AIによるセキュリティ情報解析支援システムの提案

竹原 一駿[1] 檜垣 龍徳[1] 本部 建大[1] 楠目 幹[1] 西岡 大助[1] 西山 賢[2] 合田 翔[2] 最所 圭三[1]
喜田 弘司[1]

概要：近年，企業をはじめとした組織へのサイバー攻撃は日々高度化し，セキュリティ管理者をはじめとしたセキュリティ人材への負担が高まっている．セキュリティ管理者は，サイバー攻撃に対するリスク評価と戦略の策定のために，“表層解析作業” を行う必要がある．この作業は，日々 Web ページなどの情報源から多種多様に公開されるセキュリティ情報を収集し，類似する情報をグループ化した後に，内容の精査や確認が必要であり，セキュリティ管理者に多大な負担をかけている．特に日本では，セキュリティ人材は不足しており，表層解析作業の負担を減らさなければ，サイバー攻撃に十分に対応できない．そこで我々は，セキュリティ管理者の表層解析作業における負担を軽減することを目的とした，AI によるセキュリティ情報解析支援システム “CSICOS” を開発している．CSICOS では，情報源となる大量の Web ページからセキュリティ情報を収集し，正規化を経て AI を用いてグループ化やタグの付与を行い，セキュリティ管理者に整理された情報を提供する．本論では，CSICOS の提案，設計，実装と，（株）STNet で行った試作版の評価実験の結果，得られた課題について述べる．

Proposal of Security Information Analysis Support System by Artificial Intelligence

Ichitoshi TAKEHARA[1] Tataunori HIGAKI[1] Tatsuhiro MOTOBU[1] Motoki KUSUME[1]
Diasuke NISHIOKA[1] Masaru NISHIYAMA[2] Shou GOUDA[2] Keizo SAISHO[1] Koji KIDA[1]

1. はじめに

近年，コンピュータの発達によりサイバー攻撃は日々高度化している．情報通信白書によると，日本ではサイバーセキュリティ人材は圧倒的に不足しており，特に「セキュリティ戦略・企画を策定する」人材が不足していると述べられている [1]．企業をはじめとした組織のサイバーセキュリティ人材（セキュリティ管理者）は，リスク評価と戦略の策定のために，組織に関するセキュリティ情報を調査する “表層解析作業” が発生する．表層解析作業とは，セキュリティ情報を様々な粒度や情報源によって収集し，組織に関連する情報を抽出，類似する情報のグループ化を行い，組織のセキュリティ管理者が内容を精査，確認する作業である．表層解析作業には以下の 3 つの課題があり，セキュリティ管理者の大きな負荷となっている．

情報の収集における課題（①）： セキュリティ管理者は，多種多様で Web ページを始めとした情報源で公開されている膨大なセキュリティ情報を，横断的に収集しなければならない．それらから日々，収集するページは 1 日あたり 1 万件に上り，非常に手間がかかる．さらに，Web ページには，以下の 3 パターンがあり，それぞれ異なる手法で収集しなければならない．

RSS 有

JPCERT など RSS として配信している Web ページ

RSS 無

個人ブログなど RSS を配信していない Web ページ

Twitter

Twitter の単一アカウントや，提供されるリスト機能による複数アカウントから投稿される内容

特にパターン（ RSS 無 ）では，その日の最新情報を毎日同じ URL で公開している場合がある．この場合，最新情報を公開している Web ページを解析し，各ページを収

1 香川大学
2 （株）STNet

集しなければならない．

さらに，キーワードにて記事を全文検索できる必要がある．たとえば，セキュリティ管理者が，個別のアプリケーションのセキュリティ情報を調べる場合に必要となる．大量の Web ページに対して横断的に検索を行う必要があり，大きな負担となる．

記事の解析における課題（②）： 収集した多種多様な情報を解析する．複数の情報源から入手した情報の中には，組織と関係の薄い情報や重複，類似した情報が含まれていることがある．セキュリティ管理者には，これらの情報の取捨選択や，関連する情報を探す手間が発生する．また，大量の情報の中から，緊急性や重要性が高い情報を狙って得ることは困難である．

情報の閲覧における課題（③）： セキュリティ管理者は，組織に関係する情報を優先的に閲覧する．しかし，どの記事が組織に関係するのかを判断する手間が発生する．複数の Web サイトで同じ情報が公開されていた場合でも，セキュリティ管理者は Web サイト毎に情報を確認する必要がある．特定のキーワードを検索する場合も，Web サイト毎に検索しなければならない．このとき，Web サイト毎で縦断的にではなく，複数の Web サイトで横断的に情報を閲覧できることが求められる．

本研究では，以上の課題を解決することで，セキュリティ管理者の負荷を減らすことを目的とした，AI によるセキュリティ情報解決支援システム “**CSICOS**” を開発している．本システムでは，セキュリティ管理者に対し，定期的に Web 上のセキュリティ情報の収集と解析を経て，整理された情報を提供する．

本論では，CSICOS の提案，設計，試作，（株）STNet にて行った評価実験，それにより判明した今後の課題について述べる．2 章にて，システムの全体構成について述べる．3 章～5 章にて，課題① ～ ③ の解決手法についてに述べる．6 章にて，（株）STNet で行った評価実験について述べる．7 章にて，本論のまとめと関連研究を述べる．

2. システム構成

開発している CSICOS の全体構成について述べる（**図 1**）．本システムは，課題①を解決する収集部，課題②を解決する解析部，課題③を解決するフロント部，それらの結果を共有するデータベースで構成されている．

本システムにおける情報収集，収集した情報解析，解析した結果をセキュリティ管理者へ提示する流れを述べる．まず，収集部にて，Web ページにて公開されているセキュリティ情報を収集する．それらの収集した情報は 1 ページを 1 つの「記事」として，データベース: “sectySystem” に格納される．次に，解析部にて，“sectySystem” から収集した記事を取得し，解析を行う．解析結果は，データベース: “analyzedDB” に格納される．最後に，フロント部の API サーバにて “analyzedDB” より解析した結果を取得し，Web ページによりセキュリティ管理者に提示する．このとき，フロント部の機能であるブックマークの情報などは，データベース: “bookmarkDB” に格納する．

各部は，データベースでは MySQL，収集部と解析部では Python，フロント部の API サーバには Node.js，Web ページの描画には Vue.js で実装されている．これらはコンテナ型仮想環境 Docker 上で動作する．

3. 収集部

3.1 概要

課題（①）を解決する，収集部について述べる．様々な Web ページから記事を収集し（3.2 節），データベースへ格納する（3.3 節）．このとき，全文検索に対応する（3.4 節）．

情報源として収集する Web ページは，1 章で示したとおり 3 パターンに分類できる．参照先の Web サイトは予めテーブル: “crawl_list” に記録しておく．それぞれのパターンに従って，（ RSS 有 ）は RSS の URL，（ RSS 無 ）は Web サイトのトップページ，（ Twitter ）の場合はアカウント名やリスト名を記録する．これらの情報を参照することで記事を収集し，ユニークな記事 ID を割り当て，データベースに記録する．さらに，記事を全文検索するためのインデックスを付与する．

実現における課題としては以下の 3 つがある．

各記事のリンクの取得： パターン（ RSS 無 ）において，ブラウザでアクセスしたときに JavaScript が動作して，始めて記事へのリンクが表示される Web サイトがある．

記事の重複や収集漏れ： 記事には，同じ URL でも前日とは違う情報が公開されている場合や，違う URL でも同じ内容の記事を収集する場合がある．たとえば，“`www.hostname.com/today.html`” という URL でその日の最新情報を載せているときである．記事の重複や収集漏れを防がなければならない．

全文検索： 記事の 2-gram へ分割，インデックスの付与を行わなければならない．また，一般的に全文検索の実行時には，時間がかかる．

3.2 記事の収集

パターン（ RSS 有 ）は，RSS から各記事の URL を収集して，1 記事毎にクローリングする．

パターン（ RSS 無 ）は，セキュリティ管理者が Web サイトごとに個別のプログラムを作成する．データベースへの記録などの動作はクラスにより実装されているため，記事の URL を連続的に取得する動作箇所のみを作成する．課題（各記事のリンクの取得）は，Docker コンテナ内で Google Chrome を起動し “chromedriver”[2] により，コン

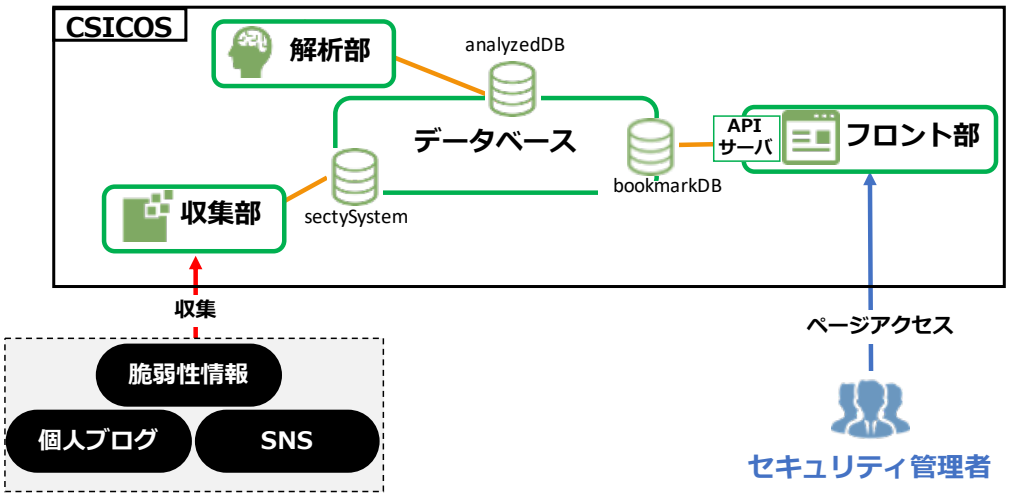

図 1　システムの全体構成
Fig. 1　Overall System Configuration.

表 1　テーブル:crawl_data
Table 1　Table:crawl_data.

項目	カラム名	型	制約
記事 ID	Info_ID	INTEGER	主キー
記事の実データの記録パス	File_PATH	VARCHAR	
記事のハッシュ値	File_HASH	CHAR	
記事のトップページ	ListID	INTEGER	“crawl_list” の ID を参照
収集元である記事自体の URL	Page_URL	VARCHAR	
収集日時	Got_DATE	DATETIME	

テナ内でブラウザを動作させることで URL を取得する．

パターン（ Twitter ）は，Twitter API を用いて収集する．単一アカウントの場合は “`statuses/user_timeline`” ，リストの場合は “`lists/list`” にアクセスすることで，収集する．また，短期間かつ大量に投稿を収集すると，API のアクセスに制限がかかる．その場合は，5 ～ 10 分程間をあけてアクセスする．

3.3　記事の記録

収集した記事は，1 記事ずつ付加情報（記事の ID，収集日，記事自体の URL，ハッシュ値など）をテーブル: “crawl_data” （**表 1**）に，小文字に変換した記事をテーブル: “full_textdata” に，記事自体を実ファイルにそれぞれ記録する．小文字に統一して記録することで，テーブル: “full_textdata” を用いた全文検索における，大文字小文字の区別を防ぐ．記事は，テーブル: “crawl_list” から外部参照されており，検索結果の記事毎に収集源を特定できる．

課題（記事の重複や収集漏れ）は，記事毎にハッシュ値を記録する．たとえば，過去に収集した記事 A と新たに収集した記事 B のハッシュ値が衝突した場合，収集した古い記事 A を破棄し，新しい記事 B を挿入することで，最新の収集日や ID を記録する．

3.4　全文検索の高速化

課題（全文検索）は，小文字に変換し記録した記事に対して，検索を行うことで実現する．一般的にデータベースに記録されているテキスト情報の全文検索は，SQL による LIKE 句を用いた文字列検索にて実現することが多い．しかし，LIKE 句による検索は非常に時間がかかる．そこで，収集した記事の記録時に MySQL の機能である “FULLTEXT” インデックスを付与する [3]．これにより，記事の 2-gram で分割された結果を同時に記録でき，高速な検索を行える．また，ストレージエンジンを MySQL 標準の InnoDB から Mroonga[4] に変更し，さらなる検索の高速化を実現する．28 万件の記事に対する検索時間を，**表 2** に示す．2 単語の検索ワードは AND 検索である．

4.　解析部

4.1　概要

課題 ② を解決する，解析部について述べる．解析部では，前処理（4.2 節）として収集記事からタイトルと本文を抽出する．組織にとって必要な情報を内容の類似した記事で集約し，重要度をつける．これにより，多くの情報から求める情報を探す手間を低減する．

タグを用いて，組織にとって必要な情報を抽出する（4.3

表 2　InnoDB と Mroonga の検索時間の比較
Table 2　Comparison of Search Times in InnoDB and Mrronga.

検索ワード	記事の件数	InnoDB		Mrronga	
		LIKE 句（sec）	FULLTEXT（sec）	LIKE 句（sec）	FULLTEXT （sec）
password	25225	31.75	30.33	35.47	0.49
脆弱性	69985	30.58	2.23	35.18	0.02
microsoft java	74403	16.75	64.40	19.54	0.86
脆弱 サーバ	23303	43.03	1.43	49.47	0.02

表 3　テーブル:analyze_data
Table 3　Table:analyze_data.

項目	カラム名	型	制約
解析番号	analyze_id	INTEGER	主キー
記事 ID	article_id	INTEGER	“crawl_data” の “Info_ID” を参照（表 1）
前処理ファイルの ID	normalize_id	INTEGER	“normalization_data” の ID を参照
記事のタイトル	title	VARCHAR	
グループ ID	similar_id	INTEGER	
記事の特徴的なタグの ID	article_primary_tag_id	INTEGER	“tag_data” の ID を参照
グループの特徴的なタグの ID	primary_tag_id	INTEGER	“tag_data” の ID を参照
記事のスコア	score	FLOAT	

節）．前提条件として，セキュリティ管理者は組織に関連する製品やベンダ名，それらに関する脆弱性情報があれば，必要な情報を手間なく探すことができることとする．一例として，組織で使用されている Web ブラウザが “Firefox” のみである場合，“Firefox” は必要な情報であるが，他のブラウザである “Internet Explorer” や “Google Chrome” は不要な情報となる．製品名やベンダ名の情報は，過去の脆弱性情報がまとめられた JVN[5] より，API を使用することで取得できる．これらの情報をタグや解析における特徴量として利用し，記事とタグを紐付けることで必要な情報を抽出する．

抽出した情報を集約する（4.4 節）．必要な情報を抽出するだけでは，その中で類似した記事を何度も閲覧してしまう．このような事象は，特定のある内容に関する記事ごとに，1 つのグループとすることで避けることができるといえる．そこで，収集記事をタグの出現の傾向の近いものをまとめることで，内容の類似した記事をグループ化する．

記事に対して，重要度を決める（4.5 節）．グループの中で，どの記事が最良の記事であるか判断することは困難である．そこで，各記事に評価値を付与することで判断する．集約された情報が多いものほど，注目度の高い重要な内容であるとして高い評価値を付ける．

これらの処理の結果は，解析結果を管理するテーブル: “analyze_data”（**表 3**）に記録する．補助的に，“analyze_data” 以外にタグを管理するテーブル: “tag_data”，収集記事とタグの関連を管理するテーブル: “link_artcle_tag”，前処理結果を管理するテーブル: “normalization_data” をそれぞれ用いる．

4.2　前処理

前処理では，収集記事からのタイトル及び本文の抽出を行う．収集部では，Web ページや Twitter をデータソース（収集源）としていて，中でもパターン（ Twitter ）をデータソースとする記事が非常に多い．しかし，Twitter からのデータは 1 記事あたりの情報量が非常に少なく，パターン（ RSS 有 ），（ RSS 無 ）などの Web ページと同列に扱うことは難しいため，本論では除外する．そのため，本論で扱う記事は HTML や XML の形式のみである．

タイトルの抽出は，各記事の `<title>`タグにおけるテキストを抽出することで実現する．しかし，タイトルを抽出するだけでは，収集元のブログタイトルなど本文と関係のない情報が含まれる場合があった．そこで，収集元ごとに一貫している部分を除外し，それ以外を抽出したものをタイトルとして採用した．たとえば，「新型コロナで CIO が注意すべき IT 予算の使い方 - ZDNet Japan」[6] と，「NEC ソリューションイノベータ、RPA ロボット派遣サービスを開始 - ZDNet Japan」[7] では，“- ZDNet Japan” が共通している．そこで，共通した文字を除外した，「新型コロナで CIO が注意すべき IT 予算の使い方」，「NEC ソリューションイノベータ、RPA ロボット派遣サービスを開始」をタイトルとして採用する．本文の抽出は，`<header>`や`<navi>`タグなど，本文と関係が少ないタグで宣言されている内容を削除することで実現する．

ここで抽出した本文は，“normalization_data” に記録する．タイトルは収集記事の ID と前処理結果の ID と紐付けて，“analyze_data” に記録する．

4.3 情報の抽出

前処理をしたデータを元に，必要な情報の抽出について述べる．本システムでは，組織に関連する機器やソフトウェアの，セキュリティ脆弱性に関する情報を得ることを目指す．そのため，JVN で確認できる脆弱性情報やベンダ名及び製品名，組織で使用されている機器やソフトウェアをタグとして利用する．前処理をしたデータに対して形態素解析を行い，そこで出現したタグがあるか確認する．タグを記事と紐付け，データベースに格納する．これにより，必要な情報をタグとして記事を抽出する．

4.3.1 辞書の更新

形態素解析を行う上で，MeCab[8] に既に登録されているシステム辞書では，検出できない固有名詞がある．一例として “情報処理推進機構” がある．MeCab のシステム辞書では，“情報処理”，“推進”，“機構” の三単語として検出される．この原因は，辞書内で設定されている語が意図しない優先度であることや，検出したいタグが辞書に含まれていないことである．そのため，JVN から収集したタグを辞書として追加し，コストを低くすることで優先度を高く設定した（**表 4**）．

表 4 辞書として追加したタグ（抜粋）
Table 4 Tags Added as a Dictionary.

表層形 (単語)	コスト	品詞	分類 1	分類 2	···
シーサート	1	名詞	固有名詞	一般	···
X.509	1	名詞	固有名詞	一般	···
標的型攻撃	1	名詞	固有名詞	一般	···
port scan	1	名詞	固有名詞	一般	···
情報処理推進機構	1	名詞	固有名詞	一般	···

4.3.2 タグの検出

形態素解析の結果からタグを検出する際に，空白を含むタグを見落としていた．形態素解析の段階で，空白を含む語を一語として認識できないことが原因である．そのため，形態素解析の結果を数語ずつ空白区切りで結合し，検出できる形の結果を作成した．タグの出現が確認された場合，収集記事の ID とタグの ID を紐付けて，“link_artcle_tag” に記録する．このとき出現しなかったタグは，解析において特徴量として使用しない．

4.4 記事の集約

前処理にて抽出した本文から学習データを作成した上で解析し，内容が類似する記事を集約する．本システムでは，収集部にて記事の収集を毎日行うことを想定しており，将来的に収集記事が数万件に上ることが考えられる．そこで，グループ化の際には大規模なデータを扱うのに適していると言われる “MiniBatchKMeans” 用いる [9]．“MiniBatchKMeans” 法とは，“KMeans” 法のように所属するグループを 1 データごとに計算するのではなく，一定数のデータごとに計算する手法である．グループ化する際に使用する特徴量には，情報の抽出（4.3 節）において用意したタグを利用する．

4.4.1 記事の選定

収集した記事は，次回の収集時に収集源が同じ Web ページであっても，記録するハッシュ値がアクセスカウンターの更新など僅かな差でも異なり，別の記事だと識別される．そのため，今までに収集した全ての記事をグループ化の対象とすると，ほとんど内容の変わらない収集記事が同一グループに多数含まれてしまう．記事の内容は，ハッシュ値が異なっていてもタイトルが同一であれば，同じ記事であるか，前回の収集後に更新されたと考えられる．そこで，タイトルの同一な複数記事の中で，収集日時が最新の記事をグループ化の対象とした．

4.4.2 学習データの作成

前処理データに対して形態素解析を行い，学習データを作成する．形態素解析の結果から，タグそれぞれの出現回数を要素とする，収集記事の ID が行，タグが列であるデータフレーム（**表 5**）を作成する．たとえば，1 行目 1 列目では，ID が 1 である収集記事の中に，“Firefox” の出現回数が 3 であることを示している．このデータフレームを元に，tf-idf 値を計算し，特徴量に傾斜をつけたものを学習データとして解析で用いる．tf-idf とは，あるデータの特徴の中で，全データを通して多く出現している特徴は価値を下げ，あまり出現していない特徴は価値を上げるというものである．

表 5 データフレーム
Table 5 dataframe.

記事 ID \ タグ	Firefox	Java	···	Microsoft
1	3	2	···	0
2	0	5	···	0
⋮	⋮	⋮	⋱	⋮
n	1	1	···	4

4.4.3 グループ化

作成した学習データを，グループ化する．まず，グループ化の対象収集記事の中でいずれのタグとも紐付けられていない記事を，1 グループにまとめる．次に，それ以外の記事を “MiniBatchKMeans” 法によってグループ化する．このときグループ数は，均等に分割された場合に，1 グループあたり 7 件程度の記事が所属するような値に設定した．グループ化の結果は “analyze_data” に記録する．

4.5 重要度の算出

グループ化した各グループ及び記事に対して，最も特徴的なタグと重要度を算出する．まず，グループの重要度を

算出するために，各グループを所属するデータ数で比較する．データ数が多いグループほど，グループの特徴的な内容が一般に重要なものであり，広く普及していると考えられるため，重要度を高く設定する．

次に，グループの最も特徴的なタグを求める．各グループは行を所属する記事の ID，列を特徴量，要素を特徴量の tf-idf 値としたデータ構造を持つ．**表 6** には例として，規模を縮小したものを示している．グループ内での各特徴量の tf-idf 値の平均を求め，平均値の最も高かった特徴量をそのグループにおける最も特徴的なタグと位置付ける．たとえば，表 6 では，それぞれの tf-idf 値の平均が，“Firefox” が 0.611，“Java” が 0.196，“Microsoft” が 0.055 となる．そのため，“Firefox” が最も特徴的なタグとなる．グループ内に所属する記事ごとの重要度は，所属するグループの最も特徴的なタグの tf-idf 値が高いものほど，そのタグについて多くの言及がなされていると考えられるため高く設定する．

また，各記事の特徴量の tf-idf 値は，記事における各タグの重要度として位置付けて，最も tf-idf 値の高い特徴量をその記事における最も特徴的なタグとした．タグの重要度を “link_artcle_tag” に，それ以外のデータを “analyze_data” にそれぞれ記録する．

表 6 グループのデータ構造
Table 6 Data Structure of the Groups.

記事 ID ＼ タグ	Firefox	Java	Microsoft
1	0.632	0.227	0
7	0.594	0.113	0.145
33	0.606	0.247	0.02

5. フロント部

5.1 概要

課題③を解決する，フロント部について述べる．データベースに格納された解析結果を，セキュリティ管理者に表示する．

多くのセキュリティ管理者は，保有している情報源を，1つずつ順番に閲覧している．課題③で述べたように，従来の表層解析作業は手間がかかり，無駄な時間が多く発生する．本システムのフロント部では，複数の Web サイトの情報を横断的に提示することで，その課題を解決できる．

図 2 に示すように，複数の Web サイトの情報を横断的に閲覧するため，“機器・構成” ・“キーワード” ・” 後で” の 3 つのタブを実装した．“機器・構成タブ”（5.3 節）では，あらかじめ登録した機器・構成情報に基づき，必要なセキュリティ情報を持つ記事のタイトルを確認できる．日々の表層解析作業において，複数の Web サイトを横断的に，記事の内容が重複せず，情報を閲覧できる．“キーワードタブ”（5.4 節）では，タグ検索と全文検索ができ，その検索結果を確認できる．収集した全ての Web ページに対し，入力したキーワードで検索できる．“後でタブ”（5.5 節）では，ブックマークに登録しておいた記事を確認できる．複数の Web サイトで別々に存在している情報を，一度に保存し閲覧することができる．これらのタブから，詳細ページ（5.6 節）に遷移することで，類似する記事や付与されているタグを確認できる．以上のタブを活用することにより、セキュリティ管理者の手間を減らすことができる．

解析結果は，API サーバを経由して取得する．API サーバでは，フロントからのリクエストに応じて SQL を発行し，データベースを操作する．

5.2 使用方法

各タブでは，それぞれの条件に合わせて，レコードが一覧で表示される．1 レコードごとに，ブックマークボタン，記事の収集日時，Web ページのタイトル，その Web ページに最も重要度の高いタグ，類似記事の数を表示する．ページタイトルをクリックすると，詳細ページ（5.6 節）に遷移し，より詳細な情報を表示する．一度クリックしたページは，レコードを薄い灰色で表示し，既読であることが一目でわかる．既読情報はデータベースに格納され，次回のアクセス時も反映される．レコードの情報は，ページを表示する際に API サーバにアクセスし，上部から 1 ページの件数分を取得して表示する．

画面下部には，next ボタンや prev ボタンを配置している．next ボタンをクリックすると次のページに進み，prev ボタンをクリックすると前のページに戻る．これらのボタンがクリックされたときに，該当ページのデータを取得する．

5.3 機器・構成タブ

“機器・構成タブ”（図 2）について述べる．自分の組織に，関係する情報を表示する．類似する内容が含まれている記事は 1 つのグループに集約され（4.4 節），グループ内で最も重要度の高い記事を代表として，表示する．1 レコードに 1 グループの代表を表示する．

5.4 キーワードタブ

“検索タブ”(**図 3**) について述べる．タグ検索では，検索ボックスに入力されたタグが，紐付いている記事を検索する．空白区切りで複数のタグを入力することで，AND 検索ができる．全文検索では，検索ボックスに入力されたキーワードが，タイトルや本文に含まれている記事を検索する．タグ検索と同じく，AND 検索ができる．全文検索における検索対象の記事は，全て小文字に変換して記録されて

図 2　機器・構成タブ
Fig. 2　Information Tab.

いる（3.3 節）．そこで，キーワードも小文字に変換されて検索される．空白を含む単語は，ダブルクォーテーションで囲むことで，一単語として検索される．たとえば，全文検索で「“Apache Tomcat” https」と入力すると，“apache tomcat” と “https” を両方含む記事を検索する．

5.5　後でタブ

“機器・構成タブ” などの各タブでは，レコードごとにブックマーク機能を提供している．“後でタブ”（図 4）では，記録したレコードの一覧を見ることができる．

レコードの左側にある登録ボタン（☆）を押すと，ブラウザから API サーバにリクエストを送り，テーブル: “bookmark_data” に該当記事の ID（表 1 の記事 ID）が記録される．同様に，レコードの削除ボタン（×）を押すと，ID が削除される．ブックマークした記録はリアルタイムで反映され，“後でタブ” に反映される．次回アクセス時は，テーブル: “bookmark_data’ からブックマークが読み込まれ，“後でタブ” に反映される．

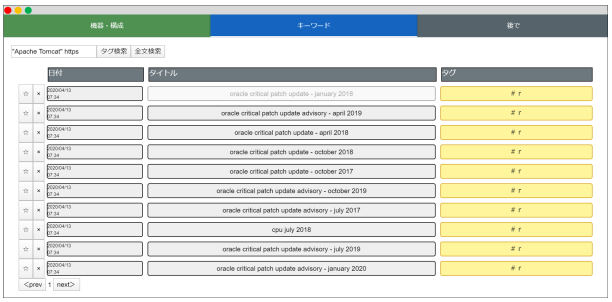

図 3　検索タブ
Fig. 3　Search Tab.

図 4　後でタブ
Fig. 4　Bookmark Tab.

5.6　詳細ページ

詳細ページ（図 5）について述べる．詳細ページには，“機器・構成タブ” や “キーワードタブ” などのレコードに表示される，記事のタブをクリックすることで，遷移し，類似記事，タグや Web ページが確認できる．その記事の情報を，より詳しく知りたい場合に利用する．類似する記事を一覧で表示することで，他の Web ページの記事を一度に確認することができる．

左側に，類似記事を表示する．各タブと同様に，レコードが一覧で表示される．1 つのレコードには，ブックマークボタン，データの取得日時，記事のタイトル，最も重要度の高いタグが確認できる．

右側に，上部にタグと下部に Web ページを表示する．タグの数が多く，画面からはみ出す場合は， 左右の矢印をク

図 5　詳細ページ*1
Fig. 5　Detail Page.

*1　参考文献 [10] より引用

表 7　選択式アンケートの内容

Table 7　Contents of Selective Questionnaire.

① 情報収集時間は短縮されましたか？	⑩ 他の利用者 (セキュリティ運用者) にシステムを勧められますか？
② 分類された情報は，類似していると思いましたか？	⑪ 操作における導線はわかりやすいと思いましたか？
③ トップページ (機器・構成) にて提供された情報は得たい情報でしたか？	⑫ ページのプレビュー機能は役に立ちましたか？
④ タグ検索結果は意図したものでしたか？	⑬ 付与されているタグは意図したものでしたか？
⑤ キーワード検索結果は意図したものでしたか？	⑭ 画面のレイアウトは分かりやすいと思いましたか？
⑥ ブックマーク機能は役にたちましたか？	⑮ 情報の並び順 (ソート) は意図したものでしたか？
⑦ 記事詳細ページにて求めていた記事は得られましたか？	⑯ 必要な機能は十分にあると思いますか？ (機能性)
⑧ 検索処理にかかる速度は許容できるものでしたか？	⑰ 処理の重さなど，本システムは常用できそうですか？ (効率性)
⑨ 本システム全体を通して，無駄なく目的の情報にたどり着けましたか？	

表 8　自由記述式アンケートの内容

Table 8　Contents of Free Description.

Ⓐ どの箇所に特にどのような不満を感じましたか？ (データの収集手法や収集元, 解析結果，ブラウザ上の描画)
Ⓑ 本システムをどのように改良するべきだと思いますか？

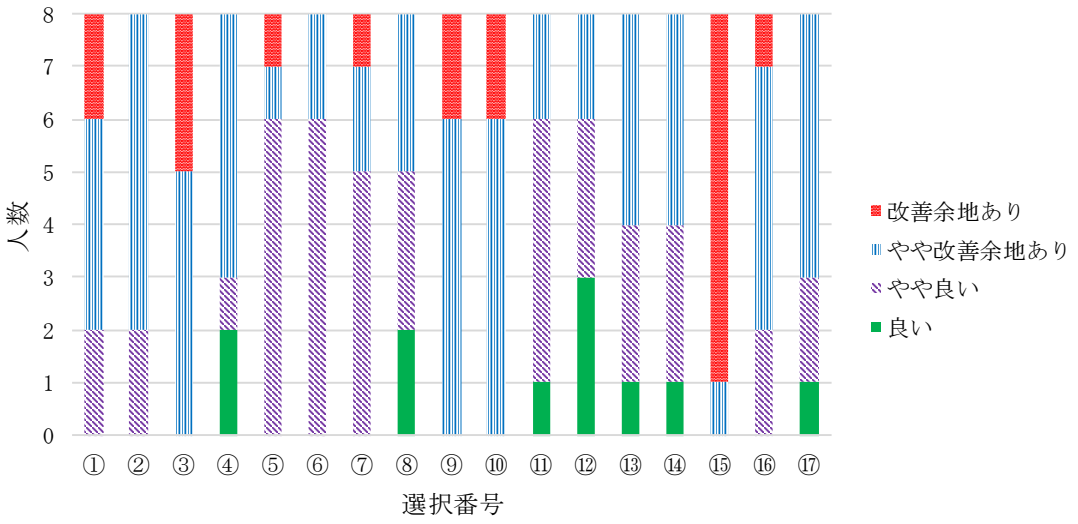

図 6　選択式アンケートの結果

Fig. 6　Results of Selective Questionnaire.

表 9　自由記述式アンケートの結果（抜粋）

Table 9　Results of Free Description.

Ⓐ	数カ月以上前の記事がトップに多く表示されているように思うので、 もう少し新しい情報が知りたいと感じました。 同じソース元で集約されているものが多く、集約されているのは同じソースの過去記事が目立ちました。他のソースから集められているケースもあるので、惜しいなと思いました。
Ⓑ	・記事は新しい順に表示されて欲しいです。 ・記事自体の日付が一覧からは分からないので表示したほうがよいかと思います。

リックすることで表示する．下部では，iframe を用いて該当記事の Web ページを表示しており，閲覧することができる．

6.　評価実験

6.1　目的と方法

2020 年 4 月 13 日～4 月 17 日に，試作した CSICOS に対して，（株）STNet のセキュリティ管理者 8 人を対象に評価実験を行った．課題① ～ ③ が解決できているか，システムを通してセキュリティ管理者の負荷を減らすための機能は十分に実現されているか，不足している機能は何かを調査した．表層解析作業を，従来の手法で行った場合と CSICOS を用いた場合で比較した．情報源の収集は，1 日あたり 1 度しか行っていないため，前日の従来の作業結果と比較した．5 日間作業を続け，最終日に選択式（**表 7**）と自由記述式（**表 8**）のアンケートを行った．

6.2　結果

選択式における結果を**図 6** に，自由記述式における解答の抜粋を**表 9** に示す．

選択式では，全体的に多くの質問で，“やや改善の余地あり” や “改善余地あり” が目立った．特に，③より，セ

キュリティ管理者が得たい情報を提供できていないと言える．課題①に関しては，自由記述式より，同じ記事ばかり収集されているように見えるとの意見が多かった．課題②に関する②，③，⑨，⑯などは，あまり良い評価を得られていない．課題③に関する④，⑧，⑪～⑭など，比較的良い評価を得ているが，⑮の情報のソートは，7人が“改善余地あり”と回答している．

自由記述式では，不足していると感じる機能の要望が多かった．特に，記事の収集日ではなく，記事自体の公開日を表示してほしいという意見が多かった．また，組織の機器に関係なく，セキュリティ情報の傾向や流行などの“トレンド”機能が欲しいとの希望があった．

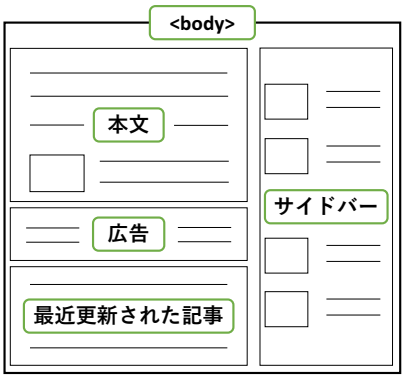

図 7　収集サイト概略図
Fig. 7　Overall of Crawl Site.

6.3　考察

①と⑨の結果から，現状では，本研究の目的であるセキュリティ管理者の負担を減らすに至っていないことがわかった．⑯にて機能面がまだ十分ではないという意見が多い．そこで，自由記述式の要望にあった，トレンド機能の試験導入の検討を決めた．トレンド機能を用いることによる，表層解析作業への影響を調査することにした．

結果を踏まえた，課題①～③に対する考察を述べる．

課題①： 同じ記事ばかり収集されているように見える原因は，実際にその情報源が1日毎に記事を追加する回数が多いためである．

記事の公開日の取得に関しては，記事のパターン（RSS有）やパターン（Twitter）では，RSSやAPIの取得時にメタデータとして記録されていることが多く，抽出は比較的手軽である．しかし，パターン（RSS無）の場合は，記事自体を解析し，公開日付を抽出しなければならない．さらに，公開日付のフォーマットは記事ごとに異なり，抽出が難しい．

全文検索に関しては，⑧より，十分な速度が出ていると判断できる．

課題②： 現状の解析手法としては，前処理において本文の抽出が十分にできていないことが考えられる．同一のWebサイトから収集している記事では，`<body>`タグの中に，本文以外に最近更新された記事などの要素を，共通して持つ場合がある（**図 7**）．そういった要素に付与されているタグが，グループ化に影響を及ぼしていることが考えられる．

主要なタグに関しても，④より意図したものが表示されていないことが多かった，という評価を得た．タグは，JVNから確認できる情報を全て用い，必要以上にタグ付けがされたことで，このような印象につながったのではないかと考えられる．

課題③： ユーザ認証の導入により，ユーザごとにブックマークや既読を分ける要望が多かった．レイアウトに関しても，⑭にあるとおり，改善の余地がある．たとえば，詳細ページで，類似記事のタイトルを表示する幅が狭いため，タイトルを確認しにくいという問題がある．類似記事の表示をドロワーにしたり，Webページの表示をポップアップにしたりすることで，表示領域を拡張することが求められる．

また，⑮から，解析結果のソートに関しては，厳しい評価を得た．現状のグループの大きさ降順ソートではなく，日付降順，同一日付の中のグループの大きさ降順のソートの方が良いという意見が主であった．

7.　おわりに

我々は，セキュリティ管理者の表層解析作業における負担を減らすことを目的として，AIによるセキュリティ情報解析支援システム“CSICOS”を開発している．本論では，システムを試作し，（株）STNetにて評価実験を行った．その結果，現状では，セキュリティ管理者の負担を減らすに至っていないことがわかった．今後は，課題①～③の解決手法の改良や不足している機能を追加し，さらなる評価実験を行う．

関連研究としては，香川大学の楠目らが開発しているシステム [11][12] や，広島大学の田島らが試作したシステム [13] がある．これらのシステムは，主にデータベース化された製品識別子“CPE”や共通脆弱性評価システム“CVSS”などの情報を用いて，組織にある機器の脆弱性を自動判定することを目標としている．CSICOSでは，ニュースサイトやTwitterをはじめとしたSNS，個人ブログなどの情報源から組織に関連する情報の抽出や解析を行い，セキュリティ管理者に提供する．また，AIの1つである機械学習を用いてグループ化することで，対策案が講じられていないセキュリティ情報に対しても，類似する事例を提案できる．

謝辞　本研究は（株）STNetとの共同研究として行っており，提案や実装，評価に至るまで多大な御協力の元，本論文という1つの区切りに辿り着きました．この場で，

厳しくも優しいアドバイスや評価をしていただいた（株）STNet の皆様に感謝致します．

参考文献

[1] 総務省：総務省 | 令和元年版情報通信白書 | サイバーセキュリティに関する現状と新たな脅威，総務省（オンライン），入手先 ⟨https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r01/html/nd113310.html⟩（参照 2020/04/28）．

[2] chromedriver users: ChromeDriver - WebDriver for Chrome, Google (online), available from ⟨https://chromedriver.chromium.org/⟩ (accessed 2020/04/28).

[3] MySQL: MySQL :: MySQL 5.6 リファレンスマニュアル :: 14.2.13.3 FULLTEXT インデックス，Oracle Corporation and/or its affiliates（オンライン），入手先 ⟨https://dev.mysql.com/doc/refman/5.6/ja/innodb-fulltext-index.html⟩（参照 2020/04/28）．

[4] mroonga: Mroonga - MySQL で高速日本語全文検索，Mroonga Project（オンライン），入手先 ⟨https://mroonga.org/ja/⟩（参照 2020/05/05）．

[5] JPCERT/CC: Japan Vulnerability Notes, JPCERT/CC (online), available from ⟨http://jvn.jp/⟩ (accessed 2020/05/06).

[6] Staff, Z. J.: 新型コロナで CIO が注意すべき IT 予算の使い方 - ZDNet Japan，ZDNet Japan Staff（オンライン），入手先 ⟨https://japan.zdnet.com/article/35153558/⟩（参照 2020/05/11）．

[7] Staff, Z. J.: NEC ソリューションイノベータ、RPA ロボット派遣サービスを開始 - ZDNet Japan，ZDNet Japan Staff（オンライン），入手先 ⟨https://japan.zdnet.com/article/35153546/⟩（参照 2020/05/11）．

[8] taku910: MeCab: Yet Another Part-of-Speech and Morphological Analyzer, taku910 (online), available from ⟨https://taku910.github.io/mecab/⟩ (accessed 2020/05/11).

[9] Sculley, D.: Web-Scale k-Means Clustering, *Proceedings of the 19th International Conference on World Wide Web*, WWW ’10，Vol. www2010, No. April26-30, New York, NY, USA, Association for Computing Machinery, pp. 1177–1178 (online), DOI: 10.1145/1772690.1772862 (2010).

[10] NEXT, S.: 【セキュリティニュース】オムロン製の制御システムソフト「CX-Supervisor」に 5 件の脆弱性（1 ページ目/ 全 1 ページ）：Security NEXT，ニュースガイア株式会社（オンライン），入手先 ⟨http://www.security-next.com/101833⟩（参照 2020-05-07）．

[11] 楠目幹，最所圭三，喜田弘司：脆弱性情報を利用したゼロデイ攻撃対策システムにおける DB 構築，第 81 回全国大会講演論文集, 4ZA-03，pp. 3–441–3–442 (2019).

[12] 楠目幹，喜田弘司，最所圭三：脆弱性情報を利用したゼロデイ攻撃対策システムにおける構成情報収集機能の実装及び脆弱性評価機能の設計，電子情報通信学会技術研究報告，Vol. 119, No. 140, pp. 1–6 (2019).

[13] 田島浩一，岸場清悟， 近堂徹，渡邉英伸，岩田則和，西村浩二，相原玲二：脆弱性診断と脆弱性情報公開サイトを用いた脆弱性更新通知機能の試作，マルチメディア, 分散協調とモバイルシンポジウム 2017 論文集， Vol. 2017, pp. 1661–1664 (2017).