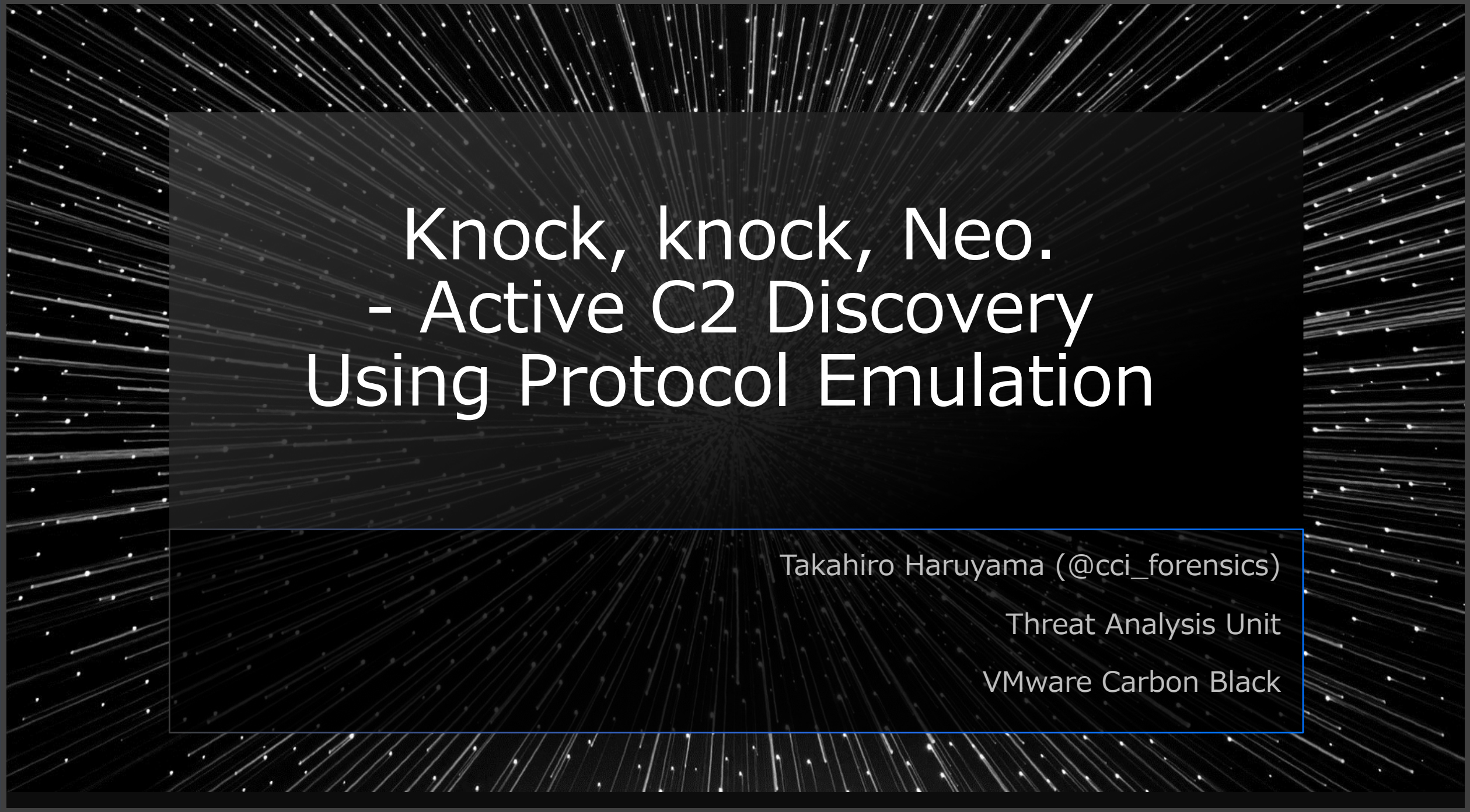
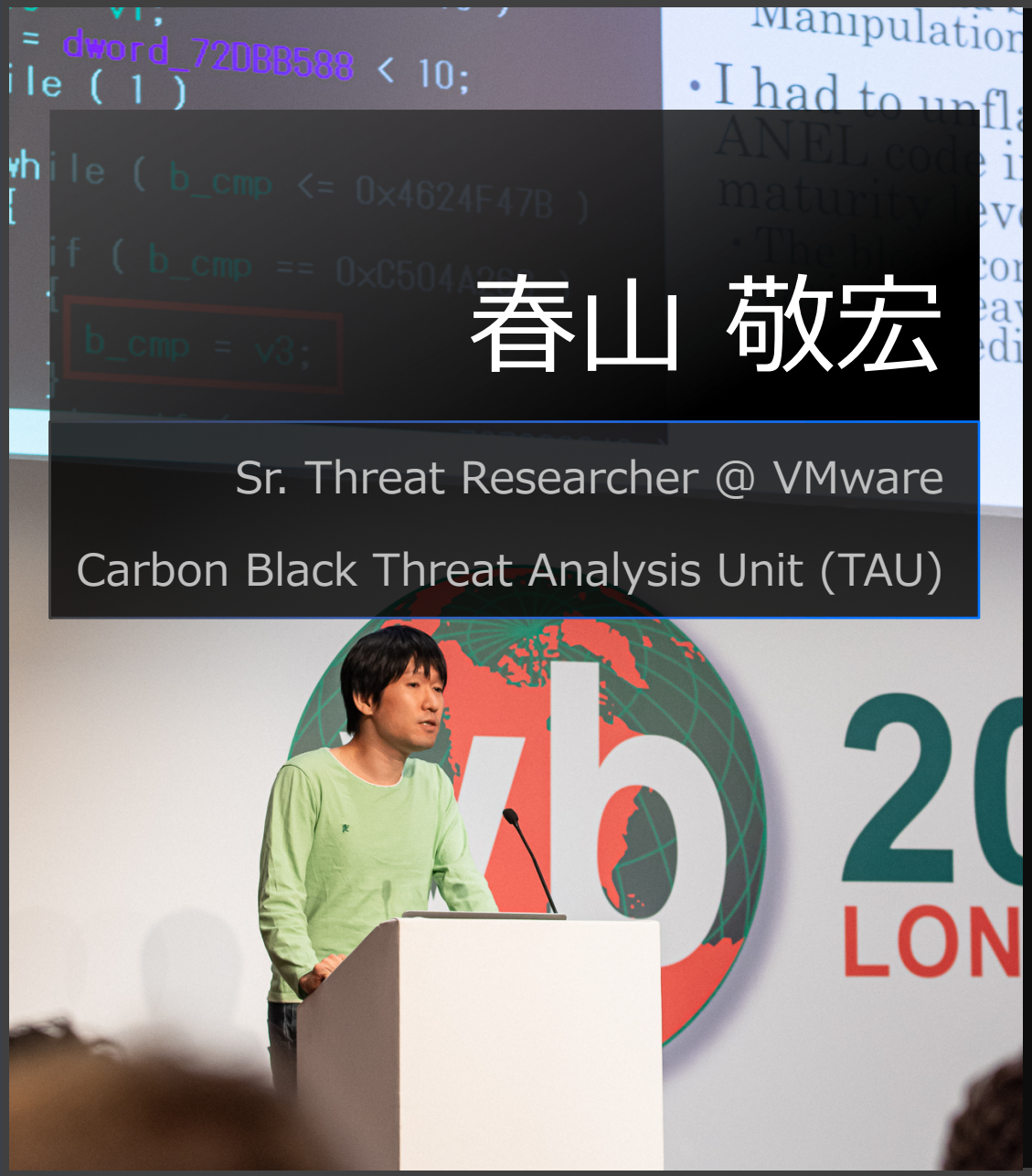


Knock, knock, Neo.
- Active C2 Discovery Using Protocol Emulation

Takahiro Haruyama (@cci_forensics)

Threat Analysis Unit

VMware Carbon Black



- サイバーエスピオナージを目的とした標的型攻撃の分析に従事
- 過去のリサーチ例
  - フォレンジック解析
    - メモリフォレンジクス
    - アンチフォレンジクス
  - マルウェア解析
    - PlugX
    - Winnti
  - リバースエンジニアリング
    - バイナリ差分解析
    - コンパイラレベル難読化の解除
  - etc..

モチベーション

- レピュテーションに頼るC2のIoCの限界
  - 欲しいのは、IoCではなくIoA
- 独自のテレメトリーを持ちたい
  - IRのチームを組織内に持っていない、組織のテレメトリーが成熟していない
- 何をターゲットにするか？
  - 当時リサーチ済み（HydSeven NetWire [1], Winnti 4.0 [2]）
  - より多くの脅威アクターに使われる＆標的型攻撃で使われるもの

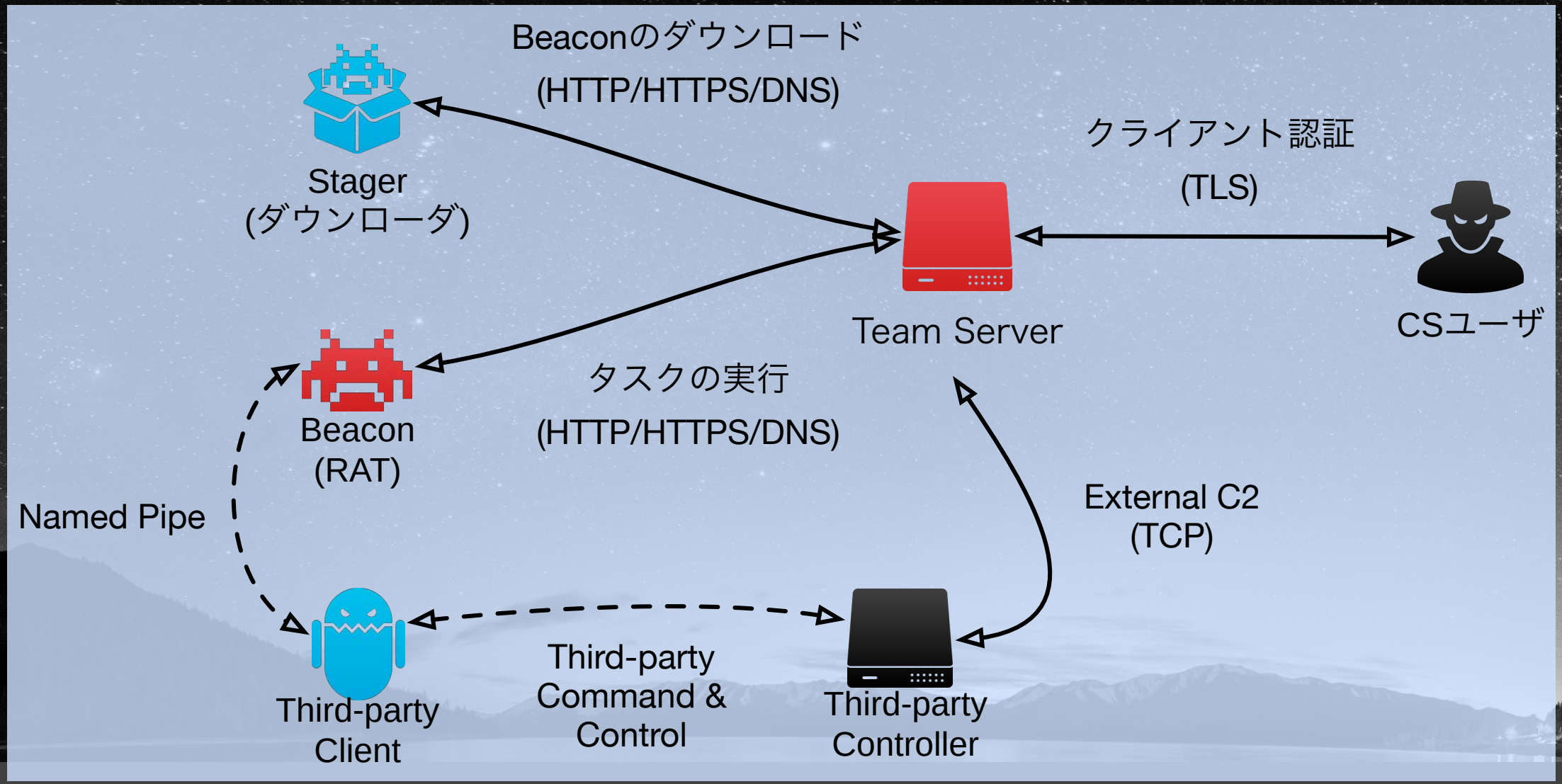
  → Cobalt Strike [3]

本日の構成

- Cobalt Strikeの各プロトコル詳細
- スキャナの実装
- 収集した脅威情報の活用
- まとめ


Cobalt Strikeの
各プロトコル詳細

プロトコル概観


Beaconのダウンロード
(HTTP/HTTPS/DNS)
Stager
(ダウンローダ)
クライアント認証
(TLS)
Team Server
CSユーザ
タスクの実行
(HTTP/HTTPS/DNS)
Beacon
(RAT)
Named Pipe
External C2
(TCP)
Third-party
Client
Third-party
Command &
Control
Third-party
Controller

分析したバージョンと利用ツール

- Cobalt Strike version 3.13 – 4.2

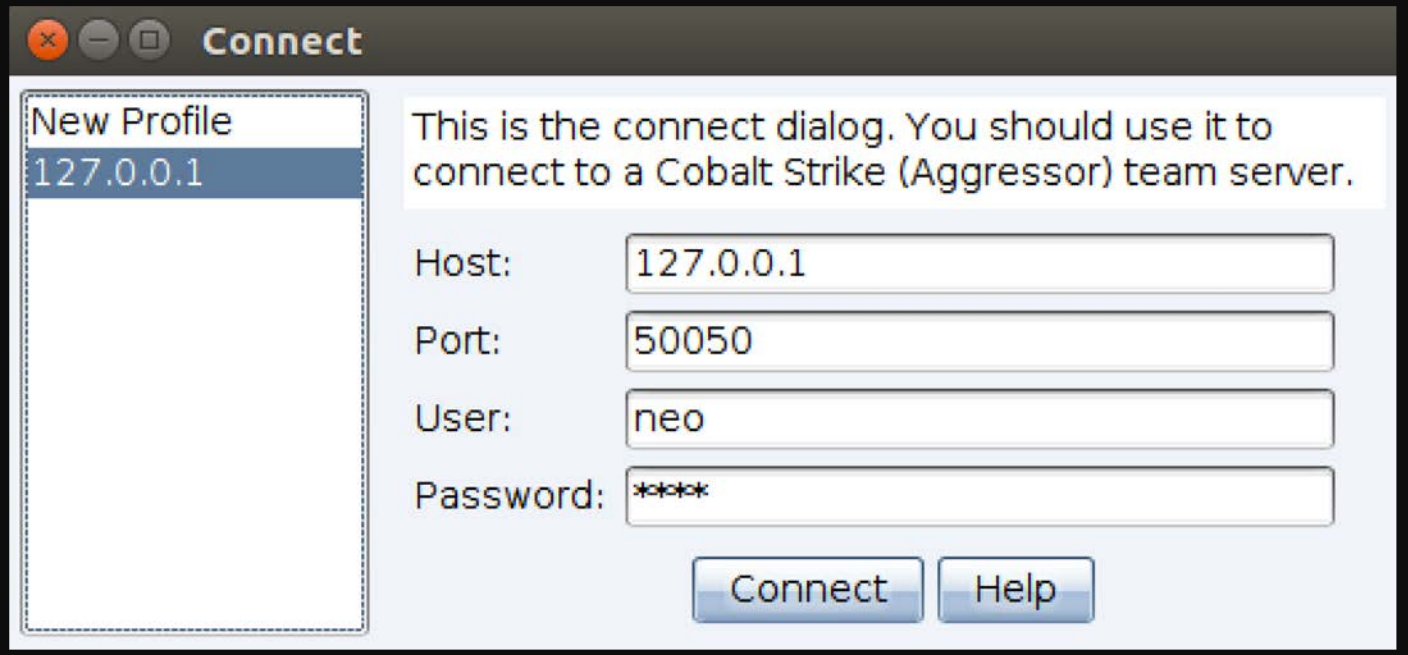
ターゲット	ツール
Team Server	Procyon
Beacon/Stager/shellcode	IDA Pro
通信データ	Wireshark

クライアント認証プロトコル

Cobalt Strikeの各プロトコル詳細

クライアント認証プロトコル

- TLSを通したパスワード認証
  - デフォルトポートは50050



プロトコルフォーマット

- 下記リクエストパケットに対して、サーバは４バイトの応答を返す
  - 0xcafe＝OK, 0＝NG

```
struct struc_auth_req_pkt
{
  int signature; // 0xbeef (big endian)
  char pass_size;
  struct_auth_payload payload;
};
struct struct_auth_payload
{
  char pass[8]; // flexible length (e.g., 8 bytes)
  char padding[248]; // padding up to 0x100 bytes (248 bytes in this case)
};
```

Stager プロトコル

Cobalt Strikeの各プロトコル詳細

HTTP/HTTPS Stager

- Team Serverに、checksum8ルール [4] を満たすURIパスへGETリクエストを送ることで、Beaconを含むシェルコードを取得
  - e.g., 3xQa, i4Jv, qB1y, ...
  - Malleable C2 profile [5] のuri_*で任意のURIを指定可能だが..
- Webクローラと思われるUser-Agentは弾く (lynx/curl/wget)

```
        while(True):
            tmp = random.sample(string.ascii_letters + string.digits, n)
            if sum([ord(x) for x in tmp]) % 256 == 93:
                return ''.join(tmp)
```

```
http-stager {
        set uri_x86 "/get32.gif";
        set uri_x64 "/get64.gif";
```

DNS Stager

- DNS TXTレコードをリクエストすることでシェルコードを取得
  - DNS Stagingはx86のみサポート
- DNS TXTレコードは255バイトしか持てないので、複数回リクエストを送って得たバイナリを結合
  - [オフセットを表す3文字].stage.[CSユーザが所有するドメイン名]
    - e.g., aaa.stage.hoge.com, baa.stage.hoge.com, ...
  - v4.1以降では任意のサブドメインを指定可能だが..
    - C2 profileのdns_stager_subhost

DNS Stager (続)

- 結合後、シェルコードがbeaconをデコードして実行
- v3までは、DNSはHTTPとのハイブリッドプロトコル
  - Stagerに”reverse_dns_txt”を選択しても、HTTPも動作した状態
  - v4.0以降は、Stager/Beacon共に完全なDNS実装

C2 Profileの
dns_stager_prepend
で先頭にゴミを追加可能

Beacon プロトコル

Cobalt Strikeの各プロトコル詳細

Beaconプロトコル基本フロー

1. Beaconがセッションメタデータ（感染端末から収集した情報）をTeam Serverに送信
   - HTTP/HTTPSでは設定に基づいて定期的に送信
   - DNSでは明示的なコマンド実行が必要（IDのみ定期的に送信）
     - e.g., checkinコマンド
2. Team Serverが応答
   - 実行させたいタスク（コマンド）があればそれを含める
3. Beaconがタスク情報を受信、その実行結果を送信

セッションメタデータの構造

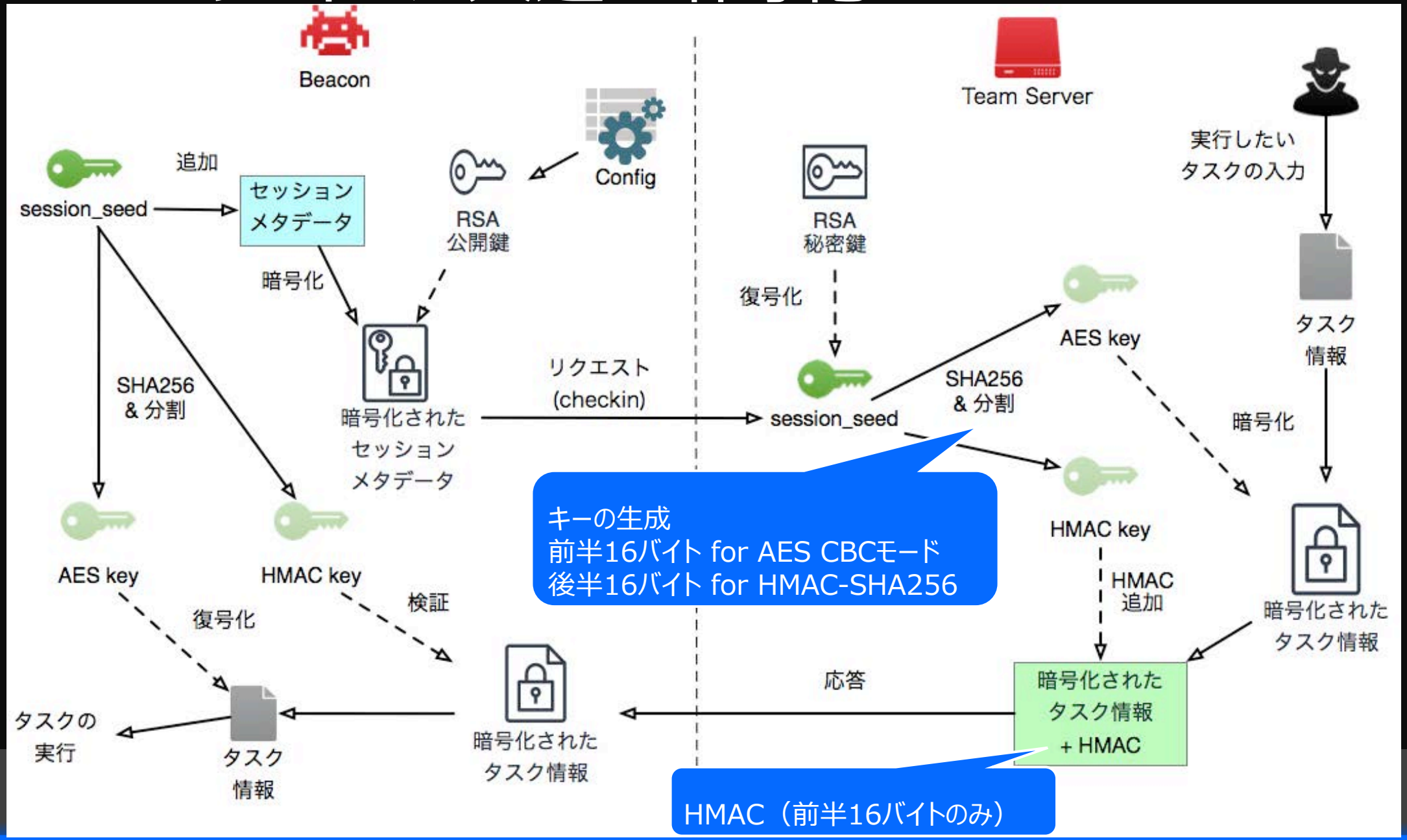
- バージョンの更新に伴い、可変長データが減る傾向
- beacon_IDで感染端末を一意に識別
- タスクの要求・応答データはsession_seedを元にAESで暗号化（後述）
- セッションメタデータはRSAの公開鍵で暗号化

```
struct struc_metadata_314 // version 3.14 metadata
{
  int big_signature; // 0xbeef
  int big_size;
  char session_seed[16];
  __int16 little_ANSI_codepage;
  __int16 little_OEM_codepage;
char victim_info; /* flexible length, tab delimited
                     "%d\t%d\t%d.%d\t%s\t%s\t%s\t%d\t%d",
                     beacon_ID_0to100000,
                     pid,
                     dwMajorVersion,
                     dwMinorVersion,
                     ipaddr,
                     computer_name,
                     user_name,
                     is64,
                     beacon_arch (1:x64, else:x86) */
};
```

```
struct struc_metadata_40 // version 4.0 metadata
{
  int big_signature; // 0xbeef
  int big_size;
  char session_seed[16];
  __int16 little_ANSI_codepage;
  __int16 little_OEM_codepage;
  int big_beacon_ID; // up to 0x7FFFFFFE
  int big_pid;
  __int16 big_port;
  char flags; // 1:None, 2:beacon_x64, 4:is64, 8:is_admin
  char victim_info; /* flexible length, tab delimited
                       "%d.%d\t%s\t%s\t%s\t%s",
                       dwMajorVersion,
                       dwMinorVersion,
                       ipaddr,
                       computer_name,
                       user_name,
                       process_name */
};
```

```
struct struc_metadata_41 // version 4.1/4.2 metadata
{
  int big_signature; // 0xbeef
  int big_size;
  char session_seed[16];
  __int16 little_ANSI_codepage;
  __int16 little_OEM_codepage;
  int big_beacon_ID; // up to 0x7FFFFFFE
  int big_pid;
  __int16 big_port;
  char flags; // 1:None, 2:beacon_x64, 4:is64, 8:is_admin
  char dwMajorVersion;
  char dwMinorVersion;
  __int16 big_build;
  char bytes[4]; // GetModuleHandleA/GetProcAddress
char gmh[4]; // GetModuleHandleA (low dword)
  char gpa[4]; // GetProcAddress (low dword)
  int little_IP_addr;
  char victim_info; /* flexible length, tab delimited
                       "%s\t%s\t%s",
                       computer_name,
                       user_name,
                       process_name */
};
```

Beaconプロトコル共通の暗号化



HTTP/HTTPS Beacon

- デフォルトではセッションメタデータはGET、タスク実行結果はPOSTで送信
- CSユーザは、Malleable C2 profile [5] を用いて、リクエストメソッドやURIパス・パラメータ、ヘッダの値、送信データのエンコーディング等を指定
- マルウェアアナリストは、Beaconに含まれるコンフィグをパースすることでその設定を把握できる (コンフィグの構造については後述)

```
buf C2_REQUEST (http-get.client) transform at 0x584:
- BUILD at 0x58a: 0 = metadata or id
- BASE64 at 0x592
- HEADER (Store data in an HTTP header) at 0x596: b'Cookie'
```

```
GET /activity HTTP/1.1
Accept: */*
Cookie: CY6sAk8xdKjnwXx2nl+2WZeXLjONY6qeOLlhhsE3ClwOdXIlrsGfavdfaqwknCpKeBtF8HMVHMAWSlKYJONqL
+OM17TzTJzh4KqC/RKp+RiKDvWnCbFANy54hq935YNzMGTLELA6SpD57IrHntvbc1dPSYn10Uxur0/IZLKjc40=
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; NP07; NP07)
Host: 172.16.24.2
Connection: Keep-Alive
Cache-Control: no-cache
```

DNS Beacon

- DNS Aレコードをリクエストして、タスク実行をポーリングする
  - [beacon ID].[csユーザが所有するドメイン名]
  - セッションメタデータは毎回送信しない
- タスクの有無に応じて応答のアドレス値が変化
  - タスク無 -> C2 profileのdns_idleに設定された値
    - デフォルトは0.0.0.0
  - タスク有 -> 上記値をDNS Beaconの“mode”に応じてxorした値

タスクダウンロード時のmode	プロトコル	サブドメイン
http (version 4で廃止)	HTTP	-
dns (version 3のデフォルト)	DNS Aレコード	cdn
dns-txt (version 4のデフォルト)	DNS TXTレコード	api
dns6	DNS AAAAレコード	www6

DNS Beacon (タスクのダウンロード)

- タスクがある場合、第４オクテットをxor
- xorする値の計算
  - ベース値は240
  - セッションメタデータの送信が必要 -> ベース値 |= 1
  - mode dns-txt ->ベース値 |= 2
  - mode dns6 ->ベース値 |= 4

Team Serverが
セッションメタデータ送信と、
タスクのダウンロード
(dns-txt mode) を要求

```
Standard query 0x97bb A 308b3dfe.cs42.test.com
Standard query response 0x97bb A 308b3dfe.cs42.test.com A 0.0.0.0
Standard query 0xab97 A 308b3dfe.cs42.test.com
Standard query response 0xab97 A 308b3dfe.cs42.test.com A 0.0.0.243
```

DNS Beacon (タスクのダウンロード、続)

- e.g., dns-txt modeの場合、サブドメインにapiをつけてリクエスト
  - Aレコード（タスク情報のデコード後のサイズ）
  - TXTレコード（タスク情報）

dns_idleとサイズをxor

```
Standard query 0x3831 A api.0e3d3d2b.1ad07cfe.cs42.test.com
Standard query response 0x3831 A api.0e3d3d2b.1ad07cfe.cs42.test.com A 0.0.0.48
Standard query 0x481e TXT api.1e3d3d2b.1ad07cfe.cs42.test.com
Standard query response 0x481e TXT api.1e3d3d2b.1ad07cfe.cs42.test.com TXT
```

```
Type: TXT (Text strings) (16)
Class: IN (0x0001)
Time to live: 1
Data length: 65
TXT Length: 64
TXT: 7RNFcrJV6LL8XquyiwxbS3de1jKLFpCzDORT0x9RAAvrGg9fbeUziuKZynYcW3K1
```

暗号化されたタスク情報
(base64-encoded)

DNS Beacon (データの送信)

- セッションメタデータ or タスク実行結果を、DNS Aレコードのリクエストとして送信
  - データはサブドメインに含まれる

セッション
メタデータ
送信

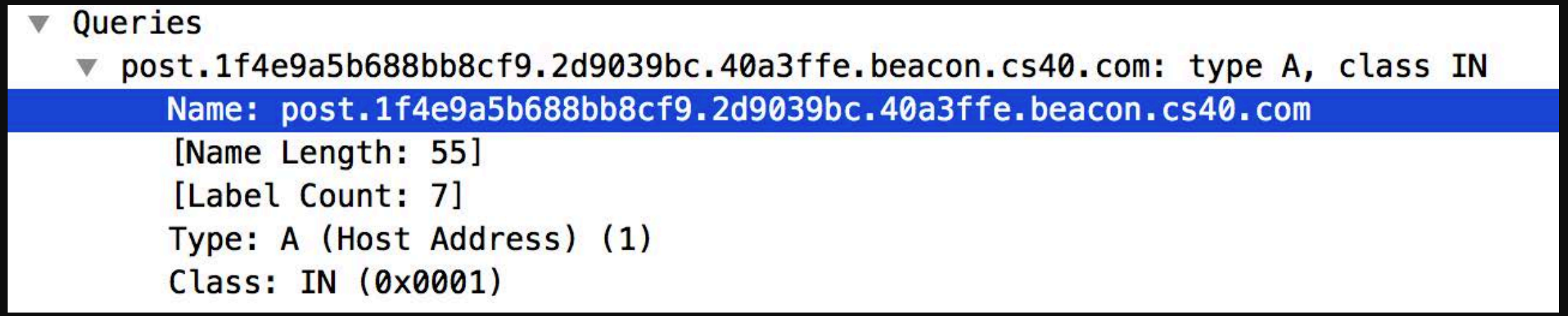
タスクの
ダウンロード

タスク実行
結果送信

```
Standard query 0x83ee A www.180.0d8f494b.1ad07cfe.cs42.test.com
Standard query response 0x83ee A www.180.0d8f494b.1ad07cfe.cs42.test.com A 0.0.0.0
Standard query 0xe8ce A www.41bfa81c1e641296221af4d2ea402e1a4be72f6a2.cfe75b630a286b69
Standard query response 0xe8ce A www.41bfa81c1e641296221af4d2ea402e1a4be72f6a2.cfe75b6
Standard query 0xa9c5 A www.13d46c1532710a55bcdef88fc0ab1df19b7a7ba90663531cc.2d8f494b
Standard query response 0xa9c5 A www.13d46c1532710a55bcdef88fc0ab1df19b7a7ba90663531cc
Standard query 0x3831 A api.0e3d3d2b.1ad07cfe.cs42.test.com
Standard query response 0x3831 A api.0e3d3d2b.1ad07cfe.cs42.test.com A 0.0.0.48
Standard query 0x481e TXT api.1e3d3d2b.1ad07cfe.cs42.test.com
Standard query response 0x481e TXT api.1e3d3d2b.1ad07cfe.cs42.test.com TXT
Standard query 0x996e A post.130.0448209c7.1ad07cfe.cs42.test.com
Standard query response 0x996e A post.130.0448209c7.1ad07cfe.cs42.test.com A 0.0.0.0
Standard query 0x249b A post.24cfc87073aca60d6d6f7241579129491ad24f76a0e214629.da2442
Standard query response 0x249b A post.24cfc87073aca60d6d6f7241579129491ad24f76a0e2146
```

DNS Beacon (データの送信、続)

- データ送信時のサブドメインの形式



post = タスク実行結果送信

1f4e9a5b688bb8cf9 = [データに使われるサブドメインの数を表す 1 文字] + [データ]

2d9039bc = DNS cache data ID

40a3ffe = Beacon ID

External C2 プロトコル

Cobalt Strikeの各プロトコル詳細

External C2 プロトコル

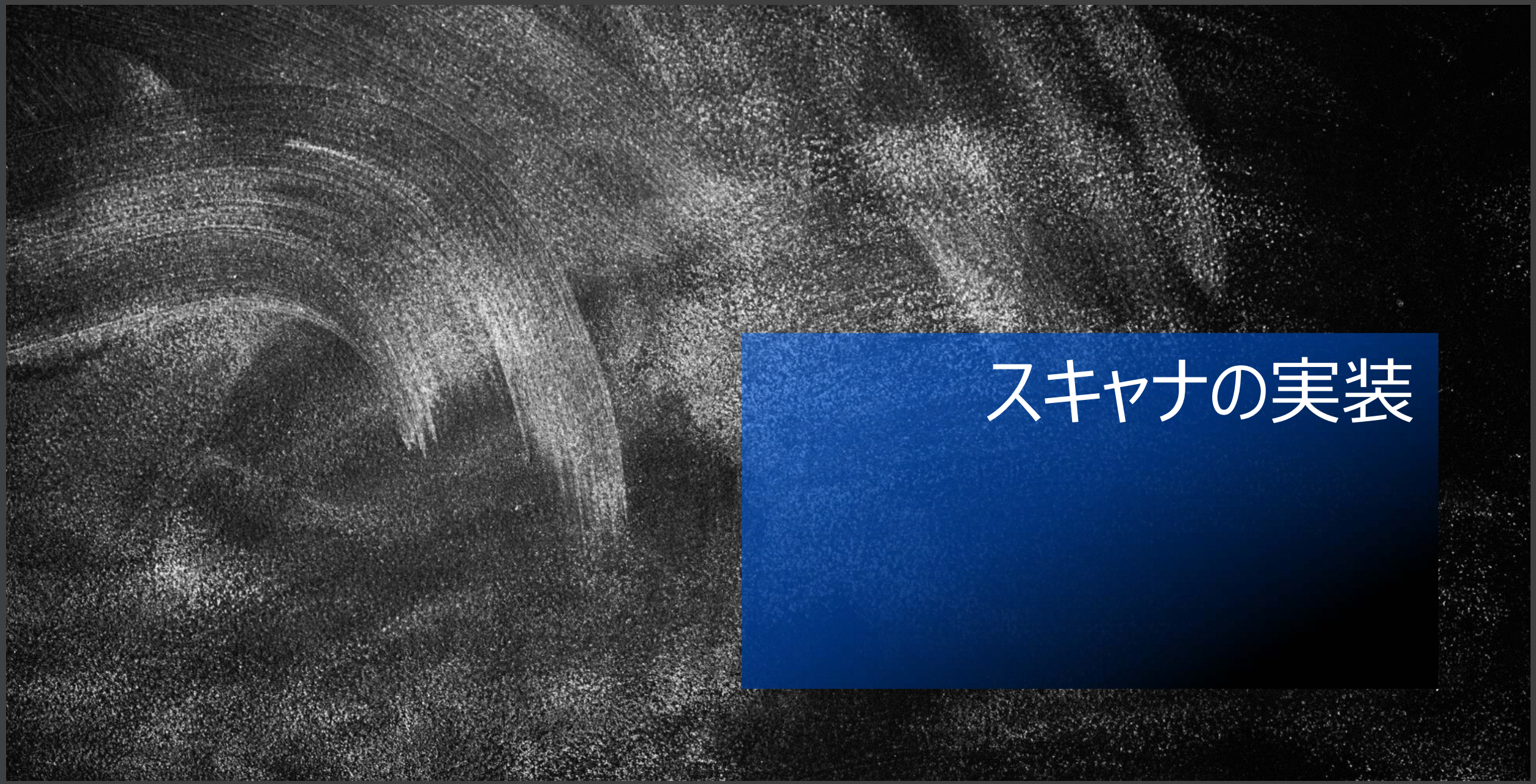
- third-party C2プロトコルはユーザが自由に決めて良い
  - e.g., DNS over HTTPS [6]
- third-party controllerとTeam Serverの間のプロトコルはTCP
  - frameと呼ばれるシンプルなフォーマットを使う
    - バッファサイズ (リトルエンディアン４バイト) + バッファ
  - 通信開始時、”キー＝値”の文字列バッファを含むオプション情報を送信

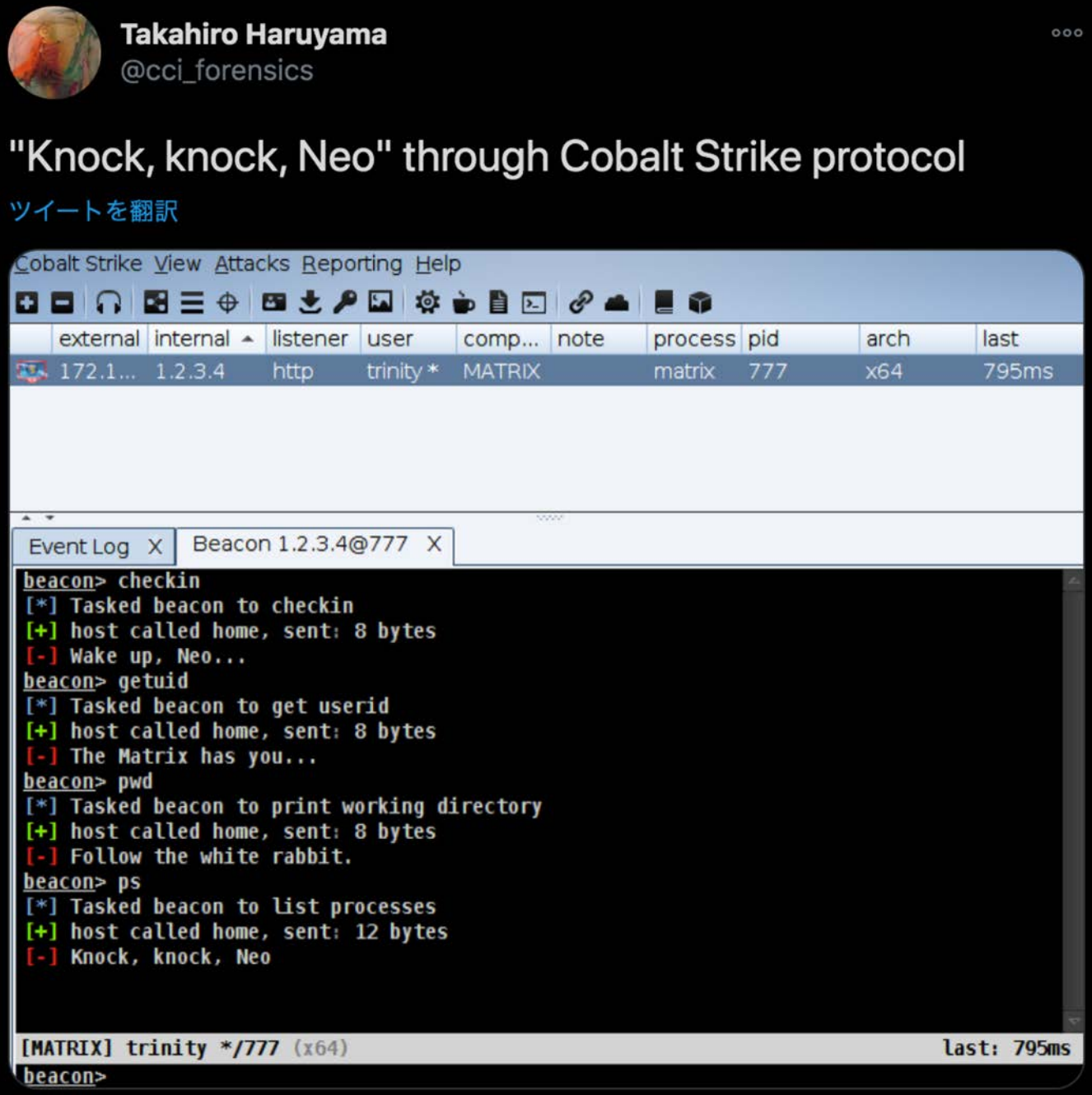
キー	説明	デフォルト値
arch	Beaconのアーキテクチャ (x86 or x64)	x86
pipename	名前付きパイプの名前	externalc2
block	応答前にタスク情報を待つ時間 (ミリ秒)	100
type	未使用	rdll

External C2 プロトコル (続)

- オプション情報の送信後、third-party controllerが“go” frameを送信することで、Beaconをダウンロード
- セッションメタデータ、タスク情報とその結果のやりとりも、同じframe形式でやりとりされる

```
00000000  08 00 00 00 61 72 63 68  3d 78 36 34                ....arch =x64
0000000C  0f 00 00 00 70 69 70 65  6e 61 6d 65 3d 66 6f 6f   ....pipe name=foo
0000001C  62 61 72 09 00 00 00 62  6c 6f 63 6b 3d 31 30 30   bar....b lock=100
0000002C  02 00 00 00 67 6f                                  ....go
    00000000  00 de 03 00                                    ....
    00000004  4d 5a 41 52 55 48 89 e5  48 81 ec 20 00 00 00 48   MZARUH.. H.. ...H
    00000014  8d 1d ea ff ff ff 48 89  df 48 81 c3 a0 49 01 00   ......H. .H...I..
    00000024  ff d3 41 b8 f0 b5 a2 56  68 04 00 00 00 5a 48 89     A    V h    ZH
```


スキャナの実装


Takahiro Haruyama
@cci_forensics
"Knock, knock, Neo" through Cobalt Strike protocol
ツイートを翻訳
Cobalt Strike View Attacks Reporting Help
external internal listener user comp... note process pid arch last
172.1... 1.2.3.4 http trinity * MATRIX matrix 777 x64 795ms
Event Log X
Beacon 1.2.3.4@777 X
beacon> checkin
[*] Tasked beacon to checkin
[+] host called home, sent: 8 bytes
[-] Wake up, Neo...
beacon> getuid
[*] Tasked beacon to get userid
[+] host called home, sent: 8 bytes
[-] The Matrix has you...
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[-] Follow the white rabbit.
beacon> ps
[*] Tasked beacon to list processes
[+] host called home, sent: 12 bytes
[-] Knock, knock, Neo
[MATRIX] trinity */777 (x64)
last: 795ms
beacon>
午前10:37 · 2020年6月15日 · Twitter Web App

本手法のアプローチ

- Stagerプロトコルにフォーカスする [4]
  - 静かに実施する
    - BeaconプロトコルのエミュレーションはCSユーザに気づかれやすい
  - 認証のようなセキュリティ機構を回避しない
    - クライアント認証プロトコルへのスキャンはログイン試行の誤解を招く
  - 誤検知を避ける
    - HTTPレスポンスやTLSのハンドシェイク、DNS Aレコードの応答に基づく単純な判断では誤検知を 0 にできない

本手法のアプローチ（続）

- StagerプロトコルでBeaconをダウンロードし、そのコンフィグを抽出する
- Stagerを無効にするC2 Profile設定（version 3.5.1以降）
  - HTTP/HTTPS/DNS Stagerが無効になる
  - External C2のstagingは無効にできない
- 無効にした場合は、Beaconを配置する仕組みが別途必要
  - 大半のTeam Serverは利便性のために有効にしたまま？

```
set host_stage "false";
```

従来手法

概要	公開元	対象プロトコル	誤検知	備考
NanoHTTPDの挙動に基づく検出	Fox-IT [7]	HTTP/HTTPS	あり (NanoHTTPD)	3.13で修正済み
HTTPレスポンスヘッダに基づく検出	ZoomEye [8] FireEye [9]	HTTP/HTTPS	あり	
TLS Server Helloレスポンスに基づく検出	Salesforce [10]	HTTPS	あり (同じバージョンのJava web server)	JARMを公開
Stagerプロトコルを使ったコンフィグ抽出	[11] [12] [13] [14] [15]	HTTP/HTTPS	なし	450-781 (1スキャン)

DNS/ExternalC2未対応、HTTP/HTTPS Stagerは後で性能を比較

コンフィグパーサーの実装

スキャナの実装

Beaconのコンフィグ構造

```
struct struc_config_param_header {
    __int16 id; // 0 is end
    __int16 type; // 1:word, 2:dword, 3:buf
    __int16 size;
};
```

- ネストしたバイナリ構造
  - 各コンフィグ値は６バイトのヘッダを持つ
  - typeがbufの場合、以下のいずれか
    - 文字列
    - transform data
      - HTTP設定
      - プロセスインジェクション設定

transform data

- プロセスインジェクション transform data はシンプル
  - インジェクションコードのprepend/append
  - 各々は、４バイトサイズ情報＋データ
- HTTP transform data は若干カオス
  - コンフィグと異なり、各transform値のヘッダは４バイトのIDのみ
    - IDごとにタイプ・サイズが異なる、ヘッダのみのIDもあり
  - transform IDが同じでも、異なるtransform dataであれば、その形式が異なってくる
    - e.g., サーバ応答 (C2_RECOVER) vs セッションメタデータ送信 (C2_REQUEST)

既存公開パーサー [16][17][18] との違い

- 最新バージョンのコンフィグ値に対応、各HTTP transform dataを完全にパース

```
...
buf C2_RECOVER (http-get.server.output) transform at 0x47e:
- PRINT (Send data as transaction body) at 0x484
- APPEND at 0x488: size = 1522 (0x5f2)
- PREPEND at 0x490: size = 84 (0x54)
- PREPEND at 0x498: size = 3931 (0xf5b)
- BASE64URL (URL-safe Base64) at 0x4a0
- MASK (XOR mask w/ random key) at 0x4a4
buf C2_REQUEST (http-get.client) transform at 0x584:
- _HEADER at 0x58a: b'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
- _HEADER at 0x5d9: b'Referer: http://code.jquery.com/'
- _HEADER at 0x601: b'Accept-Encoding: gzip, deflate'
- BUILD at 0x627: 0 = metadata or id
- BASE64URL (URL-safe Base64) at 0x62f
- PREPEND at 0x633: b'__cfduid=‘
- HEADER (Store data in an HTTP header) at 0x644: b'Cookie’
...
```

jquery C2 profile [20]

に対するパース結果

Internet-wideに Team Serverを 見つけるために

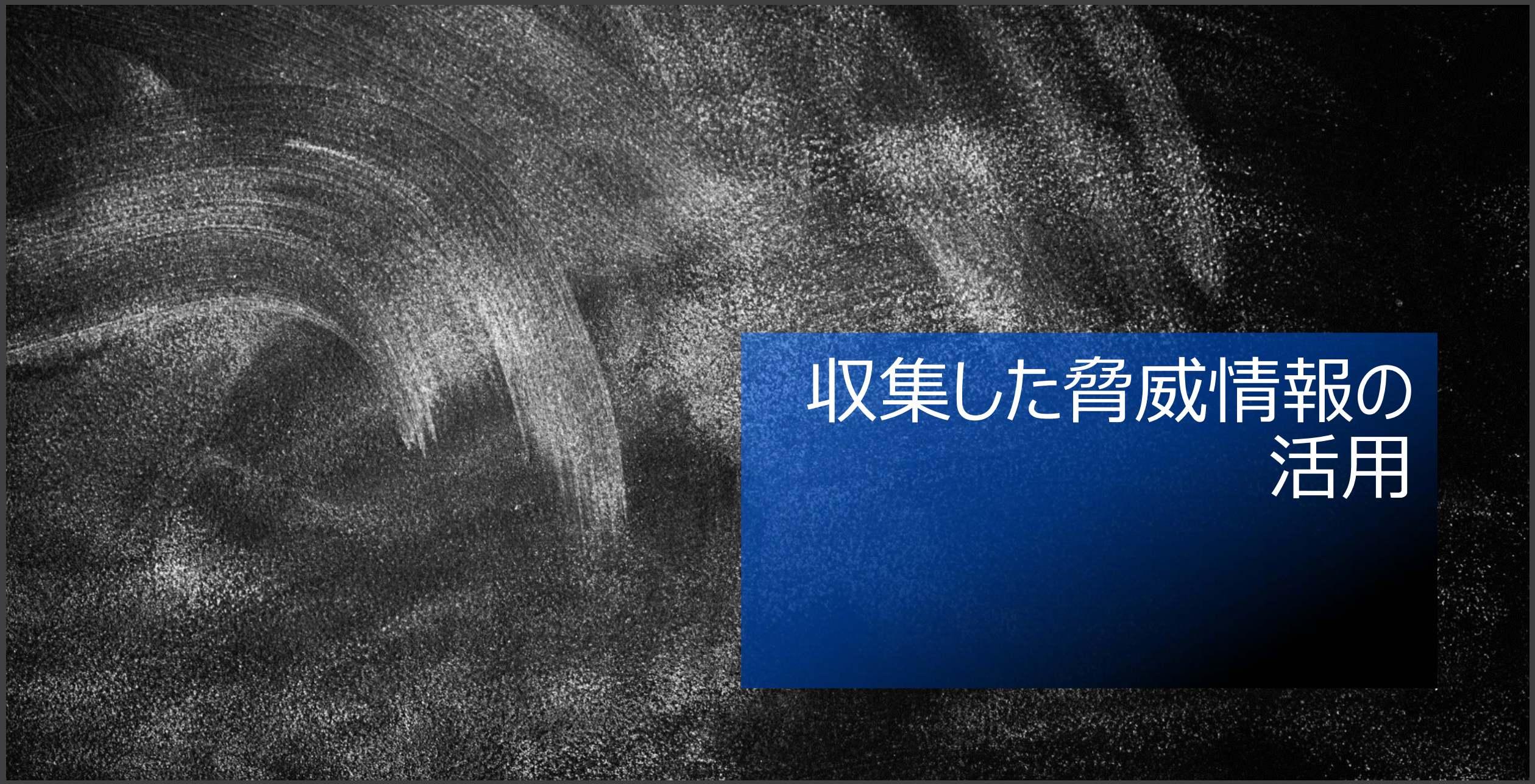
スキャナの実装

Internet-wideにTeam Serverを見つけるために

- ターゲットプロトコル・ポートの選定
  - DNS以外はポートを自由に変更可能
  - HTTP/HTTPS
    - 過去のインシデント、公開レポート
    - 過去のBeacon検体からのパース結果 [20]
    - twitterでの検索
    - 他組織によるStagerプロトコルを使ったコンフィグ抽出 [14][15]
  - External C2
    - とにかく実際にテストしてみる
  - ターゲットを増やせば増やすほど取得数は増えるが、インターバルが長くなる

Internet-wideにTeam Serverを見つけるために (続)

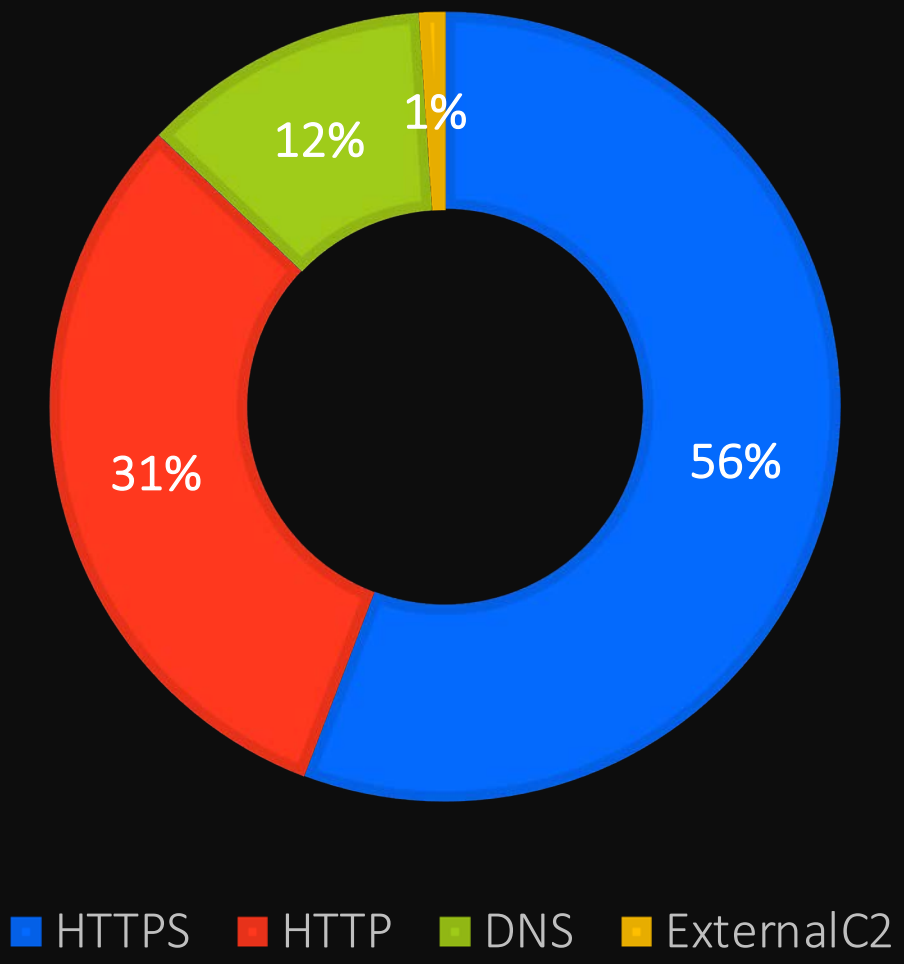
- オープンホストリストの取得
  - リアルタイム性を求めるのであれば自ら生成 (e.g., Zmap [21])
  - 省エネで行きたいならサードパーティが配布しているリストを使う (e.g., Rapid7 [22])
    - データ更新間隔の長さを許容できるかで判断
- 送信元の匿名化
  - TCPベースのリクエストはTorを通す
  - UDPベースのリクエストは匿名化VPNを通す


収集した脅威情報の
活用

観測データ

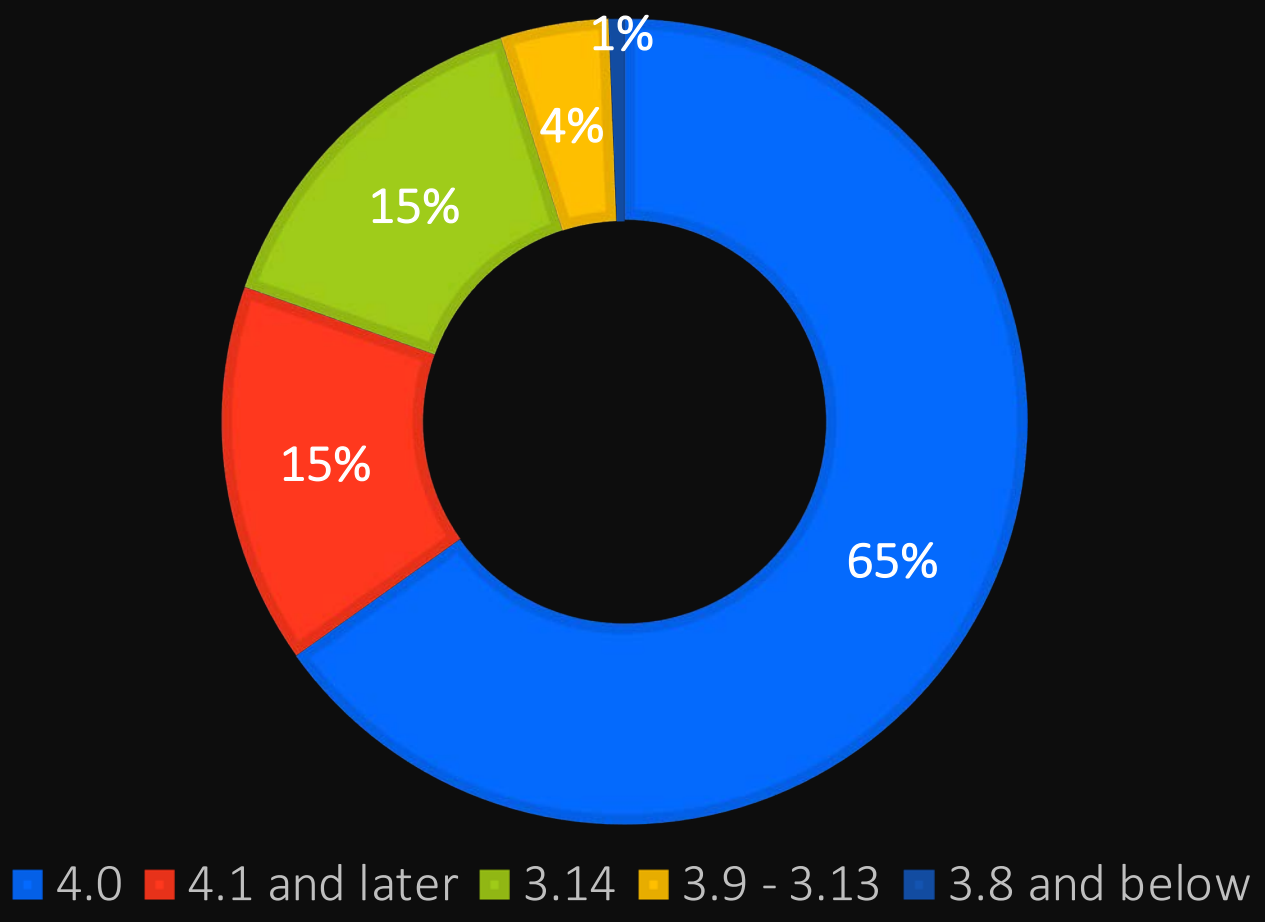
収集した脅威情報の活用

累計とプロトコル別割合



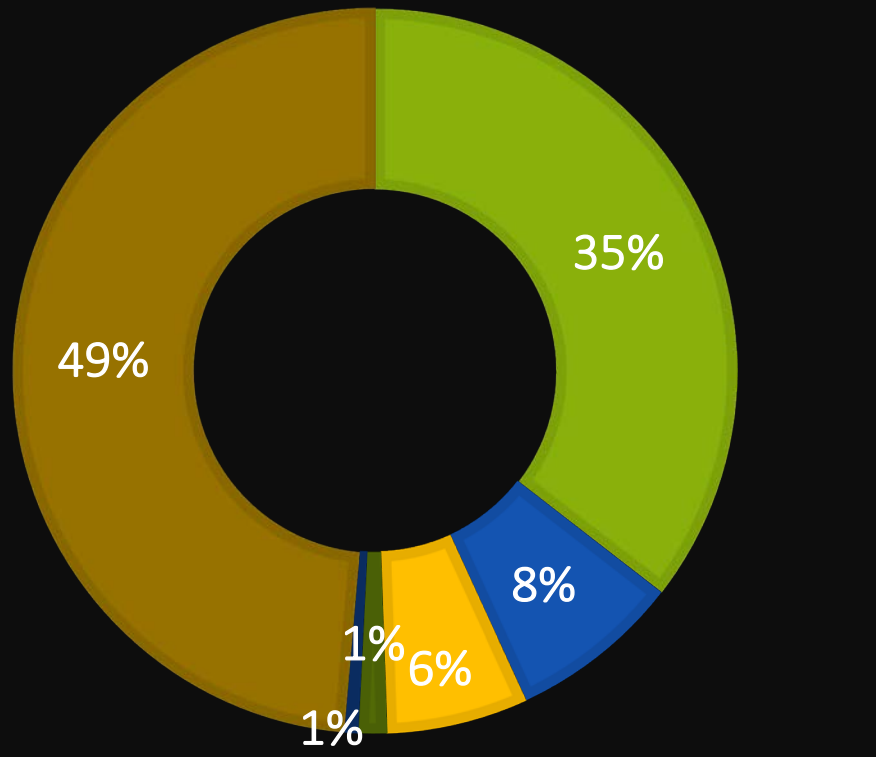
- 観測期間
  - 2020/02から現在まで
- Team Serverの累計数
  - 6,336 (IP/portユニーク)

バージョン別割合



- コンフィグ値に基づく分類
  - version 3.8 以下は、WATERMARK (Customer ID) なし [19]
  - version 4.1 以降は、実際は多い可能性あり

リーク・クラック版の割合



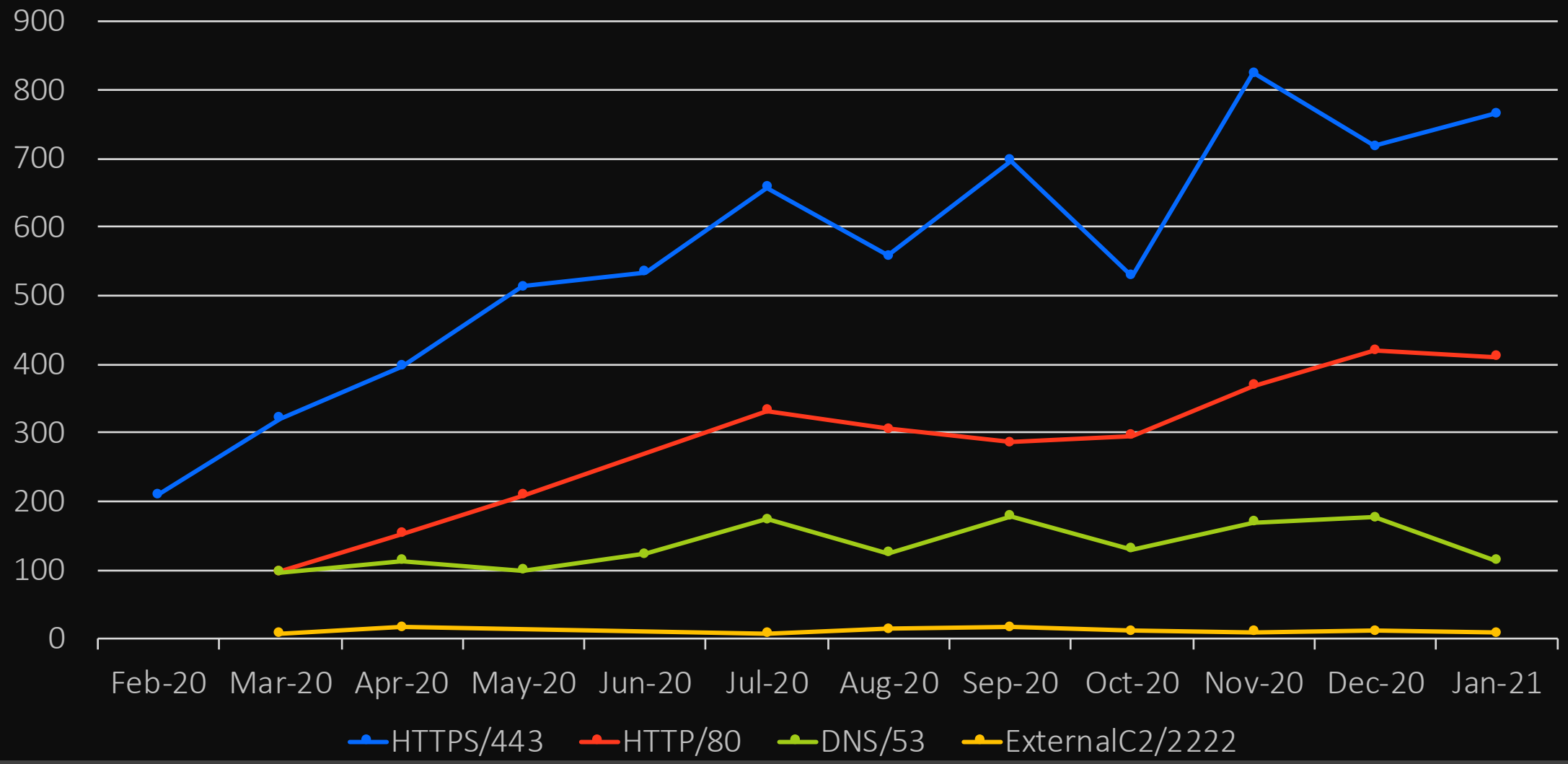
- 以下の４つのCustomer IDは、リーク・クラック済み
  - 1873433027
  - 305419896
  - 16777216
  - 1359593325
- customer ID = 0
  - トライアル版のはずだが..

クラック済みトライアル版

- トライアル版の場合、version 3.6以降は、Beaconのタスク実行の暗号化がoffになる [23]
  - コンフィグ値の SETTING_CRYPTO_SCHEME = 1
- トライアル版＆3.6以降で、当該値が0になっているTeam Serverがin the wildに多数存在

```
...
word CRYPTO_SCHEME (1 = disable encryption) at 0x746: 0 (0x0)
...
dword WATERMARK at 0x798: 0 (0x0)
...
```

１スキャン毎の取得数の推移（well-knownのみ）



HTTP/HTTPS 1スキャン性能比較 (2020/12)

Michael Koczwara @MichalKoczwara · 2020年12月13日

I mapped active Cobalt Strike servers in the wild (over 450). Some of them could be legit Red Team Ops. However, the majority probably belongs to APT/Ransomware groups.

docs.google.com/spreadsheets/d

[11] 450
複数ポートをスキャン
(HTTP/HTTPS)

¬ whickey @notwhickey · 2020年12月5日

I attempted to extract #CobaltStrike Beacon configs from #JARM scan data (11-25 & 12-04) collected by @silascutler on port 443 for hash `07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1`
I extracted 503 configs in JSON; you should block C2s:

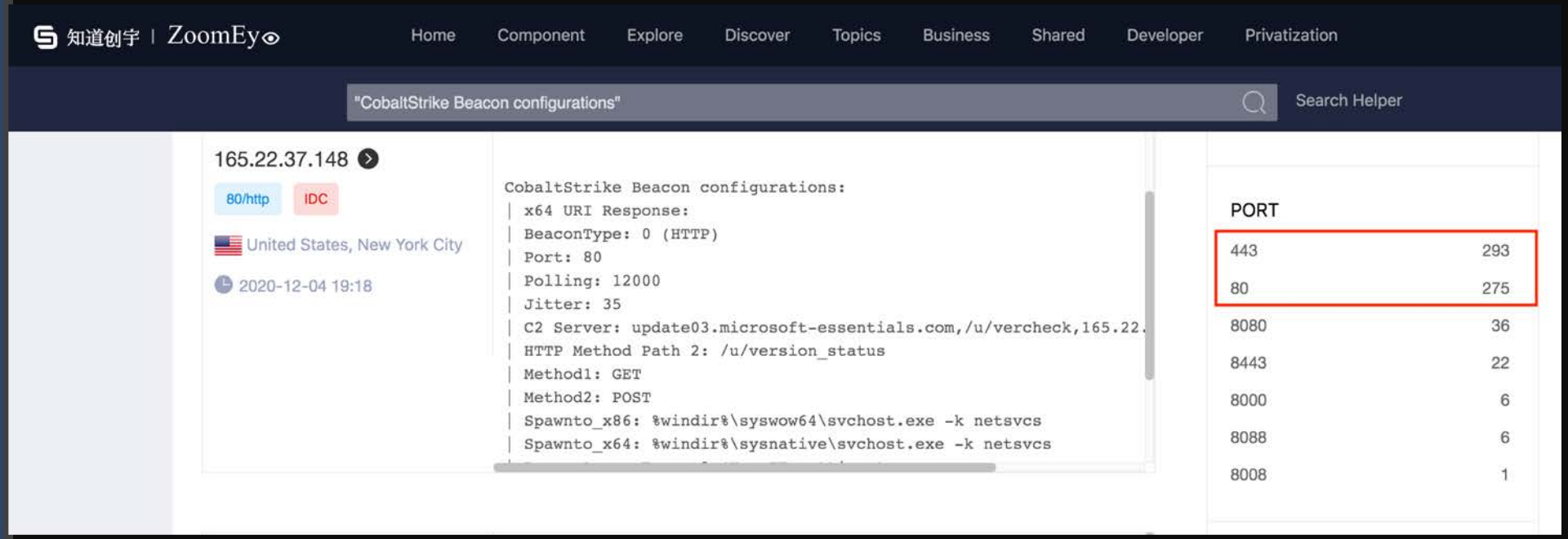
[12] 503
443のみスキャン
JARMベース
(HTTPS)

Based on the same code, I have scanned the 3424 servers identified with JARM in Shodan, I have scanned them all using this script and found 520 serving Cobalt Strike beacons.

https://www.randhome.io/blog/2020/12/20/analyzing-cobalt-strike-for-fun-and-profit/

[13] 520
複数ポートをスキャン
JARMベース
(HTTPS)

HTTP/HTTPS 1スキャン性能比較 (2020/12、続)



[14] 568
443 & 80スキャン
(HTTP/HTTPS)

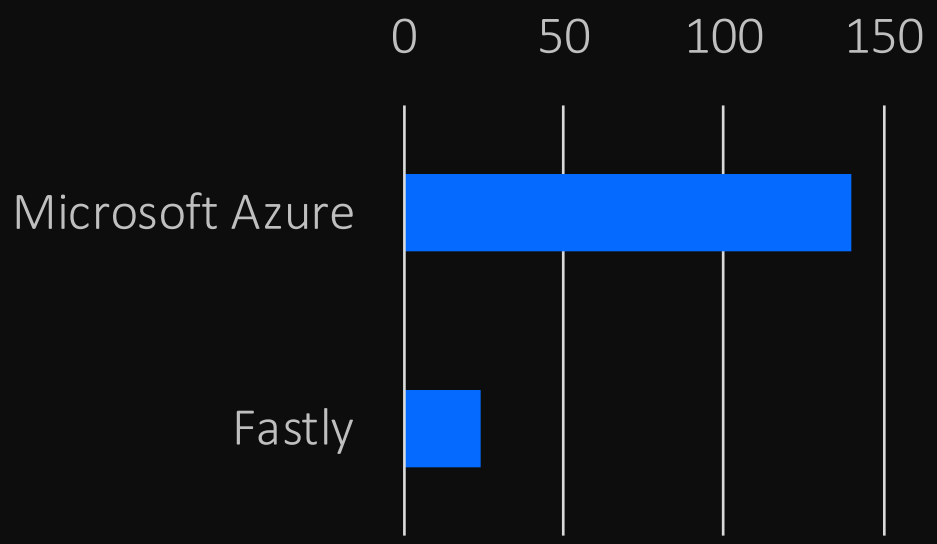
[15] 781
複数ポートをスキャン
(HTTP/HTTPS)

ドメインフロンティング

- CDN等でのルーティングを悪用するテクニック [42]
  - HTTPS通信で、DNSクエリとTLSのSNIフィールドに無害なCDNのドメイン、HTTPのHostヘッダに攻撃者のサーバドメインを指定する
  - 双方のドメインが同じCDNから提供されていれば、HTTPSの接続先が無害なCDNであっても、実際の通信を攻撃者のサーバに転送できる
- Cobalt StrikeでのHostヘッダ設定
  - Malleable C2 Profile [43] -> HTTP transform data
  - GUIメニュー（C2 Profile設定を上書き）-> コンフィグ値 (ID=54)

```
...
buf DOMAINS (C2 URLs) at 0x13a: ajax.microsoft.com,/gp/cerberus/gv
...
buf C2_REQUEST (http-get.client) transform at 0x412:
- _HOSTHEADER at 0x418: 'Host: cdndev94.azureedge.net'
...
```

観測データで見るドメインフロンティング



期間中観測された
ドメインフロンティングされている
Team Serverの数

- 過去の古いBeacon検体の調査 [44]によると、以下の悪用がメジャーだったが..
  - Amazon CloudFront (cloudfront.net)
  - Google App Engine (appspot.com)
  - Microsoft Azure (azureedge.net)
- AmazonとGoogleは2年前に対策済み [45][46]
  - 現在１、２件程度でほぼ絶滅
- Microsoft Azureが、観測期間中のドメインフロンティング悪用のほとんどを占める
  - Azure = 140, Fastly = 24

なぜMicrosoftは対策しない？

Tim MalcomVetter ❤️ @malcomvetter · 2019年3月21日

Opinion: Red Teams use domain fronting more than FIN/APT groups.

Why? Because red teams have to follow more rules when acquiring callback domains. We can't reuse third party victims.

Also, because categorization and aging are hard, requiring discipline & planning.

11 30 112

Andrew Thompson
@anthomsec

返信先: @malcomvetterさん

I agree with your opinion. I see domain fronting with Cobalt Strike and Empire use regularly, and almost always it is authorized actors. We can also consider the fact that a lot of domain front-able domains exist within "legitimate" IP space, where bad largely doesn't operate.

コンフィグ値に基づいた攻撃者のグルーピング

収集した脅威情報の活用

コンフィグ値に基づいた攻撃者のグルーピング

- コンフィグ値の共通点から、特定の脅威アクター・キャンペーンに結びつくクラスタを作れないか？
- 有効な情報
  - PUBKEY
  - WATERMARK
  - その他
    - HTTPヘッダ情報 (USERAGENT, HOST_HEADER, transform data)
    - PROCINJ_STUB (jarファイルパスのハッシュ値)

PUBKEY

- セッションメタデータ暗号化に使われる公開鍵
  - コンフィグ内にDERフォーマットで保存
- 元となるRSAのキーペアファイルは、Team Serverのディレクトリに .cobaltstrike.beacon_keys として初ログイン時に生成
- 仮にそのディレクトリが別ホストにコピーされた場合、そのキーペアファイルは使い回されることになる

→PUBKEYが同一なTeam Serverは、同じ脅威アクターによって管理されている可能性が高い

WATERMARK

- customer ID [19]
  - authorization file (cobaltstrike.auth) から抽出
  - version 3.9以降で実装された仕組み
  - アップデートプログラムを実行するたびに値が変わる
- リーク・クラック版でなければ、同一アクターであることを保証
- リーク・クラック版であっても、レッドチームか攻撃者かを区別可能

公開事例1: Mustang Panda [24] (2020/03)

- COVID-19をテーマとしたファイルを囮に、Cobalt Strikeを感染させるエスピオナージキャンペーン
- 公開Team ServerのPUBKEYに一致: 185.62.56[.]217
  - 資料作成時点で未だアクティブ

公開事例2: ランサムウェア インシデント

- MAZE ランサムウェア [25] (2020/05)
  - 公開Team ServerのPUBKEYに一致: 152.89.244[.]48
- Sodinokibi ランサムウェア [26] (2020/06)
  - 公開Team ServerのPUBKEYに一致: 5.101.1[.]202
  - 公開Team ServerのCustomer ID (452436291) に、別の19のIPが一致

公開事例3: PyXie [27] (2020/11)

- Vatet ローダー、PyXie RAT、Defray777 ランサムウェアのコンビネーションを使うサイバークライムキャンペーン
  - PyXie RATが使われる前は、ほとんどがCobalt StrikeのBeaconやStager
- 公開Team ServerのPUBKEYに、別の151のIPが一致

公開事例4: Blackrota [28] (2020/11)

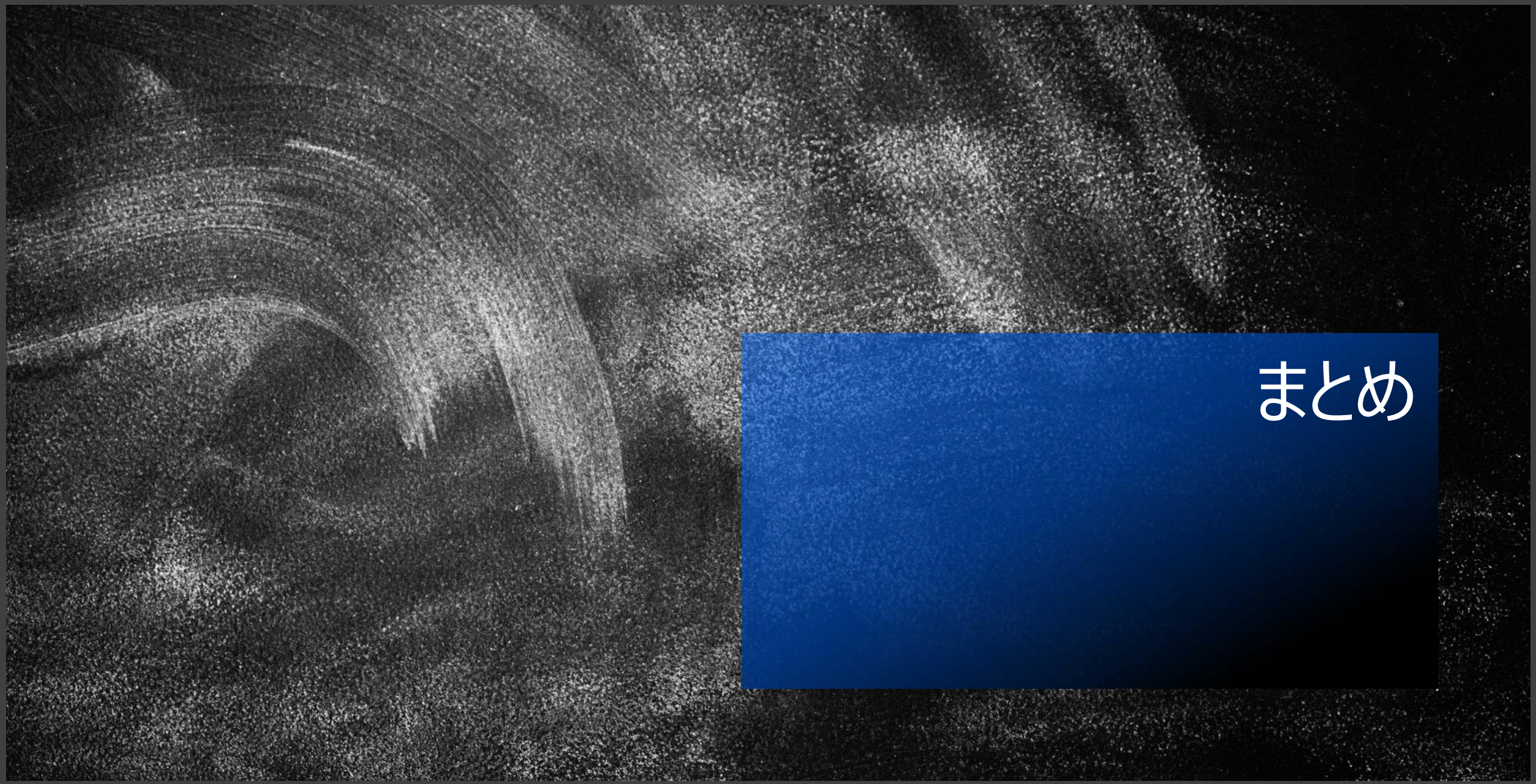
- Docker Remote APIをexploitするバックドア
  - Geacon [30] ベースの実装
- 公開Team ServerのPUBKEYに一致: 187.33.236[.]62
  - 資料作成時点で未だアクティブ

公開事例5: LuckyMouse/TA428 [29] (2020/12)

- Able Desktopがサプライチェーン攻撃に悪用された事例
- インストーラ実行時にデプロイされるKorplugのC2に、同時期にCobalt Strike Team Serverが稼働していた [31]
- 当該Team ServerのPUBKEYに、別の253のIPが一致

その他の公開事例

- ヒットするが非公開のIP無し
  - Kiya [32], Chimera [33], IndigoDrop [34], RYUK, etc..
- ポートが変則的で当時見逃し
  - APT41 [35], CVE-2020-14882 スキャンキャンペーン [36]
- Stager無効
  - UNC2452 [37]
- 観測期間外の、時間が経ち過ぎた攻撃との関連付けは難しい
  - APT10 [38], TA505 [39], OceanLotus [40][41]


まとめ

まとめ

- Cobalt Strikeプロトコルの分析結果を共有
- プロトコルエミュレーションによって、インターネット上のTeam Serverを発見する手法の実装
  - 観測期間中に稼働していたTeam Serverを、ほぼリアルタイムで捕捉
  - 攻撃開始前のサーバも、デプロイした段階で検出
  - １年間で6,300超のTeam Serverが稼働、過半数がリーク・クラック済み
- 今後の課題
  - External C2の独自プロトコルの調査・対応
  - ハニーポットによるCSユーザの実行タスクパターンの収集
- あなたのソリューションは？

リファレンス (1/4)

[1] https://www.carbonblack.com/blog/active-c2-discovery-using-protocol-emulation-part1-hydseven-netwire/

[2] https://www.carbonblack.com/blog/threat-analysis-active-c2-discovery-using-protocol-emulation-part2-winnti-4-0/

[3] https://www.cobaltstrike.com/

[4] https://blog.cobaltstrike.com/2019/02/19/cobalt-strike-team-server-population-study/

[5] https://www.cobaltstrike.com/help-malleable-c2

[6] https://github.com/SpiderLabs/DoHC2

[7] https://blog.fox-it.com/2019/02/26/identifying-cobalt-strike-team-servers-in-the-wild/

[8] https://80vul.medium.com/identifying-cobalt-strike-team-servers-in-the-wild-by-using-zoomeye-part-2-acace5cc612c

[9] https://www.fireeye.com/blog/threat-research/2020/07/scandalous-external-detection-using-network-scan-data-and-automation.html

[10] https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a

[11] https://twitter.com/MichalKoczwara/status/1338110430498197505?s=20

[12] https://twitter.com/notwhickey/status/13350318394304798 72?s=20

リファレンス (2/4)

[13] https://www.randhome.io/blog/2020/12/20/analyzing-cobalt-strike-for-fun-and-profit/

[14] https://www.zoomeye.org/searchResult?q=%22CobaltStrike%20Beacon%20configurations%22

[15] https://quake.360.cn/quake/#/reportDetail?id=5fc6fedd191038c3b25c4950

[16] https://github.com/Sentinel-One/CobaltStrikeParser

[17] https://blog.didierstevens.com/2020/11/07/1768-k/

[18] https://github.com/sysopfb/malware_decoders/blob/master/cs_beacon/proper_beacon_decoder.py

[19] https://www.cobaltstrike.com/help-authorization-files

[20] https://github.com/threatexpress/malleable-c2

[21] https://github.com/zmap/zmap

[22] https://opendata.rapid7.com/

[23] https://blog.cobaltstrike.com/2015/10/14/the-cobalt-strike-trials-evil-bit/

[24] https://www.anomali.com/blog/covid-19-themes-are-being-utilized-by-threat-actors-of-varying-sophistication

リファレンス (3/4)

[25] https://www.fireeye.com/blog/threat-research/2020/05/tactics-techniques-procedures-associated-with-maze-ransomware-incidents.html

[26] https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/sodinokibi-ransomware-cobalt-strike-pos

[27] https://unit42.paloaltonetworks.com/vatet-pyxie-defray777/

[28] https://blog.netlab.360.com/blackrota-an-obfuscated-backdoor-written-in-go-en/

[29] https://www.welivesecurity.com/2020/12/10/luckymouse-ta428-compromise-able-desktop/

[30] https://github.com/darkr4y/geacon

[31] https://twitter.com/KorbenD_Intel/status/1217121790872424448

[32] https://insight-jp.nttsecurity.com/post/102fz2k/kiya

[33] https://www.cycraft.com/download/%5BTLP-White%5D20200415%20Chimera_V4.1.pdf

[34] https://blog.talosintelligence.com/2020/06/indigodrop-maldocs-cobalt-strike.html

[35] https://www.fireeye.com/blog/threat-research/2020/03/apt41-initiates-global-intrusion-campaign-using-multiple-exploits.html

[36] https://isc.sans.edu/diary/rss/26752

リファレンス (4/4)

[37] https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html?1

[38] https://www.lac.co.jp/lacwatch/people/20180521_001638.html

[39] https://www.fsec.or.kr/user/bbs/fsec/163/344/bbsDataView/1382.do?page=1&column=&search=&searchSDate=&searchEDate=&bbsDataCategory=

[40] https://www.paloaltonetworks.jp/company/in-the-news/2019/tracking-oceanlotus-new-downloader-kerrdown

[41] https://www.volexity.com/blog/2020/11/06/oceanlotus-extending-cyber-espionage-operations-through-fake-websites/

[42] https://attack.mitre.org/techniques/T1090/004/

[43] https://blog.cobaltstrike.com/2017/02/06/high-reputation-redirectors-and-domain-fronting/

[44] https://paper.seebug.org/1190/

[45] https://aws.amazon.com/blogs/security/enhanced-domain-protections-for-amazon-cloudfront-requests/

[46] https://blog.torproject.org/domain-fronting-critical-open-web