチェーン・リスクマネジメントへの挑戦

船引 裕司
サイバー・グリッド・ジャパン　GM

令和の時代を迎え、社会システムのIT化がますます加速しているように感じられます。特に、FinTechの台頭は著しく、最近やや乱立気味の“〇〇ペイ”といった名称のスマホ決済サービスは、我々庶民の日々の購買活動におけるツールとして利用されるが故、爆発的に普及し始めています。これに伴い、事故の発生も見られるようになり、先般発生した事例では、相当数のスマホ決済サービス利用者が「身に覚えのない利用」に遭遇し、大きく報道されました。一方、代用通貨というより投資の対象として取り上げられることが多い暗号資産に関しても、億円単位の“流出”事故が相変わらず起きているのが現状です。旧来、金融サービス・勘定系のシステムにおいては安全性の確保が絶対条件であり、プラットフォームが進化してもこの要件は保持されるべきです。しかし、オープン化、複雑化の流れの中でシステムに潜在するリスク、特に、システム全体を構成する膨大な要素に潜むリスクを完全には検証し得ない、という現状において、リスクを完全に排除できないという事実は、今後も社会的に大きなダメージを与えるインシデントとして顕在化してくるのではないか、と考えられます。加えて、システムに組み込まれる大量の機能モジュールもますますブラックボックス化しています。例えば、スマホアプリのソフトウェアトークンにより本人認証が実現されている実態を、利用者はほとんど意識せずに使っているのが実情でしょう（そういった意味では、本人宛の電子メールおよびSMSの活用や、手元カードに記された乱数表の数字入力等の方が、2段階認証の実感を持ちやすいかもしれません）。

さて、こうした潜在リスクに対して、ID/パスワードの管理やウイルスソフト更新など利用者で自己防衛することは当然として、システムを供給・運営する側としても、利便性を極力落とさずこのリスクを極限まで低減化する、という課題はビジネスの命運をも左右しかねない大きな要素になっています。そして、数年前に登場した概念「サプライチェーン・リスク」への対応については、重要インフラの保護や国家安全保障の観点での特定ベンダー製品の排除といった動きだけでなく、もっと身近なコンシューマー向け製品の品質管理等においても議論がなされている状況にあります。例えば、サイバー・グリッド・ジャパンの研究テーマの一つでもあるIoTセキュリティの分野にも関連するものとして、いよいよ今秋には、高速道路上での自動運転(*)を可能とする車が発売されますが、1台の車両は1,000を超える供給者から約3万点の部品が調達されて組み上げられているとされます。これはまさにサプライチェーンの塊のようなものと言えましょう。

こうした現況を踏まえ、本号の特集ではサプライチェーン・リスクの問題、特に、システム開発の側面においてどうセキュリティを担保していくか、というテーマを取り上げました。サイバーセキュリティ対策のリーディングカンパニーであると同時に、システムインテグレーションを事業の柱の一つとする当社ならではの視点やアプローチが皆さまのご参考になれば幸いです。

(*)メーカー発表は「自動運転ではなくハンズオフドライブが可能な運転アシスト」という位置付け

「創る」と「衛る」の合わせ技で防ぐ、見えない脅威「サプライチェーン・リスク」

サイバー・グリッド・ジャパン
サイバー・グリッド研究所長 仲上 竜太

レストランで楽しむことができるさまざまな料理、スーパーや食料品店で販売される豊富な食材、手軽かつ安価なファストフード、コンビニエンスストアで24時間いつでも購入することができる弁当など、私たちの生活と食は、密接な関わりを持っています。

普段、私たちが口にする食品は、農業や漁業、畜産業などの生産に始まり、加工や製造、そして流通・販売と、多くの過程を経て私たちの食卓へ届いています。

このような生産、加工、流通を通して製品が消費者の手に届くまでの過程は、「供給の連鎖」を意味する「サプライチェーン」と呼ばれています。

普段何気なく購入している食品や食材は、さまざまなサプライチェーンを通じて、私たちの手元まで届いています。

世界的規模の分業によって構築されたサプライチェーンは、その経済的合理性によって、我々現代人に安価に、安全で健康的な食生活をもたらしました。

その一方、「異物混入」や「食品偽装」などの食品にまつわるトラブルが、たびたび報道されています。

重大な場合には集団食中毒などの極めて大きな健康被害が発生することもあり、その都度、改めて食の安全性が見直されています。

商品を製造するサプライチェーン上のどこか一カ所で異物混入や故意の偽装が行われ、発覚した場合、その商品全てに対する顧客からの信頼は、一瞬にして失われてしまいます。

自分たちがどれほど気をつけていたとしても、サプライチェーンのどこかにトラブルが混入されてしまうことで、そのトラブルの当事者となってしまうのです。

これらのリスクは、「サプライチェーン・リスク」と呼ばれ、全ての製造業において対処すべき課題となっています。

情報システムのサプライチェーン・リスク

デジタルトランスフォーメーションによって、社会的活用の幅が限りなく広がっている情報システムにおいても、この「サプライチェーン・リスク」に対応せざるを得ない状況が生まれています。

これまでのサイバーセキュリティでは、公開されているホームページへの攻撃、OSや基盤ソフトの脆弱性をついた攻撃、メールを使用した攻撃など、インターネットを通してシステムの外部から侵害が行われることを想定して対策が行われてきました。

しかし昨今では、システム外部からの侵害だけでなく、正規のサプライチェーンを通してシステム内部に、しかも知らない間に攻撃の糸口が埋め込まれてしまう事象が発生しています。

2017年9月、無料で利用可能なWindows用端末管理ソフトで、メーカー正規の自動更新機能を通して、利用者の環境にマルウエアに汚染された更新ソフトが配布される事態が発生しました。

2019年には台湾のパソコンメーカーでも、パソコン管理ソフトの自動更新を通じて、一部のパソコンに外部から侵入可能なバックドア（裏口）が作成されるという事件が発生しています。

このように、情報システムにおいてのサプライチェーンの弱い部分に付け込むことで、正規のルートを乗っ取り、不正を行うサイバー攻撃のことを「サプライチェーン攻撃」と呼んでいます。

システム開発の現場にも広がるサプライチェーン攻撃

2017年には、インターネットにソースコードが公開されているオープンソースソフトウエアに、巧妙に混入したマルウエアが問題となりました。

システム開発時に使用する統合開発環境プログラムである「Eclipse（エクリプス）」の人気プラグイン「Eclipse Class Decompiler」では、利用者である開発者のパソコン内の情報が不正に窃取される機能が意図的に混入されていました。

このプラグインでは、正規に公開されているソースコードとは異なる不正コードが配布用の正規のバイナリファイル（人間が読めないデータ形式のファイル）に含まれており、開発者コミュニティに報告されるまでの間、公式の開発環境パッケージに含まれて配布されるなど幅広いユーザーに影響を及ぼしました。

オープンソースとして公開されているソフトウエアでは、ソースコードがインターネット上に公開されています。そこでは、世界中のボランティアエンジニアによるオープンな協力開発体制が組まれており、プログラムの挙動はある程度担保されています。

しかし、作者によって作成・配布されたバイナリファイルが公開されているプログラムコードと異なるというケースは想定されていませんでした。

この問題では、情報システムやミドルウエアそのものの脆弱性ではなく、正規の配布サイトを通じて、悪意を持ったプログラムが配布されることでソフトウエア開発環境が狙われてしまう可能性が証明されました。

多重構造に潜むサプライチェーン・リスク

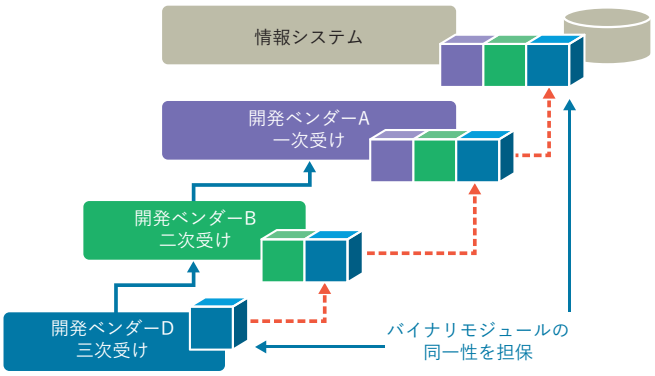
情報システムの開発では、一部のモジュール開発を外部のベンダーに委託することはよくあります。

その際、契約条件によってはバイナリファイルによる納品が行われます。

ソースコード上での検証ができないバイナリファイルで提供されたモジュールは、その挙動を確認することによって動作を検証することは可能ですが、意図的な脅威が含まれた場合に検知することは非常に困難です。

日本の情報システム開発で多くみられる多重下請け構造は、高度に複雑化したソフトウエアサプライチェーンといえます。

納品されたバイナリファイルが正常でも、複数の工程を経る過程のどこかで悪意を持って改ざんが行われ、脅威が混入する可能性が存在しています。



D社の開発したモジュール

どう対応する？サプライチェーン・リスク

IPA❶が2019年6月に発表した「情報セキュリティ10大脅威2019❷」では、組織に対する脅威として「サプライチェーンの弱点を悪用した攻撃の高まり」が、初めてランクインし、注目を集めました。

IPAの提示したガイドラインでは、再委託先・再々委託先の管理は困難としつつも、業務委託先に適切なセキュリティ管理を要求することを対策として求めています。

米国では、米政府機関が調達を行う際のセキュリティガイドラインとしてアメリカ国立標準技術研究所（NIST）が定めた、NIST SP800-171❸が運用されています。

この基準は、安全保障の観点からサプライチェーンに対してセキュリティ基準に対する準拠を求めるもので、政府だけでなく産業界全体の基準として、米国以外でも採用する動きが広がっています。

日本政府においても、令和元年度に実施する防衛調達において、NIST SP800-171に準拠した新情報セキュリティ基準を制定することを防衛装備庁が方針として発表しています。

サプライチェーン・リスクに対応する対処として、SP800-171のようなセキュリティ基準を、自組織だけではなく、発注先のベンダーに求めることで、セキュリティへの取り組みが可視化され、サプライチェーン全体のセキュリティ意識向上を図ることが可能になります。

ソフトウエア・サプライチェーン・リスクに対しては、米国商務省電気通信情報局（NTIA）において、「Software Bill of Materials / SBoM＝ソフトウエア部品表」の採用が議論されています。

SBoMは、ソフトウエアを構成する部品を一覧表として管理することによって、そのソフトウエアがどのようなモジュールで構成されているのかを可視化する仕組みです。

昨年順位	個　人	順位	組　織	昨年順位
1位	クレジットカード情報の不正利用	1位	標的型攻撃による被害	1位
1位	フィッシングによる個人情報等の詐取	2位	ビジネスメール詐欺による被害	3位
4位	不正アプリによる スマートフォン利用者への被害	3位	ランサムウエアによる被害	2位
NEW	メール等を使った脅迫 詐欺の手口による金銭要求	4位	サプライチェーンの弱点を悪用した 攻撃の高まり	NEW
3位	ネット上の誹謗・中傷・デマ	5位	内部不正による情報漏えい	8位
10位	偽警告によるインターネット詐欺	6位	サービス妨害攻撃による サービスの停止	9位
1位	インターネットバンキングの 不正利用	7位	インターネットサービスからの 個人情報の窃取	6位
5位	インターネットサービスへの 不正ログイン	8位	IoT機器の脆弱性の顕在化	7位
2位	ランサムウエアによる被害	9位	脆弱性対策情報の公開に伴う 悪用増加	4位
9位	IoT機器の不適切な管理	10位	不注意による情報漏えい	12位

「情報セキュリティ10大脅威 2019」　NEW：初めてランクインした脅威

現代のシステム開発では、オープンソースの利用や多重化された受注構造によって、ステークホルダーが指数関数的に増大しています。このような現状において、前出の仕組みを活用した管理強化は、サプライチェーン・リスクに対して一定の効果を得られるものと考えられます。

「創る」と「衛る」を一体化したモノづくりへ

サプライチェーン・リスクの高まりに対する対応は、ソフトウエア開発現場にとって新たな管理項目の増加となり、開発現場の負担増大につながります。

開発現場においては、従来の管理項目に加え、SBoMへの対応や適切なモジュール管理、オープンソースソフトウエアのコミュニティからの情報収集など、ソフトウエア開発者を支援する仕組みを整えていくことで、サプライチェーン・リスクに対応することが重要です。

また、開発を行う仕組みそのものを変えることで、サプライチェーン・リスクを軽減することも可能です。

開発と運用を一体化し、ソフトウエアを含めたシステム全体の開発プロセスを運用と同時並行して実施するDevOpsの取り組みは、運用で行うセキュリティチェックの仕組みを開発段階に取り入れることで、より安全な開発環境を実現することができます。

これらの取り組みにより、システムそのもののセキュリティと、開発環境のセキュリティを同時に担保することによって、これから開発に求められるセキュリティレベルを効果的に達成することが可能になります。

複雑化する課題に挑戦するエンジニア

高度化・複雑化した情報システムの開発、運用を担うエンジニアは、次々に登場する新たなテクノロジーに対応しながら、サプライチェーン・リスクをはじめとした増え続ける脅威と日夜戦っています。

企業や組織における資産がデジタル化され、データそのものが価値を生み出す現代社会において、情報システムは私たちの生活と切り離すことのできない存在となっています。

24時間、365日動き続けるデジタル社会を創り、衛り続けるエンジニアの不断、不休の取り組みが、私たちの生活を支えています。

❶ IPA:独立行政法人情報処理推進機構https://www.ipa.go.jp/　❷ 情報セキュリティ10大脅威2019（IPA）https://www.ipa.go.jp/security/vuln/10threats2019.html
❸ セキュリティ関連NIST文書（IPA）https://www.ipa.go.jp/security/publications/nist/index.html

システム開発とセキュリティのこれから

アジャイル開発やDevOps時代に企業はどう取り組むべきか

倉持 浩明　株式会社ラック 常務執行役員　CTO

はじめに

あなたの会社では、新機能のシステム開発にどれくらいの時間がかかるだろうか。新機能のリリースに１カ月から半年以上の期間を要するなら、残念ながらあなたの会社のデジタルトランスフォーメーション(DX)対応能力は「低い」と言わざるを得ない❶。インターネット専業企業が「１日に何回も新機能をリリースする」という話を聞いて自社とは関係ないと思うだろうか。今日においては、インターネット専業の企業だけでなく、金融業や製造業など、あらゆる企業がインターネットを用いてビジネスを行っている。UberやAirbnb、Netflixなどのディスラプター(破壊者)は従来の産業構造の間隙を突き、インターネットとソフトウエア並びにビジネススピードを武器にしてあらゆる企業の前に立ちはだかっている。彼らのように「１日に何回も新機能をリリースする」能力を持った企業と闘っていくために、徹底的にITを活用する「デジタルトランスフォーメーション」に関する組織能力を強化しなければならない。こうした圧倒的なデジタルトランスフォーメーションへの対応能力は、「アジャイル開発」や「DevOps(デブオプス)」で支えられている。本稿では特に昨今注目を集めるDevOpsに焦点を当て、DevOps時代に求められるセキュリティについて解説する。

図１:新機能のリリース頻度と企業のIT活用能力 (「2016 State of DevOps Report」を基に作成)

	Highパフォーマー	Mediumパフォーマー	Lowパフォーマー
リリース間隔	オンデマンド(1日に複数回のリリースが可能)	1週間から１か月に１度	1週間から6か月に１度
変更までの所要時間	1時間未満	1週間から１か月	1か月から6か月
障害が発生した場合の修復に要する平均時間(MTTR)	1時間未満	1日未満	1日未満
変更が失敗する確率	0－15%	31－45%	16－30%

DevOpsとは何か

DevOpsとは、「システム開発チームと運用チームが協調しあうことで、確実かつ迅速にユーザーにシステムがもたらすビジネス価値を届け続ける」という概念である。DevOpsという言葉が生まれるきっかけは、2009年にカリフォルニア州サンノゼで行われたVelocity2009というイベントで、写真共有サービスのFlickrのエンジニアが発表したプレゼンテーション(「毎日10回以上のリリースを行う、Flickrにおける開発と運用の協調」❷)だといわれている。

ここで、DevOpsの対極にあると思われるような、これまでの一般的なシステム開発と運用の現場を考えてみよう。システム開発チーム(Dev)と運用チーム(Ops)との間にはどのような緊張関係が存在するだろうか。システム開発はプロジェクトとして進行管理されるのが一般的である。そして、そのプロジェクトは常に納期を意識し、納期までにシステムをリリースすることに大変に重きを置いている。納期を重視する開発チームとしては、納期を優先するために開発後期に追加された仕様変更や、テストフェーズで発見された軽微な不具合については「運用で回避」することを考えがちである。一方、運用では、

アジャイル開発やDevOpsでは開発・運用チームだけでなく営業やセキュリティチームも協調していくことが大切

1 「2016 State of DevOps Report」https://puppet.com/resources/whitepaper/2016-state-of-devops-report
2 「10+ Deploys Per Day: Dev and Ops Cooperation at Flickr」https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr

システムが利用される限りは円滑に機能するように保守運用を継続する必要がある。運用チームにとっては増え続けるシステムを一定のコストで運用していくことを求められているから、こうした「運用回避策」を嫌って当然である。こうしたことの積み重ねが、開発チームと運用チームの間に大きな緊張関係を生じさせているのが現実である。同じような緊張関係は、営業チームと開発チーム、あるいは開発・運用チームとセキュリティチームとの間でも同様に生じている。

「1日に何回もシステムの新機能をリリースできる」ようにしていくためには、迅速にシステムの開発やテストを行い、開発チームと運用チームでの情報共有を進めていく必要がある。それを支える方策はたくさんあるが、単にツールを導入して解決できるわけではない。「システムがもたらすビジネス上の価値を継続してユーザーに届け続ける」ためには開発チームと運用チームだけでなく、営業チームとセキュリティチームを含めた各チームが協調する組織文化を、経営者のリーダーシップのもとで構築していく必要がある。あなたの会社のシステムがうまく機能するために、ビジネスオーナーや顧客との関係強化が重要であればあるほど、DevOpsは有効な手段である。そして、その実現は情報システム部門だけに任せるのではなく、経営者がリーダーシップを発揮して体制を作る必要があるほど、難しく、課題の多いものなのである。

アジャイル開発とは何か

ここで、DevOpsと一緒に語られることの多い「アジャイル開発」について簡単に解説しておこう。アジャイルとは直訳すれば「機敏な」「迅速な」という意味の英単語である。アジャイル開発においては、1週間や2週間といった非常に短いサイクルでシステム開発のリズムを維持しながら、ユーザーに価値を届けていく。初期リリースの段階から、機能は少ないながらも「完全に動作する」システムを提供していく。こうすることで、開発チームとしてはPDCAサイクルを小さく回し、ビジネスの視点では開発しているものが本当に正しいのか、ユーザーに価値を提供できているのかという大きなPDCAサイクルも回していく。アジャイル開発とは、経験則から導き出された「ソフトウエアを正しく開発していくための方法論」なのである。

DevOpsが開発チームと運用チームの協調に焦点を当てているのに対し、アジャイル開発では主に開発チームに焦点が当たっている。企業がDevOpsの組織能力を獲得していく過程（すなわち、開発と運用の協調により組織能力を最大化していく過程）においては、前提として開発チームがアジャイル開発手法を習得している必要がある。世界中のDevOps導入企業の実態を調査した「2018 State of DevOps Report❸」においてもDevOpsの導入・定着に至る道程の第1ステップとして「開発チームでアジャイル開発手法が適用されること」から始める企業が多いとされている。アジャイル開発を導入するには、ソフトウエアの構成管理システムを導入し、システムのビルドやテストの自動化を実現するシステム基盤を構築することが必要になる場合が多い。こうしたシステム基盤を整備することで、1週間や2週間という短いサイクルでシステムをリリースしていくことが可能になるからである。

DevOpsの導入に先立ち開発チームがアジャイル開発を導入している企業が多い

図2：企業におけるDevOps適用の5つのステージ（「2018 State of DevOps Report」を基に作成）

Stage 1	**Normalize the technology stack** 開発チームでアジャイル開発手法が適用される 構成管理システムが導入され、継続的インテグレーションを始める基礎ができる
Stage 2	**Standardize and reduce variability** 開発チームと運用チームの協調が始まる OSやミドルウエアの標準化が進み、適用範囲を拡大させる準備が整う
Stage 3	**Expand DevOps practices** リリース時のシステム切替方法の再利用、システム基盤変更の事前テスト リリースにまつわる苦痛・失敗が減少していく
Stage 4	**Automate infrastructure delivery** システム設定やシステム資源の割当の自動化 セキュリティポリシー適用の自動化
Stage 5	**Provide self-service capabilities** 各種リソースをセルフサービスで利用可能、障害対応の自動化 セキュリティチームが設計やリリースに関与するようになる

❸「2018 State of DevOps Report」https://puppet.com/resources/whitepaper/state-of-devops-report

アジャイル開発とDevOpsのセキュリティ

これまでの話を踏まえ、企業がデジタルトランスフォーメーションに向き合い、アジャイル開発やDevOpsを導入していく上で必要なセキュリティ対策の考え方について説明しよう。なお、技術的なポイントについてはP.10のコラム「アジャイル開発やDevOpsにセキュリティを導入していく上で有効な技術的施策」にまとめてあるので必要に応じて参照していただきたい。

セキュリティ対策の「シフトレフト」に取り組む

従来型のシステム開発において、セキュリティ機能が正しく実装されたかどうかは、テストフェーズにおいて検証されることが多い。しかし、システム開発においては後工程になって検出された課題ほど、対処に時間とコストを要する。開発や設計、場合によっては要件定義まで戻って対応を行う「手戻り」が大きくなるからだ。当社のセキュリティ診断サービスの結果をまとめた「セキュリティ診断レポート 2018 陽春」❹では、診断したシステムの半数を超える54％のシステムにおいて重要な問題点（High/Mediumリスク）を検出したとしている。品質を担保するためには、テスト工程で品質を「検査」するのではなく、設計・開発の工程において品質を「作り込む」のが合理的である。ソフトウエアのセキュリティについても同様である。

システムのセキュリティ品質を高めるためには、開発の早い段階（工程表のなるべく「左側」）からセキュリティに関するレビューやテストを実施するなどの対策が有効である。アプリケーションセキュリティに関する世界中のプロフェッショナルが非営利で参画するOWASP（Open Web Application Security Project）❺がまとめた「OWASP Top 10 Proactive Controls（安全なWebシステム開発への事前の対策ガイドライン）❻」では、システム開発者が注意すべき重要な項目の第1位として「セキュリティ要件の定義」を挙げている。セキュリティ対策の力点を、工程表上の左側に移動（シフト）させていくことから、こうした動きを「シフトレフト」という。システム開発の初期段階からセキュリティについて考慮していく取り組みは、アジャイル開発やDevOpsにおいても有効である。アジャイル

図3：重要な問題点（High/Mediumリスク）を検出した診断対象
（ラックの「セキュリティ診断レポート 2018 陽春」より）

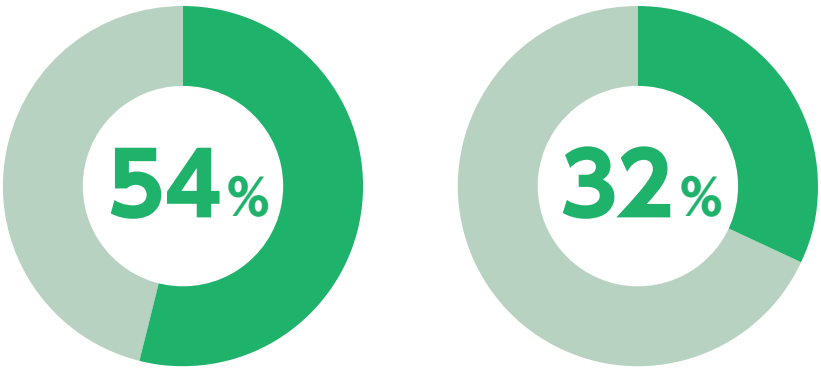


図4：「セキュリティ要件の定義」が最も大事
（「安全なWebシステム開発への事前の対策ガイドライン」より）

C1:	セキュリティ要件の定義
C2:	セキュリティフレームワークやライブラリの活用
C3:	セキュアなデータベースアクセス
C4:	エンコーディングおよびエスケープ
C5:	すべての入力値の検証
C6:	アイデンティティと認証管理の実装
C7:	適切なアクセス制御の実装
C8:	すべてのデータの保護
C9:	ロギングとモニタリングの実装
C10:	エラー処理と例外処理

❹「セキュリティ診断レポート 2018 陽春」https://www.lac.co.jp/lacwatch/pdf/20180405_sec_report_vol2.pdf
❺「OWASP」https://www.owasp.org/ ❻「OWASP-Top10-Proactive-Controls-2018-JP」https://github.com/shonantoka/OWASP-Top10-Proactive-Controls-2018-JP

開発においてシフトレフトを実現するには、開発チームがセキュリティ対策を「プロジェクトの初期段階から取り組むべきもの」と認識する必要がある。そのためにはセキュリティ担当者を1週間や2週間のサイクルで行われるシステムのレビュー（デモ）に初期段階から招集し、セキュリティ面からの指摘や支援を得ることが効果的である。同時にセキュリティ担当者は開発チームのレベルに応じて必要なセキュリティ教育を企画・実施すると良い。開発の初期段階で開発チームを軌道修正することができれば、将来的な手戻りリスクを大幅に低減することができる。OWASPにはシステム開発者に対するセキュリティ対策の教育で有用なコンテンツも提供されている。

OWASP Top10

OWASP Top 10 は、Webアプリケーションの脆弱性のカタログであり、最も重大なWebアプリケーションリスクを示している。
https://www.owasp.org/images/2/23/OWASP_Top_10-2017%28ja%29.pdf

OWASP Top10 Proactive Controls （安全なWebシステム開発への事前の対策ガイドライン）

OWASP Top10 Proactive Controlsは、OWASP Top10で示された脆弱性を作り込まないようにするためのアプローチをまとめた開発者向けのドキュメントである。OWASP Japanの有志による日本語訳もGitHubで公開されている。
https://github.com/shonantoka/OWASP-Top10-Proactive-Controls-2018-JP

OWASP Cheat Sheet

OWASP Cheat Sheetは、認証や認可をはじめとしたセキュリティに関わる機能を実装する際に考慮すべきセキュリティ対策についてまとめたドキュメントである。設計や開発時のチェックシートとして用いることでセキュリティを高めることができる。
https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series

セキュリティ対策をエンジニア全員の仕事にする

もともとDevOpsという言葉が生まれた背景には、開発チームと運用チームの組織が縦割りで、お互いに言い争いをしているという状況がある。同じような状況は開発・運用チームとセキュリティ担当者の間でも存在する。ソフトウエア開発の現場がアジャイル開発やDevOps能力を獲得していくと、開発・運用チームとセキュリティ担当者の間の緊張関係はこれまで以上に高まってくる。開発・運用チームが非常に短いサイクルでソフトウエアをリリースできるようになってくると、セキュリティ担当者がボトルネックになる場合がある。開発や運用のエンジニアに比べてセキュリティ担当者の人数は圧倒的に少ないのが現状だからだ。

セキュリティ担当者が持つ優れた知見や能力を最大限に活かすためには、セキュリティ対策をエンジニア全員の仕事にしていく必要がある。セキュリティ対策をエンジニア全員の業務に組み込んでいくことで、組織全体としてのセキュリティレベルが向上する。そしてセキュリティ担当者はより高度かつ専門性の高い業務に集中することができるようになるだろう。

つまり、デジタルトランスフォーメーション時代を勝ち残っていくためには、企業はDevOpsにセキュリティの協調を加えた「DevSecOps能力」の獲得が必要であると言える。

最後に

アジャイル開発やDevOpsのムーブメントは、一部のユーザー企業だけでなく、システムインテグレーション事業者の間でも着実に広がりを見せている。当社でも2018年7月にアジャイル開発センターを設立し、アジャイル開発案件の受託やエンジニアに対する教育活動を行っている。しかしながら、長年にわたって蓄積された開発チームと運用チームの関係性の変革や、ユーザー企業とシステムインテグレーション事業者の間の関係性を変革するのには時間を要する。

アジャイル開発やDevOps時代においてセキュリティは各プロセスに組み込まれていく必要があるが、それによりこれらの手法が持つ「協調する」という本質が損なわれてはならない。アジャイル開発やDevOpsという新たな組織能力を獲得した先にある未来を、エンジニアだけでなく経営者やユーザー企業も巻き込んで議論していき、ワクワクする世界を創造していこうではないか。

アジャイル開発やDevOpsにセキュリティを導入していく上で有効な技術的施策

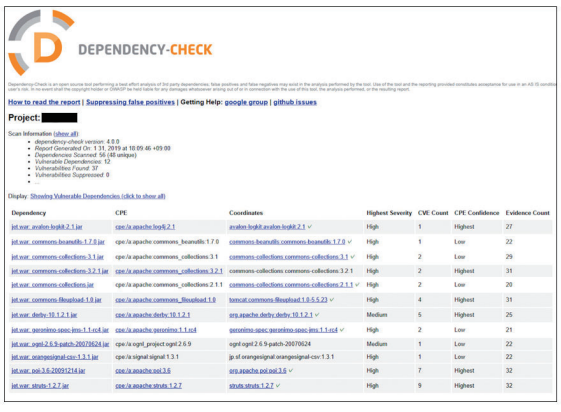
アジャイル開発やDevOpsは、ツールやソリューションを導入すれば実現するものではない。しかし、アジャイル開発やDevOpsを実現するための基盤整備や自動化の取り組みはセキュリティ対策レベルを向上させることに寄与する。このコラムでは、アジャイル開発やDevOpsにセキュリティを導入していく上で有効な技術的施策について解説する。

その1 セキュリティ対策のセルフサービス化を推進する

エンジニア自身が日々の業務の中で日常的にセキュリティ対策を行うことができれば、開発・運用チームはセキュリティに関するより早いフィードバックを得ることができる。ソフトウエアのセキュリティテストで使うツールなどを、開発・運用エンジニアが使えるように環境を整備していくことから取り組むと良いだろう。

アプリケーションのセキュリティをテストする方法としては静的セキュリティテスト（Static Application Security Testing、SAST）と動的セキュリティテスト（Dynamic Application Security Testing、DAST）がある。静的セキュリティテストはアプリケーションのソースコードを検査することでセキュリティ上の問題点を検出する手法であり、開発が終了していない段階から実施することができる。一方、動的セキュリティテストは実際にアプリケーションを動作させることで脆弱性を確認する手法であり、セキュリティ担当者が行う侵入テストに近い。静的セキュリティテストと動的セキュリティテストには一長一短があるので、組み合わせて使うと良い。ただし、こうしたセキュリティテストツールによるテスト結果には、誤検知や過剰検知が含まれることが多い。テスト結果についてはセキュリティ担当者が確認し対応が必要なものと不要なものをトリアージし、ツールをチューニングしていく。また、こうしたツールのテスト結果の意味と対応方法についてエンジニアに対してアドバイスを施していく必要もある。

図5：OWASP Dependency Checkの実行結果
アプリケーションの依存関係を解析し脆弱性のあるソフトウエアが含まれているかどうかをレポートする。



システム運用の観点では、自社で開発したものではないソフトウエアに脆弱性が存在しないかどうかにも目を配る必要がある。具体的にはミドルウエアやフレームワークそしてライブラリやコンポーネントなどである。こうした製品の脆弱性情報を把握し、既知の脆弱性があれば速やかにパッチを適用しアップデートを行う。アプリケーションが利用しているライブラリやコンポーネントを洗い出し、既知の脆弱性が含まれているかどうかをチェックするにはOWASP Dependency Check[7]などのツールが活用できる。

セキュリティ担当者はこうした一連のプロセスを自動化するツールを導入することでセキュリティ対策を開発・運用チームの日常的な業務に組み込んでいく。開発の初期からテストを自動化するための環境を構築し、テストコードを記述していくのは従来型の開発テスト手法に比較して初期投資がかかる。一見無駄なことのように感じるかもしれないが、こうした自動化に対する投資対効果はセキュリティ対策の点でも非常に大きいのである。

その2 システムへの不正アクセスの傾向をモニタリングする

セキュリティ担当者の主要な業務の一つに、システムへの不正アクセスをモニタリングするセキュリティ監視がある。多くの企業ではすでに何らかのセキュリティ監視を導入していると思われるが、セキュリティ監視から得られた自社のシステムに対する不正アクセスの傾向を開発チームや運用チームと日頃から共有しておくことも忘れてはならない。可能であれば運用チームが用いているダッシュボードなどのツールにセキュリティ監視のアラートも表示されるように統合する。自社のシステムに対して実際に行われている不正アクセスの傾向を、エンジニア全員はもとより経営者を含むビジネスサイドが目にする場所に置くことで、セキュリティに対する意識が向上していくことが期待できる。

その3 開発環境や自動化されたデプロイメントパイプラインを防御する

開発環境を構築する作業の自動化やテスト自動化をはじめとして、本番環境へのアプリケーションのデプロイや、インフラ構成など、自動化の対象は多岐にわたる。

こうした自動化されたプロセスにおいて、開発環境や本番環境が悪意のあるソフトウエアによって汚染されてしまうことを防ぐ必要がある。dockerなどのコンテナを利用したシステム開発が増えているが、コンテナで利用するイメージファイルをインターネットからダウンロードして利用している場合には、イメージファイルが不正なものに置き換えられていないかどうかをチェックする仕組みを考慮する。

併せて考慮しておきたいのが、パスワードなどの認証情報の管理である。自動化に当たり、スクリプト中に構成管理サーバーやインテグレーションサーバーにログインするためのパスワードや暗号化キーなどの秘密情報が必要になる場合がある。またクラウド環境においては、数十台から数百台に及ぶサーバーの認証情報を管理する必要がある。こうした認証情報を安全に管理していくためにはVault[8]などのツールを使った運用の導入を検討することを推奨する。

DevOpsとセキュリティの関係について、「2018 State of DevOps Report」では「DevOpsのハイパフォーマーは、ローパフォーマーな組織よりもセキュリティ対応に費やす時間が半分で済んでいる。セキュリティ対策に日々取り組んでいるからだ」としている。アジャイル開発やDevOpsに正面から取り組んでいくことが、結果的にセキュリティの向上にもつながるのである。

[7]「OWASP Dependency Check」https://www.owasp.org/index.php/OWASP_Dependency_Check
[8]「HashiCorp Vault」https://www.lac.co.jp/service/product/vault.html

セキュリティ・ケイパビリティ

設計段階からセキュリティを織り込む打開策

OWASP Japan チャプターリーダー　岡田 良太郎

「シフトレフト」という言葉を聞いたことがありますか。これは、原義的には、後工程と前工程の連携によって改善を進めるためのパターン・ランゲージです。システム開発においては、ソースコード、コンポーネント、アーキテクチャ設計、また開発プロセスの生産技術そのものを改善する取り組みなど、システム開発のもっと早い段階でセキュリティを織り込もうというメッセージとして使われてきました。

OWASP Top 10-2017をじっくり読んでみると、セキュリティ脆弱性に対応するためには、実装以前に、システムの設計段階で明確に決めておくべきことが多いことに気が付きます。例えば、「認証の不備」「アクセス制御の不備」「不十分なロギングとモニタリング」などは設計時の重要な要素でしょう。実際の話、システムの弱点や脆弱性は、半分は実装の問題、

OWASP Top 10 - 2017

- A1:2017 – インジェクション
- A2:2017 – 認証の不備
- A3:2017 – 機微な情報の露出
- A4:2017 – XML 外部エンティティ参照 (XXE)
- A5:2017 – アクセス制御の不備
- A6:2017 – 不適切なセキュリティ設定
- A7:2017 – クロスサイトスクリプティング (XSS)
- A8:2017 – 安全でないデシリアライゼーション
- A9:2017 – 既知の脆弱性のあるコンポーネントの使用
- A10:2017 – 不十分なロギングとモニタリング

半分は設計レベルの欠陥に起因しています。設計段階において、セキュリティについては対策をするかしないか程度の、あいまいなものであってはいけないということです。どんな具体的な検討材料があるでしょうか。

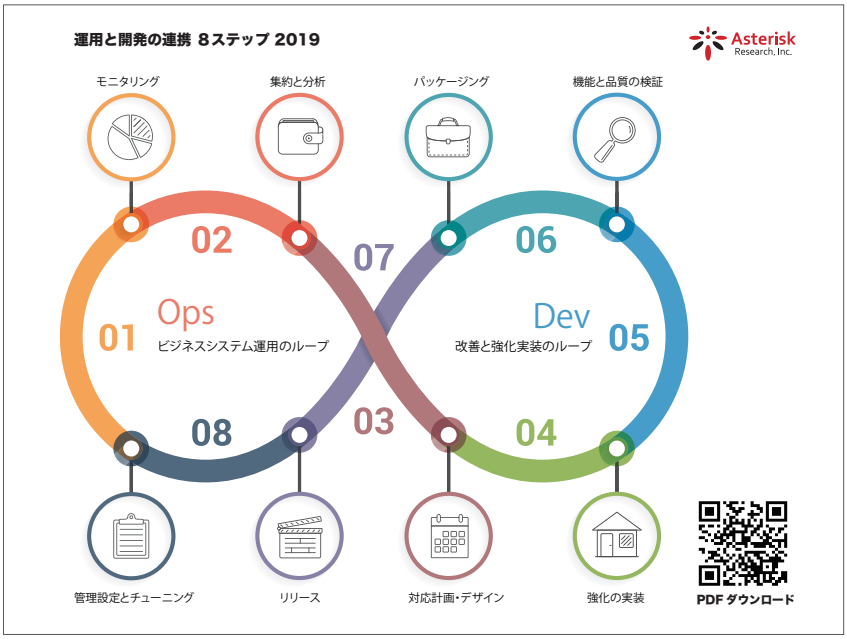
ここで、高速なシステム開発と運用サイクルを、F1レースだと見立てますと、F1マシンがサーキットでスロットル全開で走っている状況は、「運用」、すなわちDevOpsのうちOpsサイドです。ピットはDevでしょう。サーキットは人々の注目を浴び、ひいては事業利益を生みます。経営陣が最も重要視する部分ですから、マシンがクラッシュしないようにするため、マシンの状況を正確にモニターし、問題をアラートできる機能が不可欠であることは言うまでもありません。それにピットが対応することになります。

同様に、セキュリティの検討をクリアにする打開策のひとつは、攻撃や障害に対応する対策の有無だけでなく、ケイパビリティ（性能）だ、ということです。つまり、現状のサイバー攻撃情勢、さらにシステム特有のリスクを狙う攻撃に対し、防御力、検知スピード、さらに被害拡大しないための対応機能や復旧能力に注目します。

例えば、システムにリスト型攻撃があった場合、それを「何秒」で検出し、どの範囲までダメージを止めるか。何分で復旧できるか、要件を数値化していきます。また、日々のモジュールのアップデートハンドリングも考慮に入れるべきでしょう。こうした検討により、ネットでユーザーに指摘されるまで、セキュリティ被害に何週間も気付かない、という状況を避ける設計要件が見えてきます。顧客の指摘や、別の場所から情報漏えいが発覚して初めて気が付くのは現場にとっても大変高くつきます。

この検討を実践するにあたり、本番環境にセキュリティテストをかけることは役立ちます。現状の運用のケイパビリティを調べるための侵入テストは、レッドチーミングといわれることもあります。これにより、今後のシステムのセキュリティ対応力の不足、さらなる要件としてどのようなことを盛り込む必要があるのか、明らかになることでしょう。

さて、今回はシステム設計に盛り込む要素であるセキュリティ・ケイパビリティというコンセプトをご紹介しました。どんなシステムであっても、運用と開発の関係、それを支えるセキュリティの仕事は円滑で柔軟なものとなって初めて機能します。高速なDevOpsは、全速力のサーキットとピットの強い信頼関係を基盤としているべきで、セキュリティはその両方を支えるエンジニアリングに位置付けられるということです。OWASP コミュニティには、設計、テスト、実装、運用に貢献し、また相互の連携の助けにもなるドキュメントやツールがたくさんあります。ぜひ、ご覧いただき、あきらめずに実践に取り組んでみてください。シフトレフト！



第5回 DevSecOpsのSecは

ここ数年、移り変わりの早い世の中に機敏に対応できる力「ビジネスアジリティ（企業の機敏さ）」の重要性がますます高まっています。ビジネスアジリティ向上に寄与するITシステム開発の概念としてDevOpsやDevSecOpsというワードもよく聞くようになってきました。

DevOpsについては特集「システム開発とセキュリティのこれから」で深掘りしていますので、ここではセキュリティチーム（Sec）がどのようにシステム開発チーム（Dev）と運用チーム（Ops）に貢献できるのかについて、私が所属するサイバー・グリッド研究所での開発経験を踏まえつつ考察したいと思います。なお、開発規模が小さいため、執筆者はDev、Sec、Opsのすべてに属しています。そのほか、DevとSecに兼任で1名、Opsに専任で1名の計3名のチームです。

開発（Dev）

サイバー・グリッド研究所では、CTF（Capture The Flag）環境を提供するWebアプリケーション（以降、スコアサーバーと呼ぶ）を内製しています。CTFは情報セキュリティの技術や知識を競う競技です。ラックでは、毎年ラックグループ全体を対象としたCTF「LACCON（ラッコン）」や社外向けの教育コンテンツに内製のスコアサーバーを利用しています。

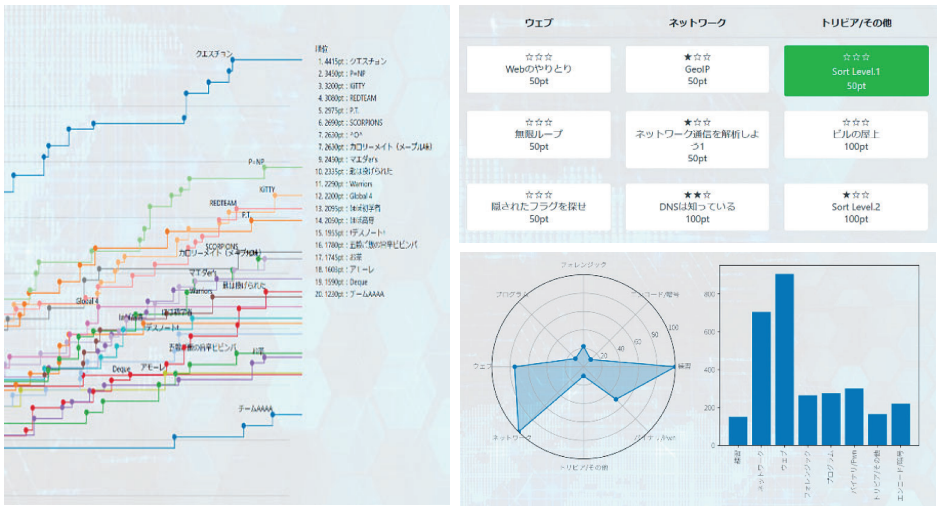


図1：スコアサーバーの画面キャプチャ

スコアサーバーは、Python＋Django（Webアプリケーションフレームワーク）で開発し、下表のような機能を実装しています。

ユーザー種別	機能名	機能概要
一般	ユーザー作成	競技参加者のユーザー作成
	ログイン	ユーザーログインの成否判定
	フラグ投稿	回答（フラグ）投稿の正否判定
	成績表示	自身の成績の詳細表示
	順位表示	全体の順位を折れ線グラフで表示
	運営相談	不明点を運営に問い合わせる
	Writeup投稿	問題の解説記事の投稿機能
管理者	ユーザー管理	アカウントの編集、個別の成績抽出、チーム別の成績抽出
	データ管理	CTF問題の編集、開催データのインポート、エクスポート
	開催管理	設定した時間により、競技前・競技中・競技終了のステータスを管理 ステータスによって挙動を変更

表1：機能一覧（一部抜粋）

開発ルールは「セキュアコーディングの手法にのっとっていること」「Python標準のコーディング規約をもとにコーディングすること」「Webアプリケーションフレームワークを活用し独自実装しないこと」とし、ソースコード管理はGitLab、UIテストはSelenium、セキュリティテストはOWASP ZAPを使うようにしました。

必要か？

谷口 隼祐
サイバー・グリッド・ジャパン
サイバー・グリッド研究所
チーフリサーチャー

セキュリティ（Sec）

今回は、小規模開発であるということに加えて開発チームのメンバーがセキュリティエンジニアであるということもあり、開発チームとセキュリティチームのメンバーは兼任としました。手が空いているメンバーが手動でWebアプリケーション診断やソースコードチェック、仕様のチェックを行いました。

その他、ライブラリやフレームワークのメジャーバージョンがあがった際には、セキュリティ面から更新の要否を判断しました。

運用（Ops）

スコアサーバーは、下図のように複数のコンポーネントから構成されています。一度構築した後に再構築が手間になってしまい、バージョンアップが疎かになったため、スコアサーバーに必要な各種コンポーネントをコンテナ化するようにしました。コンテナ化することによって、スクリプト一発で簡単に検証環境や本番環境でスコアサーバーを動作させられるようになりました。死活監視はZabbixを使い、大幅なバージョンアップをした際には自社のWeb診断を受けています。

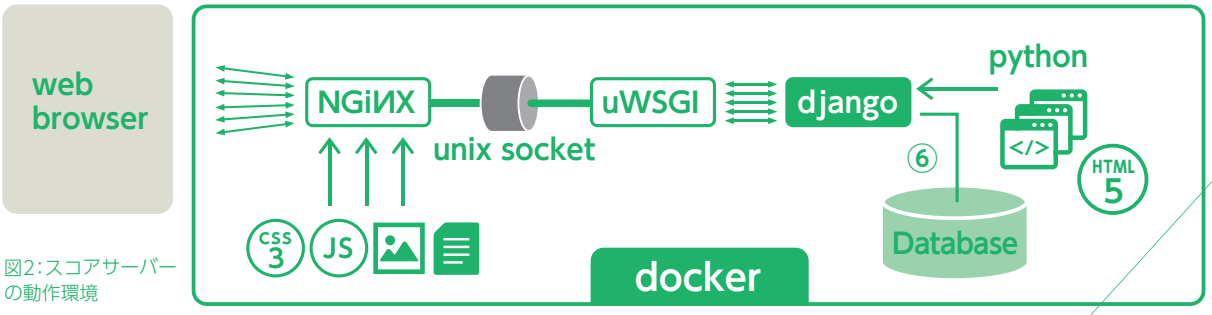


図2：スコアサーバーの動作環境

セキュリティチームは必要か？

前述した開発ルールで進めていくと、開発当初はセキュリティ上の問題点はあまり検出されませんでした。XSSやSQLインジェクション、CSRFなどの脆弱性はWebアプリケーションフレームワークを活用することで防げますし、統合開発環境（IDE）を活用することでコーディング規約の遵守やWebアプリケーションフレームワークを適切に利用することができました。

また、スコアサーバーの環境をコンテナ化してから、利用している各種ライブラリやフレームワークのバージョンを細かくアップデートすることが楽になりました。環境の作り直しが容易にできるようになったため、バージョンアップによるエラーが出た場合でも一つ前のバージョンにすぐに戻せるようになりました。

セキュリティを意識した開発・運用であれば「セキュリティチームは不要では？」と思い始めましたが、その矢先に深刻度の高い脆弱性が見つかりました。認証・認可やアクセス制御に関わる脆弱性です。これらはツールで見つけにくい脆弱性で、仕様に照らし合わせないとバグなのかセキュリティ上の問題なのかが判断しにくいものです。見つけたタイミングは、スコアサーバーの管理機能の追加実装部分（表1：開催管理機能）の動作確認中です。

ラックで内製しているスコアサーバーには、競技前・競技中・競技終了によってアクセス可能な範囲を変える機能があり、競技前にはCTF問題は表示されない、Writeupの表示は競技終了後のみといった制御が求められます。また、利用者の権限によってもアクセス可能な範囲が変わります。この部分には、認証・認可に関する脆弱性が作り込まれやすく、実際に利用者が閲覧できてはいけないページが閲覧できてしまう脆弱性が見つかりました。仕様が複雑になればなるほど、セキュリティの観点から仕様や仕様に基づいた実装に不備がないか、片手間ではなく専門的にチェックするチームが必要だと感じました。

まとめ

先端を行くDevSecOps環境ではありませんでしたが、その中でもセキュリティチームの必要性を再確認することができました。セキュリティチーム（Sec）は、開発チーム（Dev）と運用チーム（Ops）のパフォーマンスを維持する潤滑油のような存在であるべきだと考えます。仕様の段階でセキュリティ上の不備を指摘したり、脆弱性を作り込みにくくする開発サイクルを支援したり、開発現場そのもののセキュリティを確保する活動を通じて、「ビジネスアジリティ（企業の機敏さ）」向上に寄与するセキュリティチームづくりを目指していきたいと思います。

ラックの

顔

第8回

システム開発現場の最前線における セキュリティ対策の現在と未来

ラックの主軸の事業である「セキュリティソリューションサービス」。セキュリティ、と一言でいうとシステムそのものに対するセキュリティをいかに担保するかという部分に焦点を向けがちだが、ソフトウエアサプライチェーンの拡大によって、開発現場にもその脅威が迫っている。環境や技術など、進化や変化が激しいシステム開発現場におけるセキュリティ対策の課題と解決策について、ラックの先鋒ともいえる二人の社員が、それぞれの立場から意見を述べた。

開発者からアーキテクトへ。挑戦する意欲がキャリアの幅を広げた

大学を卒業後、ラックとは別の会社でSEとしてキャリアをスタートさせた藤本。汎用機でのシステム開発を皮切りに、Webやクライアントサーバー(C/S)などさまざまな環境における開発経験、さらにはシステムの全体構想を設計するアーキテクトといわれるポジションやプロジェクトマネジャー(PM)など、数多くの顔を持つ。その多様な経歴は、ラックの中でもかなり特殊だといえるだろう。

「若い頃は基本的に“仕事を断らない”こと、そして“何にでも首を突っ込む”ことをモットーにしていました。そして、やりたいことがあれば“やらせてくれ”とはっきり言う。随分生意気だったなと思います(笑)(藤本)」

知識欲旺盛な藤本が、自らのキャリアの転機だったと振り返るのは、2010年にJSOC(Japan Security Operation Center／セキュリティ監視・運用サービスの拠点)のシステム開発にアーキテクトとして参画したことだ。

「当時は、ラックのグループ会社であるエー・アンド・アイ システムに在籍していました。当然のことながら“開発セキュリティ”に対する知識は持ち合わせていましたが、“JSOCが何を守っているのか”という本質的なことについては、そのプロジェクトに関わったことできちんと理解できたような気がします。開発者、アーキテクト、PMとしての経験に、セキュリティに関する実務の実績が加わったその頃から、業務に関連するさまざまな分野の中で、苦手と感じるものがなくなってきたように思います(藤本)」

一方、「他のSIerと異なり、ラックがセキュリティ事業を行っていることに興味を持った」という大沼は、そのキャリアをラック一本で形成してきた。入社後は、

大沼重成

shigenari ohnuma

顧客先に常駐して旅行会社のサイト保守や開発を行うSEとして従事してきたが、入社5年目に、環境が変化したという。「入社当時は、主流の技術であったMicrosoft社のClassic ASPがASP.NETという技術へと切り替わり始めた頃でした。ASP.NETに詳しい人間が社内にあまりいない中、チャレンジしてみたら結構できてしまって(笑)。社内で“ASP.NETに強い人間”として認知され始めてきた2008年頃に大型のプロジェクトでアプリケーションのアーキテクトを任されたことは大きな経験でした(大沼)」

同プロジェクトには優秀なメンバーが集結していたこともあり、大沼は「心身共に大きく成長できた時期だった」と当時を顧みる。この経験が、開発者にとどまることなく技術リーダーやアーキテクトなどさまざまな立場を歴任する契機となったことは明白だろう。そしてその後、クラウドの台頭によってさらなるキャリアの拡大があったと大沼は言う。「オンプレミス(自社運用)のデータセンターをMicrosoft社のAzureというクラウドサービスへ移行するプロジェクトに参画した際、Azureの実績を積んだことが、それまで主としていたアプリケーション開発から、インフラの開発へと少しずつ軸足を移すきっかけになりました。Azureをベースにインフラ構築をしながらアプリケーション開発を支援していたところに社内でアジャイル開発センターが発足し、今に至っています(大沼)」

ビジネス環境の変化に柔軟に対応し、「価値」を見出すアジャイル開発

藤本が所属するエンタープライズ事業部の配下に、2018年7月に発足したアジャイル開発センター。アジャイル開発手法を活用したSIサービスにラックが注力していることは、同センターが発足から3カ月後に事業部階層へと格上げされたことでも明白だ。

アジャイル開発センターのミッションについて、大沼は次のように語る。

「お客さまのデジタルトランスフォーメーションの推進――業務のIT化によって生産性の向上を図ることを目的としています。アジャイル開発に対する顧客ニーズが高まっている今、ラック社内のノウハウを集結させ、展開していこうという思いから発足した組織です(大沼)」

「アジャイル開発」という言葉を見聞きするようになって久しいが、その定義については、開発フェーズを上流から下流へと段階的に進めていく従来のウォーターフォールモデルと異なり、短いサイクルで開発を進めながらトライ&エラーを繰り返す、という言い方が一般的だ。しかし大沼は、「ウォーターフォールとアジャイルの違いは、単にプロセスにとどまらない」と話す。「ビジネスのスピードが日々加速している今、数年をかけてシステムを作っても、設計時とリリース時の市場は大きく変わっている可能性が高い。お客さまのフィードバックやビジネスのコンテクストを常に反映させながら、ユーザーが真に必要としているもの、ビジネスとして価値のあるものを随時提供し続けていく。“ビジネスの変化にきちんと対応していく”ことがアジャイルの本質だと考えています(大沼)」

藤本も、次のように補足する。「ビジネス環境の変化が著しい今、企業にとって、サービスを提供するメリットは未知数です。ですから、最初から完全な形でサービスをローンチするよりも、まずは小さくスタートして、サービスニーズを捉えながらサービスを育てていこうと考える企業が多い。アジャイル開発では、品質を安定させる仕組みが外側にあり、その中で開発を繰り返します。一定の安全性を担保したうえで、まずはシステムをサービスに適用する、ということが可能であり、かつ、お客さまにとっても我々にとっても、大沼が言う“価値”がどこにあるのかを追究し、攻めていくことができる手法だと思います(藤本)」

藤本博史

hiroshi fujimoto

せキュリティの専門家が不足する開発現場は、個々の意識改革で変わる

現在はセキュリティコンサルタント業務に従事する藤本。プログラミングのソースコードを解析して脆弱性などの問題を洗い出したり、顧客が作成した設計書が必要なセキュリティ対策を満たしているかなどを評価したりと、セキュリティのスペシャリストとして活動している。数多くの開発現場でセキュリティを支援する傍ら、藤本は「システム全体のセキュリティ対策や方式を考えることができるアーキテクトが不足している」ことを危惧していると話す。「“セキュリティ”というと、実際にコーディングする側のタスクだというイメージを持たれがちなのですが、そもそも上流の工程、つまり設計の段階で考慮されていなかったことは下流工程に落ちてきたときに致命的に残るわけです。設計の段階できちんとレビューをしておけば、バグに対するリスクやコストの発生を未然に防ぐシフトレフトが可能になる。それをするためにはかなり高いレベルの知識を持つスーパーマンのような人材が必要になってくるのですが、そのような人材を確保するのは難しいと感じています（藤本）」

アジャイル開発の現場に身を置く大沼も、同様のことを感じているという。「アジャイル開発の場合、インフラ、アプリケーションそしてもちろんセキュリティも含め、さまざまな要素を一つのチームで完結していきます。設計からコーディング、テストなどの実作業はもちろん、新たな技術も習得しなければなりませんし、さらにはヒアリングや見積もりなどもこなしています。セキュリティは新たな攻撃手法や脅威、リスクが日々見つかる分野なので、多くの責任を抱える開発チームでそれを追い続けるというのは正直に言って難しい。私としては、チーム内で基本的なセキュリティを担保しつつも、セキュリティの専門家が第三者的な視点でレビューをするといったような仕組みが必要だと考えています（大沼）」

このような課題をどのように解決していくべきか。藤本は、「現場の意識の拡大」がそのカギになると示唆する。「残念ながら、開発の現場はそれぞれが自分の役割だけを追いかける縦割り型のコミュニティになっていることが多いのが現状なのですが、その文化を変えることが一つの手段になると思います。まずは設計をした人間が開発者にきちんとロジックを説明し、開発チームの中の溝を埋めていくこと。インフラ、アプリケーション、ネットワークと、外側にはまだまだ多くの溝があるわけですが、それぞれが少しずつ輪を広げ、重なる部分を増やしていけば、スーパーマンなんていらないんじゃないか、そう思います（藤本）」

「情報を守る」ことは変わらぬミッション。ただしその手法や視点は進化を続ける

技術面の進展が著しいシステム開発の世界。すべての機能を一つのプロセスで稼働させる、一枚岩的な「モノリシック」から、それぞれの機能が独立したプロセスとして稼働する「マイクロサービス」へと、システムのアーキテクチャはここ数年で大きく変化しつつある。それに伴い、セキュリティ対策における考え方に変化はあるのか。尋ねると、両名は口をそろえて「原理原則は変わらない」と答えた。「つまるところ、守るべき大切な情報があるからセキュリティ対策を施すということに尽きるので、アーキテクチャが変わっても、本質的な部分は変わりません。ただし、脅威というものの姿は日々変わってくるので、手段をどう選ぶかということは非常に重要だと思います（藤本）」

セキュリティ対策においては、アプリケーションに加えてファイアウォール（不正アクセスをブロックするシステム）やIPS（Intrusion Prevention System／侵入防止システム）などのセキュリティ装置を装備する、「多段」の形をとることが一般的だ。それでも事故が起こるリスクはゼロにはならない。どこまでを想定すべきかという問いに対して藤本は「率直に言ってしまえば、流出までは想定すべきだ」と断言する。「事故は起きない、という前提での対策では甘い。事故が起きることを前提に準備をする必要があると考えています（藤本）」

加えて大沼は、アーキテクチャがマイクロ

大沼重成
shigenari ohnuma
アジャイル開発センター

2003年ラック入社。旅行会社およびアパレル会社等のWebサイト開発・保守業務を通じて、アプリケーション、クラウド、インフラのアーキテクトとしての経験を積む。2018年10月より当時在籍していたエンタープライズ事業部と兼任でアジャイル開発センターに所属、2019年4月より専任。趣味はジャズでサックスとフルートを演奏する。現在は２児の子育てに追われ演奏活動休止中。

サービスへと変わったことで気づきがあったと話す。「モノリシックの場合、一つのシステムですべての処理が行われるのでログは集約されています。ただ、マイクロサービスではシステムごとにログが分散するので、それぞれのログがどう関連付いているのかを追うことができるユニークIDのようなものが必要になってくるかと思います。複数に分割されたシステム間では、API（Application Programming Interface）を介して認証の引き継ぎや同期を行っていかなければならないわけですが、システムが細分化したことでセキュリティを考慮すべきエンドポイントは大幅に増えました。複数のシステムが複雑に連携し合うAPIの脆弱性は見つけにくく、脆弱性が顕在化したときのビジネスインパクトも大きい。APIに対するセキュリティへの意識が高まってきたことも、最近の傾向だと思います（大沼）」

新しいことを追いかけるチャレンジングな精神がラックの将来を担う

システム開発のSEから、セキュリティコンサルタントやアジャイル開発など、多様なキャリアを築いてきた藤本と大沼。そのキャリアパスは、ラックの社員にとってのモデルケースになる、と経営層も太鼓判を押す。社内でも目立つ存在であることは間違いないが、大沼は自分の性格を「内気でシャイ」だと表現する。「もともと人前で話すのは苦手な性格ではあるんです。ただ、入社して12年たった2015年に、思い切って社内技術者のコミュニティを立ち上げたことで、晴れやかな気持ちになったんです。それまではこの仕事が自分に合っているのか、漠然としながら孤独に開発者を続けていましたが、思いを共有できるつながりができたことで、自分がやりたいことはやっていいんだと気づけた。そしてラックがそれを許してくれる会社だというのはありがたいと思っています。今後もまず行動を起こすことを基本に、アジャイル開発の推進や、強みであるAzureの技術を生かして会社のためにできることを見つけていきたいと考えています（大沼）」

一方の藤本も、「今後もやりたいことは全部やっていく」と積極的な姿勢を見せる。「気になっていること、関心があることは山ほどあります。もともとが、いろいろなことをやりたい性格なのかもしれません。興味があれば、何かについて学んだり調べたりすることは自分にとって“勉強”という認識にはならない。自分の若い頃がそうだったように、若手のエンジニアにも“興味があることにはとりあえず首を突っ込め”と言いたいですね。何かにトライしたときに、それを成功と捉えるか失敗と捉えるかは自分次第。たとえ10年かかってでもやり続けて、最後に結果が出れば成功だし、逆に途中でやめてしまったら失敗だと自分では思っています（藤本）」

仕事ではほとんど接点がないという二人。しかし、開発現場における課題意識や仕事に対する向き合い方には、多くの共通点が垣間見えた。

藤本博史
hiroshi fujimoto

エンタープライズ事業部 インフラソリューションサービス部
セキュリティサービスグループ 担当部長

1998年、ラックの前身であるエー・アンド・アイ システム入社。2010年にJSOCのシステム開発に携わり、2012年よりラック統合により転籍、システムサービス本部産業統括部にてアーキテクトとして従事。2015年より開発者向けの教育コンテンツの開発やセキュリティ教育も担当。2017年より現職。週末は、妻とハンドメイドフェスや神社に風鈴を見に出かけたり、外出を楽しんでいる。

ラックは、1986年にコンピューターのシステム開発サービスの提供を目的として誕生し、以来30年以上にわたってお客さまとともにさまざまなシステムを開発してまいりました。

そして、情報ネットワークの普及拡大に伴って現実社会の課題となり始めたセキュリティ対策事業をスタートしてから、20年以上が経過しました。

過去には、情報システムとセキュリティが開発コストの観点から反駁し合う関係にあった時代もありました。しかし現在は、IoTデバイスの普及に伴い、インターネットがそれこそ「どこにでも」存在する時代となっています。急速にデジタル化する時代を迎えた昨今、開発とセキュリティは両輪で進めていくべきものという考え方が、徐々に浸透しつつあります。

サイバー・グリッド・ジャパンは、高度に巧妙化するサイバー攻撃とそれによる被害発生を防ぐための研究開発部門として、2014年に発足しました。
また2018年には、デジタルトランスフォーメーション時代のビジネスに根差したシステム開発コンセプト「アジャイル開発」の普及を推進する組織として、アジャイル開発センターを設立しています。

Society5.0の実現に向けたデジタル化が進み、高度に複雑化した標的型サイバー攻撃や、今号の特集でもある「見えない脅威」サプライチェーン攻撃が現実的に対処すべき経営課題として降りかかる中、情報システムとセキュリティは切っても切り離せない関係となっています。

ラックは、情報システムの「作り手」としての目線を持ったセキュリティ企業として、これからも日本の情報社会の安全と発展に寄与すべく、邁進いたします。

CYBER GRID JOURNAL VOL.8

サイバー・グリッド・ジャパンは株式会社ラックの研究開発部門です。
サイバー攻撃や各国のセキュリティ事情、セキュリティ防御技術などに関する最先端の研究のほか、
複数のセキュリティ企業との連携や新たな製品・サービスの開発、各種啓発活動などにより
日本のセキュリティレベルと情報モラルの向上に貢献しています。

株式会社ラック サイバー・グリッド・ジャパン
〒102-0093 東京都千代田区平河町 2-16-1 平河町森タワー
E-MAIL：sales@lac.co.jp　https://www.lac.co.jp/

株式会社ラック
サイバー・グリッド・ジャパン