Vol.2019-IOT-46 No.11
2019/6/14

クローラ等のHTTPリクエスト自動収集/簡易解析システム reqhack

白倉 大河[1,a)]　長谷川 皓一[1]　山口 由紀子[1]　嶋田 創[1]

概要：Web サーバは絶えず悪意のある攻撃やスキャンに晒されており，対策にはログの収集や解析が重要となる．そのためには，調査に必要な情報を収集する事前準備として，情報収集機能を持った Web サーバ，情報収集用ドメイン，SSL 証明書などを用意しなければならず，また，収集したログを解析する際にもツールの準備や収集した情報に関する HTTP の仕様を確認するといった手間が必要となる．本研究では任意のサブドメインを持つ記録用 URL を発行し，クライアントの接続情報やリクエストの内容の記録，閲覧をブラウザ上で一貫して行えるシステム reqhack を提案し構築した．このシステムを利用したリクエスト収集/解析の運用例として，悪意のあるボットが閲覧している可能性のある Web サイトに記録用 URL を記載してアクセスの収集と解析を行った．

reqhack: Automated Collection and Analysis System for Web Crawling HTTP Requests

Shirakura Taiga[1,a)]　Hasegawa Hirokazu [1]　Yamaguchi Yukiko[1]　Shimada Hajime[1]

Abstract: Web servers are exporsed to continuous malicious attack and scans so that it is important to collect and analyze access logs for diffencive security. In order to analyze, we have to prepare Web servers with collection functions, SSL certificates and domains. In addition, we will have to prepare analyzer tools and check HTTP specifications when we analyze collected logs. In this research, we constructed a system named reqhack that can create recording URL with arbitrary subdomain to record connected client information, HTTP request, and view them on a browser with several analyze help. We operated reqhack to collect and analyze malicious accesses by posting recoding URLs to web sites that might be visited by a malicious bots as trial.

1. はじめに

Web サーバにはコンテンツを閲覧する Web ブラウザや，検索エンジンのクローラなどの正当なアクセスがある一方で，悪意のある攻撃やスキャンにも晒されている．正当なクローラからのアクセスを識別したり，悪意のあるアクセスを調査するためにはログを収集し，解析する必要がある．そのためには，調査に必要な情報を収集する事前準備として，情報収集機能を持った Web サーバ，情報収集用ドメイン，SSL 証明書などを用意しなければならず，また，収集したログを解析する際にもツールの準備や収集した情報に関する HTTP の仕様を確認するといった手間が必要となる．この問題を解決するために，いくつかの HTTP リクエスト収集/解析手法が提案されているが，サブドメインレベルの解析ができないこと，HTTPS 対応が容易ではないこと，Web ブラウザ上からは簡易な解析しか行えないなどの問題点が残っている．

本研究では，既存の提案の問題を解消し，HTTP と HTTPS のリクエストやクライアントの接続情報などを記録し，ブラウザ上から簡便に閲覧可能なシステム reqhack の提案および構築を行った．reqhack では，サブドメインレベルや HTTPS の解析環境を容易に構築できるとともに，閲覧時にはログを整形表示したり，簡易的なフィルタリング機能を用意するなど，解析を補助する機能を提供す

[1] 名古屋大学 Nagoya Uniersity
[a)] shirakura@net.itc.nagoya-u.ac.jp

る．このシステムを利用したリクエスト収集/解析の運用例として，悪意のあるボットが閲覧している可能性のあるWebサイトに記録用URLを記載してアクセスの収集と解析を行った．

2. 既存のHTTPリクエストの記録・解析手法

2.1 既存サービスとその問題点

HTTPリクエストを記録し，ブラウザ上で整形表示するサービスやソフトウェアはすでにいくつか存在する．著名なものでは，RunScope社のrequestbin [*1]がある．requestbinは主にWebhookのテストやブラウザからのアクセスを記録することを想定して設計されており，インターネットからのスキャンや悪意のあるアクセスの解析で使う際には機能不足な点がある．以下の各節で，個々の機能不足な点について説明する．

2.1.1 サブドメイン形式の記録用URL

requestbinなどの既存サービスでは，`http://example.com/in/ID`のようにパスの部分に個別のIDを含んだ形式の記録用URLを発行する場合が多い．しかし，クローラやスキャナはドメイン直下のrobots.txtやsitemap.xmlにアクセスし，その情報や指示を確認したり，攻撃やデータ収集の足がかりにする場合がある[1] [2]．また，.htaccessなどの設定ファイル，WordPressのような著名なコンテンツ管理システムを構成するファイルに直接アクセスを行い，権限設定の不備等を確認するスキャンを行うツールも存在する．

このような多種多様なアクセスの記録に対応するためには`http://ID.example.com/`のようなサブドメイン形式の記録用URLへの対応が必要となる．

2.1.2 HTTPS対応

HTTPSはサーバクライアント間で安全なやり取りができるよう，通信をTLSで暗号化したプロトコルである．近年では機密情報や個人情報を含まないページも含め，サイト全体に暗号化を適用する常時HTTPS化を採用するWebサイトが増加している[3]．ブラウザでもHTTPコンテンツの規制が強化されている．HTTPSのページにHTTPの画像やスタイルシートを読み込んでいる場合は混合コンテンツとして警告が表示される．また，Webカメラや位置情報のような中間者攻撃によって悪用されたときに被害が大きくなる可能性のあるAPIへのアクセスが制限される[4] [5]．通常のブラウザからのアクセスを想定する場合や，ブラウザのヘッドレスモードを利用するクローラを想定する場合，HTTPSへの対応は非常に重要な要件となる[6][7]．

2.1.3 解析ページの機能

記録したHTTPリクエストを閲覧するページは，サービスによって機能は異なるが，あまり高度な機能は提供されていない．例えば，requestbinでは，画像などのファイルを含めたメッセージ本文の解析やプレビュー表示，同様のリクエストをcURLで送る方法の提示，生のリクエストヘッダの表示などには対応していない．

2.2 既存のWebサーバのログ機能

既存のWebサーバやリバースプロキシを用いてログを収集することも考えられるが，いくつか問題がある．第一に，記録される情報が限定的で，Webサーバソフトウェアによってはログ出力できない項目も存在する．第二に，エンコードされた長大な文字列やバイナリデータなどが含まれる場合があり，人間が読む際に解析用のツールを用意する必要が出てくる．加えて，nginxはログを記録する際やバックエンドにある別のサーバにリクエストを転送する際に，HTTPの仕様に基づいてリクエストをパースしたり書き換える．これによって，nginxのログ出力やnginxを経由して通信するアプリケーションサーバではリクエストの中の空白や大文字小文字のようなデータの特徴が失われる可能性がある．HTTPヘッダインジェクションで不当なヘッダを挿入できないかといった検証を行う場合，この挙動は解析の邪魔になる可能性がある．

3. reqhack実装と試験運用

2節で記した既存のHTTPリクエスト解析方法の問題を解決するため，reqhackを提案し実装した．reqhackはWebサーバ等を1から設定するのと比べ，小さな手間でWebブラウザ上からHTTPリクエストの記録，解析を行えるシステムである．

3.1 reqhackによる環境構築コスト削減

reqhackはHTTPリクエスト環境の構築と運用に必要となる，以下の複数の機能に対して1つのパッケージで対応できるシステムである．

- 必要な情報を記録するWebサーバおよびデータベースサーバ
- HTTPのヘッダや本文を人間が見やすい形で整形，表示するフロントエンド
- より詳細な解析を行うためにプログラムから処理しやすい形式でログを出力するエクスポート機能
- 上記ソフトウェアやモジュールの管理（Docker Compose[*2]により複雑な設定を行わずにデプロイが可能）

これにより，以下の準備のみでreqhackを稼働可能となる．

- リクエストを受け付けるドメインやSSL証明書の用意
- インターネットからアクセス可能なサーバの用意
- Docker Composeに必要なパッケージの導入とreqhack

[*1] `https://github.com/Runscope/requestbin`

[*2] `https://www.docker.com/`

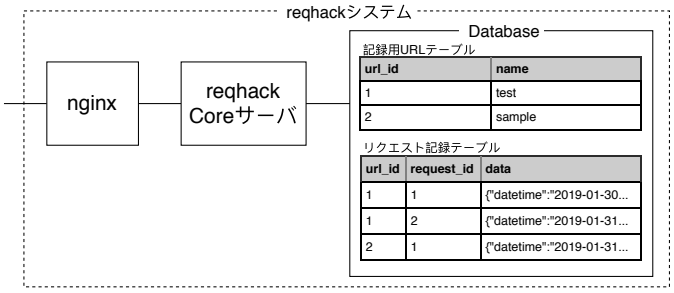


図 1 reqhack のシステム構成図

のビルド

- サーバに対するポート解放などのネットワークの設定
- HTTP リクエストを受け付けるドメインの DNS 設定

さらに，reqhack をサービスとして提供しているサーバが存在すれば，上記の手間は全て不要となりブラウザで記録用 URL の発行から解析までを完結できる．複数人のチームで解析を行う場合には，1 つの reqhack で解析ページの URL を共有することも可能なため，収集結果の解析作業の分担も容易である．

3.2 reqhack システムの構成と運用時のデータフロー

reqhack のシステム構成図を図 1に示す．フロントエンドに Web サーバである nginx，発行した URL 情報や各 URL へのアクセス情報を保持するデータベースである MySQL をバックエンドとする構成である．nginx は直接インターネットに接続されており，HTTPS のエンドポイントやリバースプロキシ，静的コンテンツの配信サーバなどの役割を持つ．nginx の設定の詳細は 3.3節に示す．reqhack Core サーバは Go 言語を用いてフルスクラッチで作成したモジュールであり，nginx から処理を受け取って情報の整形やデータベースへの読み書きを行う．

一般的な reqhack の利用方法は以下の通りである．

1. 解析者は Web ブラウザ上から任意のサブドメインを持った記録用 URL を発行する．reqhack システム内では，記録用 URL テーブルに作成されたサブドメインのレコードが追加される．
2. 発行した記録用 URL を SNS に公開するなどして攻撃者がアクセスするように仕向ける．
3. 攻撃者が記録用 URL にアクセスすると，アクセスの日時や HTTP リクエストの内容がリクエスト記録テーブルに記録される．
4. 任意の時間の経過後，解析者は解析ページを閲覧する．記録用 URL ごとにリクエスト記録テーブルの中身を参照し，必要に応じて整形表示できる．

サブドメイン形式の記録用 URL で HTTPS をサポートするためには，ドメインのワイルドカード SSL 証明書を取得する必要がある．今回は無料で証明書を発行している Let's Encrypt [*3] を利用した．

3.3 nginx が担う機能

図 1に示すように reqhack では，DNS を除くと nginx のみがインターネットから直接接続可能とな構成となっている．nginx は複数の機能を担う．

3.3.1 リバースプロキシ

nginx が Core サーバのリバースプロキシとして動作する．単にインターネット側からのリクエストを Core サーバにリダイレクトするだけではなく，Core サーバが必要とする情報に合わせて 2 パターンのヘッダの書き換えを行う．

1 つ目はサブドメイン形式の URL から Core サーバが想定する URL への書き換えである．Core サーバは REST 形式の API でリクエストを受ける形で実装しており，API に適合するように書き換えを行う．例えば，`http://aaa.example.com/bbb` へのアクセスは `/v1/aaa/in/bbb` へのアクセスに変換される．もう 1 つがリバースプロキシを経由する際に失われる情報を専用のヘッダで引き渡すように書き換えている．リバースプロキシを経由する際に失われる情報には以下のようなものがある．

IP アドレス リバースプロキシに到達した地点で，IP ヘッダにはクライアント IP アドレスが指定されている．しかし，Core サーバがレスポンスする相手はリバースプロキシであるため，Core サーバに渡す前に nginx の IP アドレスに書き換えられる．

ポート番号 リバースプロキシに到達した地点で，TCP ヘッダにはクライアントのポート番号が指定されている．しかし，Core サーバが必要とする情報はリバースプロキシのポート番号となるため，Core サーバに渡す前に nginx に対するポート番号に書き換えられる．

スキーマ reqhack は http と https のスキームに対応しているが，3.3.2節に詳細を記すように，nginx において https を復号して Core サーバに http として引き渡す．Core サーバが元のアクセスが http か https かを判別するために追加の情報が必要となる．

生のリクエストヘッダ nginx では上記の情報に加え，仕様に基づいて許容される範囲で不要な空白の除去や大文字小文字の変換を行って，Core サーバにデータを引き渡す．しかし，2.2節で述べたように HTTP ヘッダインジェクションの確認などで生のリクエストヘッダを確認するのが有用な場合があるので，これも Core サーバに引き渡す．これらの情報を `X-Reqhack-Real-IP-`（乱数）といった HTTP ヘッダに指定し，Core サーバに引き渡す．Core サーバではこれらの情報からクライアントの IP アドレスや本来の URL などを復元できる．ここで，追加する HTTP ヘッダの一部に乱数

[*3] `https://letsencrypt.org/`

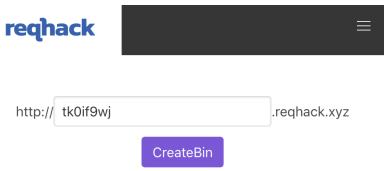


図 2　記録用 URL 発行ページのスクリーンショット

を使用するのは，HTTP ヘッダはアプリケーションレイヤーの通信の情報であり，IP ヘッダや TCP ヘッダと異なり，攻撃者が容易に任意の値に設定できるためである．ここを固定されたフィールドとした場合，攻撃者がリクエストの送信先が reqhack であると考えた時に，reqhack のサーバを騙すような HTTP ヘッダをもったリクエストが送信することで攻撃者の接続元 IP アドレスを偽装したり，解析を混乱させることができる．そこで，今回はサーバで生成した乱数を HTTP ヘッダの一部に埋め込むことで，乱数が分からない攻撃者はヘッダの偽装ができないようにした．この乱数を含む HTTP ヘッダは解析画面には表示されないように Core サーバで削除してデータベースに保存している．

3.3.2　HTTPS の復号

reqhack 実装に必要となるリバースプロキシの機能として HTTP ヘッダの書き換えがあるが，HTTPS では暗号化されているため書き換えを行えない．そこで，nginx のコンテナで秘密鍵を保持し，リクエストの復号とレスポンスの暗号化を行う．これにより，nginx から接続されている Core サーバは HTTP のみに対応すればよくなるため，構成もシンプルになるという利点もある．

3.3.3　静的コンテンツ配信サーバ

nginx は一般的な Web サーバの機能として静的コンテンツ配信に対応しており，reqhack では解析ページを静的ファイルとして配信している．このプログラムは JavaScript で記述されたシングルページアプリケーションとなっている．

3.4　記録用 URL 発行ページ

図 2に記録用 URL 発行ページの表示例を示す．サブドメイン部分の ID はデフォルトではランダムな文字列がフォームの入力欄に表示されているが，任意の値に書き換えて設定することもできる．ただし，すでに使用されている記録用 URL の ID と同一の記録用 URL は発行できない．記録用 URL を発行すると，3.5節で詳細を説明する解析ページに自動で移動する．

3.5　解析ページの機能

解析ページは記録用 URL で記録された HTTP リクエストを表示するページである．

3.5.1　ヘッダ

解析ページのネットワーク系情報のスクリーンショットを図 3 (a) に示す．以下，丸付き数字で示す部分に対応した説明を行う．

①メソッド　リクエストで指定されたメソッドを表示する．一般的に HTTP で使われるメソッドについてはそれぞれ異なる色をつけて識別しやすいようにした．ただし，nginx の制限でアルファベットの大文字，アンダーバー，ハイフン以外の入ったリクエストは受け付けることができない．

②URL　アクセスのあった URL をパス部や URL クエリなども含めて表示する．

③リクエストの ID　記録されたリクエストの ID を表示する．クリックでこのリクエストの ID を表示するページの固定リンクをクリップボードにコピーできる．

④リモートアドレス　接続してきたクライアントの IP アドレスを表示する．マウスオーバーでクライアントのポート番号も確認できる．横には，OSINT（Open Source INTelligence）情報を公開しているサイトへのリンクをアイコンで示している．同様に，アイコンをクリックすることで，Censys，Shodan，VirusTotal の各サービスでリモートホスト名や動いているサービスの確認などが素早く行える．

⑤アクセス日時　リクエストのあった日時と，現在時刻との差異を表示する．

3.5.2　HTTP ヘッダ

HTTP ヘッダの一覧を表示する箇所のスクリーンショットを図 3 (b) に示す．データの加工方式の異なる 3 つの表示形式を用意してあり，図 3 (b) は Table 形式を選択したときのものである．以下，3 つの表示形式を説明する．

Table　表形式で表示する．HTTP ヘッダのフィールド名の左のボタンから MDN Web Docs[*4] でそれぞれのフィールドの説明を参照できる．同じフィールド名のヘッダが複数存在する場合，別のセルに分けて複数行で表示する．

JSON　エクスポート用に JSON 形式で表示する．

Raw　nginx に渡ってきた際の生の HTTP ヘッダを表示する．これには HTTP ヘッダだけではなくリクエストラインも含まれている．この情報は 2.2節で述べたように特定の解析で有用であると考えられる．

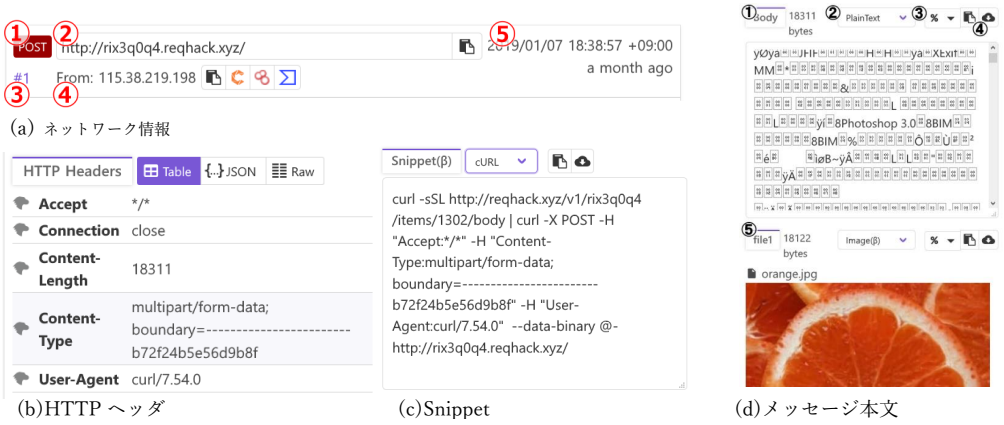
3.5.3　Snippet

記録された HTTP リクエストと同じ HTTP ヘッダやメッセージ本文を持つリクエストを発行するコマンドなどを表示する．図 3 (c) に cURL 版を表示した例を示す．

3.5.4　メッセージ本文

POST や PUT で使われるメッセージ本

[*4] https://developer.mozilla.org



(a) ネットワーク情報
(b)HTTP ヘッダ
(c)Snippet
(d)メッセージ本文

図 3 解析ページ

文と，`application/x-www-form-urlencoded` や `multipart/form-data` 形式のデータを解釈した結果を表示する箇所のスクリーンショットを図 3 (d) に示す．以下，図中の丸付き数字で示す部分に対応した説明を行う．

①Body メッセージ本文全体を表示する項目であり，本文全体のバイト数を表す部分もここに示す．

②表示形式の選択ドロップボックス メッセージ本文の表示形式を以下に示す形式として指定する．

- **PlainText** 単純に文字列と解釈して表示する．他にふさわしい表示形式が判別できなかった場合のデフォルトの表示形式でもある．
- **JSON** Content-Type が `application/json` の場合のデフォルトで，本文が JSON であると解釈し，整形して表示する．JavaScript の JSON.parse 関数によるパースができない場合，エラーを表示する．
- **Form** Content-Type が `application/x-www-form-urlencoded;` から始まる場合のデフォルトであり，パラメータ及び URL のクエリ文字列を表形式で表示する．この形式は HTML の form で POST する際に広く用いられており，キーと値のペアを `key1=value1&key2=value2` のように並べて，パーセントエンコーディングしたものである．
- **XML** Content-Type が `application/xml` か `text/xml` である場合のデフォルトで，XML と解釈して整形表示する．XML としてパースできなかった場合，エラーを表示する．
- **Image** 画像として表示する．`image/`で始まる Content-Type がある場合，デフォルトでこの表示形式になる．

③Encode/Decode ボタン メッセージ本文はパーセントエンコーディングや Base64 でエンコーディングされていることがある．ボタンに応じて decodeURI, decodeURIComponent, atob（Base64 デコード）, encodeURI, encodeURIComponent, btoa（Base64 エンコード）の JavaScript の関数でエンコードやデコード処理を行うことができる．

④コピー，ダウンロードボタン クリップボードに本文をコピーするボタンとファイルとしてダウンロードするボタン．③のボタンによるエンコード/デコードやビューの切り替え結果も一部を除き反映される．

⑤multipart/form-data アップロードフォームのようにファイルを送信した際に使われる multipart/form-data の内容を個別に表示できる．画像ファイルの表示などにも対応している．

3.6 解析ページへの Content Security Policy の適用

reqhack の解析ページでは攻撃者の送信した情報やファイルが表示する機能を備えているが，この表示処理において外部サーバから画像やスクリプトを読み込むリクエストを reqhack が受け付ける可能性がある．そのため，攻撃者が解析の可能性について確認するために攻撃者のサーバから画像やスクリプトを読み込ませ，解析者の存在や解析環境の IP アドレスなどの情報窃取をしかけてくる可能性がある．また，万が一，reqhack 側にクロスサイトスクリプティングの脆弱性が存在した場合，解析ページの中身やトークンなどの情報を奪われる危険がある．

解析ページは外部から渡ってきたデータを HTML や JavaScript として解釈しないように最大限の注意を払って実装しているが，完全に脆弱性がないことを保障するのは難しい．そこで，万一脆弱性が存在した場合でも被害を最小限に抑えられるように CSP（Content Security Policy）を適用している．

CSP はリソースの読み込みやスクリプトの実行が可能な範囲を指定するブラウザのセキュリティ機構である．reqhack では以下のようなヘッダを解析ページに付与することで，CSP の設定をブラウザに通知している．

```
Content-Security-Policy:
    "default-src 'self';
    img-src 'self' data:;
```

```
style-src 'self' 'unsafe-inline'";
```

この指定により，ブラウザはスクリプトや iframe などのリソースの読み込みに対しては以下のような制限をかける．

- インラインの `<script>` 要素，`javascript:` の URL，インラインイベントハンドラーのようなインラインリソースの使用を許可しない．
- eval のような文字列をスクリプトとして解釈することを許可しない
- 解析ページと異なるオリジンへのアクセスを全て遮断する．オリジンとは，スキーム，ホスト名，ポート番号の組み合わせである [8]．

画像のリソースは上記に加えて `data:`スキームによる埋め込みを許可する．これはリクエストに含まれる画像ファイルを表示するために `data:`スキームを使用する必要があるためである．

スタイルシートのリソースには `unsafe-inline` を許可している．これは`<style>`タグでスタイルシートを埋め込むことを許可する指定であり，使用しているライブラリの一部が必要としているために使用を許可している．

4. リクエスト収集・解析の試験運用

4.1 実験目的および実験方法

ワンアクションで情報を発信可能な SNS や Web サイトには機密情報やパスワードなどが操作ミスなどで公開されることがあり，また，ソースコード共有サイトではうっかり秘密鍵等を含めたソースコードを push して公開する事例も見られる．このような情報の収集を目的としたクローラの存在が確認されている．特にクラウドサービスを展開している AWS（Amazon Web Services）の API キー等を悪用し，仮想通貨のマイニングや攻撃サーバとして使用する攻撃が報告されている [9]．

そのような悪意のあるクローラの一部には，投稿に含まれる URL にアクセスしてくるものもあると予想される．reqhack を使うことで，どこから収集したデータであるかを区別して攻撃者がクローラを動かしているサーバの IP アドレスやプログラムの構成情報などを確認できると考えられる．そこで，2019/5/6-2019/5/20 の 2 週間，以下の手順で悪意のあるクローラからのアクセスを reqhack で収集，解析の試験運用を行った．記録用 URL の作成およびサイトへの投稿は 1 日 1 回の実施としている．

1. 学内設置ホスト（rh.net.*.ac.jp）と，学外のクラウド設置ホスト（reqhack.xyz）で reqhack サービスを用意
2. Twitter，GitHub，pastebin.com で新規アカウントを準備
3. 2 つのホストと接頭語（AWS, Secret, Key）を 15 分間隔 [*5] で以下の 4, 5 の手順を実施
4. 半角英数字 10 桁のランダムな文字列を生成し，それに 1, 2, 3 を付与した記録用 URL を発行
5. 以下のサイトに「接頭語 記録用 URL」の形で投稿
   - Twitter
   - GitHub Gist （タイトルやファイル名はデフォルト）
   - pastebin.com （匿名，ファイル名は空白，1 日で削除のオプションを指定）
6. 24 時間以上待ってアクセスを確認

4.2 観測結果

4.2.1 記録用 URL 公開サイトごとの差異

Pastebin 2019/5/9 に AWS を接頭語にした rh.net.*.ac.jp の記録用 URL だけにアクセスがあった．

GitHub Gist 2019/5/6 に AWS の接頭語で公開した 2 件に個別のクローラからアクセスがあった．また，2019/5/12 に公開した記録用 URL には，ロシアから 13 回まとまったアクセスがあった．こちらのアクセスは，`/`，`/favicon.ico`，`/asd` といういう 3 つのパスに何度かアクセスをしてきた．HTTP ヘッダや通信の流れ，リモートホストに ppp とを含むことからも，機械的なアクセスではなく人手でブラウザで開いていくつかの URL にアクセスしてきたのではないかと考える．その後，5/15 に公開したデータにもブラウザのようなアクセスが複数記録された．

Twitter 記録用 URL へのアクセスの大部分は Twitter に公開したものだった．日毎のアクセス数の合計と，一部のクラウド事業者のリモートホストからのアクセスを表 1にまとめた．それぞれのクラウド事業者からのアクセスについて，4.2.2節で述べる．

4.2.2 高頻度で特徴的なアクセス

GCP `*.googleusercontent.com` からのアクセスは GCP（Google Cloud Platform）からのものである．複数のボットが動いていることが予想されるが最短では 15 秒以内，最長では約 2 時間 4 分後にアクセスがあった．平均では約 20 分でアクセスがあった．特徴として，期間中のアクセスは 1 つの GET を除くと，全て HEAD メソッドであった．User-Agent（以下 UA）フィールドの有無のみが異なる 2 つのアクセスを同じタイミングで行ってくるボットが頻繁にやって来た．Connection と Twitter のリファラ（`https://t.co/`から始まる URL）のフィールドのみを指定してアクセスしてくる別のボットも頻繁に観測された．

AWS `*.amazonaws.com` からのアクセスは，AWS（Ama-

[*5] 2019/5/8 まではホストごとの間は 15 分だったが，サーバの問題で 2019/5/9 からは 1 時間 45 分の間をおいた

表 1　Twitter に公開した記録用 URL へのアクセス (すべて 2019 年)

	5/6	5/7	5/8	5/9	5/10	5/11	5/12	5/13	5/14	5/15	5/16	5/17	5/18	5/19	5/20
AWS	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
GCP	9	10	9	7	3	3	3	1	0	3	3	4	3	3	3
Apple	12	14	14	12	14	12	12	14	12	18	12	12	16	12	14
ahrefs.com	30	27	24	30	27	30	30	31	26	27	28	24	24	24	21
trendiction.de	2	4	2	4	2	10	4	4	6	4	6	4	4	4	6
その他	14	27	13	13	15	7	13	11	12	12	9	16	13	18	12
合計	71	86	66	70	65	66	66	65	60	68	62	64	64	65	60

zon Web Service）からのものである．計測期間中，毎回同じボットからアクセスがあった．reqhack.xyz には接頭語にかかわらず，毎回アクセスがある．最小で 1 分 3 秒，最大で 5 分 42 秒，平均 2 分 34 秒．rh.net. . . .ac.jp では AWS の接頭語の際にだけ 18 秒〜19 秒程度でアクセスがある．reqhack.xyz にあるアクセスとは UA やメソッドが大きく異なり，別のクローラと推測される．

Apple　reqhack.xyz で発行した URL に毎回アクセスが来る．UA に `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/600.2.5 (KHTML, like Gecko) Version/8.0.2 Safari/600.2.5 (Applebot/0.1; +http://www.apple.com/go/applebot)` が指定されているように，Apple 社の Bot である．UA に含まれる文字列は Apple 社公式のドキュメントであり，17.0.0.0 のネットワークブロックからのアクセスが正当であることが記述されている．

ahrefs.com　UA には `http://ahrefs.com/robot/` という URL が含まれており，ahrefs というマーケティングの調査関連のサービスからのアクセスである．ahrefs.com に IP アドレスブロックが記載されており，すべてのアクセスがこのブロック内であることが確認できた

4.2.3　低頻度で特徴的なアクセス

UA にサービスの URL が含まれるアクセス　以下のようなサービスから UA にサービス名や URL を入れてアクセスがあった．

1. **omgili.com** omgili.com という SNS などの発言の検索などを提供するサービスのクローラのようであるが，検証方法などは示されておらず，本物かどうか検証はできなかった．
2. **Livelap**　数回アクセスがあり，`http://site.livelap.com/crawler` に説明のあるソーシャル情報を扱う Web サービスのクローラである．ただし，IP アドレスはクラウドコンピューティング事業者のもので，本当にこのサービスのものかを検証することはできなかった．
3. **Linkfluence** ソーシャル関係を取り扱う Web サイトである．検証方法などは示されておらず，本物かどうかは検証できなかった．

62-210-101-170.rev.poneytelecom.eu　検証中に 2 つの記録用 URL のドメインに対して，`/`，`/feed`，`/.rss`，`/atom`，`/rss` といういずれも RSS フィードで使われることのあるパスにアクセスがあった．

ロイヤル・ホロウェイ大学　検証期間中に 2 つの記録用 URL でイギリスのロイヤル・ホロウェイ大学のアドレスから HEAD メソッドでアクセスがあった．1 度目は 6 つ，2 度目は 3 つのアクセスが記録された．User-Agent が `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/601.6.17 (KHTML, like Gecko) Version/9.1.1 Safari/601.6.17` と `Googlebot` の 2 種類があった．

trendiction.de　逆引きは `p150n14.trendiction.de` や `p150n33.trendiction.de` といったホスト名によるアクセスである．`http://trendiction.de/`はソーシャルメディアの解析等を行っているサービスで，その一環でクローリングを行っているものと推測される．このアドレスブロックは hetzner というドイツのクラウドコンピューティング事業者に割り当てられたネットワークブロックで，このサイトの正式なクローラかは判別できなかった．

4.3　考察

4.3.1　GCP や AWS からのアクセス

今回の 2 週間の観測では，CMS 等の設定ファイルや管理インタフェースへのアクセスの試みなど，明らかに悪意のあるアクセスは見られなかった．以下，収集したアクセスについて考察を行う．

AWS の IP アドレスである 54.172.118.182 からのアクセスは UA に Google Chrome のものが指定されているが，Accept-Encoding や Accept-Language といったブラウザが必ずつける HTTP ヘッダが存在せず，何らかのクローラソフトウェアを使った機械的なアクセスであると考えられる．また，`Go-http-client/1.1` への HEAD メソッドアクセスなど，通常のブラウザと大きく異なるアクセスも観測された．

Amazon 社や Google 社が API キーの流出による不正利用を未然に防ぐために検出しているという可能性も考えら

れるが，UA でサービス名を明記しない，robots.txt を確認しない，中身を読み取れない HEAD メソッドでのアクセスなど，複数不審な点がある．よって，利用目的や収集者の情報を明示しない不審なクローラが Twitter を監視していると言える．

4.3.2 アクセス元が安定しない理由

記録用 URL のサブドメイン部分のみが違うツイートを投稿してもアクセスが安定しない理由として，以下が考えられる．

1. 人が人力でアクセスしており，偶然検索などで特定のツイートを見かけた人が偶然その URL にアクセスした可能性がある．ただし以下の点から不自然である．
   - 通常のブラウザと異なるメソッドや HTTP ヘッダが指定されているアクセスがほとんどである
   - favicon.ico へのアクセスがない
   - 投稿してから数秒以内にアクセスが行われている場合が多い
2. Twitter では API の制限が存在し，認証方法によるが検索の回数が 15 分で 180 回に制限されている．API の制限などによりボットが行っている監視で取りこぼしがあった可能性がある．Apple は稀に重複してアクセスしてくることがあるが，毎回アクセスしてくることから別途契約があり制限の少ないデータを提供されているのではないかと予想される．

4.3.3 大文字を含むサブドメインへのアクセス

公開する記録用 URL にはランダムで大文字が含むものを発行した．ブラウザやクローラソフトウェアの一部は全てを小文字にしてアクセスを行う．しかし，今回記録されたアクセスの一部には大文字小文字を保ったままのものもあり，調査時に特徴として使える可能性がある．

4.3.4 サブドメイン形式の記録用 URL によるメリット

今回の実験では`/robots.txt`，`/favicon.ico`，`/rss` などドメインのルートにあるパスへのアクセスが複数記録された．他のソフトウェアと比べてパスではなくサブドメインに ID を含むことで収集できたデータだと考えられる．

5. おわりに

Web サーバは日々攻撃や不正なアクセスに晒されており，HTTP や HTTPS のログ解析は重要である．しかし，アクセスログを記録・解析するのは様々な手間がかかる．本研究では HTTP と HTTPS のリクエストを記録・解析する手間を減らすことを目的としたシステム reqhack を構築した．自分で nginx などの Web サーバを利用して記録，解析などを行うのと比べて少ない手間で基本的な情報の閲覧や解析が行えることが確認できた．

アクセスの解析についても，IP アドレスの調査や，HTTP ヘッダの整形表示などの機能によって効率よく行い，悪意のあるスキャンや正当な Apple などからのアクセスの判別に役立つことが確認できた．この結果を参考に，機密情報の取扱いやサーバのアクセス権限を再確認するなど，セキュリティの向上につなげていけると考えられる．

今後の課題としては，未対応の HTTP の仕様や，FTP 等の他のプロトコルへの対応を進めることが考えられる．また，任意のヘッダや本文を返せるように機能を拡張することで，例えば悪意のあるリクエストに対し攻撃者が想定していたレスポンスを返すことで攻撃者の挙動を更に引き出せるなど，より利用可能な範囲や状況が増えると考えられる．一方，その機能が C&C サーバや攻撃の踏み台などに悪用される可能性なども十分に検討しなければならない．

謝辞 本研究は国立研究開発法人情報通信研究機構が実施する，セキュリティイノベーター育成プログラム SecHack365 [*6] における研究開発の成果を利用したものである．

参考文献

[1] sitemaps. sitemaps.org - Home. `https://www.sitemaps.org/index.html`, 2008.

[2] Vangelis Banos, Yunhyong Kim, Seamus Ross, and Yannis Manolopoulos. Clear: a credible method to evaluate website archivability, 2013.

[3] 常時 SSL 化について — JPRS. `https://jprs.jp/pubcert/about/aossl/`.

[4] Secure Contexts. `https://w3c.github.io/webappsec-secure-contexts/`.

[5] MDN. Features restricted to secure contexts — MDN. `https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts/features_restricted_to_secure_contexts`.

[6] Getting Started with Headless Chrome — Web — Google Developers. `https://developers.google.com/web/updates/2017/04/headless-chrome?hl=en`.

[7] Headless mode — MDN. `https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Headless_mode`.

[8] A. Barth. The Web Origin Concept. RFC 6454, IETF, December 2011.

[9] 柏崎礼生. 彼はクラウドを愛したが，クラウドは彼を愛さなかった. 情報処理学会 研究報告, Vol. 37, No. 13, pp. 1–6, 2017 年 5 月.

[*6] 若手セキュリティイノベーター育成プログラム SecHack365, `https://sechack365.nict.go.jp/`