Vol.2020-DPS-182 No.12
Vol.2020-CSEC-88 No.12
2020/3/12

運用を考慮したセキュリティ要求分析手法の提案

小西慶浩[1]　大久保隆夫[2]

概要：DevSecOps の実現には，セキュリティチェックを前倒しする「シフトレフト」の考えが重要であり，開発の計画／設計段階で実施する「セキュリティ要求分析」が重要な役割を果たす．そこで，従来のミスユースケースによるセキュリティ要求分析を拡張し，運用を考慮したセキュリティ要求分析手法を提案した．最後に，具体的な案件事例に提案手法を適用し，ケーススタディを行った．その結果，セキュリティ機能を実装する際に運用から開発へフィードバックすべき事項を具体化することができるようになった．また，セキュリティ脅威，関係者，セキュリティ対策の関係を，運用まで含めた形でより詳細に示すことができるようになった．

キーワード：DevOps , DevSecOps , セキュリティ要求分析

1. はじめに

システム開発では，サービスを顧客に対してつつがなく提供できるよう開発したシステムの「運用」も欠かせない．開発の現場でも，「開発」と「運用」の一体化を前提としたプロジェクト体制をとり，開発者と運用者が連携して協力する開発手法「DevOps」が近年普及している．

「DevOps」の開発ライフサイクルでは，セキュリティへの考慮が欠けていることが指摘されている．開発システムのセキュリティ品質を高めるには，新たなプロセスの採用，新たなスキルの習得が必要となり，これが迅速化の妨げになると考えられていることが理由の１つとして挙げられる．

しかしながら，セキュリティ品質の向上はビジネスを進める上で必須である．そこで，「DevOps」の開発サイクルの中にセキュリティを組み込もうとする「DevSecOps」が近年注目されている．実際に，「DevOps」での利用を想定した脆弱性診断や運用監視に関する製品/ツール類が多く登場しており，セキュリティの組み込みが進んでいることが実感できる．

一方，計画段階での組み込みに対しては，製品/ツール類は少なく，組み込みが進んでいない状況がみられる．計画段階でセキュリティを組み込むとは，具体的には，セキュリティ要求分析を実施し，脅威の特定，リスクの評価，対策の策定を決定することである．「DevSecOps」では，開発者と運用者が一体となってセキュリティ要求分析を実施することが求められる．しかしながら，従来のセキュリティ要求分析はシステム化対象範囲のみにフォーカスしており，運用が考慮されていないケースが見受けられる．

本研究では，運用を考慮したセキュリティ要求分析の必要性を解説するとともに，従来のミスユースケースによるセキュリティ要求分析手法を，運用まで考慮したものに拡張した新たなセキュリティ要求分析手法を提案する．本研究が，「DevOps」で開発されたシステムのセキュリティ品質向上に貢献できることを期待する．

2. DevOps／DevSecOps

(1) DevOps

DevOps とは，開発者と運用者が連携して協力する開発手法の総称である．アジャイル開発のようなスピード開発手法を進めていく中で，「開発」と「リリース」の間に課題が浮き彫りになった．そこで，開発者と運用者がお互いに協調し合うことで，開発・運用するシステムのビジネス価値を高めるとともに，確実かつ迅速にエンドユーザに届けようとするプラクティスが普及している．

DevOps には，セキュリティへの考慮が欠けていることが指摘されている[1]．これはセキュリティ品質を高めるには，新たなプロセスの採用，新たなスキルの習得が必要となり，これが迅速化の妨げになると考えられていたためである．

しかしながら，セキュリティはビジネスを進める上で必須の要素である．セキュリティを十分に考慮していないシステムでは，ビジネスを行うことはできないからだ．そこで，近年，DevOps の開発サイクルの中にセキュリティを組み込もうとする「DevSecOps」が注目されている[2]．

1 情報セキュリティ大学院大学 博士前期課程
Graduate School of Information Security Institute of Information Security

2 情報セキュリティ大学院大学 教授
Graduate School of Information Security Institute of Information Security.

Vol.2020-DPS-182 No.12
Vol.2020-CSEC-88 No.12
2020/3/12

(2) **DevSecOps**

DevSecOps とは，DevOps の中にセキュリティを組み込もうとする開発手法の総称である．DevSecOps は，DevOps のスピード感を活かしたまま，製品/ツールなどを利用して各工程にセキュリティを組み込むことが重要である．

表 1 に開発の各プロセスで必要となるセキュリティ技術要素を，表 2 に運用の各プロセスで必要なセキュリティ技術要素を示す．DevSecOps を実現するためには，表中の技術要素を上手く組み込み，開発のスピードとセキュリティの両立を実現することが大切である．

表 1　開発プロセスのセキュリティ技術要素

プロセス	技術要素
計画/設計 （Plan）	全体計画作成，メトリクス， ツール教育等， 脅威分析，アーキテクチャ設計
コーディング （Code）	IDE プラグイン，コードレビュー
ビルド/試験 (Build/Test)	静的検査，動的検査， コンポーネント解析，ファジング

表 2　運用プロセスのセキュリティ技術要素

プロセス	技術要素
リリース/デプロイ (Release/Deploy)	署名検証，脆弱性検出， セキュア設定，堅牢化
運用/監視 (Operate/Monitor)	SOC，WAF , RASP, 脅威情報,インシデント対応手順

DevSecOps の実現には，「シフトレフト」の考え方が重要と言われている．「シフトレフト」とは，セキュリティチェックを前倒しすることで，開発の後戻りを防ぎ修正コストを抑える考え方である．中でも「セキュリティ要求分析」は，高速開発の現場でも有益なアクティビティと評価されており[3]，「シフトレフト」を実現する上で重要な役割を果たすと考えられる．

3. セキュリティ要求分析

3.1 セキュリティ要求分析とは

セキュリティ要求分析(Security requirements analysis)とは，「ソフトウェアへの意図的または意図しない不正アクセスの防止を保証する一連の仕様を満たすために，ソフトウェアが持つべき機能を識別するアクティビティ」である[4]．これを噛み砕いた表現を用いると，「セキュリティ要求仕様を策定する作業」とも言える．セキュリティ要求分析の段階では，設計仕様は未定という前提があり，セキュリティ要求分析の成果物により，セキュリティを含む設計を始めることができる．

「セキュリティ要求仕様」とは，脅威（攻撃）に対する対策の仕様に等しい．よって，セキュリティ要求仕様の策定の前段階として，想定される脅威を十分に洗い出す必要がある．そして，いかに脅威を洗い出し，そのリスクを評価することも重要である．

3.2 主なセキュリティ要求分析手法

(1) **STRIDE モデルによる脅威モデリング**

STRIDE モデルとは，「情報漏えい」，「なりすまし」などの特徴的な脅威を具体的に分類するための手法である．「STRIDE」は，「Spoofing : なりすまし」，「Tampering : 改ざん」，「Repudiation : 否認」，「Information Disclosure : 情報漏えい」，「Denial of Service : サービス拒否」，「Elevation of Privilege : 特権の昇格」の６つの脅威の頭文字を取っている

そして，脅威モデルの作成と分析を行うためのツールとして，「Threat Modeling Tool」が Microsoft 社より提供されている[5]．本ツールは，セキュリティの専門家でないユーザを想定して設計されており，すべての開発者が簡単に脅威モデリングを行うことができるようになっている．

(2) **ミスユースケースによるセキュリティ要求分析**

ミスユースケースとは，Sindre と Opdahl によって提案された，ユースケース図とユースケース記述を用いて，セキュリティ脅威とその関係者，および，セキュリティ対策の関係を明確にする要求分析手法である[6]．これらの関係を明確にすることで，プロジェクト内におけるセキュリティ目標の共有やセキュリティ知識の向上にも貢献できることが期待できる．また，UML ユースケースの記述を拡張したものであるので，「Threat Modeling Tool」同様，セキュリティの専門家でないユーザでもセキュリティ要求分析を行うことができる．

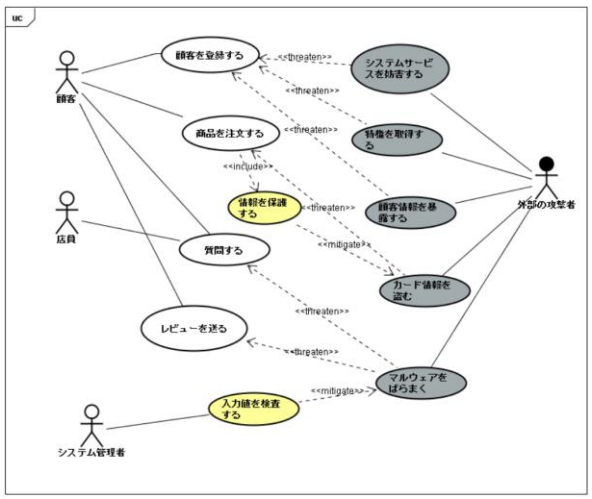


図 1　ミスユースケース図

3.3 セキュリティ要求分析の課題

DX 時代の到来，DevOps の普及と技術の発展に伴い，様々な作業の自動化が進んでいる．セキュリティにおいても，システム監視や脆弱性検査において多くの作業が自動化され

Vol.2020-DPS-182 No.12
Vol.2020-CSEC-88 No.12
2020/3/12

ている．今後も，人工知能(AI)技術の発展などにより，応用的な作業を自動で行うことが期待できる．しかしながら，「セキュリティ要求分析」の作業は手作業が残ると考えられる．これらの作業は，自動化を適用しようにもデータの様式化や収集が困難な作業である[7].

DevSecOps を実現するには，セキュリティ仕様を満たしたシステムのあるべき姿を，「企画ユーザ」「開発者」「運用者」の三者が一体となって創案することが必要であると考える．しかしながら，セキュリティ要求分析手法の多くは，システム化対象範囲のみにフォーカスを当てており，ミスユースケース図に記載されている対策のユースケース（セキュリティユースケース）にも，運用の観点は入っていないことが多い．私は，従来のセキュリティ要求分析を，「DevOps」に適した形に改善することが必要であると考える．

3.4 運用を考慮したセキュリティ要求分析の必要性

山田らは「Microsoft Threat Modeling tool の軽減策」のカテゴリを拡張し，セキュリティ対策のタイプを一覧にまとめた[8]．山田らのまとめによると，セキュリティ対策のカテゴリは表 3 の 10 つに分類できる．これらの中には，運用で実施する対策も存在する．セキュリティ要求分析を実施する際は，運用のナレッジを計画段階から取り入れることが望ましいことが予想される．

表 3　セキュリティ対策のカテゴリ

No.	対策のカテゴリ
1	Authentication：認証
2	Session Management：セッション管理
3	**Input Validation：入力検証**
4	**Auditing and Logging：監査とログ記録**
5	**Communication Security：通信のセキュリティ**
6	Cryptography：暗号化
7	Sensitive Data：機密性の高いデータ
8	**Configuration Management：構成管理**
9	Authorization：認可
10	**Configuration Management：構成管理**

※**太字**:運用で実施することが予想されるセキュリティ対策

具体的には，「Input Validation：入力検証」「Auditing and Logging：監査とログ記録」「Communication Security：通信のセキュリティ」「Exception Management：例外管理」「Configuration Management：構成管理」は運用のナレッジが存在すると考えられる．特に，「Auditing and Logging：監査とログ記録」「Communication Security：通信のセキュリティ」は，運用のナレッジがより多く存在することが考えられる．これらのセキュリティ対策を計画する際は，運用を考慮することが求められ，運用者の価値発揮も期待できることが考えられる．

4. 提案手法

4.1 提案手法の概要

DevOps の普及により，開発と運用の垣根をなくすことが重要視されている．また，運用で得られたノウハウやナレッジを開発へフィードバックすることも重要視されている．そこで，従来のミスユースケースによるセキュリティ要求分析を，運用まで考慮したものに拡張する手法を提案したい．

具体的な提案手法を解説する前に，ミスユースケースによるセキュリティ要求分析に着目した理由について説明する．ミスユースケースによるセキュリティ要求分析は，ミスユースケース図を作成することによって，セキュリティ脅威とその関係者，および，セキュリティ対策の関係を明確に示すことが可能となる．これは，プロジェクト内におけるセキュリティ目標の共有やセキュリティ知識の向上につながることが期待できる．つまり，ミスユースケースのメリットは，「企画ユーザと開発者」や「開発者同士」の相互理解への貢献に深く結びついたものであると考えられる．よって，ミスユースケースによるセキュリティ要求分析を，運用まで考慮したものに拡張することで，「開発者と運用者」の相互理解への貢献が期待できるのである．以上が，ミスユースケースによるセキュリティ要求分析に着目した理由である．

4.2 提案手法の詳細

従来のミスユースケースによるセキュリティ要求分析を運用まで考慮したものに拡張する手法を提案する．具体的な内容は以下である．

（1）運用者のアクターの記載

ユースケースの関係者に運用者のアクターを追加し，運用者を明示的に示す．

（2） 運用セキュリティユースケースの記載

運用で実施するセキュリティ対策を運用セキュリティユースケースとして追加する．具体的には，「脆弱性情報の入手」，「SSL/TLS サーバの構築」など運用者によって行われるセキュリティ対策のことを指し，アプリケーションに実装するセキュリティ対策機能とは異なるものである．

（3）運用フィードバックの記載

運用で得られたノウハウやナレッジを開発にフィードバックすべきことを明示的に示す．「運用セキュリティユースケース」 から 「セキュリティユースケース」 に 点線矢印で示す．

Vol.2020-DPS-182 No.12
Vol.2020-CSEC-88 No.12
2020/3/12

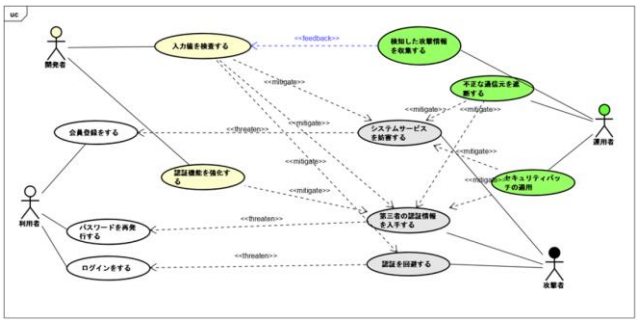


図 2 提案手法を用いたミスユースケース図の一例

図 2 に提案手法を用いたミスユースケース図の一例を示す．Web システムのログイン画面開発を想定した簡易シナリオにてミスユースケース図を作成した．図の詳細について解説する．「システムサービスを妨害する」「第三者の認証情報を入手する」「認証を回避する」の 3 つのミスユースケースを特定し，「入力値を検査する」「認証機能を強化する」の 2 つのセキュリティユースケースを対策として記載した．ここまでが今までのミスユースケースによるセキュリティ要求分析である．

しかしながら，「サービス妨害」や「情報漏えい」に関するセキュリティ対策は，運用で何らかのセキュリティ対策を実施することがある．本ミスユースケース図では，(1) 運用者のアクターを追加し，(2)「検知した攻撃情報を収集する」「不正な通信元を遮断する」「セキュリティパッチの適用」の運用セキュリティユースケースを記載した．また，運用中に得られた攻撃情報を開発側にフィードバックするよう，(3)「検知した攻撃情報を収集する」の運用セキュリティユースケースから「入力値を検査する」のセキュリティユースケースに対して運用フィードバックを記載した．

同一ユースケース上に開発と運用のセキュリティ対策を示すことは，開発と運用の相互理解に繋がることが期待できる．

5. ケーススタディ

具体的な案件を事例に，提案手法を適用し評価する．適用に利用する事例は，近年セキュリティ対策の不備により不正アクセスを許した実際のインシデント事例を参考に作成した[9][10].

5.1 検証事例 1

既存の認証情報を流用した新決済サービスの開発

(1) 背景およびシステム要件

- 企画ユーザは顧客の満足度向上のためオムニチャンネルサービスを提供している小売業である．
- 顧客満足度の向上のため，新決済サービスの開発を企画している．
- オムニチャンネルサービスに会員登録済みのユーザは，新決済サービスの会員登録は不要である．

(2) 適用結果

本事例にて提案手法を適用した結果を以下に示す（図 3,4,5,6). 最初に，利用者のアクターを追加し，システムの要件に従ってユースケースとして「オムニチャンネルサービスの認証情報でログイン」を抽出した．次に，攻撃者のアクターを追加し，ミスユースケースとして「なりすましログイン」，「Dos 攻撃でサービス妨害」を抽出した．続いて，開発者のアクターを追加し，これらの脅威を軽減するセキュリティユースケースとして，「入力検証機構を実装」，「不正アクセスを検知する機能を実装」を抽出した．最後に，提案手法を適用するため運用者のアクターを追加し，運用を考慮したセキュリティ要求分析を実施した．その結果，「入力検証機構を実装」のセキュリティユースケースへフィードバックする運用セキュリティユースケースとして，「オムニチャンネルサービス運用時の攻撃ログ情報を提供」を抽出した．さらに，「Dos 攻撃でサービス妨害」のミスユースケースに対するリスクを軽減する運用セキュリティユースケースとして「不正な IP アドレスを遮断」を抽出した．

適用の結果，セキュリティ機能を実装する際に運用から開発へフィードバックすべき事項を具体化することができた．また，運用側で実施するセキュリティ対策が抽出されたことで，セキュリティ脅威，関係者，セキュリティ対策の関係を，運用まで含めた形でより詳細に示すことができるようになった．

本事例の適用結果は，入力値を伴う認証機能全般に一般化可能と考える．認証機能が持つべき仕様を決める際は，運用者のアクターを追加し，運用まで考慮して検討することが望ましい．

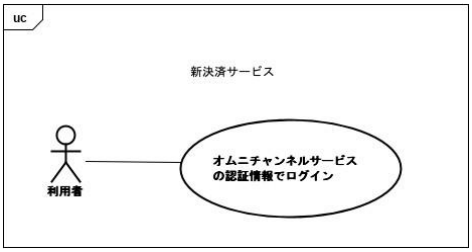
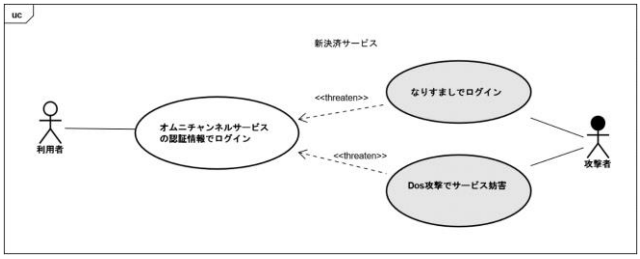


図 3 ユースケース図【検証事例 1】

Vol.2020-DPS-182 No.12
Vol.2020-CSEC-88 No.12
2020/3/12



【凡例】

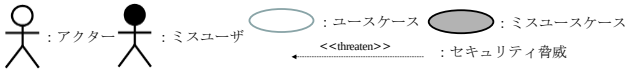
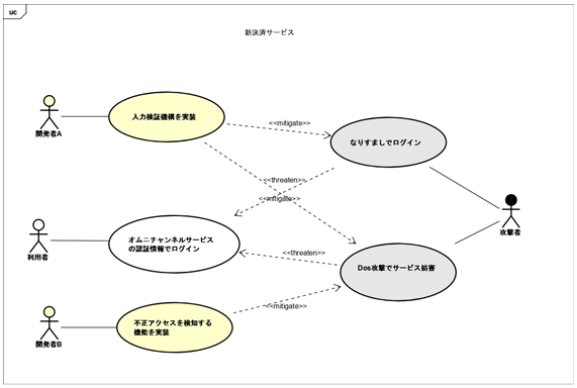


図 4 ミスユースケース図【検証事例 1】



【凡例】

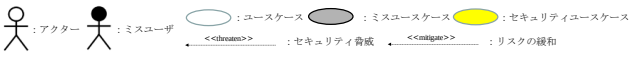
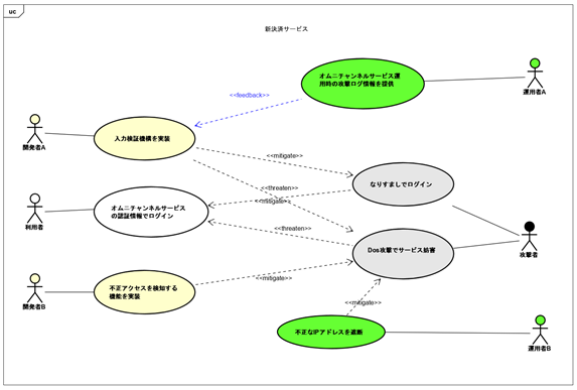


図 5 ミスユースケース図
（セキュリティユースケース有り）【検証事例 1】



【凡例】

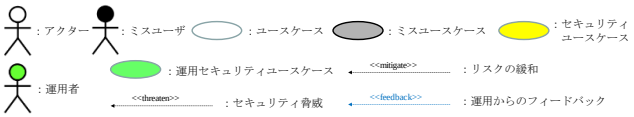


図 6 提案手法によるミスユースケース図【検証事例 1】

5.2 検証事例 2

クラウドメールサービスを利用した新メールシステムの開発

（1）背景およびシステム要件

- 企画ユーザは国立の研究開発法人である．
- 社内情報システムの利便性向上のため，新たなメールシステムの開発を企画している．
- 新メールシステムでは，クラウドメールサービスの利用を企画している．
- その他の社内情報システムは内部のイントラ基盤に構成されている（クラウドサービスは利用無）．
- への認証は，別途外部に構築した認証サーバ経由で行われる．
- 社内情報システムには，新メールシステムと同一のID・パスワードで利用できることとする．

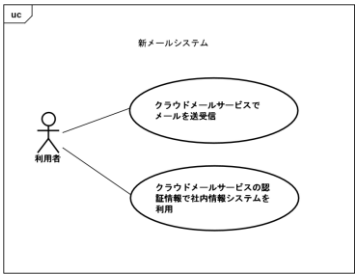
（2）適用結果

本事例にて提案手法を適用した結果を以下に示す（図7,8,9,10）．最初に，利用者のアクターを追加し，システムの要件に従って「クラウドメールサービスでメールを送受信」「クラウドメールサービスの認証情報で社内情報システムを利用」のユースケースを抽出した．次に，攻撃者のアクターを追加し，ミスユースケースとして「認証回避／権限昇格による不正利用」「不正入手したメールを元に社内情報システムへ侵入」を抽出した．「認証回避／権限昇格による不正利用」の「不正利用」の一例として，「別アカウントのメールを参照して第三者の機密情報を盗み出す」が挙げられる．続いて，社内情報システム開発者のアクターを追加し，不正利用のリスクを軽減するセキュリティユースケースとして「認証機能／認可制御を設定」を抽出した．最後に，提案手法を適用するため運用者のアクターを追加し，運用を考慮したセキュリティ要求分析を実施した．その結果，「認証機能／認可制御を設定」のセキュリティユースケースへフィードバックする運用セキュリティユースケースとして，「既存社内情報システムのセキュリティポリシーの棚卸し」を抽出した．さらに，「不正入手したメールを元に社内情報システムへ侵入」のミスユースケースに対するリスクを軽減する運用セキュリティユースケースとして「ネットワーク監視の強化」を抽出した．

適用の結果，検証事例（1）と同様に，セキュリティ機能を実装する際に運用から開発へフィードバックすべき事項を具体化することができるようになった．また，セキュリティ脅威，関係者，セキュリティ対策の関係を，運用まで含めた形でより詳細に示すことができるようになった．

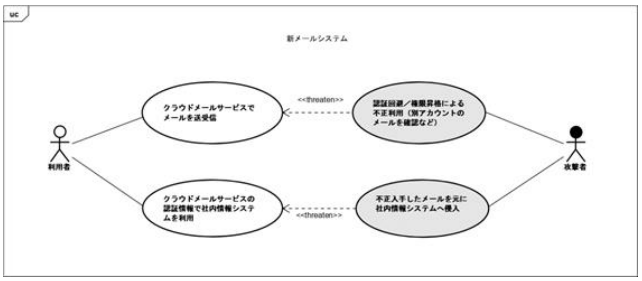
近年，基幹システムを再構築する際にクラウドサービスを利用するケースが増えている．本事例の適用結果は，基幹システムのクラウド移行全般に一般化可能と考える．基幹システムのセキュリティインシデントは重大な事故に繋がる傾向が高い．クラウド移行の際は，PSIRT(Product Security Incident Response Team)構築など，運用まで考慮した検討が望ましい．

Vol.2020-DPS-182 No.12
Vol.2020-CSEC-88 No.12
2020/3/12



【凡例】

図 7 ユースケース図【検証事例 2】



【凡例】

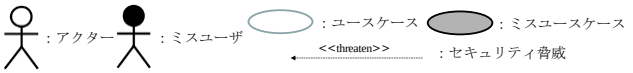
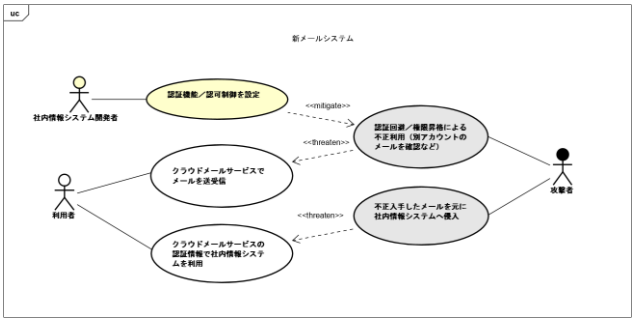


図 8 ミスユースケース図【検証事例 2】



【凡例】

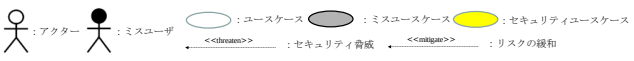
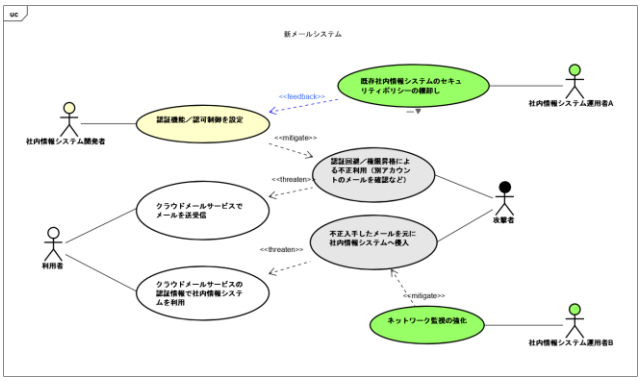


図 9 ミスユースケース図
（セキュリティユースケース有り）【検証事例 2】

6. まとめと今後の課題

開発者と運用者が一体となってシステム開発を行う DevSecOps では，運用まで考慮したセキュリティ要求分析が重要であることを解説し，従来のミスユースケースによるセキュリティ要求分析を拡張した，運用を考慮したセキュリティ要求分析手法を提案した．そして，具体的な案件事例に提案手法を適用し，評価を行った．その結果，セキュリティ機能を実装する際に運用から開発へフィードバックすべき事項を具体化することができるようになった．また，セキュリティ脅威，関係者，セキュリティ対策の関係を，運用



【凡例】

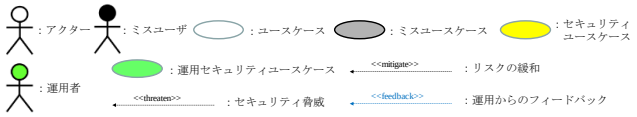


図 10 提案手法によるミスユースケース図【検証事例 2】

まで含めた形でより詳細に示すことができるようになった．

セキュリティ要求分析は重要なセキュリティアクティビティにも関わらず，作業の自動化が困難で実施コストが増大してしまう傾向がある．今後の課題としては，企画ユーザの要求を素早く抽出する要求把握の手法や，少ないインプットで高品質な分析結果をアウトプットできるコスト負担の少ないセキュリティ要求分析手法の開発が求められる．

参考文献

[1]Shannon Lietz , 「The Journey to DevSecOps」, RSA Conference 2016 , USA, 2016
[2]NTT テクノクロス株式会社, 「事例でわかるサイバーセキュリティの最新トレンド」, サイバー出版センター, 2018
[3]Carol Woody,「Agile Security - Review of Current Research and Pilot Usage」,SEI White Paper,2013.
[4]Akond Ashfaque Ur Rahman, Laurie Williams , 「Software Security in DevOps: Synthesizing Practitioners' Perceptions and Practices」 , International Workshop on Continuous Software Evolution and Delivery ,2016
[5]Microsoft Corporation , 「Microsoft Threat Modeling Tool」, 2017 , https://docs.microsoft.com/ja-jp/azure/security/develop/threat-modeling-tool
[6]Guttorm Sindre and Andreas L.Opdahl. Eliciting , 「security requirements with misuse cases」, Requir.Eng., Vol. 10, No. 1, p.34–44, 2005
[7]ZDNet Japan , 「『アジャイル』と『自動化技術』の限界」, 松岡真功,渡辺幸三, 2019, https://japan.zdnet.com/article/35144808/
[8]山田侑樹, 櫨山淳雄, 吉岡信和, 「ソフトウェアセキュリティ知識ベースを用いた要求分析及び設計における知識提示手法の提案」 , 情報処理学会第 81 回全国大会講演論文集 2019(1), 2019,p.253-254,
[9]株式会社セブン＆アイ・ホールディングス,「7pay（セブンペイ）」 サービス廃止のお知らせとこれまでの経緯，今後の対応に関する説明について, 2019 https://www.7andi.com/company/news/release/201908011500.html
[10]国立研究開発法人産業技術総合研究所,「産総研の情報システムに対する不正なアクセスに関する報告」について, 2018, https://www.aist.go.jp/aist_j/news/announce/au20180720.html