Vol.2018-CSEC-82 No.5
Vol.2018-SPT-29 No.5
2018/7/25

Infrastructure as Code によるシステム試験の自動化を用いた情報セキュリティリスク検出手法の提案

沼田晋作[†1]　中井悠人[†]　橋本昭二[†]　柏大[†]

概要：情報システムに対する変更適用時に潜在リスクが混入し，システムの可用性を損なう情報セキュリティインシデントが発生している．潜在リスクの検出はリリース前の試験で検出されることが望ましいが，試験環境および試験設計に課題があり困難である．本研究では，情報セキュリティに関するテストにおいて，システム開発等で使用される Infrastructure as Code を用いて試験環境の自動構築，試験の自動実施を行い，試験結果の定量的な取得と一覧表示をすることで，潜在リスクの検出を実施する手法を提案する．

キーワード：Infrastructure as Code, ansible，自動構築スクリプト，変更要求，潜在リスク, ISMS

Proposal of information security risk detection method by Automated System Test using Infrastructure as Code

SHINSAKU NUMATA[†]　YUTO NAKAI[†]
SHOJI HASHIMOTO[†] DAI KASHIWA[†]

Abstract:
Information security incidents that compromise the availability of the system is occurred by potential risks mixed in applying changes to the information system. Detection of potential risks in change applications is desirable to be detected in pre-release tests, but it is difficult for some problem in constructing test environment and test design. In this research, we propose the　the method to detect potential risks using list and indication by using Infrastructure as Code used in system development for automatic construction of a test environment, automatic test execution, and making the list of quantitative test results.

Keywords: Infrastructure as Code, ansible, automation script, request for change, Potential Risk, ISMS

1. システムの変更管理と潜在リスク

1.1 情報システムと変更管理

IT サービスに各種ステークホルダが要求を行い，要求への対応として，サービスを支える情報システムに対し必要な変更が適用されることは，一般的に発生する事象である．例として,ユーザからのサービスに対する機能追加要望や，情報セキュリティ対策としてソフトウェアのバージョンアップ実施などである．これら変更要求に基づいて，システムには各種変更が加えられるが，この変更の内容や適用方法，適用前の試験環境における各種試験，適用日や適用結果などを管理する変更管理は，システムの安定運用に非常に重要である．

1.2 JIS Q 27001 における変更管理と潜在リスク

企業では情報資産を守り事業を継続するため，情報セキュリティ対策を実施している．また，情報セキュリティに関する基準も複数存在し，その中で ISMS 適合性評価制度にも使用される，JIS Q 27001:2014[1] および JIS Q 27002:2014[2]は，企業で多く採用される情報セキュリティ基準である．この基準の中では，変更管理と潜在リスクへの対応として，求められる内容が記載されている．基準では変更の特定および変更作業の計画策定と事前テスト，情報セキュリティ上の影響を含む潜在的な影響についてアセスメントの実施が求められている．(JIS Q 27002:2014 12.1.2 変更管理)

1.3 潜在リスクと情報セキュリティインシデント

独立行政法人 情報処理推進機構 技術本部 ソフトウェア高信頼化センターがまとめた，情報処理システム高信頼化教訓集[3] において，実際に発生したインシデントの事例[4]とそれらに対する分析と教訓が示されている．この教訓集にはシステムのインシデント一覧が示されている．その一覧からは，システムに対する変更を原因としたインシデントを複数確認することができる．それらインシデントの発生原因の中には，リリース前に試験によって確認されるべき潜在リスクを発見できず，商用システムにおいてリスクが顕在化し，インシデントとなったものが存在する．教訓集ではそれらのインシデント原因を分析し，商用環境と試験環境との間に差分が存在した事例や，不適切なテスト実施や試験項目の不足，試験の未実施など，様々な原因と取るべき教訓を示している．これらの事例および教訓から，未だに多くのシステムへの変更管理において，潜在リスクを検出できず，長時間のシステム停止やサービス停止などの原因になっていることがわかる．

2. 潜在リスク検出における課題

2.1 事前のテストにおける課題

変更管理における潜在リスクは，商用環境へのリリース前に，試験環境におけるテストで検出され、リスクアセスメントを実施し，対策を実施することが JIS Q 27002:2014 にて要求されている．しかし，IPA が実施した情報セキュリティインシデント調査の結果からは，事前のテストで潜在リスクを検出し，リスクアセスメントを行えていない事例を見ることができる．IPA はこれらの課題に対し，高信頼化教訓集[3]において，技術的な教訓として 29 のポイントを記載している．この中で，変更管理における潜在リスクを発見する際に有用な，テストに関連する教訓が複数取り上げられている．教訓の中で，タイトルに「変更管理」「テスト」が含まれる項目は 4 つある．1 つは試験環境について，残り 3 つは試験設計についての教訓である．1 つめの試験環境における教訓の中では，試験環境と本番環境の差分について述べられている(教訓内項番 T6)．試験環境と本番環境が異なってしまうと，試験結果をそのまま本番環境でも得られる結果として解釈することは難しい．教訓集では差分を吸収しての試験の実施が求められている．しかし，差分の影響を完全に把握することは困難である．えられたテスト結果が，差分を加えると商用環境に発生するかどうかは，エンジニア等が机上検討しなければならない．システムは複雑に関連していることが多く，差分が存在するシステムに対し，その影響を見落としてしまうと，得られたテスト結果の解釈を誤ることになってしまい，インシデントの発生につながってしまう．次に，試験設計についての教訓は，テストシナリオの不足や検討漏れについての教訓である(教訓内項番 T3,T13)．たとえ試験環境を商用環境と同等にしたとしても，そこで行われるテストの項目が不足していれば，潜在リスクを検出することができず，その潜在リスクが商用環境において顕在化してしまう可能性がある．このため，網羅的に抜け漏れなく試験項目をリストアップ必要があるが，試験項目は，システムとしての技術的なテストだけでなく，サービス提供としての視点から必要な試験や，システムに発生する障害に対する耐久性を確認する障害試験など，項目が多岐に渡るため，人為的に網羅性を担保することは困難である．

項番	内容
T6	本番環境とテスト環境の差異に関する教訓
T3	テストパターンに関する教訓
T5	サービス視点での変更管理に関する教訓
T13	業務シナリオテストに関する教訓

表 1　変更管理およびテストに関する技術領域の教訓[a]

2.2 リスクアセスメントに関する課題

IPA の高信頼化教訓集[3]でも重要性が述べられているシステムへの変更適用の影響把握(教訓内項番 T5)について，リスクマネジメントの視点でのテスト技法が存在し，多く用いられている．SQuBOK 策定部会が策定したソフトウェア品質知識体系ガイドにも記載がある，リスクベースドテスト手法[5]と呼ばれる手法である．プログラムが障害を起こしうる状況やその影響について分析を行い，テストを設計する．これによって教訓 T5 を実現することができる．しかし，リスクベースドテスト技法では，事前にリスクの分析を行うため，適用される変更の影響を正確に把握する必要がある．しかし，変更適用の影響範囲を完全に机上検討することは困難である．IPA における障害調査においても，予備系への切り替わりの影響を想定しきれずに発生した障害(事例 1007,1523)や, 作業員が障害の影響を正しく運用員が把握していないことによる障害(事例 1406)も発生している．このため，ソフトウェア品質体系ガイドにも，他の補足的なテスト設計技法を用いることが留意点として述べられている．

障害事例番号	内容
1007	ディスク装置故障によりシステムがダウンした際にエラーメッセージが大量に発生し、連携先のシステムもダウンした
1523	故障したサーバを切り離したが、未処理チェック機能の対象が想定以上のデータ量となり、長時間のサービス停止となった
1406	バッチ処理が期限内に終わらなかった場合の影響をシステム運用担当者が正しく認識しておらず、カード会員への請求が遅延

表 2　事例

3. 本研究の目的

3.1 Infrastructure as Code とテスト

本研究では，2 章で述べたテストにおける課題と，リスクベースドテストを補足するためのシステム試験を，Infrastructure as Code による自動テストによって解決することを試みる．Infrastructure as Code とは，従来の設計書，手作業によるシステム構築から，ソフトウェアのコードのように定義された情報を用いて自動でシステムを構築する技術である．ansible[b]などの構成管理ツールと組み合わせることで，開発環境やテスト環境を誤りなく速やかに構築することが出来る． Infrastructure as Code によるシステム自動構築は，構築結果の再現性が担保でき，構築されるシステムが一意に定まることと，自動的に構築が可能であるため構築コストの削減，構築結果の品質向上というメリットがある．沼田らは，ansible と開発に使用される各種ツー

a 情報処理推進機構 情報処理システム高信頼化教訓集「技術領域の教訓」

†1 NTT コミュニケーションズ株式会社

b ansible は、米国およびその他の国において登録された Red Hat, Inc.の商標です。

ルを組み合わせ，構成管理情報から開発環境などを ChatOps で自動構築[6]し, DevOps を円滑に進める手法を実装し評価した．構成管理情報から構築された環境は，商用環境および開発環境全て同一な環境となり，実システムの正常性の確認や RCA(Route Cause Analysys)への適用などを行えることを示した[7].

3.2 テストにおける課題への対策

(1) 試験環境の課題に関する対策

教訓集[3]によれば，テストの環境が商用環境と異なる場合には，環境差分を吸収し，テストを実施する必要があるとされている．しかし，2.1 節でも述べたとおり，環境差分を完全に把握して試験を実施し，結果を解釈することは非常に困難である．例えば，商用環境ではクラスタリング構成を組み，VIP を用いたシステムを用いているが，試験環境では VIP を使用せずにシングル構成で構築したとする．この場合，アクセスする IP アドレスが異なるだけでなく，試験対象機器が保有している IP アドレスの数が異なるという環境差分が発生する. この IP アドレス数の違いという差分の影響を考慮すると，IP アドレスをサービスだけでなく，他の目的での用途を兼ねるため，使用目的が重複するという影響が考えられる．次に，IP アドレスの数が異なるため，firewall で定義されている ACL もこれに合わせて変更しなければならず，ACL が正しく記載されているかの確認も必要である．また，VIP と異なり実 IP アドレスはネットワークインターフェースに紐付くアドレスとなるため，ネットワークインターフェースに関連付けて記載されたルーティングなどの影響を受けることが考えられる．このように, 単に VIP の構成を実 IP アドレスの構成にしただけで，これだけの影響を考慮して，環境の整備や試験結果の解釈などを行う必要がある．これらの環境差分を全て洗い出し影響を考慮するのは困難であり，不適切な試験が行われる原因となる．

また, 試験環境を構築し, 必要なタイミングで試験を行い，試験完了と共に環境を商用に合わせて修正するなどの環境維持は, 非常に大きな人的コストがかかる．本研究では 3.1 節で述べた技術を適用し，テスト用の試験環境を構成管理情報から自動構築する．さらに，それら試験環境においてテストを自動実行する．構成管理情報から構築されるシステムは、商用環境と同一の構成であり，試験結果に対して差分の影響などを考慮する必要がない．また，テスト用の環境を自動構築し，対象の環境に対して自動的に試験を実施する．システム試験に必要な環境は，試験対象のシステム，試験対象と関連するシステムの動作を代替するスタブシステム，試験を実行する試験システムであり[8]，それらを自動的に構築し試験を実行し，構築および環境維持のコストを削減することを可能にする．

(2) 試験設計の課題に関する対策

IPA による情報セキュリティインシデントの調査結果[4]からは，テストにおける試験項目の課題として、試験の実施不足や項目不足，変更影響の考慮不足があげられている．試験項目の不足原因として，教訓集からは原因として試験項目の蓄積不足，検討不足があげられている．そこで，本取り組みでは試験項目をシナリオとして管理し，システムのデータベースに保管することで，抜け漏れを防止する．また，試験の実施不足については，事例 1506(負荷テスト未実施)および 1541(機能テスト未実施)にも示されるとおり，試験実施そのものが行われていない事例が紹介されている，試験はシステムを占有して行われるため，マシンタイムを調整し実施する．このため，任意のタイミングで実施が困難となることが多い．本研究では自動的に構築したテスト環境に対し，過去に実施されたテスト全てを自動的に実施することとした．試験環境を自動で構築するリソースプールを用意し，試験環境の構築と試験の実施を自動で行う．これにより，必要に応じて試験が実施され，かつ，人的コストを抑えることができる．

3.3 リスクアセスメントの課題に関する対策

リスクベースドテストを補完するために必要な試験を行う．IPA による教訓集[3]において，変更の影響を確認する教訓として，変更された箇所以外の試験も行い，正しく動作すること(教訓集 T5)および, ユーザ視点での確認(教訓集 T13)をすることが求められている．本研究ではこの教訓を実現するため，0 節で述べたテストにおける変更適用の影響把握と試験結果の判定について，網羅的な自動試験を行い変更された箇所以外を確認し，ユーザ視点での確認として，比較可能な定量的な試験結果の取得および結果の比較で解決することを試みる．

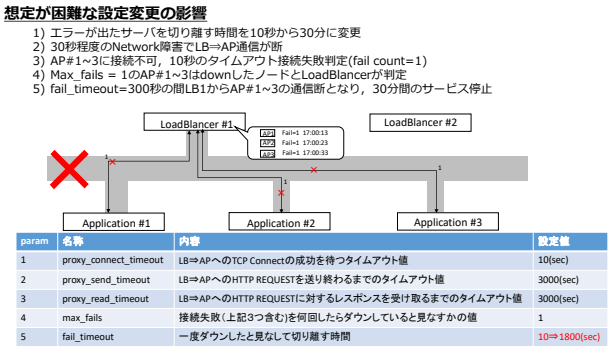
(1) 変更適用の影響把握

変更適用における影響把握のため蓄積された試験項目を網羅的に実施する．本研究では特に，机上検討することが困難な障害影響を再現することによる潜在リスク検出を行う．変更適用が障害時に与える影響を机上検討が難しい例を図 1 に示す.

LoadBalancer から Application サーバへのバランシングを行っており，アクセスがエラーとなったノードを切り離し一定時間後に LoadBalancer に組み込む間隔が設定されている．この値のデフォルト値が 10 秒であったとする. この値では, Application サーバのダウン時に 10 秒に 1 回必ずエラーが発生しユーザ影響が大きいため, 平均的な復旧時間の 30 分に値を変更した．これにより，Applicaation サーバのダウン時に発生するユーザ影響を低く抑えることができた. しかし，この設定変更後，ネットワークに 30 秒の断が発生した．LoadBlancer と Application サーバとの通信が途絶したため，Application サーバが LoadBalancer サーバのバランシング先から切り離された．デフォルトの設定であればネットワーク断のあと 10 秒で再接続するが，設定を変更したため 30 分間は接続が行われずサービス断となってしまう. これは，

10 秒から 30 分への変更に際して，ノード障害という事象は想定しているが，ネットワークの全断という事象は想定されていないためである．設定変更時に，ネットワークの全断という障害試験を実施しないと，この潜在リスクは発見できない．

本研究では，実際にシステムにおける障害ポイントに対し，網羅的に障害を発生させ[9]，テストを行うことで変更適用における影響把握の解決を試みる．障害ポイントは，各サーバで動作しているサービス，ネットワークのインターフェース，ネットワークである．適用される変更を施したシステムに対し，各サーバのそれぞれの障害ポイントについて障害を発生させ，試験シナリオを実行する．これによって，変更を施したシステムに対する障害が発生した際の網羅性も試験結果として取得可能となる．



param	名称	内容	設定値
1	proxy_connect_timeout	LB⇒APへのTCP Connectの成功を待つタイムアウト値	10(sec)
2	proxy_send_timeout	LB⇒APへのHTTP REQUESTを送り終わるまでのタイムアウト値	3000(sec)
3	proxy_read_timeout	LB⇒APへのHTTP REQUESTに対するレスポンスを受け取るまでのタイムアウト値	3000(sec)
4	max_fails	接続失敗(上記3つ含む)を何回したらダウンしていると見なすかの値	1
5	fail_timeout	一度ダウンしたと見なして切り離す時間	10⇒1800(sec)

図 1　パラメータ変更と影響範囲

(2) テストの定量評価

IPA が示している教訓集において，変更適用の影響を把握するためには，システムとしての変更された箇所の試験だけでなく，利用者側のサービス利用視点での試験が必要であると述べられている．そこで，本研究では実施したテストの結果を定量的に取得し，過去の結果との比較によって利用者視点でどのような変化が起きたのかを自動的に評価して潜在リスクを検出することを試みる．本研究では WEB-API システムを対象として開発を行った．WEB-API システムにおいては，どのようなレスポンスをシステムが何秒で返したのか，それはどう変わったのかという点での比較を行うこととした．例えば，ある WEB-API をコールした際に，試験項目としては 10 秒以内に完了すれば合格としていたとしても，ユーザ視点，サービス視点としては以前何秒でレスポンスを取得し，変更後に何秒で完了したのかという点を試験項目とする．従来 2 秒で完了していた物が 9 秒で完了したとすると，システムの試験としては合格であるが，ユーザ視点での試験とすれば，9 秒に変化したという変更の影響が出ており，その原因や影響を考慮しなくてはならない．本研究では，テストにおける結果について，各試験シナリオに対して評価対象となる定量的な指標としてレスポンスコードとレスポンス取得時間を記録する．そして，それらを自動試験の中で取得し，それら定量的な試験結果を過去の試験結果の履歴管理と比較する．えられた結果の比較を行い，過去の値と大きく異なる値を GUI でわかりやすく表示し，潜在リスクの可能性として表示する．これにより，変更によるユーザ視点での影響を視覚的に認識することが出来る．

4. 実装

4.1 自動テスト

4.1.1 試験環境自動構築

本研究では，試験用のシステムを自動構築するトリガを 2 つもうけた． 1 つはバージョン管理ソフトウェア Git[c]にソフトウェアが commit されたタイミングである．Git にソフトウェアが commit されるタイミングは，ソフトウェアの変更が確定したタイミングであり，このタイミングで自動的にテストを行うことで，ソフトウェアの変更による影響を確認することが可能となる．もう 1 つは ChatOps や WebUI によって呼び出されるタイミングである．これは任意のタイミングで実行される試験である．システムに対する変更要求は，開発されたソフトウェアの更新だけではなく，システム設定値などの変更でも発生する．これらについての対応のため，任意のタイミングで試験を呼び出す仕組みを取り入れた．

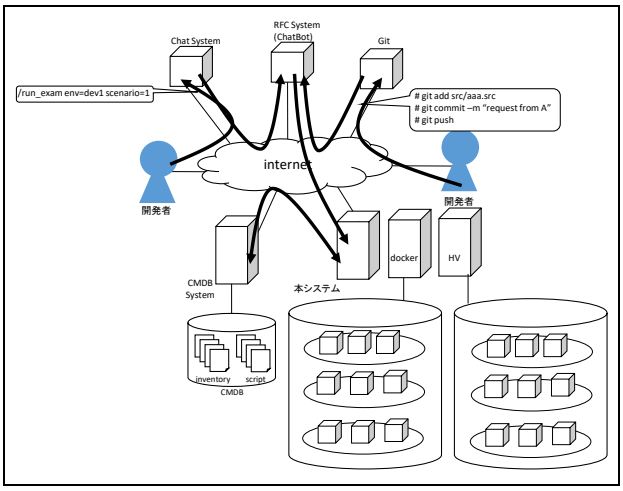


図 2　システム自動構築システム図

4.1.2 テスト自動駆動

試験を実施するタイミングは，4.1.1 のシステム構築が完了したタイミングで実施する．実施する内容は，構築された環境に応じて確定する[8]．試験シナリオそのものも Git 管理されており，試験実施のタイミングで最新の試験シナリオが実施される．本研究では，API コールを受けて処理を行い，結果をレスポンスで返すシステムを対象に実装を行った．特に，可用性についての潜在リスク検出において，システムの構成要素に障害を発生させた状態で API コールを実行する．構成管理情報から障害ポイントを自動生成[9]し，障害を発生させる．これによりシステムに障害を発生させた状況でシステム試験を実施し，結果を定量的に取得

c https://git-scm.com/

する．
本研究の目的として，試験の結果を定量的に取得して評価する．このため、試験環境のリソース状態を均一に保つ必要がある．なぜなら，試験環境において多くの仮想マシンが動作している状態と，ほとんど仮想マシンが動作していない状態では，試験結果が試験対象のアプリケーションや設定によらず変わってしまうためである．このため，仮想環境を構築する際にリソースの上限を設け，また，試験実施を行う環境も制限を設けて，試験環境の結果の均一性を担保した．

4.1.3 試験履歴保管

3.3 節の(2)で示したとおり，ユーザ視点の試験においては，過去とどう変わったのかの変更を把握することが重要である．このため，本取り組みでは試験で定量的に取得した結果を履歴保管するため，試験を実施する毎に試験 ID を払い出して付与する，さらに試験のシナリオのシナリオ ID ごとにえられたレスポンスとレスポンスタイムを保持し，試験 ID-シナリオ ID に紐付けて保管する．
試験シナリオは機能の追加と共に増加する，教訓集にも述べられているとおり，これら試験シナリオの抜けなどが試験不足による情報セキュリティインシデントを引き起こす．このため，試験シナリオも一つの管理対象として Git 管理する．試験シナリオは機能追加だけでなく，商用システム運用上発生した際にも追加されることがある．これらについては経緯なども合わせて記載し，履歴管理に役立てることとした．

4.2 潜在リスク検出とリスクアセスメント

4.1 節で示した実装に基づき，試験で使用される個々の API コールについてレスポンスおよびレスポンスタイムを取得し，得られた結果について一覧を高めて表示する．試験項目の網羅性および障害ポイント網羅性を高めているため，項目数が多くなってしまう．多くの項目につて実施した結果，確認が漏れてしまうと潜在リスクの検出が行えない．このため，得られた試験結果について，わかりやすく比較結果を GUI 表示し，大きく異なる箇所を色づけして表示した．これにより，全ての結果を目視確認することなく，自動的に比較し，一覧性高く確認することができ，試験項目の結果判定を抜け漏れなく実施することができた．

Results
2018/06/21 15:23:54 called by ChatOps
Examination ID - 00013

Scenario ID	Response(last)	Time(last)	diff
1	200(200)	08.01(09.01)	-12%
2	201(201)	13.07(02.04)	+550%
3	403(200)	13.21(03.25)	Different Action
4	200(200)	08.31(07.01)	+18%
5	401(401)	01.21(00.81)	+49%
6	201(201)	23.16(19.04)	+21%
7	201(201)	02.01(02.03)	-1%

図 3　テスト結果の一覧表示（一部）

4.3 リスクアセスメントのための情報表示

本研究では，障害ポイントの網羅性を担保してシステム試験を行う．しかし，システムの障害ポイントはテキストによって示されても，その発生箇所を誤って認識してしまうことがある．例えば，図 4 で示すシステムとその障害箇所は，テキストで説明した場合には「DMZ 面に面するアプリケーションサーバの中で動作する daemonize ソフトウェアの停止」であるが，その文章からシステムの構成における障害がどこに起きたのか，どのような状態かを想定し，どれくらい起こりうる事象なのか，可能性として大きいのかを検討することが難しい．そこで本研究では，4.2 節で示した障害ポイントとして，事象を図として表示することで，リスクアセスメントにおける事象を把握しやすくする機能を備えた．これにより，発生する事象を把握しリスクアセスメントに活用することが可能である．

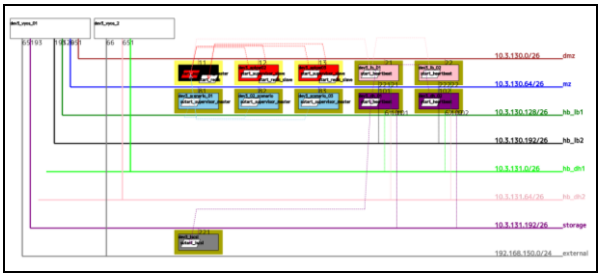


図 4　システム構成図および障害発生図

5. 評価

5.1 テストの自動実施

テストの自動実施が，期待通り行えたのかの評価を行った．開発者が能動的にテストを ChatOps から呼び出した際に，問題なくテストシナリオが動作し，結果を履歴として保存することが出来た．また，Git に commit が行われた際に，対象のソフトウェアを使用したテストが自動的に行われた．また，試験実施の環境数の上限に達した際には，正しく試験の実施を止め，他の試験が完了してから試験の実施が行われたことが確認でき，試験結果は試験 ID-シナリオ ID-レスポンスコード-レスポンスタイムの形で履歴として保管され，同一の試験シナリオについてレスポンスの差分とレスポンスタイムの増減を一覧表示することができた．
試験結果の活用について，課題点が見つかった．簡易に差分のある試験項目のみを表示しているため，試験シナリオがどのような背景で実施されている試験なのかが記録されていない．試験項目の中には、商用システムにおいて発生した不具合が再発していないことを確認するための試験や，特定のバージョンのソフトウェアだけで実施する項目のように，特定の理由から実施されている試験項目が存在する．これらの試験項目は，システムの構成が変更されていく中で削除されていくべき試験項目である．試験項目の蓄積は重要であるが，単調増加では試験の実施時間が非常に長くなり，都度実施の実現が困難になる．これら試験項目の定

期的な見直しが課題である.

5.2 潜在リスク検出およびリスクアセスメント

テストから情報セキュリティの潜在リスク検出と、リスクアセスメントのためのテスト結果を表示できるかを評価した．具体的な潜在リスクとして，過去に実システムに発生した変更適用における各種障害を検知できるかで評価した.リスク検出結果の一部を表 3 に示す. 項番 3 の潜在リスクが検出できなかった．これは，BUM 通信の帯域制御を，テスト環境の仮想スイッチの機能では実現できなかったためである.

検出できた 1,2,4,5 については，実施したシナリオの状態を図として表示することが出来た．項番 1 の潜在リスクについては，事前にリスクアセスメントを行い，発生確率からリスクを受容すべきか，対策として設定値の変更を取りやめるかの議論を行うことが出来た．本取り組みを用いるためには、把握した障害ポイントについて，発生確率を付与してリスクを算出する工程が必要である．この発生確率を並列する長期安定試験などから実データを得て実施するなどで, 試験の精度を向上させることが可能であると考える.

項番	内容	結果
1	LoadBalancer のノードチェック間隔を変更し, 一度は以下から外れたノードの復帰を 10 秒から 300 秒に変更, ネットワークの全断が 30 秒発生し, システムが 5 分全断	検出
2	内部 DB への接続数を 8 に制限，多重接続が行われた際に, 当初設定していたタイムアウト値を超えて, タイムアウトを返却	検出
3	ネットワーク機器が BUM 通信の帯域制御を設定，クラスタリングの Heartbeat 通信が制限され切り替わりが発失敗	非検出(変更再現できず)
4	データベースソフトウェアをバージョンアップ, PID ファイルおよびコントロールコマンドが変更され, 内部スクリプトが動作せずバックアップ取得できず	検出
5	snmp のコミュニティ名を変更したところ，内部監視の snmpwalk コマンドの認証が失敗し, 内部監視スクリプトが動作せず.	検出

表 3　リスク検出結果

6. まとめと今後の予定

Infrastructure as Code を用いた試験自動化により，システムに変更が加えられた際の情報セキュリティの潜在リスクを検出できることを確認した. 自動的にテスト環境を構築し，用意されたテストシナリオを実施，定量的に取得した結果を過去の値と比較することで潜在リスクを検出することが出来た.

今後は，よりリスクアセスメントを行いやすい情報を提示するため，事象の発生確率を試験における障害等にパラメータとして付与し，リスクアセスメントのためのデータ一覧を自動出力するなどの機能を追加していく予定である.

参考文献

[1] 情報技術－セキュリティ技術－情報セキュリティマネジメントシステム－要求事項, 財団法人日本工業規格
[2] 情報技術－セキュリティ技術－情報セキュリティ管理策の実践のための規範, 財団法人日本工業規格
[3] 情報処理推進機構, 情報処理システム高信頼化教訓集（IT サービス編）, IPA
[4] 情報処理推進機構, 「注意すべき観点」に基づいた障害事例の分類, IPA
[5] リスクベースドテスト,ソフトウェア品質知識体系ガイド SQuBOK Guide V2, SQuBOK 策定部会, 2017/11/15
[6] 基盤情報を含む構成管理情報を用いたシステム構成要素の関連性把握システムの開発と評価,沼田晋作 神谷法正 橋本昭二 柏大（NTT コミュニケーションズ）, 電子情報通信学会, ICM 研究会(2017/03/10)
[7] 自動構築スクリプトを用いた CMDB による障害原因自動特定・復旧機能の実装と評価,沼田晋作 神谷法正 橋本昭二 柏大（NTT コミュニケーションズ）, 電子情報通信学会, ICM 研究会(2017/01/18)
[8] 多拠点展開されるシステムにおける構成管理情報を利用した試験自動化の一考察, 中井悠人・神谷法正・沼田晋作・橋本昭二・柏 大（NTT コミュニケーションズ）, ICM 研究会(2018/03/08)
[9] The Implementation and Evaluation for Automatic Platform Failure Test System Using CMDB From Automation Script, Shinsaku Numata, Norimasa Kamiya, Shoji Hashimoto, and Dai Kashiwa(NTT Communications Corporations), APNOMS 2017