2019-09-18 OWASP Night

SSRF基礎

OWASP Kansai ボードメンバー

はせがわようすけ

長谷川陽介 (はせがわようすけ)

(株)セキュアスカイ・テクノロジー 取締役CTO
hasegawa@securesky-tech.com
http://utf-8/jp/
千葉大学 非常勤講師
OWASP Kansai ボードメンバー
OWASP Japan ボードメンバー
CODE BLUEカンファレンス レビューボードメンバー

自分たちの直面するWebセキュリティの問題を自分たちの手で解決したい！

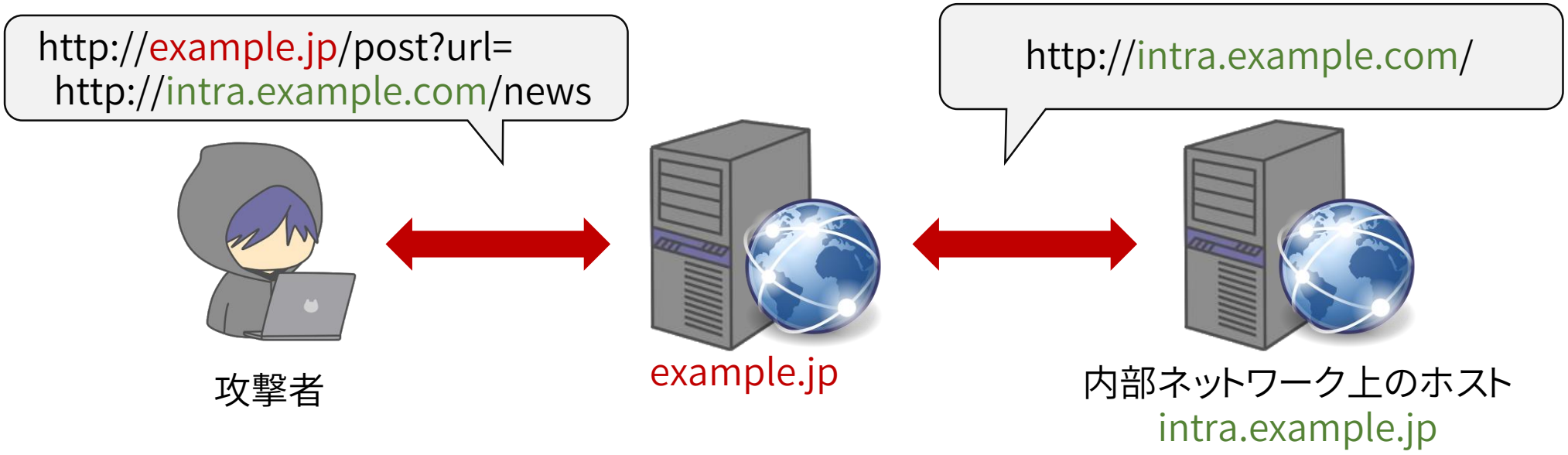
- 日本で2番目の OWASP Local Chapter
- 2014年3月から 京都・大阪・神戸・奈良 で Local Chapter Meeting (勉強会) を開催
- Webセキュリティの悩み事を気楽に相談し情報共有できる場
  - スキル、役職、業種、国籍、性別、年齢に関係なし

SSRFって?

10分でわかるSSRF

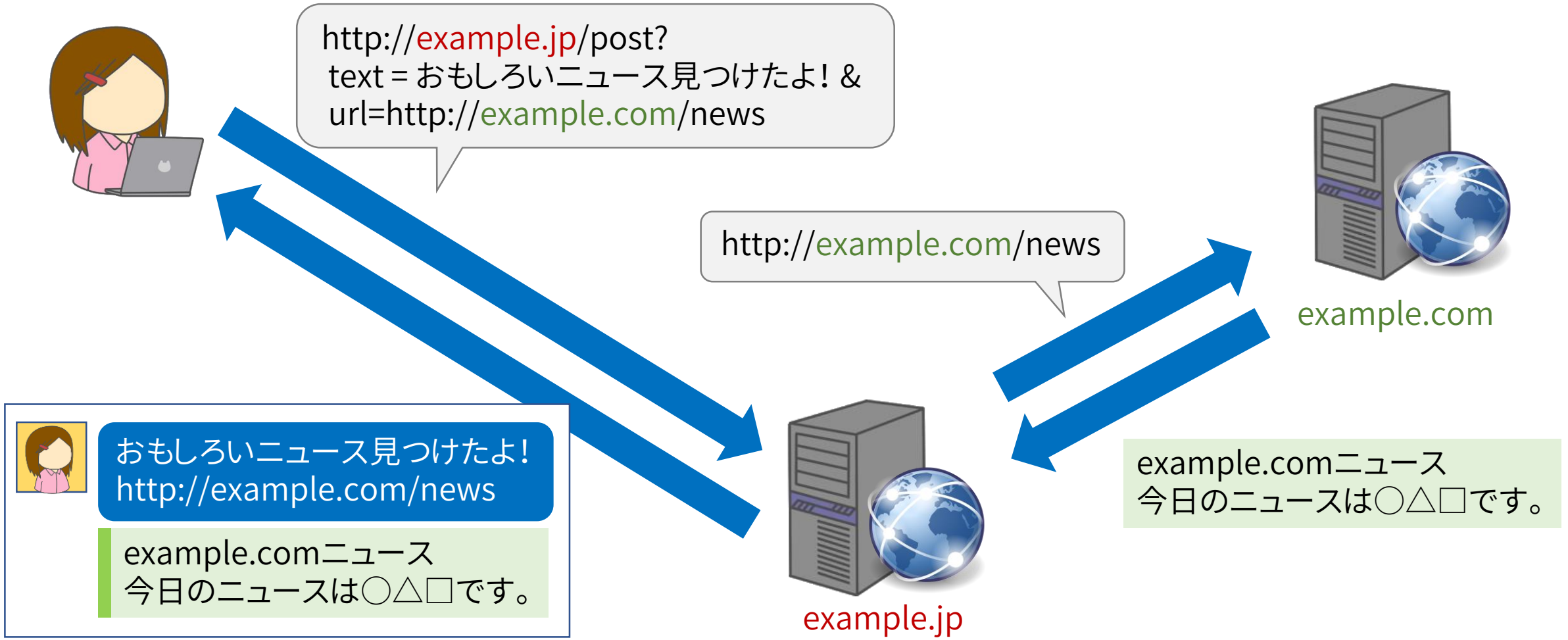
SSRFって?

- サーバーから他のサーバーへリクエストを発行するときに、リクエスト先を攻撃者が指定することができる脆弱性
  - 内部ネットワーク上のサーバーへ間接的にアクセス可能になる



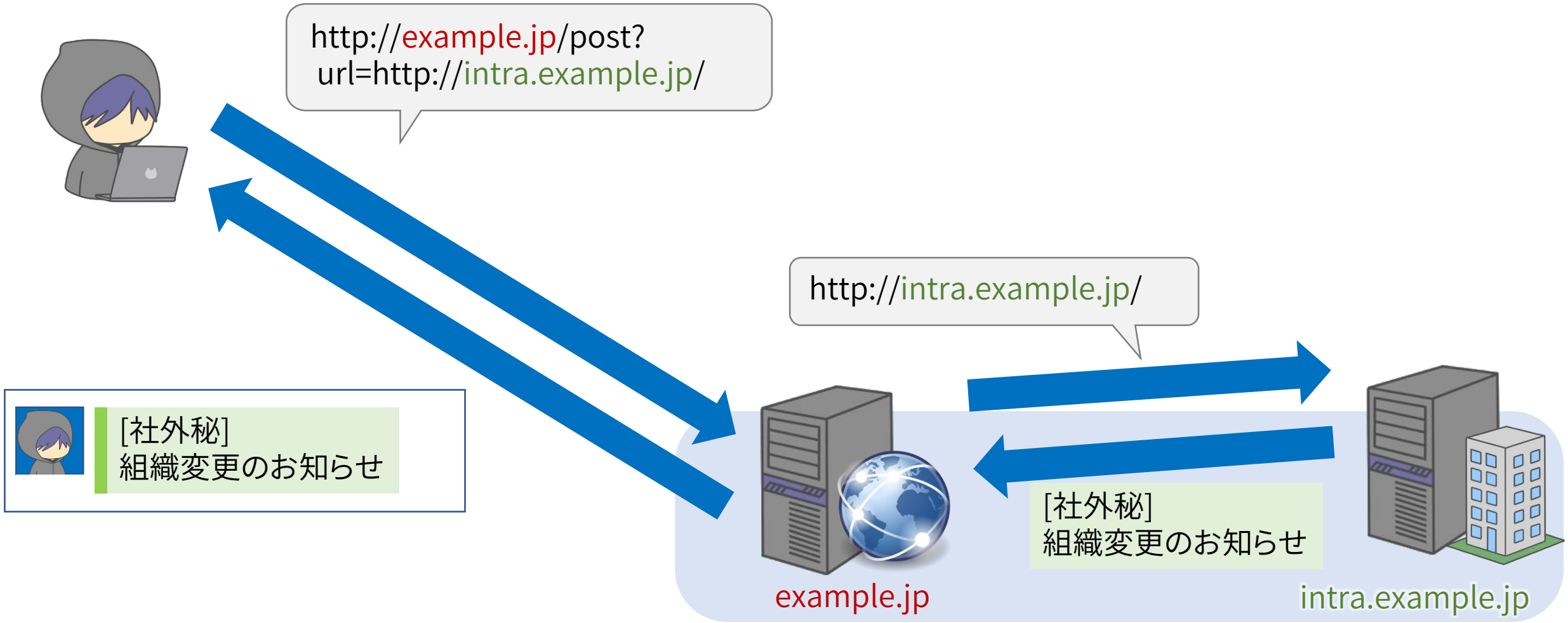
SSRFって? (例)

正規の処理 - メッセージ内でURLが指定されるとサムネイルを表示する



SSRFって? (例)

攻撃 - 内部ネットワーク内の情報が外部から取得される



クラウドのメタデータ漏えいが特に狙われる

- 各クラウドサービスはインスタンス上からHTTP GETでメタデータが取得可能
  - AWS

```
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/meta-data/iam/security-credentials/ロール名
```

  - GCP

```
http://metadata.google.internal/computeMetadata/v1/
※リクエストヘッダーに Metadata-Flavor: Google が必要
```

  - Azure

```
http://169.254.169.254/metadata/
※リクエストヘッダーに Metadata: true が必要
```

※これ以外のURLもあります

SSRF対策

- 接続可能なURLをアプリケーション内で事前に定義しておく

```
$target_list = array('http://example1.jp/', 'http://example2.jp/', 'http://example3.jp/');
if (isset($_GET['url']) {
  $url = $_GET['url'];
  if (in _array($url, $target_list, true)) {
    $c = curl_init($url);
    ....
  }
}       // ※ここまで固定である必要はないが、攻撃者が接続先をコントロールできないようにすることが重要
```

- iptables等でOS/コンテナレベルで接続先を制限する

```
iptables -A OUTPUT -m owner ! --uid-owner root -d 169.254.169.254 -j DROP
```

- XXEの脆弱性をなくす

```
<!ENTITY value SYSTEM "http://169.254.169.254/latest/metadata/">
```

SSRF対策

- こういうコードはよくない(いわゆるブラックリスト的な対策)

```
if (isset($_GET['url']) {
  $url = $_GET['url'];
  $target = substr($url, 0, 23);
  if ($target !== 'http://169.254.169.254/') {
    // 169.254.169.254でなければ通信を許可
    curl_init($url);
    ...
  }
}
```

良くないコード例

- 簡単に回避されてしまう
  - http://2852039166/ とか http://0xa9fea9fe/ とか
  - DNS Rebindingとか
  - 302 w/ Location: http://169.254.169.254/latest/meta-data/ とか

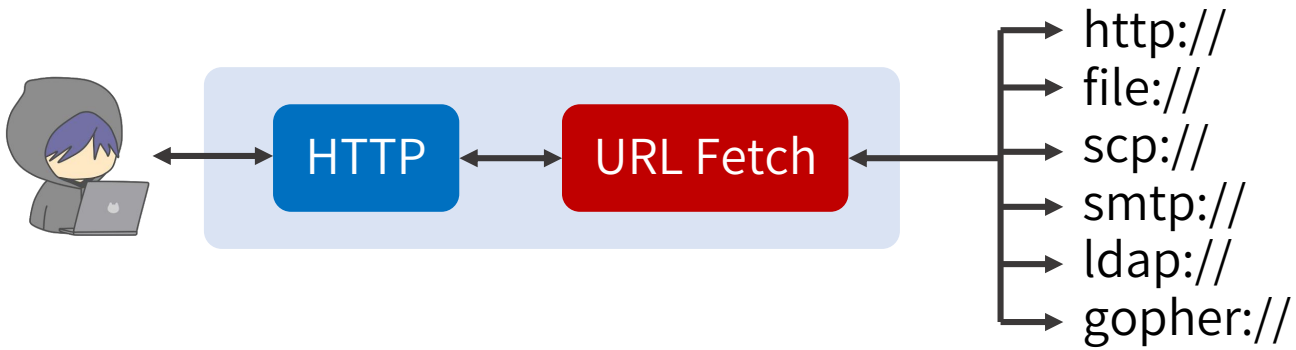
まとめ

- SSRF概要
  - サーバーから攻撃者が指定した他のサーバー/サービスへリクエストが飛ぶ攻撃
- SSRFの脅威
  - 機密情報の漏えい、他のプロトコルへの攻撃、任意コード実行など
  - 近隣でどんなサービスが動いているかに依る
- 対策
  - 接続可能なURLをアプリケーション内で事前に定義しておく
  - iptables等でOS/コンテナレベルで接続先を制限する
  - URLの検証は保険的な対策にしかならない(やらないよりはマシ)

SSRFもうちょい深追い

HTTP以外にも注意

- 通信処理系によってはHTTP以外も通信可能



- 例えば PHP cURL関数は様々なプロトコルをサポートしている

良くないコード例(PHP)

```
$url = $_GET['url'];
$c = curl_init($url);
curl_exec($c);
```

  - 攻撃者がurlとして `file:///etc/passwd` などを指定可能

PHPのcURL関数

CURLOPT_PROTOCOLS

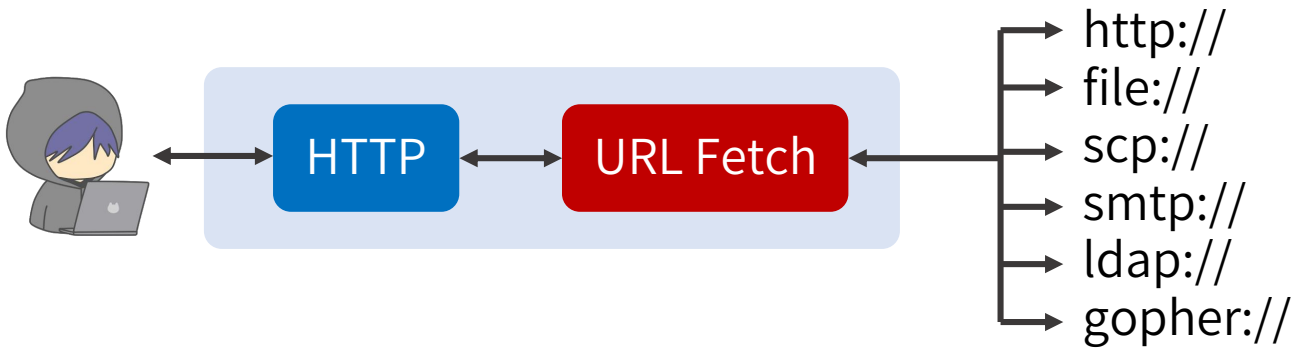
Bitmask of CURLPROTO_* values. If used, this bitmask limits what protocols libcurl may use in the transfer. This allows you to have a libcurl built to support a wide range of protocols but still limit specific transfers to only be allowed to use a subset of them. By default libcurl will accept all protocols it supports. See also CURLOPT_REDIR_PROTOCOLS.

Valid protocol options are: CURLPROTO_HTTP, CURLPROTO_HTTPS, CURLPROTO_FTP, CURLPROTO_FTPS, CURLPROTO_SCP, CURLPROTO_SFTP, CURLPROTO_TELNET, CURLPROTO_LDAP, CURLPROTO_LDAPS, CURLPROTO_DICT, CURLPROTO_FILE, CURLPROTO_TFTP, CURLPROTO_ALL

PHP: curl_setopt Manual https://www.php.net/manual/ja/function.curl-setopt.php

HTTP以外への攻撃

- SSRFを用いて内部のHTTP以外のサーバーへも攻撃可能



  - file:、scp:、ftp: - ローカルフィル、ファイルサーバーへの攻撃
  - smtp:、pop3: - メールサーバーへの攻撃
  - ldap: - LDAPサーバーへの攻撃
  - gopher: - ???

gopher:プロトコル

- 改行などを含む任意文字をURLに載せて送出可能

```
$ curl gopher://localhost:1337/-Hello%0aWorld%0afrom%20curl
```

```
$ nc -l 1337
Hello
World
from curl
```

gopher:プロトコル

- 改行などを含む任意文字をURLに載せて送出可能

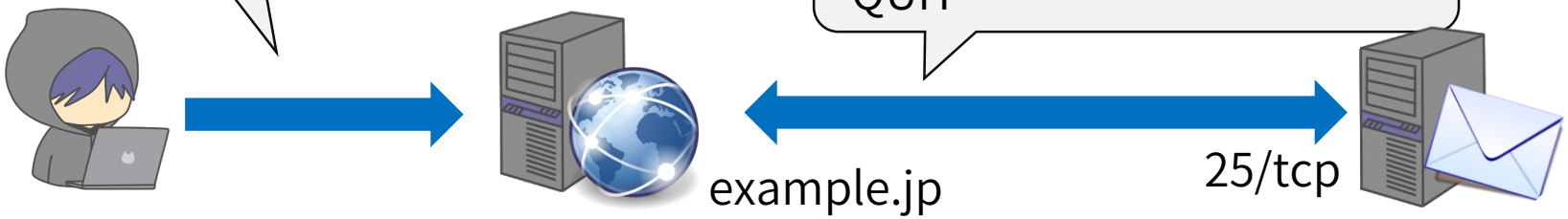
```
$ nc -l 1337
Hello
World
from curl
```

```
$ curl gopher://localhost:1337/-Hello%0aWorld%0afrom%20curl
```

- テキストベースの任意プロトコルのクライアントとして利用可能

http://example.jp/post?url=
gopher://mail.example.jp:25/-
MAIL%20FROM%3a%20a@example.jp%0a
RCPT%20TO%3a%20b@example.jp%0a
DATA%0a...%2E%0aQUIT%0a

MAIL FROM: a@example.jp
RCPT TO: b@example.jp
DATA
....
.
QUIT



SSRF with gopher

- gopherにより攻撃が行いやすくなる
  - gopherでなくてもHTTP通信ライブラリの実装で改行を送出できる場合は同様の攻撃が可能
- Redis (6379/tcp) への攻撃
  - 場合によっては任意コード実行
- SMTP/POP3等 - メールの送信、取得
- 任意のリクエストヘッダー付きのHTTP GET
  - AzureやGCPのメタデータ取得

クラウドのメタデータ漏えいが特に狙われる

再掲

- 各クラウドサービスはインスタンス上からHTTP GETでメタデータが取得可能
  - AWS

```
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/meta-data/iam/security-credentials/ロール名
```

  - GCP

```
http://metadata.google.internal/computeMetadata/v1/
※リクエストヘッダーに Metadata-Flavor: Google が必要
```

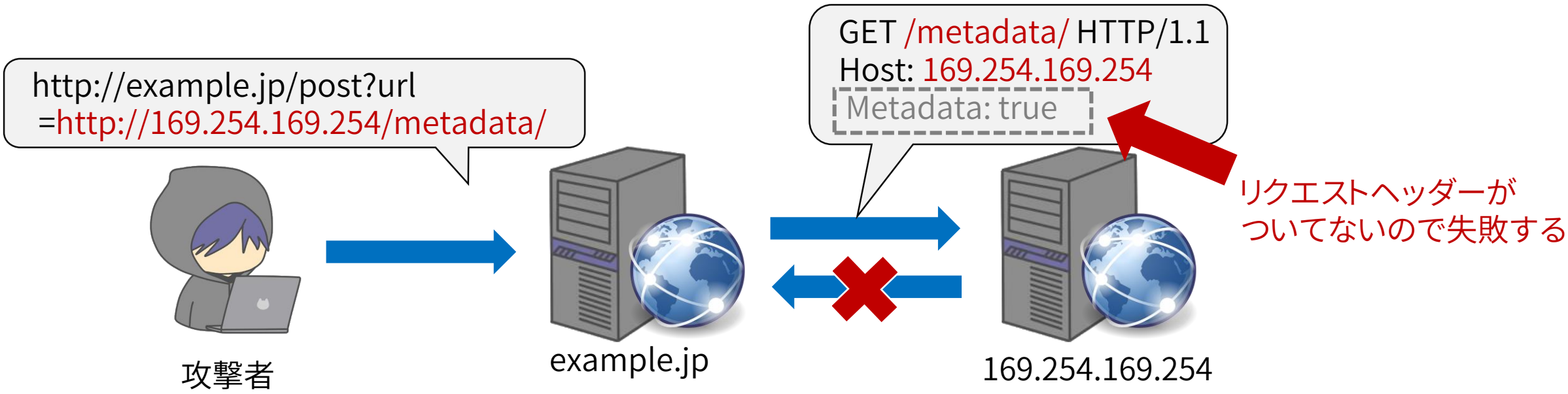
  - Azure

```
http://169.254.169.254/metadata/
※リクエストヘッダーに Metadata: true が必要
```

※これ以外のURLもあります

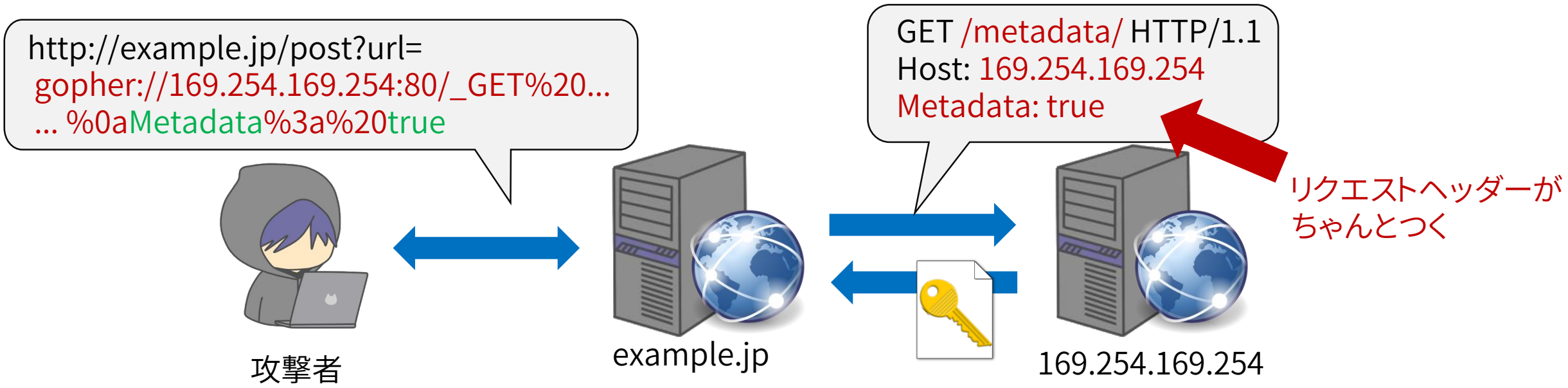
リクエストヘッダー付きのSSRF?

- アプリケーションサーバーからのリクエストには特殊なリクエストヘッダーは付与されないはず!?
  - リクエストヘッダーが付いていなければSSRFは失敗する



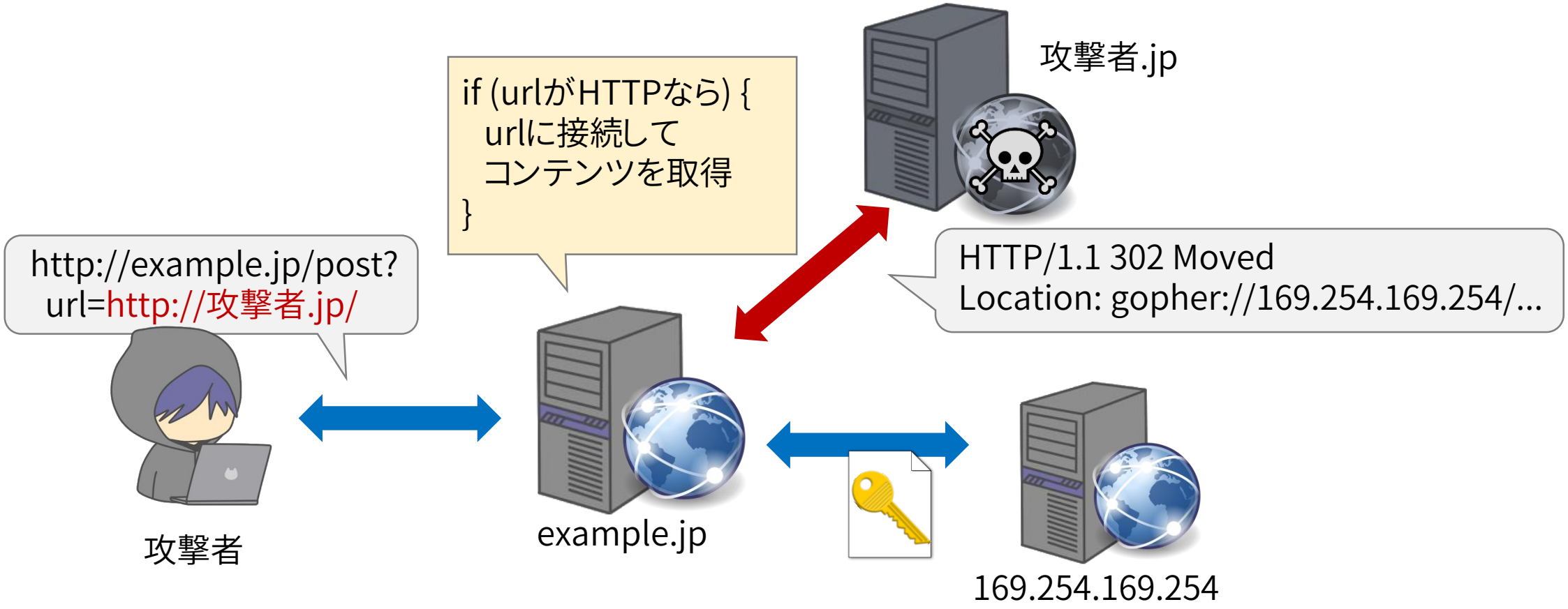
リクエストヘッダー付きのSSRF?

- gopher経由で80/tcpへ送ることで任意のリクエストヘッダーを送信することができる



gopherによるSSRF

- プロトコルをhttp/httpsに制限すればいいのか？
  - HTTPで罠サイトにつなぎ302でgoopherにリダイレクトすることも可能



gopherによるSSRF

- HTTPからgopherへのリダイレクトを受け入れるかどうかは実装による
  - curlコマンド - オプションによりgopherへもリダイレクト
    `curl -L http://redir-to-gopher.example.jp/`
  - PHP curl関数 - 設定によりgopherへもリダイレクト
    `curl_setopt($c, CURLOPT_REDIR_PROTOCOLS, CURLPROTO_ALL);`
  - Node.js httpモジュール - gopherへのリダイレクトは不可

まとめ

- SSRF概要
  - サーバーから攻撃者が指定した他のサーバー/サービスへリクエストが飛ぶ攻撃
- SSRFの脅威
  - 機密情報の漏えい、他のプロトコルへの攻撃、任意コード実行など
  - 近隣でどんなサービスが動いているかに依る
- 対策
  - 接続可能なURLをアプリケーション内で事前に定義しておく
  - iptables等でOS/コンテナレベルで接続先を制限する
  - URLの検証は保険的な対策にしかならない(やらないよりはマシ)

質問?

hasegawa@securesky-tech.com

@hasegawayosuke

http://utf-8.jp/