Vol.2012-CSEC-57 No.20
Vol.2012-IOT17 No.20
2012/5/11

状況をつぶやくセキュリティシステム

糸魚川 竜士[1,a)]　大山 恵弘[1,b)]

概要：サーバのセキュリティログを頻繁に検査することはセキュリティの確保のために重要である．しかし，監視のための仕組みがなかったり貧弱であったりすると，管理者は外出時などにログを効率的に検査できない．そこで本研究では，サーバの監視状況をツイートとして Twitter に送信するセキュリティシステムを提案する．そのシステムのユーザは，サーバのログを様々な場所からいつでも検査できる．また，決まったコマンド文字列を含むツイートにより，ユーザはシステムの挙動を制御できる．現在，本システムは Apache Web サーバを対象としている．実験では，本システムが Web サーバの監視状況をツイートとして送信したことや，与えられたコマンド文字列に関係づけられた動作を実行したことを確認した．

キーワード：Twitter

A Security System that Tweets Monitoring Status

Ryuji Itoigawa[1,a)]　Yoshihiro Oyama[1,b)]

Abstract: Checking the security logs of a server frequently is critical to ensuring the security. However, the administrator cannot check the logs efficiently when he or she is out, if the monitoring facility is absent or poor. In this paper, we propose a security system that tweets monitoring status to the Twitter Web service. The user of the system can check the logs of a server at any time from various places. The user can also control the behavior of the system through tweets that include predefined command strings. Currently, the system works with the Apache Web server only. We conducted several experiments in which the system tweeted the monitoring status of a Web server and executed operations associated with a given command string.

Keywords: Twitter

1. はじめに

Web サーバなどのセキュリティが重要なサーバは，セキュリティシステムによって監視しながら実行することがある．セキュリティ確保のための一つの有効な対策は，セキュリティシステムが出力するログを頻繁に検査することである．しかし，外出先などでは，ログを外部から確認できる仕組みを整えていない限り，管理者はログを確認することができない．その結果，サーバが攻撃を受けていることに気づかないまま攻撃を受け続ける可能性がある．

そこで，本研究では，サーバの管理者が様々な場所からセキュリティログを監視できるセキュリティシステムを提案する．提案システムはログを Twitter に投稿し，管理者は Twitter にアクセスすることにより，ログを確認する．Twitter は個々のユーザが「ツイート」(tweet) と称される短文を投稿し，閲覧できるサービスである．Twitter は PC，スマートフォン，携帯電話などから使えるので，ネットワーク環境さえあればいつでもどこでもツイートを見ることができる．デバイスに合わせて様々な Twitter クライアントも存在しており，ユーザの好みに合わせて Twitter を快適に使うことができる．これらのことから，Twitter はセキュリティログを確認するツールとしても実用性が高い可能性があると我々は考え，その実用性を検証するため

[1] 電気通信大学
The University of Electro-Communications
[a)] itoigawa@ol.inf.uec.ac.jp
[b)] oyama@inf.uec.ac.jp

表 1 Twitter API の命令の例

Twitter API	命令の説明
`POST http://api.twitter.com/1/statuses/update.json status`	ツイートを行う
`GET http://api.twitter.com/1/statuses/user_timeline.json`	ユーザタイムラインの取得
`POST http://api.twitter.com/1/direct_messages/new.json userID&text`	DM の送信
`POST http://api.twitter.com/1/friendships/create.json userID`	ユーザのフォロー

の研究を行っている．

現在，提案システムは Apache Web サーバのログを監視することに特化している．提案システムは Apache が生成するアクセスログ（`access.log`）とエラーログ（`error.log`）を監視し，重要なログが出力されると，それを自身の Twitter アカウントに投稿する．また逆に，その Twitter アカウントに対してユーザがツイートを送ることにより，システムの動作を制御することができる．提案システムは Java で書かれた Twitter bot であり，Twitter4J という Twitter にアクセスするためのライブラリを用いて実装されている．

2. Twitter

2.1 概要

Twitter では個々のユーザが 140 字以内の文章であるツイートを投稿したり閲覧したりすることでコミュニケーションを行う．ツイートが時系列に並べて表示される画面をタイムラインと呼ぶ．タイムラインには 3 種類がある．世界中のツイートが表示されるパブリックタイムライン，指定したユーザによるツイートが表示されるユーザタイムライン，個々のユーザがフォロー（後述）している全ユーザの投稿が表示されるホームタイムラインである．

ユーザが自分のホームタイムラインにユーザを追加することをフォローと呼ぶ．個々のユーザがそれぞれ好きなユーザをフォローすることで，フォローしたユーザのツイートが自分のホームタイムラインに表示されるようになる．逆に，自分をフォローしているユーザをフォロワーと呼ぶ．

他のユーザに宛てた投稿をリプライと呼ぶ．「@ユーザ名 投稿内容」の書式のツイートは，そのユーザ名のユーザに宛てたツイートとみなされる．リプライは，送り主と送り先の両ユーザ以外の第三者も見ることができる．第三者が見ることができないツイートを他のユーザに送ることもできる．そのようなツイートはダイレクトメッセージ (direct message, DM) と呼ばれる．

ユーザのツイートなどを非公開にすることを，プロテクトすると言う．プロテクトされたユーザのツイートなどは，そのユーザのフォロワー以外には公開されなくなる．

2.2 Twitter API

2.2.1 概要

Twitter が提供するサービスは Web ブラウザから利用することも，Web ブラウザ以外から利用することもできる．Twitter 向けの Web API（Twitter API）が公開されている．デスクトップアプリケーションや，Twitter 以外の Web サイトが，この API を通じて Twitter のサービスを利用できる．提案システムも Twitter API を使用する．

2.2.2 OAuth 認証

Twitter API では OAuth 認証を使って認証を行う．OAuth 認証は，Web サービスで標準になりつつある認証方式で，Flickr，Twitter，Google などが 採用している．OAuth 認証は，ユーザが第三者アプリケーションにパスワードを渡すことなく，アカウントへのアクセスの可否を決定できる認証方式である．ユーザは，Twitter を利用する第三者アプリケーションの初回利用時などに，Twitter の Web サービスを通じて，自身の Twitter アカウントへのアクセスをアプリケーションに対して許可する．ユーザは Twitter を経由していつでも，アカウントへのアクセスを許可したアプリケーションの確認とアクセスの取り消しができる．

2.2.3 Twitter API の種類

Twitter API の種類は 4 種類に分けられる．REST API，検索 API，ストリーミング API，Web サイト向け API である．Twitter に対する負荷軽減のために，これらの API にはそれぞれ使用回数制限（レートリミット）が定められている．

提案システムは REST API と検索 API を用いている．REST API は Twitter が一番古くからサポートしている API である．タイムラインやソーシャルグラフなどを取得する参照系のメソッドや，ツイートやダイレクトメッセージを送信する更新系のメソッドが含まれる．**表 1** は REST API の中で，代表的な命令の例を示したものである．表内のように命令を記述し，送信することで，Twitter API を利用できる．

2012 年 3 月 31 日の時点では，参照系のメソッドのレートリミットは，認証なしで呼び出す場合は 1 時間当たり 150 回，OAuth で認証している場合は 1 時間当たり 350 回である [3]．更新系のメソッドにはレートリミットは設けられていない．しかし，一部の更新系のメソッドには個別に制限が設けられている．ツイートの投稿を行うメソッドでは，同じテキストを連続して投稿した場合には 2 つ目以降のツイートが無視される．また，ツイートの数は 1 日 1000 件に制限され，ダイレクトメッセージの数は 1 日 250 件に

表 2 Twitter4J の命令の例

命令	説明
updateStatus(status)	ツイートを行う
getUserTimeline()	ユーザタイムラインの取得
sendDirectMessage(userID,text)	DM の送信
createFriendship(userID)	ユーザのフォロー

制限されている．

検索 API はツイートを検索するための API である．様々な検索形式が用意されているので，それらを組み合わせることで柔軟な検索ができる．検索 API のレートリミットは明示されていないが，REST API のレートリミットより緩く制限をされているので，多くの利用場面ではレートリミットには達しないと推測される．

2.3 Twitter 4J

Twitter4J [4] は Java 向けの Twitter API ライブラリである．Twitter4J では，Twitter API が提供している全機能がサポートされており，Java らしく静的に型付けされたプログラミングを行えるのが特徴である．

表 2 は，表 1 で示した命令を Twitter4J を使用して記述したものである．Twitter4J を使用することで，Twitter API をより簡便に利用することができ，プログラムを効率的に作成できる．例えば，パブリックタイムラインを取得するには `http://api.twitter.com/1/statuses/public_timeline.json` を参照することにより，ツイートの属性情報がツイートごとのリストとして取得され，**図 1** の様な JSON 形式のデータが取得される．図 1 には，パブリックタイムタイムラインに表示されるツイートのオブジェクトがリストの形で表示されている．例えば，`"place":null` は，このツイートに位置情報が付属していないことを表し，`"in_reply_to_user_id":null` は，このツイートはリプライ先となっているユーザ ID がない，つまり，このツイートはリプライではないことを表している．この JSON 形式のデータを扱うと，プログラムが煩雑になりやすい．一方，Twitter4J の `getPublicTimeline()` メソッドを使用すると，JSON 形式のデータを解析し，戻り値をツイートごとのリストオブジェクトとして返してくれるので，データを扱いやすくなる．さらに，リストからスクリーン名を取得する `getScreenName()` メソッドや，ツイート本文を取得する `getText()` メソッドが用意されているので，データも簡単に取得できる．

3. 提案システム

3.1 概要

提案システムは以下の 3 つの部分から構成されている．

投稿部 OAuth 認証と Twitter への投稿を行う部分

ログ取得部 Apache が生成するログから異常なログやアクセスログを取得する部分

リプライ処理部 提案システムに関係づけられた Twitter アカウントに送られたリプライを取得し，それに従った動作を実行する部分

```
[{
"in_reply_to_user_id":null, "favorited":false,
"place":null,
"created_at":"Mon Jan 09 12:09:46 +0000 2012",
"retweet_count":0, "in_reply_to_screen_name":null,
"in_reply_to_status_id_str":null, "retweeted":false,
"in_reply_to_user_id_str":null, "geo":null,
"user":{"lang":"ja", ... } ...
}]
```

図 1 Twitter API で受け取る JSON 形式のデータ例

提案システムは，10 秒に 1 度，繰り返し実行される bot プログラムとして実現されている．10 秒に 1 度実行しているのは，Twitter API の使用をレートリミットに達しない回数に抑えるためである．Bot プログラムの動作の流れを述べる．まず，Apache のログを取得し，異常なログやアクセスログがあるかどうかを検査する．ある場合には，その内容を Twitter へ投稿する．次に，リプライを取得する．リプライに特定の語句が含まれる場合には，その語句に合わせた動作を実行する．最後に，bot プログラムが特定の回数実行されたら，Apache のログから最新のエントリを取得して Twitter へ投稿する．すなわち，異常の発生やアクセスがない場合にも，定期的に，最新のイベントを投稿する．

特定の管理者グループのみがログを読めるようにしたい場合には，当該 Twitter アカウントをプロテクトする．世界中に公開しても良い場合には，プロテクトせずに運用する．

Bot プログラムは，監視に必要な最小限の権限を与えて実行する．一般ユーザ権限で読めるログのみを監視する場合には，一般ユーザ権限で実行し，管理者権限が必要である場合には管理者権限で実行する．

3.2 投稿部

Twitter API のいくつかのメソッドを使用するためには，OAuth 認証を行う必要がある．そこで，bot プログラムの開始時に OAuth 認証を行い，それらのメソッドを使えるようにしている．ユーザは，まず，提案システム用の Twitter アカウントを作成し，そのアカウントの OAuth 認証を行うための情報（コンシューマーキー/シークレット，アクセストークン/シークレット）を Twitter から取得する．ユーザがその認証情報を提案システムに与えることにより，提案システムが OAuth 認証をできるようになる．

Twitter への投稿には Twitter4J の `updateStatus()` メ

ソッドを利用する．このメソッドの引数にツイートのテキストを与えて呼び出すことにより，OAuth 認証したアカウントに，そのテキストを投稿する．ログの投稿の際には，内容と共に注意喚起のための文を投稿する．例えば，アクセスログが投稿された場合，「アクセスがありました．確認してください」という文を投稿する．

Twitter には特有の制限があるので，それ回避するための 2 つの対策を施している．第一に，連続して同じ内容のツイートを投稿することを防ぐために，各ツイートの末尾に投稿時のタイムスタンプを追加している．第二に，長いテキストを分割して投稿している．140 字を超えるテキストを与えて `updateStatus()` メソッドを呼び出すとエラーが返り，投稿ができない．それを回避するために，140 字から末尾のタイムスタンプ分である 20 字を引いた 120 字を超えるテキストが渡された場合，120 字ごとに分割し，別のツイートとして投稿する．

残念ながら，120 字を超えるログが多い場合や，長大な文字列のログがある場合には，ツイートの投稿数が増加する．しかし，前述のように，ツイートは 1 日 1000 件までしか投稿できない．そこで今後は，上記の場合にはログをある程度省略して投稿数を減らすなどの対策を組み込む必要がある．

Bot プログラムは，OAuth 認証されたユーザ自身によるツイートとしてログを投稿することもできるが，OAuth 認証されたユーザから他ユーザへの DM としてログを投稿することもできる．Bot プログラムはどちらの方法で投稿するかについてのフラグを持っており，それに従って投稿を行う．このフラグはリプライによって動的に変更できる．

3.3 ログ取得部

提案システムは，Apache のログが出力されるファイル `error.log`，`access.log` の内容を取得，検査する．`error.log` は Apache のエラー情報が記録されるログファイルである．`access.log` は Apache のアクセス記録が残されるログファイルである．`access.log` にログが新しく追加されていると，ログを投稿部へ渡して Twitter へ投稿する．`error.log` の検査はログのセキュリティレベルを照合し，設定したセキュリティレベル以上のログならば投稿を行う．`error.log` の各ログには，固有のセキュリティレベルが設定され，ログに付記されている．**表 3** はセキュリティレベルを表す文字列とその説明を示したものである．

3.4 リプライ処理部

リプライ処理部では，提案システムの Twitter アカウントに対して送られたリプライを取得し，そのリプライに沿った動作を実行する．なお，動作を実行させることができるのは，そのアカウントがフォローしているユーザからのリプライのみである．

表 3 セキュリティレベル

レベルを表す文字列	説明
emerg	緊急，システムが利用できない
alert	直ちに対処が必要
crit	致命的な状態
error	エラー
warn	警告
notice	普通だが，重要な情報
info	追加情報
debug	デバッグメッセージ

まず，bot プログラムは，自身のアカウントに送られた新しいリプライリストを取得する．取得には，検索 API である `search()` メソッドを利用する．このメソッドの引数には検索の条件が入る．自身のアカウントに対するリプライを取得し，過去に既に読み込んだツイートより後のツイートの検索を行う．

その後，取得したリプライリストの中の一番古いツイートから順に 1 件ずつ内容を検査する．まず，リプライの送り元のユーザが，自身の Twitter アカウントがフォローしているユーザであるかどうかを `existsFriendship()` メソッドにより調べる．そうであるなら，所定の語句がツイートに含まれているかどうかを調べる．含まれていれば，その語句に関連づけられた動作を実行する．動作を実行するかどうかは語句を含むかどうかのみによる．語順の違いや，関係のない単語が含まれているかどうかには影響されない．これにより，連続して同じ命令を bot に与えるときにも，リプライの文章を変更できるので，同内容の投稿として Twitter API に拒否されることがなくなる．また，複数の語句を組み合わせることによって，1 ツイートで複数の命令を実行する事ができる．

図 2 は，リプライによる bot への命令に用いられる語句とそれに関連づけられた動作を示している．例えば [設定][レベル][error] の様に語句を組み合わせたリプライを送ると，ログレベルを error レベル以上に設定するという「ログレベルの設定」ができる．[設定][アクセス] というリプライを送ると，「アクセスログの出力切替」を実行できる．すなわち，アクセスログを出力するかどうかを切り替えられる．「間隔の設定」では，数字を引数に取ることによって，与えた数字の分数の間隔でログを出力するように設定できる．「今のツイート間隔の出力」では，現在設定されている出力間隔を出力する．「ツイート投稿の停止」では，提案システムによる投稿をすべて停止させる．投稿されたくない状況になった場合に使うことを想定している．なお，システム自体は動作し続けているので，「ツイート投稿の再開」を指定すると，投稿を再開させることができる．「ツイートを DM へ変更」では，ログの出力先を，ユーザタイムラインではなく，特定のアカウントに対しての DM に変えることができる．他のフォロワーには投稿を見られたく

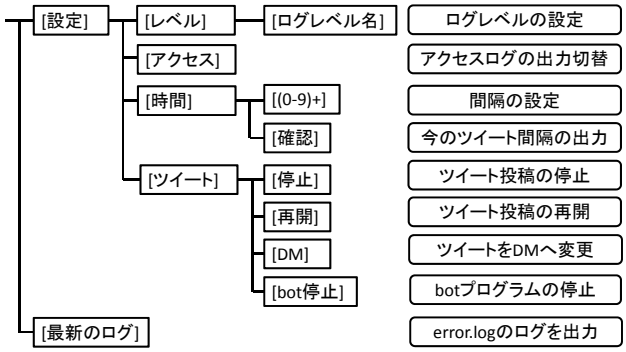

図 2 命令に用いられる語句とそれに関連づけられた動作

ないが，ユーザ自身はログの投稿を確認したい場合などを想定した機能である．これも「ツイートの再開」を指定することでユーザタイムラインへの投稿に再び切り替わる．「bot プログラムの停止」では，提案システムの bot プログラム自体を停止する．停止すると，ログの投稿やリプライの取得などはできなくなるので，システムを再び起動しなければ操作ができなくなる．投稿させる必要がなくなった場合などを想定した機能である．[最新のログ] を投稿すると，「`error.log` のログを出力」が実行される．すなわち，`error.log` 中の最新の部分を出力させることができる．

4. 実験

提案システムの動作を確認する実験を行った．実験では，OS は Windows 7 Professional，CPU が Core 2 Quad Q6600 2.4 GHz，メモリが 4 GB のマシンを使用した．Apache 2.2.21 を提案システムにより監視した．1 つの Windows OS 上に Apache と提案システムの両方を動作させた．この実験のために ol_test という Twitter アカウントを取得し，それを bot のアカウントとして利用した．また，そのアカウントへリプライを送るために，ryuji_itoigawa という Twitter アカウントを利用した．

まず，定期的なログの投稿の動作を確認する実験を行った．投稿間隔を 5 分に設定し，ログが 5 分ごとに出力されることを確認した．また，ログが 120 字以上の文字列からなる場合，そのログが分割されて投稿されたことを確認した．

次に，異常ログの投稿の動作を確認する実験を行った．まず，4 分おきに定期的な投稿を行うという設定で提案システムを実行した．その後，セキュリティレベルが error であるログを `error.log` に出力させ，それに対する提案システムの挙動を観測した．その際の `error.log` を**図 3** に，異常ログを**図 4** に示す．図 4 からわかるように，22 時 2 分 12 秒に `error.log` に書きこまれたログが，22 時 7 分 22 秒と 22 時 11 分 22 秒に，4 分の間隔で出力されている．しばらく `error.log` が更新されなかったため，同じログが 2 度，最新のものとして出力されたものである．その後，22 時 12 分 24 秒に error レベルのログを出力させた．そのログは，図 4 を見ると，22 時 12 分 32 秒に投稿されている．なお，ツイートの先頭の [最新] は，定期的な投稿もしくは命令を与えた結果の投稿であることを表しており，[検査] は，設定したセキュリティレベル以上のレベルのログが発見された結果の投稿であることを表している．

```
[Thu Jan 19 22:02:12 2012] [notice] Child 256:
  Child process is exiting
[Thu Jan 19 22:12:24 2012] [error] [client 127.0.0.1]
  File does not exist:  C:/Program Files/Apache...
```

図 3 error レベルのログを出力した時の `error.log`

図 4 異常ログの投稿

また，提案システムが同様に 4 分おきに投稿している状況において，Apache が `access.log` を更新した場合の挙動を見た．その際の `access.log` を**図 5** に，アクセスログと異常ログが投稿された様子を**図 6** に示す．図 6 の下の 2 つのツイートは，`error.log` の最新のログを 4 分ごとに投稿したものである．上の 2 つのツイートは，`access.log` の更新部分を取得し投稿したものである．なお，ツイートの先頭の [アクセス] は，アクセスログの更新であることを表している．

最後に，リプライにより bot プログラムを操作する実験を行った．通常の命令の操作，複数命令の同時操作，語順の違う命令の操作を行った．

通常の命令の操作として，[最新のログ] という命令により `error.log` 中の最新のログを出力させた．[設定][レベル][notice] という命令により，セキュリティレベルの設定を行った．**図 7** は，それらの操作が実行されたことの確認のために bot が行った投稿を示している．

複数の動作を行わせる命令として，[設定][時間][確認][60] を指定した．この命令により，設定時間間隔を確認する動作と，時間間隔を 60 分に設定する動作を同時に実行できる．図 7 より，これらの動作を上記の一命令で指示できていることがわかる．なお，時間間隔の設定が先に実行されたのは，提案システムでは図 2 で上にある命令から実行す

```
xx.xx.xx.xx - - [19/Jan/2012:23:05:12 +0900]
  "GET /a.html HTTP/1.1" 304 -
xx.xx.xx.xx - - [19/Jan/2012:23:39:41 +0900]
  "GET /favicon.ico HTTP/1.1" 404 209
```

図 5 アクセスログの内容

図 6 アクセスログと異常ログの投稿

るようにしているからである．

語順の違う命令として，[warn][設定][レベル] を与えてセキュリティレベルの設定を行った．同じく図 7 に示すように，語順の違う命令であっても正しく動作することがわかる．

実験で観測されたレスポンス時間について述べる．Bot プログラムを 10 秒に 1 度動作させているため，`error.log` や `access.log` の更新，リプライの取得などは比較的早く行えている．しかし現在，Twitter API 側の反映に 20 秒から 30 秒程度かかるため，レスポンスは 30 秒後程度になっている．

5. 関連研究

LetterTwitter [2] は，本研究と同じく，Twitter を通知に応用した研究である．この研究では，ポストの中に配置されたカメラと PC が，ポストに投函された手紙やチラシを撮影し，画像に合わせたメッセージと共に画像を Twitter 上にアップロードする．ユーザは PC や携帯電話などを利用してツイートを見ることにより，自分のポストへの投函を知る．LetterTwitter では，本研究で提案したシステムと同じく，Twitter API を使用してメッセージを投稿することにより，イベントの発生をユーザに通知する．提案システムでは，Twitter を通じて通知の確認だけでなくプログラムの挙動を操作できる点で LetterTwitter と異なる．

6. まとめと今後の課題

本研究では，Apache Web サーバのログを取得し，定期的にもしくは異常があった際に Twitter にそのログを投稿するシステムを提案した．ユーザは，提案システムによって投稿されたツイートを見ることにより，サーバの状態を

図 7 リプライによるプログラムの操作

知ることができる．また逆に，ユーザが特定の語句を含むツイートを提案システムの Twitter アカウントに対して送信することにより，提案システムに所定の動作を実行させることができる．提案システムを実装して実験を行い，システムが正しく動作することを確認した．

今後の課題を以下に述べる．第一に，現在，140 文字の制限により，出力できる情報が少ないので，ログの情報をわかりやすい形に可視化することを検討している．可視化されたグラフなどを Twitter に投稿することにより，より分かりやすい情報を得られる可能性がある．文献 [1], [5] で提案されているような，セキュリティ向上のためのログ可視化システムと組みわせることも有望と考えられる．第二に，現在のシステムでは，ログの監視や出力，システムの制御のみができるようなっているが，ツイートによって監視対象のサーバ自体を制御できるような拡張を組み込むことも検討している．たとえばツイートの投稿によって Web サーバを再起動できるようにすることなどを検討している．ただし，その際には bot プログラムに大きな権限が与えられるため，攻撃による乗っ取りやバグによる異常動作などの理由により，逆にセキュリティ上の危険が増す可能性があることも考慮する必要がある．

参考文献

[1] Takada, T. and Koike, H.: MieLog: A Highly Interactive Visual Log Browser Using Information Visualization and Statistical Analysis, *Proceedings of the 16th Systems Administration Conference (LISA 2002)*, pp. 133–144 (2002).
[2] Tsukada, K., Mizushima, Y., Ogata, A. and Siio, I.: LetterTwitter: Smart Mailbox for Spam-filtered Notification of Received Letters, *Adjunct Proceedings of Ubicomp 2010 (Video/Poster)*, pp. 439–440 (2010).
[3] Twitter: Rate Limiting, `https://dev.twitter.com/docs/rate-limiting`.
[4] Twitter4J, `http://twitter4j.org/`.
[5] 江端真行，小池英樹：不正侵入調査を目的とした複数ログの時系列視覚化システム，情報処理学会論文誌，Vol. 47, No. 4, pp. 1099–1107 (2006).