Vol.2022-CSEC-97 No.15
Vol.2022-IOT-57 No.15
2022/5/20

KEKにおける脆弱性自己点検PDCAサイクル高速化

與那嶺　亮[1,2,a)]　鈴木 聡[1,2,b)]　一井 信吾[1,2,c)]

概要：KEK ではこれまで 10 年以上にわたり，外部公開サーバの各機器管理者がアプライアンスを使用して脆弱性診断・自己点検を行える環境を整備してきた．自己点検を導入した当初はアプライアンスが高速スキャンをすれば DoS になり，低速であると時間がかかるため非同期バッチ処理のような操作が必要であること，また脆弱性の説明文が分かりくいなどの理由でそのままでは受け入れられづらい面があったためアプライアンスの直接操作させることは避け，独自の UI サーバを仲介して運用していた．
10 年の間にアプライアンスの説明文も改善され，サイバーセキュリティに対する構成員への啓蒙も遙かにすすみ，説明文を直接管理者が読んでもほぼ問題が無い状態となったことから，2021 年度からは各管理者がアプライアンスの直接操作を行う方式に切り替えた．これによりユーザが発見した脆弱性への対応が十分であったかのイテレーション操作が簡便になり，従来よりも短い期間での自己点検が完了できるようになった．本稿はこのワークフローの移行によって得られた知見についてまとめたものである．

Improving PDCA cycle on vulnerability self-inspection at KEK

Abstract: For more than a decade, KEK has been developing an environment in which administrators who run public servers can perform vulnerability scans and self-inspections using an appliance. We had introduced our own UI server because firstly asynchronous batch processing was required at that time to avoid a DoS by a high-speed scan by the appliance, and secondly the vulnerability descriptions were difficult to understand and hardly acceptable.
During the past 10 years, the issue on such descriptions have been improved, and also the awareness for cyber security among the administrators has been intensified. Now it seems the administrators can readily understand the descriptions directly provided by appliances, and in fact from FY2021 we have decided to make use of a UI of an appliance itself instead of using our own UI server. This made it easier for administrators to confirm whether discovered vulnerabilities were adequately addressed or not, and thus to shorten time of period for self-inspections than in the past. This paper summarizes the findings of this migration.

1. はじめに

1.1 背景

高エネルギー加速器研究機構（以下，KEK）では，組織外からアクセス可能な機器 (DMZ 機器) に対して脆弱性診断を管理者自身が実施して状態を把握するための “DMZ User’s Portal”[1] を内製し，機器管理者各々でセキュリティレベルを維持する PDCA サイクルを 10 年以上に渡って回してきた．脆弱性診断自体はオンプレミスのアプライアンスによって実施され，結果を DMZ User’s Portal が回収して整理し，一覧性を高めている．脆弱性診断は定期的かつ自動的に実施され，緊急度の高いものは直ちに対処を取るよう管理者に通知される．管理者は通知を元に対処を行い，手動で診断を実行し脆弱性が解消されたかを自分のタイミングで確認出来る．これに加え毎年脆弱性の総点検を管理者に義務づけて結果を回収し，機構全体としても PDCA サイクルを回している．脆弱性診断はそれなりに負荷の高いタスクであるため，DMZ User’s Portal は関連システムが過負荷にならないよう，バッチシステムのように診断指令を投入していた．また，診断完了後に診断データから PDF レポートを生成するタスクも手動で投入する必要がある．管理者の立場から見るとレポートに示される脆弱性情報では対処方法が明確ではないことが多々あり，設定変

[1] 高エネルギー加速器研究機構 計算科学センター
High Energy Accelerator Research Organization, Tsukuba, Ibaraki 305–0801, Japan
[2] 総合研究大学院大学　高エネルギー加速器科学研究科
School of High Energy Accelerator Science, The Graduate University for Advanced Studies (SOKENDAI) Tsukuba, Ibaraki 305–0801, Japan
[a)] ryo.yonamine@kek.jp
[b)] soh.suzuki@kek.jp
[c)] ichii@post.kek.jp

更・手動診断・レポート生成タスク投入・ダウンロードと確認というイテレーションを繰り返して試行錯誤せざるを得ない．それぞれの間に待ち時間が発生するため，対象機器が多い管理者は提出締切前の心理的負荷が高い．このような場合にポータルから現在走行中の診断状況が確認出来たり，制御可能であれば試行錯誤中の管理者の負荷を多いに軽減出来る．

15 年に渡る PDCA サイクルと構成員に対する啓発活動によって，アプライアンスが直接出力する診断レポートも DMZ 機器管理者に抵抗なく受け入れられつつある．2018 年には脆弱性診断アプライアンスの nCircle IP360 から Tenable.sc へと変更された．現行の Tenable.sc では脆弱性の検出するだけではなく，診断対象機器と Tenable.sc 内アカウントの紐付けも管理できるようになっている．また，脆弱性一覧を AJAX を使用したダッシュボードで確認出来たり，特定の脆弱性について対応策が出来たかどうか個別のスキャンを実施・キャンセルできるなど，UX が大きく向上している．アプライアンス側の API とその出力フォーマットはバージョンアップによって変更されることが多々あり，追従する人的コストが非常に高かった．今回は DMZ User's Portal を Tenable.sc の API に追従するよう更新するのではなく，自己点検のワークフローを見直して Tenable.sc が提供する Web UI をそのまま活かすこととなった．

1.2 解決すべき問題点

Tenable.sc が提供する Web UI を軸としたワークフローに変更するにあたり，KEK として解決すべき課題は以下の点であった：

(1) 機器管理者 (約 80 名) ごとに Tenable.sc の Web UI 用のユーザアカウントを初期化して配布する必要がある．当初自己点検のワークフローに供するのに適切な認証基盤が存在しなかったため，DMZ User's Portal はその内部に利用者のアカウント/認証情報を格納していたが，これを Tenable.sc のアカウントにまるごと移行することは不可能だった．
(2) 管理者・DMZ 機器 (約 200 台) ごとに脆弱性診断が行えるように，診断のコンフィグレーションを個別に用意する．設定を正しく行うためには Tenable のマニュアルをよく理解する必要があり，設定を間違えると正しい診断ができない．
(3) 新しいワークフローのマニュアルを用意
(4) DMZ User's Portal で提供していた脆弱性の週次メール通知の継続と改善．それまでの Critical や High の脆弱性だけでなく，Medium 脆弱性についても通知するように変更．また通知メール文のタイトルや本文の簡単な修正すら困難な状況であったため，この点も改善する．

これらの課題に対して，機器管理者の負担を最小限に抑えることを念頭に，Web フレームワークと Tenable.sc で提供されている Web API を利用して，準備をできるだけスクリプト化して課題解決を目指した．

2. ワークフローの再構築

2.1 各課題解決への基本方針

課題 (1) に関しては，Tenable.sc の認証には LDAP を使用することとした．Tenable.sc の場合はまず当該ユーザ名が内部に登録されていることが前提であって，LDAP を利用しても Tenable.sc 上のユーザの登録操作は必要となる．LDAP については DMZ User's Portal よりも後に整備された IP アドレス管理システムの認証情報を利用することとした．こちらは初期パスワード確認，リセットの手順が業務フローとして既に整備されている．このシステムは LDAP での認証を提供していなかったが，本件のために新たに LDAP での照合を可能にした．

課題 (2) に関しては，Tenable.sc の Web API を利用して，各管理者・各機器ごとの診断のコンフィギュレーションを事前に用意し，機器管理者は Web UI ログイン後，作成された診断定義を実行するだけ，という単純なワークフローとする．機器管理者の負担を減らし，また間違った設定を発生させるリスクを抑えることができる．ワークフローの簡潔化は，課題 (3) のマニュアルをわかりやすくすることにもつながる．

課題 (4) に関しては，Python ベースの Web フレームワークの一つ Django [2] を利用して別途構築する．Web フレームワークを利用する主な目的は，複数のユーザが利用するような一般的な Web アプリを作成するためではなく，脆弱性の週次メール通知の本文編集ページの構築や ORM を利用したデータベースの操作など，フレームワークの便利機能の一部を利用するためである．Django では，データベースを閲覧・編集できる管理サイト (Django Admin Site) が自動的に作成されるため，この仕組を利用すると，通知メール本文のテンプレートなどの情報を格納したデータベースを用意するだけで，管理者用サイトが自動的にできあがり，通知メールの内容を GUI から変更ができるようになる．また，Python では，Requests などの HTTP ライブラリが用意されているため Tenable.sc で用意されている API を利用するスクリプトも簡潔に書ける．

2.2 Tenabel.sc Web UI の LDAP 認証

Tenable.sc はオンプレミスの脆弱性管理システムで，Web ブラウザからシステムにアクセスし，図 1 のようなログイン画面からログインすると，ユーザごとに脆弱性診断を行う機器の設定や脆弱性診断の設定や実行を開始することができるようになっている．

LDAP 認証の設定は Tenable.sc Web UI のユーザ作成を

図 1 Tenable.sc Web UI へのログイン画面．ユーザはログイン後，脆弱性診断の設定，実行や結果の閲覧が可能

Fig. 1 Login screen to Tenable.sc Web UI. Logged-in users can configure and run vulnerability scans, and view those results.

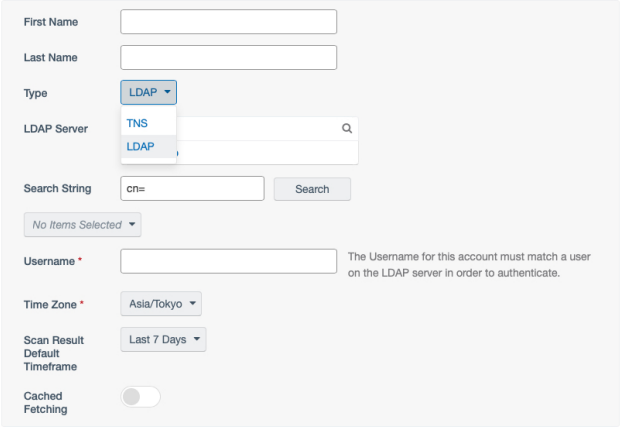

図 2 Tenable.sc Web UI のユーザ作成画面．ユーザ認証情報としてユーザ名とパスワードをここで指定する TNS 方式の他，LDAP 認証も利用可能

Fig. 2 “User Add” screen of Tenable.sc Web UI. LDAP authentication is an option as well as the TNS method, in which a user name and password are defined here as user authentication information.

行う際に，認証タイプとして LDAP を選択すると，LDAP サーバの情報を入力する項目が現れる (図 2)．まったく同様の操作を Tenable.sc が提供する Web API [3] から行えるようになっているため，KEK がもつ機器管理者のデータベースを参照する，簡単な Python スクリプトにより LDAP 認証を設定したユーザアカウント (約 80 名分) を一括作成した．

2.3 脆弱性診断コンフィギュレーションの一括作成

2.3.1 脆弱性診断のコンフィギュレーション

Tenable.sc Web UI にログインしたユーザは，自身が管理する機器の情報を “Asset” オブジェクトとして定義し，さらに脆弱性診断の設定項目を定義する “Active Scan” オブジェクトと，診断結果のレポートを定義する “Report” オブジェクトを用意する必要がある．

また，Tenable.sc Web UI では，脆弱性診断結果を他のユーザと共有する仕組みも備わっている．KEK でも，複数の管理者が一つの機器を共同管理している場合が多数を占めるため，管理者間での診断結果の共有設定を活用することにした．

Active Scan オブジェクトを用意すると図 3 で示すように一覧表に診断実行開始ボタンが表示され，ユーザはこのボタンを押すことで診断を行うことができる．図 4 は，診断実行中の進捗状況を確認する画面で，このおかげで冒頭で述べた当初の課題を根本から解決できるようになった．

2.3.2 一括作成

2.2 で述べたユーザ作成と同様に Tenable.sc Web API を利用して 2.3.1 で述べたコンフィギュレーション作成が行えるため，管理者と関連する機器をデータベースで参照しながらスクリプトによる一括生成を行うことができた．但し，各ユーザがログインして GUI で設定を行う場合と自動化して一括して設定を行う場合の異なる点として，スクリプトでは Tnable.sc の Web API を管理者権限で実行するため Asset オブジェクトや Active Scan オブジェクトの所有権を各ユーザに割り当てる設定を追加する必要があった．

2.4 操作マニュアルの作成

2.2 で述べたユーザ作成と 2.3 で述べた診断設定を事前に一括して行うことにより，各機器管理者は，ログイン後，該当する Active Scan の実行開始ボタンを押すことで診断を開始し，Report の作成ボタンを押すことで診断結果のレポートが作成される環境ができあがった．ワークフローマニュアルには，

(1) ログインに必要なユーザ名とパスワードについて，
(2) 図を用いた操作手順，
(3) 注意事項など

を 3 ページ程度に収まるように簡潔に記した．

2.5 脆弱性週次メール通知

Tenable.sc では，脆弱性診断の実行を定期的にスケジュールすることができる．各機器管理者が行う自己診断とは独立して，KEK では週一回決まった時間に，DMZ 機器に対し脆弱性診断が走っている．その週次診断結果のうち，特に早急に対策が必要な脆弱性が検知されたものを，機器管理者にメールで通知を行う．

基本的な動作の流れは

(1) Tenable.sc Web API を利用して，診断結果を取得
(2) 診断結果の解析と機器管理者情報との統合
(3) メールの題名と本文のテンプレートにデータを埋め込み，各機器管理者への通知メールを自動作成
(4) 脆弱性対応完了まで状況を追跡するためのチケットの自動発行もしくは更新

で，これらをスクリプトで記述した上で cron により定期

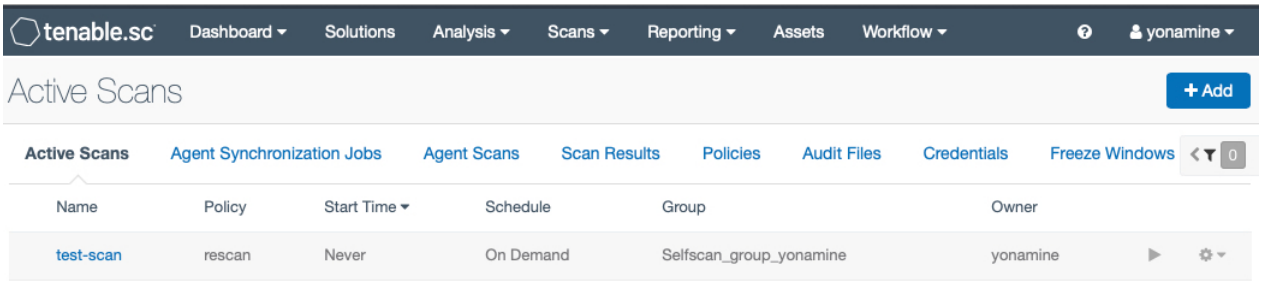

図 3 Tenable.sc Web UI の Active Scan 一覧画面．リストの右端の三角ボタンを押すことで診断が開始される．

Fig. 3 "Active Scan" list screen of Tenable.sc Web UI. Scan starts by pressing the triangle button on the right end of each item.

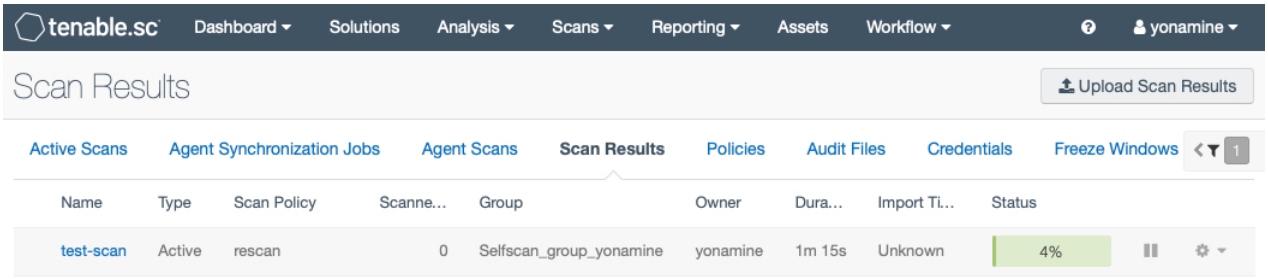

図 4 Tenable.sc Web UI の診断中の進捗状況の確認画面．リストの右端の一時ボタンを押すことで診断が一時停止される．その後再開させることもできる．

Fig. 4 Screen to check the progress during scanning in Tenable.sc Web UI. The scan can be paused by pressing the button at the right end of the list. It can be restarted afterwards.

的に実行させている．

2.5.1 診断結果の取得

Tenable.sc Web API を使って，診断結果を JSON 形式などで取得できる．KEK では，WAN, DMZ, LAN の 3 つの領域に診断装置を配置し，診断を行っているため 1 つの機器あたり 3 種類の診断結果が得られる．

2.5.2 診断結果の解析と機器管理者情報との統合

週次メール通知のための脆弱性診断は，早急な対策が必要な脆弱性の検知を目的とするため，深刻度が Medium 以上の脆弱性のみを抽出するように設定した．各機器ごとに，深刻度，WAN,DMZ,LAN の区別，脆弱性の内容，対策案などを分類し，さらに機器管理者データベースと照合しながら通知すべき機器管理者の情報をオブジェクトデータとして統合しておく．

2.5.3 通知メールの自動作成

通知メールの題名と本文のテンプレートをデータベーステーブルとして用意すると，Django Web フレームワークが自動的にデータベース管理サイト (図 5) を作成してくれる．通知メールのテンプレート以外にも設定変更の可能性があるものはデータベーステーブルに記述し，スクリプトがデータベースを参照するしくみとなっている．

2.5.4 チケットの自動発行

KEK では，検知された脆弱性をチケット管理するために，メール通知と同時にチケットの発行を行う．チケットシステム側の Web API を利用することで，スクリプトからチケット発行が可能である．週次メール通知のための脆弱性診断は，脆弱性が解消されない限り，同じ脆弱性を通知し続けることになる．チケットを管理する上では，同じ脆弱性に関してはチケットを新規で発行せずに，代わりに未対応の脆弱性として既存のチケットにコメントを追記する方が都合が良い．そのため，スクリプトでは，新規の脆弱性と未対応の脆弱性を区別して扱えるように工夫し，さらに未対応の脆弱性に対してはチケット ID を辿れるようにした．

2.6 ユーザテスト

一通り準備ができた段階で，システムの問題点の洗い出しのために，一部のユーザに対して試験的に利用してもらう 2 週間程度の試験期間を設けた．実際に数カ所の問題の発見とマニュアルの改善すべき点がフィードバックされ，本運用に入る前に修正することができ大変有意義であった．

3. 移行を終えて

3.1 課題の総括

1.2 で挙げた課題について簡単に総括する．課題 (1) については，業務フローとして既に整備されていた認証情報を

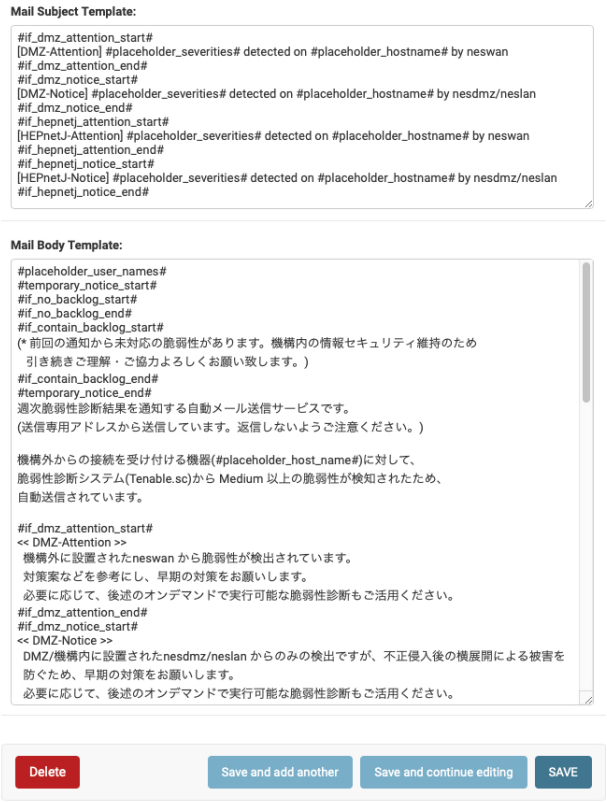

図 5 通知メールのテンプレート編集画面．管理者，機器，脆弱性内容のプレースホルダーを用意し，そこに自動的に該当するデータを挿入する仕組み．特定の条件下のみメッセージを表示させる仕組みも用意した．

Fig. 5 Edit screen for email template of notification. Placeholders for administrators' name, their device, and vulnerability details can be used in the template. The system automatically inserts the relevant data. It is also possible to display messages only under specific conditions.

活用することで，Tenable.sc の LDAP 認証の仕組みを利用することができるようになった．結果として，Tenable.sc の Web UI 用のユーザアカウントの認証情報を個別に配布する必要性自体がなくなり，課題解決となった．課題 (2) については，機器管理者に対応する全ユーザアカウントに対し，スクリプトを用いて脆弱性診断に必要となる事前設定をまとめて行った．ユーザの手間を減らすと同時に設定ミスのリスクを抑えることができ，課題解決となった．また，機器管理者のワークフローを Tenable.sc の Web UI にログインしボタンを押すだけ，というように単純化することで，結果的にマニュアルの簡略化につながり，課題 (3) も容易に解決した．課題 (4) についてもスクリプトベースの週次メール通知システムを用意することで対応することができた．最終的にできあがったシステムの各コンポーネントの関係性を図 6 に示す．

3.2 本運用開始後の状況

事前のユーザテストのおかげもあり，本運用開始後は Tenable.sc の Web UI についての問い合わせはほぼなく，思いのほかスムーズに移行することができた．また，移行前には通知対象外であった Medium 脆弱性に対しても週次ベースで通知するようになり，Medium 脆弱性に対しても即時対応する環境が整った．

3.3 Tenable.sc について

すでに述べてきた通り，KEK における当初の懸念事項であった

(1) 診断結果の記述のわかりにくさ
(2) 診断の進捗状況が把握できないこと

が解決されている．これにより，独自 UI サーバである DMZ User's Portal を欠いても自己点検のワークフローを回すことができた．また，Web API も提供されているため，必要に応じて脆弱性の週次メール通知のしくみを構築できる柔軟性も備えている点も好都合であった．

3.4 Django Web フレームワークについて

今回は以下の点でスクリプトを補助する目的で利用した：

(1) データベースをプログラミング言語上のオブジェクトにマッピングする ORM が利用できる，
(2) データベースの管理サイト (Django Admin Site) の自動作成．スクリプトの動作を GUI で設定できるようにするために利用した．

スクリプトの動作を GUI で変更できることは，システムの安定運用上重要な点で，通知メール文の変更や機器情報などの変更には，スクリプトを理解する必要や編集する必要はなく，開発者以外でも設定変更が可能となっている．また，Django Admin Site をカスタマイズし，データベース情報を一括修正もできるようにした．例えば，データベースの管理サイト上でフィルターを利用して特定の機器のリストを抽出することができ，さらにそのリストの機器に対して，脆弱性の非通知登録や機器の登録・廃止などを一括して操作ができるようになっている．

4. まとめと今後の課題

これまで KEK の DMZ 脆弱性自己診断の環境構築を支えてきた DMZ User's Portal から Tenable.sc Web UI を軸に据えたワークフローに移行し運用開始した．DMZ User's Portal の一機能として存在した，脆弱性の週次メール通知のしくみは継続，別途再構築し，Django Web フレームワークを活用して実装した．今回，スクリプトの内部を知らずとも運用できるようにシステムを構築したが，想定外の問題が起きた場合にはスクリプトを修正する必要があり，その意味では開発者への依存性は完全に取り除かれているわけではない．運用安定性を考えた場合，技術継承を含めた

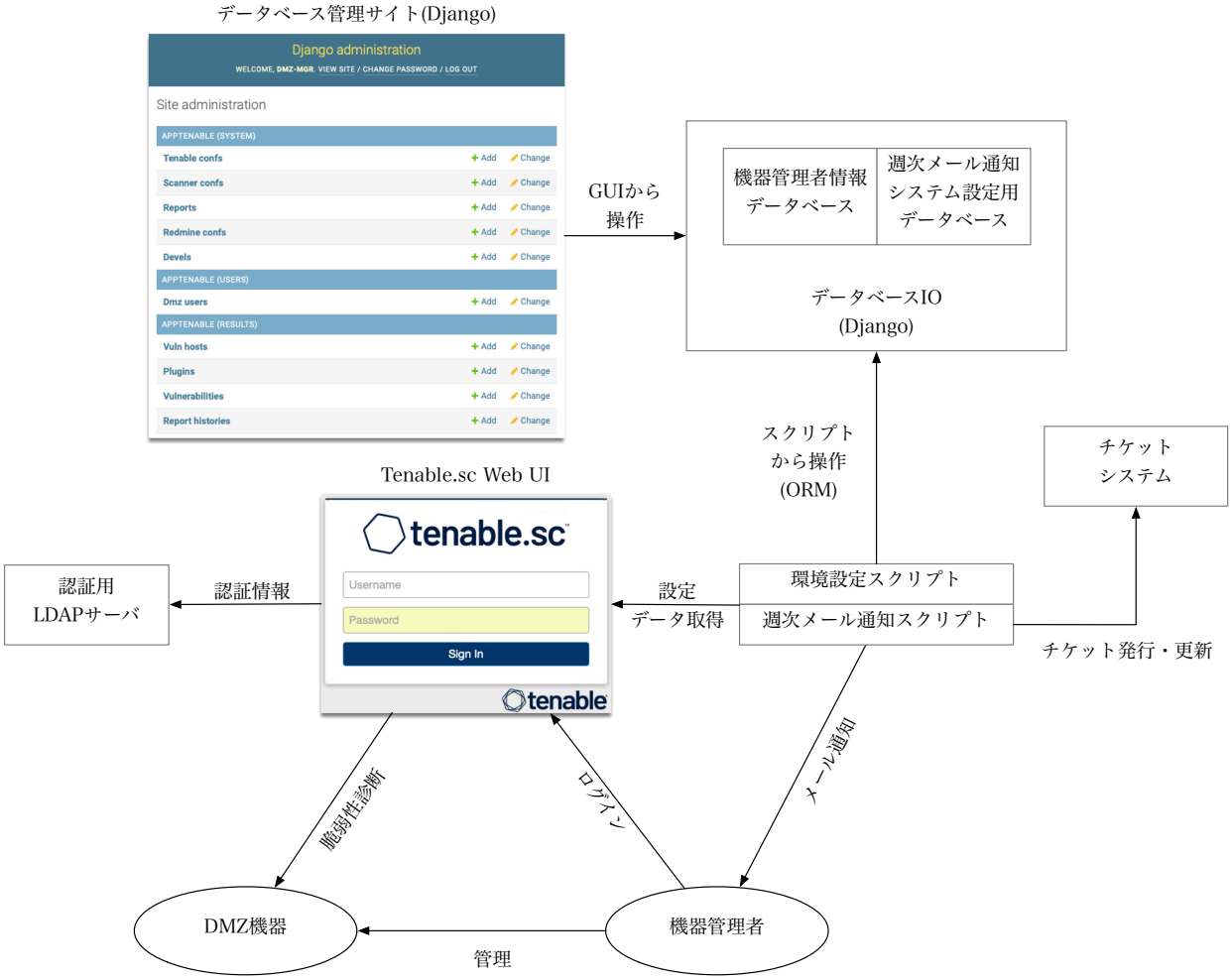

図 6 各コンポーネントの関係性
Fig. 6 Relationship between each component

システムの維持管理を組織的に行うことが必須であり，それが困難である場合はクラウドサービスへの移行を有力な手段として検討すべきと思われる．

謝辞 本稿のシステム移行において，助言・技術協力を頂いた KEK セキュリティグループの皆様に感謝の意を表します．また，機器管理者の皆様のご理解・ご協力のおかげでスムーズな移行が実現しました．厚く御礼申し上げます．

参考文献

[1] 村上 直，湯浅富久子，金子敏明：DMZ ネットワークのサーバ管理者自身による脆弱性診断，インターネットと運用技術シンポジウム 2016 論文集，Vol. 2016, pp. 41–48 (2016).
[2] : django The web framework for perfectionists with deadlines, , available from ⟨https://www.djangoproject.com⟩ (accessed 2022-04-04).
[3] : Tenable.sc API: Overview, , available from ⟨https://docs.tenable.com/tenablesc/api⟩ (accessed 2022-04-04).