CD-001

グラフ理論に基づく SSH サーバログの統合管理および リアルタイム可視化システムの提案
Proposal of integrated management and real-time visualization system for SSH server logs based on graph theory

大歳 英征[1)] 中原 崇[1)] 波多 悠輔[1)] 前田 達哉[2)]
Hideyuki Otoshi Takashi Nakahara Yusuke Hata Tatsuya Maeda

大堀 天照[2)] 岩井 陸人[2)] 森井 駿[2)] 田中 直人[2)] 小林 孝史[2)]
Takaaki Ohori Rikuto Iwai Shun Morii Naoto Tanaka Takashi Kobayashi

1 はじめに

1.1 現在のセキュリティ事情

コンピュータネットワークの発達に伴い，IoT デバイスやクラウドストレージサービスなど，様々な機器やサービスが提供されている一方で，サーバ管理者や利用者に対するサイバー攻撃が増えている．国立研究開発法人情報通信研究機構によると，過去 10 年間のサービスに対する攻撃の通信量を表すパケット数は増加傾向にある [1]．当該資料に掲載されている表の一部抜粋を表 1 に示す．

また，同資料によると，攻撃の対象となるポート番号は，23/TCP，22/TCP，445/TCP，80/TCP が順に多いとしている．23/TCP は Telnet サーバで利用され，主に IoT 機器の制御に用いられるポートである．22/TCP は SSH サーバで利用され，IoT 機器やサーバの管理などに用いられるポートである．本研究では特に SSH に対する攻撃に注目する．445/TCP は SMB サーバで利用され，主に WindowsOS におけるファイル共有に用いられるポートである．80/TCP は Web サーバで利用され，IoT 機器の Web インターフェースに用いられるポートである．

Internet Storm Center のポート番号調査 [2] によると，攻撃の対象となるポート番号は，22/TCP，80/TCP，23/TCP，2222/TCP，587/TCP が順に多いとしている．

SSH サーバに対する攻撃が多い背景として，SSH サーバの目的がサーバの遠隔操作にあり，攻撃者が一度でも認証を突破し侵入に成功すると，直接サーバを操作することが可能な事がある．また，そのサーバを起点に内部ネットワークの他サーバへ攻撃することも容易になる．SSH サーバへの侵入による攻撃イメージを図 2 に示す．

表 1 年間総観測パケット数の統計（過去 10 年間）

年	観測パケット数
2011	約 46 億
2012	約 79 億
2013	約 129 億
2014	約 257 億
2015	約 545 億
2016	約 1281 億
2017	約 1504 億
2018	約 2121 億
2019	約 3279 億
2021	約 5001 億

1） 関西大学大学院総合情報学研究科知識情報学専攻
Intelligent Informatics Major, Graduate School of Informatics, Kansai University
2） 関西大学総合情報学部
Faculty of Informatics, Kansai Uniersity

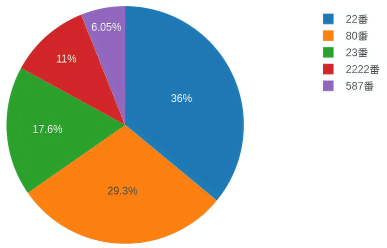

図 1 ポート番号に対する攻撃の割合 (2022 年 9 月)

1.2 ログの可視化

サーバを安全に管理するためには，ログの可視化を行い，攻撃傾向を分析することが有効である．しかし，ログの可視化は，主に 2 つの理由で困難となりうる．

1 つ目は可視化システムのバックエンドを実装するコストである．可視化システムには，データベースが必要である．しかし，データベースの構築には，テーブルの作成や，スキーマの型を 1 から作成する必要がある．

2 つ目は可視化システムのための API を実装するコストである．可視化には，データベースからデータを取得するための API を構築する必要がある．しかし，API を構築するには，SQL 文に関する知識や，API に関する知識が必要であり，学習コストが高い．

以上の問題を解決するために，本研究では MongoDB と GraphQL[3] を用いた可視化システムを提案する．MongoDB により，データ型を意識せずにデータを追加することができ，JSON オブジェクトでデータをやり取りすることが可能となる．また，GraphQL の Subscription と Query により，データ管理者は API 構築の手間を省くことができる．

1.3 SSH プロトコル

SSH（Secure SHell）とは，暗号や認証技術を用い，遠隔のサーバへの暗号化された安全な通信経路の確立とその経路を用いた通信を提供するプロトコル群である．IETF（Internet Engineering Task Force）によりその基本的な仕様が RFC（Request For Comments）4250〜4254 として策定されている [4]．図 3 に，SSH サーバの流れを

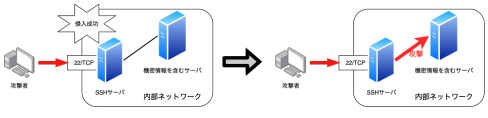

図 2 SSH サーバへの侵入による攻撃イメージ

示す．

2　関連研究

先行研究では，Zabbix と Telegram を用いた可視化システム [5] が提案されている．先行研究では，Zabbix というオープンソースのリソース監視システムを用いて，ping of death 攻撃と SYN フラット攻撃が検知された際に Telegram を用いて管理者にアラートするシステムを構築している．しかし，先行研究では，SSH サーバに対する攻撃は分析対象としていない．

既存の可視化システムとして，Splunk[6] がある．Splunk では，運用サーバにおけるトラフィックを可視化することができる．例えば，Web サーバに対するリクエストや，サーバ機器のリソース消費量を可視化することが可能である．しかし，Splunk で設定する項目には，SQL 文に関する知識が必要な部分もあるため，設定が難しくなると考えられる．

本研究では，SSH サーバに対する攻撃を分析するツールとしての可視化システムを提案する．本システムでは，GraphQL を用いることで設定を基本的に必要とせず，可視化するために必要なデータを取得する際には，簡易化したクエリを行うことができるシステムを目標とする．

2.1　API サーバのパラダイム

2.1.1　RPC

1960 年代に Remote Procedure Call（RPC）が発明された．RPC は，クライアントからサーバに対して，何らかの動作を要求するメッセージを送信する．サーバは，メッセージを受信すると，クライアントに向けてレスポンスを送信する．

2.1.2　REST

2000 年に Roy Fielding による，論文 [7] で REST が提唱された．REST API では，それぞれのエンドポイントに対して，GET，PUT，POST，DELETE という操作を行うことで，固有のレスポンスを得ることができる．

REST には，欲しい情報に対して，多くの GET リクエストを投げる必要があり，欲しい情報とくらべて，過剰な情報を取得してしまうという課題点がある．また，REST のエンドポイントを管理することにもコストがかかる．

2.1.3　GraphQL

GraphQL はオブジェクトの関係性を表すためにグラフ理論を用いている．グラフ理論により，オブジェクトの包含関係を表すことができるため，データの関係性を含めたクエリが可能となる．GraphQL は基本的に単一のエンドポイントのみで構成されるため，REST と異なり，欲しい情報を適切な量のクエリで取得することが可能となる．リスト 1，2 に本研究で用いる GraphQL サーバ [8] へクエリした例とその結果を示す．リスト 1 を REST でのリクエストに変えた場合を本研究では，データベースの関係を表し，データベースに対するクエリを変わり行うための API として，GraphQL を用いる．

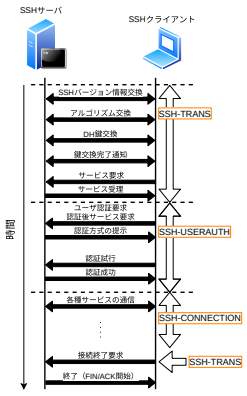

図 3　SSH プロトコルにおける通信の流れ

リスト 1　クエリの例

```
query {
  authLogIp(range: { from: "2020-12-10", to:
    "2020-12-10 0:01" }) {
    ip
    asn {
      organization {
        userList
        passwordList
        ipList {
          ip
          user
          password
        }
      }
    }
  }
}
```

本研究で使用するグラフを図 4 に示す．図 4 を見ると，リスト 1 は 2020 年 12 月 10 日 0 時 0 分から 0 時 01 分までの SSH サーバに対するアクセス情報のクエリである．クエリしている情報にはアクセス元の IP アドレス，IP アドレスを所有している組織が使っているユーザ名リスト，パスワードリスト，組織が所有している IP アドレスリスト，その IP アドレスが使っているユーザリスト，パスワードリストが含まれる．クエリの結果はリスト 2 に示す．

リスト 1 に示したように，GraphQL を用いることで，データベースを直接操作することなく，データの取得を行うことが可能になる．例えば，IP アドレスから組織，組織から IP アドレスリスト，IP アドレスリストから個々の IP アドレスが使うユーザ名とパスワードのリストを導出することが可能である．GraphQL により複雑なデータ構造を図 4 のように一連の流れとして扱うことができる．

GraphQL を使用するメリットは 2 つある．1 つ目は，連鎖的なクエリである．GraphQL によりオブジェクトの関係を，グラフとして表せるため，情報の関係性があるものをクエリすることができる．2 つ目は，GraphQL の API を利用できることである．API を用いることで，ユーザはデータベースの存在を意識せずに，複雑なクエリを行うことができる．

GraphQL には，問い合わせ方法として，Query，Mutation，Subscription の 3 つがある．Query はデータを取得するときに用いる．Mutation は，データを追加するときに用いる．Subscription はデータの変更をリアルタイムで取得するときに用いる．

3　データ収集と可視化

本章では，SSH サーバから収集できるログと，収集したログを可視化する方法について述べる．

3.1　ログの収集

本研究のシステムでは，関西大学小林研究室が運営している 3 つの SSH サーバから出力されるログを収集している．3 つの SSH サーバのうち，2 つは日本にホスティングされている．残り 1 つは，ムンバイにホス

リスト 2　クエリの結果

```
{
  "data": {
    "authLogIp": [
      {
        "ip": "A.B.C.D",
        "asn": {
          "organization": {
            "userList": ["louwg"],
            "passwordList": ["louwg"],
            "ipList": [
              {
                "ip": "A.B.C.D",
                "user": ["louwg"],
                "password": ["louwg"]
              }
            ]
          }
        }
      }
    ]
  }
}
```

リスト 3　REST での URL の例

```
http://example.com/2020-12-10/2020-12-10%200:01/authLogIp/ip/
http://example.com/2020-12-10/2020-12-10%200:01/organization/userList/
http://example.com/2020-12-10/2020-12-10%200:01/organization/passwordList/
http://example.com/2020-12-10/2020-12-10%200:01/ipList/ip/
http://example.com/2020-12-10/2020-12-10%200:01/ipList/user/
http://example.com/2020-12-10/2020-12-10%200:01/ipList/password/
```

ティングされている VPS を使用している．図 5 に本研究におけるシステム構成を示す．SSH サーバにアクセスがあると，SSH サーバはログを出力する．出力されたログを Go 言語により作成されたプログラムが読み取り，読み取った内容を GraphQL サーバに送信する．送信されたデータは，GraphQL サーバにおいて，パースされ，ログの内容が，リスト 4 に沿った形式である場合，MongoDB にデータを格納する．MongoDB に格納する際に，可視化システムのフロントエンドに対して，更新されたデータが送信される．

3.2　ログの格納

SSH サーバから出力されるログは，リスト 4 に示すフォーマットである．SSH サーバのログ形式を変更をする際はログを JSON に変換するコードを変更することで，独自のログ形式に対応することが可能である．

SSH サーバログに含まれている情報を表 2 に示す．ユーザ名とパスワードに関しては，SSH サーバソフトウェア内で 16 進数化されており，エスケープ文字などにより，データベースへの不正なアクセスができないようになっている．本研究で使用する SSH サーバは，出力されるログが OpenSSH のログとは異なるソフトウェア [9] を使用している．本システムで使用される項目は，ユーザ名，パスワード入力，IP アドレスである．

本研究では，SSH サーバから出力されるログを統合的に収集し，アクセス状況を可視化するシステムを提案する．図 5 に，本システムの構成を示す．本システムでは，SSH サーバにアクセスがあったときに，SSH サーバのログを GraphQL サーバに送信する．GraphQL サーバは受信したデータが，ログ形式に沿っているかどうかを確認して，ログ形式と一致する場合のみ，データベースに書き込む．また，GraphQL サーバは，ログを受信した際に IP アドレスから，国名，都市名，組織名，緯度，経度，国コード，大陸コード，ASN 番号を取得する．取得

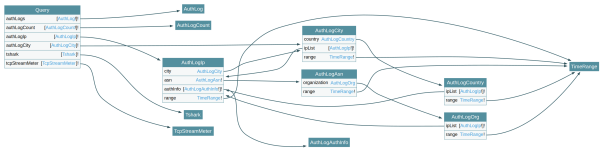

図 4　本研究でのグラフ

表 2　SSH サーバログに含まれている情報

ログの種類	説明
認証結果	認証に成功したか失敗したか
ユーザ名	認証に使用したユーザ名 (16 進数または ASCII)
パスワード	認証に使用したパスワード (16 進数)
IP アドレス	認証に使用した IP アドレス
ポート番号	SSH クライアントが使用しているポート番号
認証時間	パスワード入力要求から，パスワード返答までの時間（秒）
判断	SSH サーバが判断した結果
RTT	SSH サーバ上で独自に計測したサーバクライアント間の Round Trip Time
KEXINIT	KEXINIT メッセージの送受信時間
NEWKEYS	NEWKEYS メッセージの送受信時間

には，geojs[10] に対してリクエストを送信し，その結果をデータベースに格納している．可視化システムを表示している Web ブラウザから GraphQL サーバにアクセスがあった場合は，GraphQL サーバからデータベースにクエリを送信し，データベースから得られたデータをフロントエンドに返す．

3.3　可視化システムに使用しているソフトウェア

本研究では，グラフ等を表示する可視化システムには，ApexCharts[11] と react-simple-maps[12] を用いる．API サーバとデータベースサーバには，GraphQL と MongoDB を用いる．本節では，表示するグラフに付いて説明する．表示するグラフは，SSH アクセスがあったときの世界地図 (以下，スパイダーマップと表記)，1 時間ごとのログイン試行回数，1 日ごとのログイン試行回数，1 日の国ごとのログイン試行回数の割合，SSH クライアントバージョンの割合である．

リスト 4　SSH のログ形式

```
Sep 30 14:36:51 localhost sshd[4726]: [Auth:Fail,User:root,IP:A.B.C.D,Time:0.479649,Detect:Attack,RTT:0.000000,Year:2019,Month:09,Day:30,Hour:05,Minute:36,Second:51,MicroSec:055325]KEXINIT:0.000000,NEWKEYS:0.000000 [preauth]

Aug 1 03:37:21 localhost sshd3[19616]: Fail,root,A.B.C.D,0.120138,Attack,0.103100,2020,07,31,18,37,21,391265,0.103724,0.102476 [preauth]

Sep 1 03:16:56 localhost sshd3[3229]: Fail,6e696e61,6e696e61313233,A.B.C.D,0.348250,Attack,0.675394,1598897816,997971,0.107193,1.243594

Oct 1 11:46:59 localhost sshd3[4469]: [Auth:Success,User:root,IP:A.B.C.D,Time:0.129587,Detect:Attack,RTT:0.000000,Year:2019,Month:10,Day:01,Hour:02,Minute:46,Second:59,MicroSec:706609]KEXINIT:0.000000,NEWKEYS:0.000000 [preauth]

Nov 8 20:59:41 localhost sshd4[6409]: Fail,726f6f74,7a68656e7275696463,A.B.C.D,44422,0.136092,Normal,106.016872,1636372781,075988,109.678610,102.355134 [preauth]
```

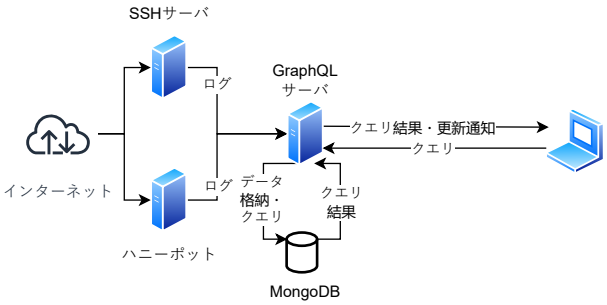

図 5　本研究でのシステム構成

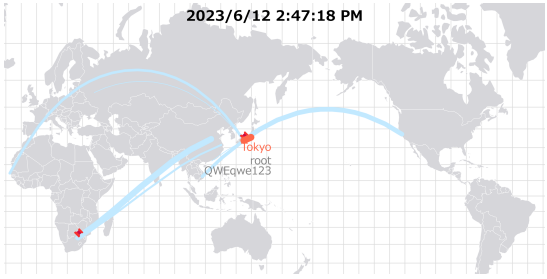

図 6　スパイダーマップ

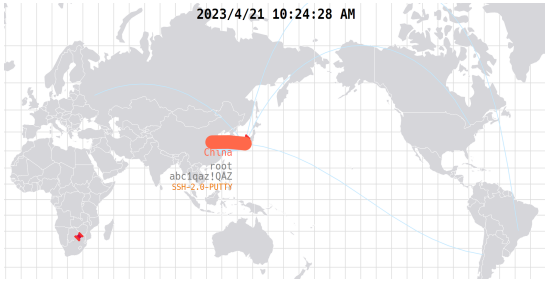

図 7　スパイダーマップ (SSH クライアントバージョン付き)

3.4　スパイダーマップ

スパイダーマップでは，SSH サーバに対してアクセスがあった場合に，アクセス元の都市名，場所，試行されたユーザ名，パスワード，SSH クライアントバージョンをプロットする．図 7 に，スパイダーマップの例を示す．スパイダーマップでは，直近のアクセスを赤線で表示している．また，直近 75 件のアクセスの内，重複しているアクセスは，太い線で表すようになっている．SSH サーバを設置している物理的な位置にピンを立ててある．

3.5　1 時間ごとのログイン試行回数

1 時間ごとのログイン試行回数は，SSH ログイン試行回数を集計して，折れ線グラフに表したものである．横軸は時刻を表し，縦軸は回数を表す．リスト 5 にクエリを，結果を 6 に示す．図 8 に表示されるグラフを示す．緑色の線は SSH サーバが出力するログの件数を示しており，青色の線はパケットキャプチャの件数を表示している．パケットキャプチャは SSH 通信全てを取得しているため，基本的にログよりも多いことがわかる．

リスト 5　1 時間ごとのログイン試行回数を取得するクエリ

```
query ($range: InputTimeRange!) {
  authLogCount(by: HOUR, range: $range) {
    _id
    count
  }
}
```

リスト 6　1 時間ごとのログイン試行回数を取得するクエリの結果

```
{
  "data": {

    "authLogCount": [
      {

        "_id": "2023-06-14T00:00",
        "count": 15
      },
      {
        "_id": "2023-06-14T01:00",
        "count": 48
      }, (中略)

      ]
  }
}
```

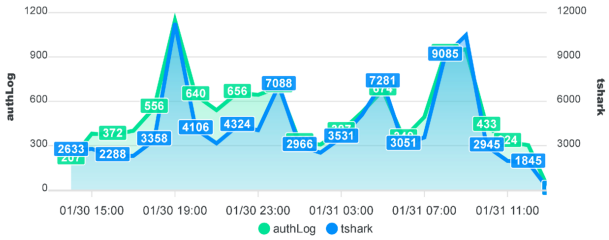

図 8　1 時間ごとのログイン試行回数のプロット

3.6　1 日ごとのログイン試行回数

1 日ごとのログイン試行回数は，現在の日付から，30 日前までのログイン試行回数を集計して，ヒートマップで表したものである．横軸は曜日を表し，縦軸は週の初めを表す．リスト 7 にクエリを，8 に結果を示す．図 9 に表示されるグラフを示す．色が濃い部分は他の日付に比べてログイン試行回数が多いことを表している．

3.7　国ごとのログイン試行回数の割合

国ごとのログイン試行回数の割合は，1 週間の，国ごとのログイン試行回数の割合を棒グラフで表したものである．地域は色で分けられ，グラフ下部に表示される地域名をクリックすることで，クリックした地域のみがハイライトされる．縦軸は合計のログイン試行回数，横軸は日付を表している．リスト 9 にクエリを，10 示す．図 10 に表示されるグラフを示す．

3.8　SSH クライアントバージョンの割合

SSH クライアントバージョンの割合は，1 日の，SSH クライアントバージョンの割合をツリーマップで表したものである．ツリーマップでは，SSH クライアントバージョンが多いほど大きい面積として表される．SSH サーバに対する攻撃は，専用の攻撃ツールが使われるため，どの SSH クライアントが攻撃に使われやすいかがわかるようになっている．リスト 11 にクエリを，クエリ結果を 12 に示す．グラフを図 11 に示す．

リスト 7　1 日ごとのログイン試行回数を取得するクエリ

```
query ($range: InputTimeRange!) {
  authLogCount(range: $range) {
    _id
    count
  }
}
```

リスト 8　1 日ごとのログイン試行回数を取得するクエリの結果

```
{
  "data": {

    "authLogCount": [
      {
        "_id": "2023-06-14",
        "count": 4604
      }（中略）

    ]
  }
}
```

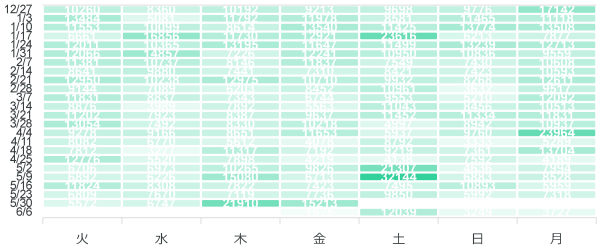

図 9　1 日ごとのログイン試行回数のヒートマップ

リスト 11　1 日の SSH クライアントバージョンの割合を取得するクエリ

```
query {
  tsharkCount(by: ssh_protocol) {
    _id
    count
  }
}
```

リスト 12　1 日の SSH クライアントバージョンの割合を取得するクエリの結果

```
{
  "data": {
    "tsharkCount": [
      {
        "_id": "SSH-2.0-libssh_0.9.6",
        "count": 291
      }（中略）

      ]
  }
}
```

図 11　SSH クライアントバージョンの割合

3.9　本システムの特徴

本システムの特徴をまとめる．本システムでは，クロスプラットフォーム，統合管理，リアルタイム性の 3 つの特徴がある．

3.9.1　クロスプラットフォーム

ログを送信するプログラムは，Go 言語で書かれているため，動作環境に依存せずに実行可能である．また，

リスト 9　国ごとのログイン試行回数の割合を取得するクエリ

```
query {
  authLogCount(by: country) {
    _id
    count
  }
}
```

リスト 10　国ごとのログイン試行回数の割合を取得するクエリの結果

```
{
  "data": {
    "authLogCount": [
      {
        "_id": "United States",
        "count": 886
      }（中略）


  ]
  }
}
```

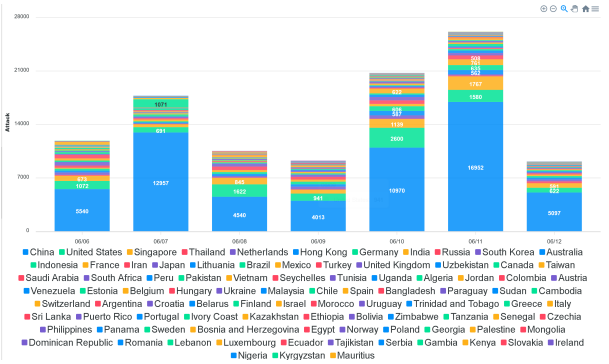

図 10　国ごとのログイン試行回数の割合

GraphQL サーバは JavaScript で書かれている，Node.js バージョン 16.13.0 以上の環境があれば，実行可能である．このため，本研究で作成したシステムは，ほとんどのプラットフォームで動作可能であり，導入コストが低いといえる．

3.9.2　複数の SSH サーバログを統合管理

本研究のシステムでは，複数の SSH サーバを管理することができる．サーバが増えた場合でも柔軟に対応することが可能である．SSH サーバを増設した際は，ログを送信するプログラムを増設した SSH サーバ上で実行することにより，GraphQL サーバに対してログを送信できる．

3.9.3　リアルタイム可視化システム

可視化システムのフロントエンドでは，GraphQL サーバに Subscription と Query を行い，情報を取得して，スパイダーマップやグラフに表示している．Subscription を使用しているため，SSH サーバにアクセスがあった場合は，即座にスパイダーマップ上に更新される．

4　評価

SSH サーバと可視化システム間での，リアルタイム性を評価するために，パフォーマンス評価を行った．評価を行った項目は，SSH サーバと GraphQL サーバ間，GraphQL サーバと可視化システム間，SSH サーバと可視化システム間，geojs による遅延である．検証環境を表 3 に示す．

4.1　SSH サーバと GraphQL サーバ間

本項目では，SSH サーバと GraphQL サーバ間でのパフォーマンス評価を行った．測定した時間は，SSH サー

表 3　パフォーマンス評価時の環境

OS	Ubuntu 20.04.3 LTS x86_64
Kernel	5.11.0-38-generic
CPU	AMD Ryzen 9 5950X (32) @ 3.400GHz
Memory	128GB

バにアクセスがあった時間と，GraphQL サーバにある MongoDB にデータが格納終了した時間の差である．平均値は，360.0 ミリ秒であった．

4.2　SSH サーバと可視化システム間

本項目では，SSH サーバと可視化システム間でのパフォーマンス評価を行った．測定した時間は，SSH サーバにアクセスがあった時間と可視化システムに更新通知があった時間の差である．平均値は 180.93 ミリ秒であった．

4.3　geojs による遅延

本システムでは，SSH ログがデータベースに格納される前に，geojs へリクエストを送信して，国名等の情報を取得している．geojs へリクエストを送信してから，応答があるまでの遅延平均時間は，30.76 ミリ秒であった．以上の結果から，SSH サーバと GraphQL サーバ間において，geojs へのリクエストが約 30 ミリ秒を占めていることがわかった．geojs へリクエストする代わりに GraphQL サーバ内に IP アドレスと国，都市の対応表を用意しておくことで，時間を短縮できると考えられる．

4.4　評価結果

以上の評価結果から，レスポンスタイム，更新通知時間，データベース格納時間全てにおいて 1 秒以内に収まっている．このことから，本研究のシステムには，リアルタイム可視化システムとして十分に運用できるものであると考える．

5　今後の課題

本システムでは，SSH サーバへのアクセスを分析して，リアルタイム性を持った可視化をすることが可能である．また，時間帯を区切って表示することで，時間帯によって攻撃数がどのように変化するのかが分かる．しかし，本システムは，SSH プロトコルについてのみしか可視化することができない．SSH アクセス以外にも．ICMP や HTTP 等のプロトコルに対応させることで，攻撃者がどのようなプロトコルを攻撃する傾向にあるのか分析できると考える．また，時間帯の表示を国ごとに分解することで，どの国がどの時間帯に攻撃が多いかが分かるようになると考える．

参考文献

[1] Nicter_report_2021.pdf. https://www.nict.go.jp/cyber/report/NICTER_report_2021.pdf. (Accessed on 06/13/2022).

[2] Dshield api - sans internet storm center. https://isc.sans.edu/api/#topports. (Accessed on 01/10/2023).

[3] Graphql. https://spec.graphql.org/October2021/. (Accessed on 01/14/2022).

[4] Openssh: Specifications. https://www.openssh.com/specs.html. (Accessed on 12/03/2020).

[5] Mohd Faris Mohd Fuzi, Nur Fatin Mohammad Ashraf, and Muhammad Nabil Fikri Jamaluddin. Integrated network monitoring using zabbix with push notification via telegram. *Journal of Computing Research and Innovation*, Vol. 7, No. 1, p. 155–163, Mar. 2022.

[6] Splunk — the data platform for the hybrid world. https://www.splunk.com/. (Accessed on 06/12/2022).

[7] Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. (Accessed on 01/12/2022).

[8] Swapi graphql api. https://graphql.org/swapi-graphql. (Accessed on 01/19/2022).

[9] 孝史小林, 柊也嶌岡, 心悦唐, 洸希嶋田, 綾雅小川. パスワード認証情報を収集する ssh サーバの構築および運用とそれを活用した bruteforce 攻撃の検知手法. Technical Report 16, 関西大学総合情報学部, 関西大学総合情報学部, 関西大学総合情報学部, 関西大学総合情報学部, 関西大学総合情報学部, may 2021.

[10] Geojs — geojs · rest/json/jsonp geoip api. https://www.geojs.io/. (Accessed on 06/13/2022).

[11] Apexcharts.js - open source javascript charts for your website. https://apexcharts.com/. (Accessed on 06/10/2022).

[12] zcreativelabs/react-simple-maps: Beautiful react svg maps with d3-geo and topojson using a declarative api. https://github.com/zcreativelabs/react-simple-maps. (Accessed on 06/14/2022).