infoblox

DECOY DOG 調査レポート

DNS に潜伏するマルウェア ツールキットの発見

目次

エグゼクティブサマリー

Decoy Dog は、Infoblox が発見したマルウェアツールキットで、ドメインネームシステム (DNS) を使用してコマンド & コントロール (C2) を実行します。侵害されたクライアントは、DNS クエリを介してコントローラと通信し、コントローラから指示を受信します。このコントローラは DNS ネームサーバに統合され、通常の解決プロセスでクエリが送信されます。2023 年 4 月に Decoy Dog の存在を開示し、4 月 23 日に初期調査結果の詳細レポートを公開しました。これは、DNS データの監視により検出されました。当時の分析では、ツールキットが Pupy として知られるリモートアクセス型トロイの木馬 (RAT) に基づいて構築されていることが確認されましたが、どのシステムが悪用されているか、ツールキットがどのように展開されているか、Pupy が変更されたかどうかは不明でした。[1] 私たちが提供した詳細情報により、コミュニティの他の人々が侵害されたマシンを見つけ出し、その全貌が明らかになることを期待していました。しかし、Decoy Dog を取り巻く謎は深まるばかりです。

4 月以来、Infoblox は Decoy Dog と Pupy に関するさらなる調査を実施しました。本レポートは、その調査結果になります。Decoy Dog は、パブリックリポジトリにないコマンドと構成を使用する Pupy への主要なアップグレードであることがわかりました。私たちは、Decoy Dog クライアントの通信を分離し、各コントローラに関する他の多くのプロパティを推測するアルゴリズムを開発しました。これにより、ツールキットが拡散し、これは少なくとも 3 人の攻撃アクターの管理下にあるとかなりの自信を持って結論付けることができます。私たちが観察した活動は依然としてロシアと東ヨーロッパに限定されていますが、コントローラ内には複数の攻撃アクターと一致する技術、戦術、手順 (TTP) の明確なグループが存在しています。

すべての Decoy Dog の攻撃アクターは、4 月の開示に対して何らかの形で反応し、そのバリエーションにより、複数の運用者がいるという私たちの評価を裏付けています。ソーシャルメディアでの最初の発表の直後、ネームサーバーの一部が停止されました。残りのすべては、最初のレポートで強調した動作を削除するために変更されましたが、これはコントローラに応じて異なる方法で実現されました。あるコントローラのグループでは、ジオフェンシングと呼ばれる技術により、発信国に応じてクエリへの応答を制限し始めましたが、他のコントローラは ping サブドメインのクエリへの応答を変更しました。

LinkedIn での当社の開示に対して、ある攻撃アクターは非常に迅速に反応したため、私たちは当初、新しいドメインがセキュリティ研究者によって模倣登録されたと考えました。しかし、さらなる分析により、それらは置換ドメインであることが判明しました。攻撃アクターは運用を終了するのではなく、侵害された既存のクライアントを新しいコントローラに転送しました。これは、攻撃アクターが既存の被害者へのアクセスを維持する必要があると感じていたことを示す普通ではない対応です。これにより、ある 1 つの Decoy Dog ドメインのセットの TTP とその他すべての TTP の間に明確な分離が作成されました。

当社の発表から数週間で、Decoy Dog の活動の足がかりとなった根本的なマルウェアと脆弱性を特定する人が誰もいなかったことには驚きました。しかし、調査が進むにつれて、通信が 1 年以上検出されなかった理由が明らかになりました。Decoy Dog を使用した攻撃は非常に標的が絞られており、各コントローラは少数のアクティブなクライアントしかいなかったためです。一部のサーバーは、一度に 4 ～ 8 台のアクティブなクライアントを数か月間一貫して維持します。同時にアクティブなクライアントの数が時間の経過とともに増加していることを確認したサーバーもありましたが、一度に観察された影響を受けたデバイスの総数は 100 台未満でした。小規模な被害者では、一般的に金銭目的の攻撃者は排除され、デバイス上で長期間にわたって存続する必要性があることは、高度な攻撃者と一致します。

私たち自身の Pupy トラフィックから署名を特定することで、Decoy Dog の通信の一部を再構築することができました。私たちはインターネット上に Pupy サーバーを確立しました。これをコードの選択的リバースエンジニアリングと組み合わせることで、DNS クエリと応答を特定の Pupy コマンドに関連付けることができました。このことから、a) Decoy Dog には Pupy にはないコマンドが含まれていることと、b) 下にある通信のほとんどを明らかにすることができました。さらに、Decoy Dog の攻撃アクターは、Pupy を利用して、キー交換などの機能のために

1 https://github.com/n1nj4sec/pupy

DNS 外の他のトランスポート層を利用しているようです。脅威アクターは、これがリモートアクセストロイの木馬 (RAT) としての Pupy の利点の 1 つであると考えている可能性があります。

Decoy Dog ツールキットの最初の導入は、2022 年 3 月下旬または 4 月上旬に行われています。5 月中旬までにアクティブになった、異なる TTP を備えた 2 番目のコントローラの出現によって示されるように、その直後にこの最初のコントローラは販売されたか、盗まれたかしました。3 番目のドメインが登録されたのは、2022 年 7 月で、9 月まで戦略的に熟成しました。これら後者の 2 つのコントローラは、ロシアの IP 空間でのホスティングを含み多くの特徴を共有しているため、同じ攻撃アクターによって所有されている可能性があります。ただし、いくつかの違いがあります。数か月後、さらに 2 個のドメインが登録されましたが、これも以前のコントローラとは異なる特徴がありました。これらのドメインを登録した攻撃アクターは、当社の開示直後にクライアントを新しいドメインに移行しました。Infoblox は現在、合計 21 個の Decoy Dog ドメインを監視し、その一部は先月に登録・展開されました。

DNS ログの分析を通じて Decoy Dog が Pupy とは大きく異なることが判明したため、VirusTotal で入手可能な関連バイナリサンプルを調べて、実行可能ファイルで違いがあるかどうかを確認しました。これらのサンプルをリバースエンジニアリングしたところ、Pupy として検出されたものの、オープンソースバージョンよりもはるかに高度であることがわかりました。サンプルには、a) クライアント上で任意の Java コードを実行する機能、b) いくつかの新しいトランスポートメカニズム、c) 永続性を確保するための新しい DNS メカニズムが含まれています。1 つのメカニズムは従来の DNS ドメイン生成アルゴリズム (DGA) に似ており、無料のダイナミック DNS プロバイダーを使用して、いわゆる緊急コントローラに接続しています。すべてのサンプルは同じ基本的な更新を共有していますが、サンプルの 1 つは、ストリーミングトランスポートの使用に関連する、他のサンプルには見られない独自の機能を備えています。

理由はまだ明らかではありませんが、Decoy Dog は、通常、敵によるコンテンツの検出と回復を回避することを目的とした秘密通信の中核となる原則に違反しています。通常の Pupy サーバーは、侵害されたクライアントから繰り返される通信クエリを拒否しますが、Decoy Dog サーバーは、再生された DNS クエリに応答するだけでなく、巧妙に作成されたクエリにも応答します。この動作は DNS のワイルドカードの構成に似ており、Infoblox による Decoy Dog の検出の重要な要因でした。Decoy Dog の高度な機能を考慮すると、再生とワイルドカードの動作は次のような設計によるものと推測されます。意図が何であれ、業界が Decoy Dog を新たなマルウェアとして認識できなかった原因の一部は、DNS の広範な再生にありました。

セキュリティベンダーによる積極的なインターネットスキャンにより、数百万の Decoy Dog 通信がグローバルネットワークを通じて再送信され、その中には当社のお客様も含まれていました。これが、ツールキットの発見につながりました。ベンダーがトラフィックをマルウェアとして識別できないため、クエリの再実行を避けるために、感染していないネットワークから Decoy Dog コントローラへの DNS 接続が作動されました。Infoblox のお客様は感染しておらず、当社のリゾルバへのクエリはすべて異常なベンダースキャンの結果であると確信しています。当社の顧客ネットワークに対する差し迫った脅威はないにもかかわらず、Decoy Dog は依然として発信元が不明確な洗練されたツールキットであり、今後も蔓延する可能性があります。

Decoy Dog は実際に新たに観察されただけでなく、私たちの知る限り、悪意のある運用で Pupy の DNS C2 コンポーネントが使用されたのはこれが初めてです。これは、おそらく、リポジトリ内のソフトウェアを変更し、DNS を適切に構成する必要がある、Pupy ネームサーバーの確立が難しいことが原因であると考えられます。露出が少ないため、セキュリティ業界が Pupy と Decoy Dog の両方を検出して防御することは困難になります。これらの C2 システムを使用する運用を混乱させるために、当社のサーバーからキャプチャした Pupy DNS トラフィックとソフトウェアの内部動作の詳細を含む調査データセットをコミュニティに提供しています。このドキュメントはこの種のものとしては初めてのものであり、他の人が検出アルゴリズムを構築したり、私たちの調査結果を再現したりできるようになります。

Decoy Dog の物語は、脅威の検出と対応のソースとしての DNS の力を明らかにしています。また、セキュリティ業界を支配するマルウェア中心のインテリジェンスエコシステムの本質的な弱点も明らかになっています。このツールキットは DNS 脅威検出アルゴリズムによって検出され、今日、それに対する唯一の防御は DNS です。さらに、すべてが共通のマルウェアを使用していることに気づく前には、いくつかのコントローラドメインに不審フラグを立て、リゾルバでブロッ

クしていました。悪意のある活動が特定される前、多くの場合は運用される前に阻止するこのタイプの保護は、DNS 検出および応答システムに特有のものです。

このレポートでは、Pupy と Decoy Dog を識別するための知識を防御者に提供します。DNS C2 について詳しく説明しますが、悪意のあるアクターが Pupy を展開するのに役立つ情報は提供しません。また、完全な Decoy Dog DNS 署名も公開しません。元のレポートで特定したいくつかの動作を説明し、Decoy Dog が Pupy とどのように異なるかに注目していきます。さらに、マルウェア自体を所有したりネームサーバーを制御したりすることなく、クライアント数とコマンドトラフィックを推定できるようにした、大量の Decoy Dog DNS トラフィックの分析についても説明します。Decoy Dog のサンプルが Pupy とどのように異なるかを説明します。最後に、Decoy Dog の運用者が当社の開示にどのように反応したかについて考察し、コントローラのサブグループ全体に共通する特徴を紹介します。付録には、追加のサポート技術情報が含まれています。

背景

Infoblox は、2023 年 4 月初旬にドメインネームシステム (DNS) を使用したコマンド & コントロール (C2) ツールキットである Decoy Dog を発見しました。これは、Pupy[2] と呼ばれるオープンソースのリモートアクセストロイの木馬 (RAT) に基づいており、ドメイン名のクエリと IP アドレスの応答を介して、クライアントとサーバー (コントローラ) 間の暗号化された通信を転送します。Infoblox リゾルバへのパッシブ DNS クエリを監視し、異常な動作がないかどうかをアルゴリズムが監視したことからこれを発見しました。Decoy Dog ドメインのクエリは、少数の顧客ネットワークのセキュリティ装置から行われていました。これらのクエリにより、永続的な低プロファイルのマルウェアビーコンと一致する署名が作成されました。DNS が明らかに機密通信チャネルとして使用されていたにもかかわらず、公開されているインテリジェンスデータではドメインが C2 として識別されていなかったため、この活動に対する人間による調査は驚くべきものでした。実際、オンラインの評判チェッカーで「評判が良い」と評価されたものもあります。コミュニティがトラフィックをブロックし、侵害の性質を特定できるようにするために、私たちは 4 月 13 日に一連のドメインを発表しました。

私たちの最初の調査中に、Infoblox は Pupy ソフトウェアから独立した固有の DNS 署名を特定しました。攻撃アクターは、非常に特殊な方法で C2 システムを展開し、運用していました。このため、私たちは Decoy Dog を別個のツールキットとして特定しました。この署名を共有しているのは世界中の少数のドメインだけであり、そのすべてが Decoy Dog ネームサーバーでした。

4 月 23 日、私たちは署名の一部、パッシブ DNS の初期分析、コントローラドメインのサブセットを「Dog Hunt: Finding Decoy Dog Toolkit in Anomalous DNS Traffic (Decoy Dog 狩り : 異常な DNS トラフィックの中の Decoy Dog ツールキットを探し出す)」と題するレポートで発表しました。[3] このレポートでは、Pupy の特定の動作に注目しています。その特定の動作は「ping」を含む特定のサブドメインのクエリに対する一連の localhost 応答を返すというものです。また、当時完全には説明できなかった、DNS 通信内の多くの傾向についても説明しました。特に、応答として返された IP アドレスの驚くべきパターンと、サーバーが再度クエリに応答したという事実を特定しました。これは秘密通信システムとしては予期せぬことです。

この発表を受けて、ベンダーやその他の組織を含むセキュリティコミュニティの幅広いメンバーから私たちに問い合わせがありました。その多くは、自社のネットワークや顧客のネットワークで関連トラフィックを確認していましたが、侵害されたデバイスを特定したり、活動の範囲を認識したりしていた人は誰もいませんでした。これらの組織の一部からの情報により、DNS がどのように生成されたかを以下の当社のネットワークと分離して、確認へと導くことができました。また、活動の範囲を確認し、仮説をテストするのにも役立ちました。この非公式でのコラボレーションは非常に有益であり、感謝しています。

2 https://malpedia.caad.fkie.fraunhofer.de/details/win.pupy

3 https://blogs.infoblox.com/cyber-threat-intelligence/cyber-threat-advisory/dog-hunt-finding-decoy-dog-toolkit-via-anomalous-dns-traffic/

わかりやすくするために、このレポートでは Pupy という用語を、一般的な Pupy ではなく、特に Pupy DNS C2 を指すために使用します。

Pupy

リモートアクセスト型トロイの木馬 (RAT)

PPupy は、複雑なモジュラートランスポートシステムを特徴とするオープンソースのポストエクスプロイト用リモートアクセスト型トロイの木馬 (RAT) です。[4] Pupy の主要なコードベースは 2015 年に GitHub で公開されましたが、DNS C2 メカニズムは 2019 年まで追加されませんでした。本レポートは、Pupy C2 に関する最初の公開文書です。さらに、他の人が私たちの研究を再現し、将来のための防御を作成できるように、GitHub でデータセットを提供しています。

Pupy はオープンソースですが、DNS C2 プロトコルの使用は稀で、実際に Decoy Dog 以外で使用されていることを特定できていません。[5] 世界中の企業や組織にサービス提供している当社のレゾルバから、Pupy DNS C2 が過去に使用されたという証拠は見つかっていません。2023 年上半期のグローバル pDNS では、Pupy 用に開発した DNS 検出器を使用したところ、Decoy Dog 以外ではソフトウェアの使用は見られませんでした。最後に、私たちは様々なベンダーに個人的に問い合わせましたが、誰もそれが使われているのを見たことがありませんでした。APT (Advanced Persistent Threat) アクターによる Pupy の使用が報告されましたが、DNS C2 コンポーネントは採用されていなかったようです。[6]

Pupy がまれにしか使用されないのは、少なくとも部分的には、システム操作の難しさが原因である可能性があります。グローバル DNS を介して Pupy 通信を確立するのは簡単ではありません。ネームサーバーを正しく構成し、GitHub リポジトリ内のコードを変更する必要があります。さらに、DNS には、Pupy ソフトウェアが正しく処理しない再帰的リゾルバによって異なるという複雑さがあります。これらの課題は、攻撃者とハッカーの両方による採用を妨げている可能性がありますが、Cobalt Strike のような一般的なツールがかなり頻繁に見られるのとは対照的です。[7]

Pupy DNS C2 は今日では稀ですが、Decoy Dog の使用が広まっており、防御側が何らかの形で Pupy と対峙する可能性が高まっています。コミュニティの準備を支援するために、Infoblox は Decoy Dog と Pupy の両方について重要な調査を実施しました。Infoblox は、インターネット上に Pupy サーバーを展開して、その動作を Decoy Dog と比較しました。次に、Infoblox リゾルバからパケットデータ (pcap) とパッシブ DNS ログをキャプチャしました。私たちは、Decoy Dog の独自の性質をよりよく理解するために、Pupy のデプロイとコードの選択的なリバースエンジニアリングを組み合わせて使用しました。ここでは、私たちの調査に関連する Pupy の構成要素について説明します。わかりやすくするために、このレポートでは IPv4 (A レコード) 応答を使用した通信に限定しますが、利用可能な場合、Pupy は IPv6 (AAAA) 応答を使用します。本レポートで説明されているクエリエンコーディングは、Pupy バージョン 2 の現在のデフォルトです (特に指定がない限り)。[8]

---

4 https://github.com/n1nj4sec/pupy

5 「実際に」という表現は、サイバーセキュリティの専門用語で、運用上導入されることを意味し、分離された侵入テストや研究の一部ではないことを意味します。

6 https://www.volexity.com/blog/2022/06/15/driftingcloud-zero-day-sophos-firewall-exploitation-andan-insidious-breach/

7 https://www.esecurityplanet.com/threats/how-cobalt-strike-became-a-favorite-tool-of-hackers/

8 Pupy C2 の以前のバージョンには、各クエリにホスト情報が含まれていませんでした。Decoy Dog はクライアントのバージョン 3 であることがわかりましたが、クエリのエンコードはバージョン 2 と同じであるようです。

PUPY の仕組み

私たちの前回のレポートでは、Pupy についての概要を説明し、Decoy Dog の通常とは異なる特徴に注目しました。[9] 本レポートでは、さらに掘り下げて、Decoy Dog との接続を実際に行い、収集されたパッシブ Pupy DNS データを利用して進行中の運用を理解する方法を説明します。

Pupy は、感染したクライアントとサーバーの間で継続的な通信を提供するように設計されているため、攻撃アクターがクライアントにリモートアクセスしたい場合に、接続はすでに確立されています。攻撃アクターは、接続されているクライアントを監視し、幅広い操作を提供するよう選択的に命令できます。DNS は C2 通信にのみ使用されます。クライアントから引き出された重要なデータは、Pupy が提供する他の多くのトランスポートオプションの 1 つを介して送信されます。その結果、Pupy DNS クライアントは、コントローラへの認証管理、コマンドの確認、システム情報の提供、その他のいくつかの役割に制限されます。サーバーからのコマンドの処理の間に、クライアントはスリープ状態になります。

DNS 通信は、クライアントによって開始され、維持されます。クライアントは、通常の DNS 解決パスを介して、または DNS over HTTPS (DoH) が有効で使用可能な場合は、DNS over HTTPS(DoH) を介してクエリを送信します。[10] コントローラは、クライアントのリクエストに応答して、暗号化された IP アドレスの形式でコマンドを送信します。すべてのクエリ応答は完全な通信であり、クライアントもサーバーも 1 つのコマンドのデータを 2 つの DNS クエリに分割することはできません。このプロトコルは、一般的な DNS トンネリングシステム (Iodine など) とは区別され [11]、クライアントは DNS を介してセッションを確立し、通信を処理するためにいずれかの側で複数のパケットを再構築することができます。クライアントはほとんどのコマンドを確認する義務があり、サーバーはすべての有効なクライアントクエリにコマンドまたは確認で応答します。クライアントの語彙は非常に限られています。セッションの管理、コマンドの確認、システム情報の送信、キーの確立を行う 9 種類のクエリがあります。カスタムコマンドは、追加の関数を記述することで追加できますが、それにはソフトウェアを完全に理解する必要があります。

クライアントは起動時に、共有キーが確立されているかどうかに応じて、2 通りの異なる方法のいずれかでサーバーにクエリを実行します。このクエリは、Pupy クライアントのシステムと状態に関する現在の情報をサーバーに提供するか、新しい暗号化セッションを開始するための単純なクエリを作成します。暗号化されたセッションを無効にすることは可能ですが、これはデフォルトではなく、Decoy Dog では確認されていません。これに応じて、コントローラはリクエストを確認するか、クライアントにキー交換の実行を要求するか、または新しいコマンドを送信します。コマンドのセットがすべて完了すると、クライアントは設定された間隔で (デフォルトでは 60 秒) スリープします。このプロセスは、クライアントが実行されている間、繰り返されます。Pupy のクライアント / サーバー通信の高レベルの概要を図 1 に示しますが、クライアントプロセスの詳細については付録 A を参照してください。

9 https://blogs.infoblox.com/cyber-threat-intelligence/cyber-threat-advisory/dog-hunt-finding-decoydog-toolkit-via-anomalous-dns-traffic/

10 Pupy は、デフォルトで DoH に Quad9 サーバーを使用します。

11 https://github.com/yarrick/iodine

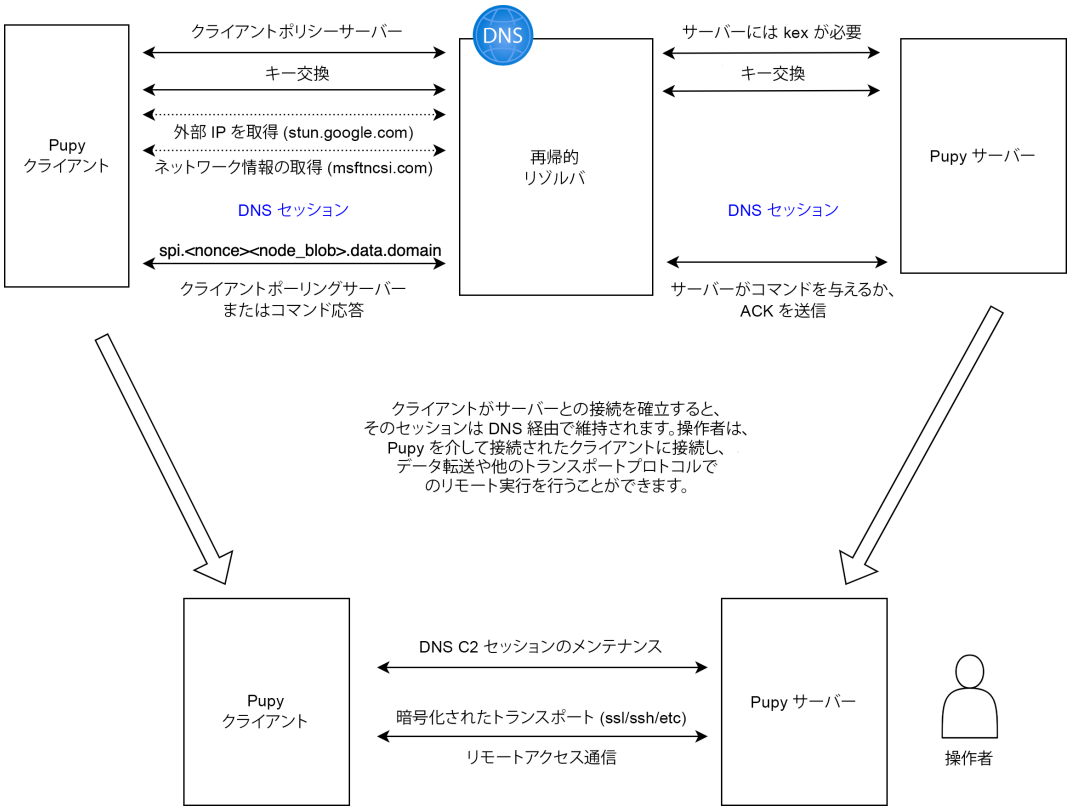

図 *1. Pupy* 通信の高レベルの概要 .

Pupy アクターは、コントローラコマンドラインユーティリティからクライアントと通信します。クライアントがコントローラに接続されると、キューに入れられたコマンドはすべて DNS 応答内でコード化されます。操作者は、クライアントの開いているポートで接続を確立し、抽出に使用するトランスポート層を指定します。サーバーの DNS 通信は依然としてかなり制限されていますが、クライアントよりも範囲は広くなっています。幅広いコマンドがあり、これらを連結してクライアントへの 1 つの応答にできます。クライアントが通信交換を開始するのに対し、サーバーは通信のセキュリティを確保する責任があります。これは、暗号化キーをローテーションする役割を果たす、各クライアントとのいわゆるセッションを強制することによって行われます。これについては、次のセクションで説明します。

セッションの開始

Pupy では、アクターのコマンドを送信する前に、クライアントとコントローラ間で暗号化されたセッションを確立する必要があります。このセッションは、クライアントの通信がタイムアウトすると期限切れになり、DNS クエリの復号時のエラーやクライアントの再起動など、他の理由で強制的に更新される可能性があります。セッションは、クエリ内のセキュリティパラメーターインデックス (SPI) ラベルの存在によって識別され、一時的な共有キーを使用して暗号化されます。通信の詳細は、クライアントが以前にサーバーに接続したかどうかなど、様々な要因によって、セッション初期化の正確なプロトコルは異なるため、観測された DNS 交換にばらつきが生じます。ただし、一般的な交換は次のとおりです。

- クライアントは、確立されたセッションなしで、または期限切れのセッションでサーバーにチェックインする (クエリ 1)。
- サーバーは、キー交換とクライアントのシステム情報を必要とするコマンドで応答する。[12]
- クライアントはシステム情報の要件を確認する (クエリ 2)。
- クライアントは、楕円曲線アルゴリズムを使用してランダムな秘密キーと公開キーのペアを生成し、これをサーバーに送信し、サーバーも同じことを行い、新しいキーで応答する (クエリ 3)。

12 これは通常、サーバー側でポリシーとポーリングと呼ばれる 2 つのコマンド形式になります。

- クライアントとサーバーは、この交換を使用して新しい共有セッションキーを確立し、このキーは AES 暗号化でパケットを暗号化し、セッションを識別するための SPI も作成する。
- クライアントは、他のサービスへの追加の DNS クエリを使用して、外部 IP アドレスを含むネットワークに関する情報を収集する。
- クライアントは、共有暗号化キーを使用してこの情報を送信し、クエリに SPI を含めてアクティブなセッションの存在を通知する (クエリ 4)。
- クライアントは追加のシステムステータス情報を送信する (クエリ 5)。

共有キーと SPI は通常、3 回のクエリ後に確立されますが、技術的にはキーの交換は 1 回のクエリと応答で行われます。セッション中、各クエリと応答はこの共有キーを使用して暗号化されます。暗号化には、クライアントによって生成され、クエリごとに変更される 32 ビットのノンスも使用されます。新しいセッションが確立されるとキーが再生成されますが、クライアントの動作中はクライアントのノンス値が継続されます。これについては、以下のセクションで詳しく説明します。

クエリのコード化

クライアントは、サーバーへの暗号化された通信を含むクエリを生成します。これらには、キー交換情報やサーバーからのコマンドへの応答が含まれる場合があります。各クエリで通信できる送信データは最大 52 バイトです。送信されたデータに加え、各クエリには以下が含まれます。

- ノンス、クライアントバージョンによって生成される 4 バイトの増分値
- Pupy DNS C2 バージョン cid を示す 1 バイトの値
- クライアント iid の作成時にランダムに生成される、クライアントの構成からの 4 バイトの値
- Pupy クライアントプロセスのノード ID の下位 16 ビットを含む 2 バイトの値
- クライアントからの、通常はデバイスの MAC アドレス、オプションで SPI からの 6 バイトの値
- キー交換中に生成され、特定のクライアントのサーバー上のセッションを表すクエリに存在する 4 バイトの値

すべてのクライアントクエリには、これら 13 バイトのクライアント情報と、基礎となるペイロード上の 4 バイトのチェックサムが含まれます。基礎となるペイロードは暗号化され、一連のコマンドと関連データで構成されます。

クライアントは、DNS プロトコルではクエリ名 (qname) と呼ばれる完全修飾ドメイン名 (FQDN) としてサーバーに送信されるデータを暗号化し、コード化します。以下の図 2 に示すプロセス全体には、送信データとサーバーが必要とする追加情報の両方の暗号化、配置、コード化が含まれます。次のように動作します。

- 送信されるデータに、ホスト固有の情報が付加される。
- この複合バイト文字列は、共有対称キーと現在のノンスを使用して暗号化される。
- 送信データの最初の暗号化バイト (最大 35 バイト) がコード化され、qname の最初 (右端) のラベルに使用される。
- 暗号化されたバイトの残りの部分には、最大 17 バイトの送信データが含まれる可能性があり、現在のノンス値が先頭に追加され、qname の 2 番目のラベルを作成するためにコード化される。
- セキュリティパラメータインデックス (SPI) がクライアントに存在する場合、それはコード化され、qname の 3 番目 (左端) のラベルに使用され、この値は、サーバーとのキー交換の後に設定される。
- ノンスは、次のクエリに使用されるクライアント内の暗号化データの長さだけ増加する。

暗号化されたバイトからドメイン名ラベルへのコード化については、前回のレポートで説明しました。カスタムマップと 32 ビットのコード化を組み合わせて使用し、最終結果が確実に有効なドメイン名であるようにします。基礎となるデータペイロード構造については、付録 B で説明しています。

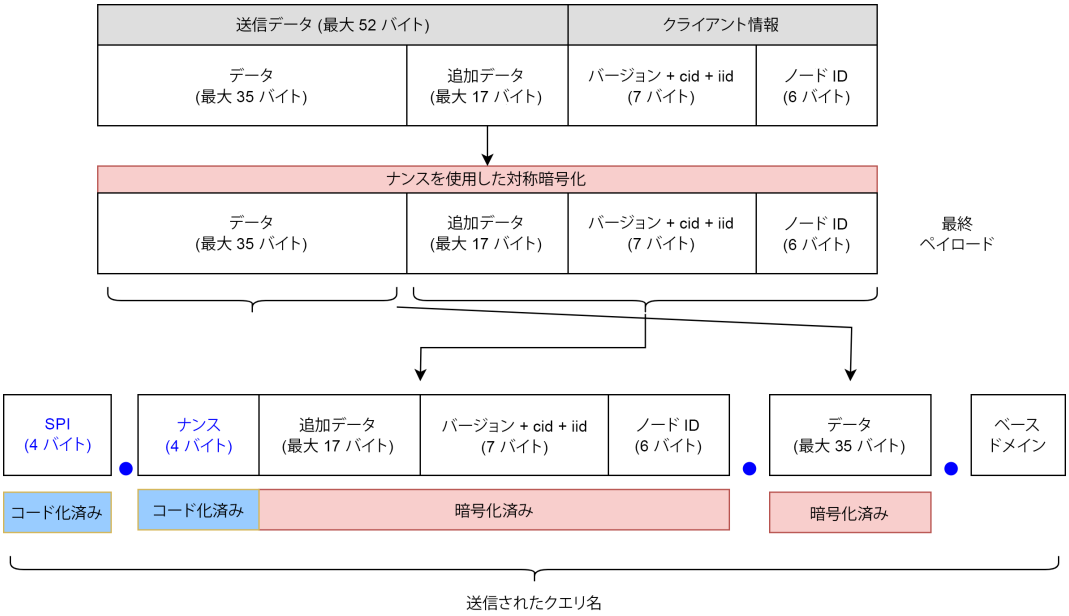

図 2. クライアントからのデータを *DNS* クエリの *qname* に変換するプロセス。ベースドメインは、*Pupy* サーバーのドメイン名です。

Pupy は、DNS クエリを暗号化するときにデフォルトで AES を使用します。クライアントとの共有キーが確立されている場合、クライアントはこれを使用して完全なバイト文字列を対称的に暗号化します。それ以外の場合は、確立された公開キーを使用します。どちらの場合も、基礎となる送信データが複数のクエリ間で同じままである場合でも、コード化されたクエリが一意であることを保証するために、現在のノンスも暗号化に使用されます。これは暗号攻撃から保護するための標準的な仕組みです。その結果、クエリされたドメイン名を復号して暗号化されたデータを明らかにすることができますが、暗号化されたデータはキーがなければ復号化できません。ノンス値はランダムな 32 ビット値で初期化され、クエリごとにペイロードの長さだけ増加します。

Pupy ネームサーバーはクエリを受信すると、ドメイン名を復号化して SPI 値、ノンス、暗号化されたペイロードを明らかにします。有効なクライアント通信を受信していることを確認するために、サーバーは SPI が存在する場合は有効かどうか、そしてクライアントに対して記録された以前のノンスよりも大きいかどうかをチェックします。ペイロード内で暗号化されているバージョン番号のチェックなど、データに対して他のいくつかのチェックが行われます。これらのチェックのいずれかが失敗すると、クライアントにエラーが返されます。

特に、Pupy は同じクエリに 2 回応答せず、変更されていない Pupy サーバーは、過去にすでに受信したクエリに対して NXDOMAIN (そのようなクエリはありません) で応答します。以前にクエリしたドメイン名のクエリを試みることで、当社独自の Pupy サーバーでこの動作を検証しました。Decoy Dog の特徴は、Pupy C2 プロトコルと一致する回答で、再生された DNS クエリに応答するため、これは重要です。

DNS クエリにはノンスの可逆のコード化が含まれており、クエリごとにノンスがペイロード長ずつ増加するため、単一のクライアントに関連付けられたクエリのスレッドを再構築できます。このレポートの後半で説明するように、Pupy または Decoy Dog ドメインのパッシブ DNS の収集を考慮すると、この再構築を使用してクライアントの数と、特定の場合の通信の性質を推定できます。

特別なドメイン名の処理

クエリを受信すると、サーバーはクエリ名を分析し、それがクライアントからの暗号化されたパケットの適切な構造と一致するかどうかを判断します。特有な処理が必要な特殊なケースもいくつかあります。これらの特殊なケースを除き、期待される形式を満たさないリクエストは拒否されます。それらの特殊なケースの 1 つは、前回のレポートで説明した ping リクエストです。サブドメイン pingN のクエリで、N は整数で、N の長さのローカルホスト応答のシーケンスを返します。ping 自体のクエリはそのような応答を 15 個返し、ベースドメインのクエリは単一のローカルホスト応答、つまり 127.0.0.1 を返します。

ping リクエスト以外に、単一の IP アドレスを使用して単一のラベルクエリに応答するようにサーバーを構成できます。この機能の目的は不明であり、クライアントでは使用されていないようです。これは、ソースコードでは DNS アクティベーションリクエストと言われます。この機能は文書化されていないため、これを利用するには、アクターはサーバーソフトウェアがどのように機能するかを理解する必要があります。

単一ラベルのサブドメインの特別な処理には、文字列のキーと値のペアである「アクティブ化」エントリを構成する必要があります。次に、この値はサーバーの秘密キーと組み合わせて使用され、応答 IP アドレスが作成されます。この応答は一方向ハッシュ関数を使用して作成され、反対にはできません。ハッシュ引数は大文字と小文字が区別され、次のように定義されます。

MD5(subdomain_label + activation_value + private_key)

応答のコード化

サーバーがクライアントからクエリを受信すると、復号して結果を確認し、クライアントデータを処理します。特に、クライアント通信は適切にフォーマットされ、クエリのコード化に関するセクションで前述したように、2 つまたは 3 つのラベルが含まれている必要があります。次に、サーバーは 1 つ以上のコマンドを含むクライアントへの応答を組み立てます。IPv4 (A) または IPv6 (AAAA) クエリを返すことができますが、簡単に説明するために IPv4 (A) クエリに限定します。

サーバーの応答は暗号化されたバイナリ文字列で、1 つ以上の A レコードにコード化されます。[13] このコード化のプロセスは、下の図 3 に示しています。応答の最大バイト数は 64 で、3 バイトのセグメントでコード化されるため、応答には最大 22 個の IPv4 アドレスが含まれます。

- 最初のステップで、サーバーは応答の長さを計算し、これを応答データの前に追加します。次にランダムバイトを追加して、長さが 3 バイトの倍数の合成文字列を作成します。[14] この合成文字列をペイロードと呼んでいます。
- 2 番目のステップでは、ペイロードの 3 バイトのセグメントから IPv4 アドレスが繰り返し作成されます。各 IPv4 アドレスは 32 ビット値で表され、ビット 0 が上位ビットになります。
- 各アドレスの上位 3 ビットはランダムです。
- 各セグメントにはインデックスがあり、これによりクライアントは受信時にデータを並べ替えることができます。これは 5 ビットで表されます。このインデックスは、結果のビット 3 ～ 7 にあります。
- ペイロードセグメントはビット 8 ～ 30 にあり、ペイロードセグメントの上位ビットが IPv4 アドレスの最初のオクテットの最下位ビットになります。
- 最後に、最下位ビットであるビット 31 は、ペイロードセグメントで生成されるチェックビットです。このチェックサムの性質により、このビットは IPv4 アドレスの 75% で 1 になります。
- 結果の 32 ビット文字列は IPv4 アドレスとして解釈され、応答に追加されます。

---

13 一連のコマンドが組み立てられてから、キー交換が完了している場合は、コード化前に共有キーと現在のノンスを使用して暗号化されます。それ以外の場合は、サーバーの秘密キーがナンスとともに使用され、公開キー楕円曲線アルゴリズムでデータが暗号化されます。

14 コードでは、このプロセスはより複雑ですが、結果は同じになります。

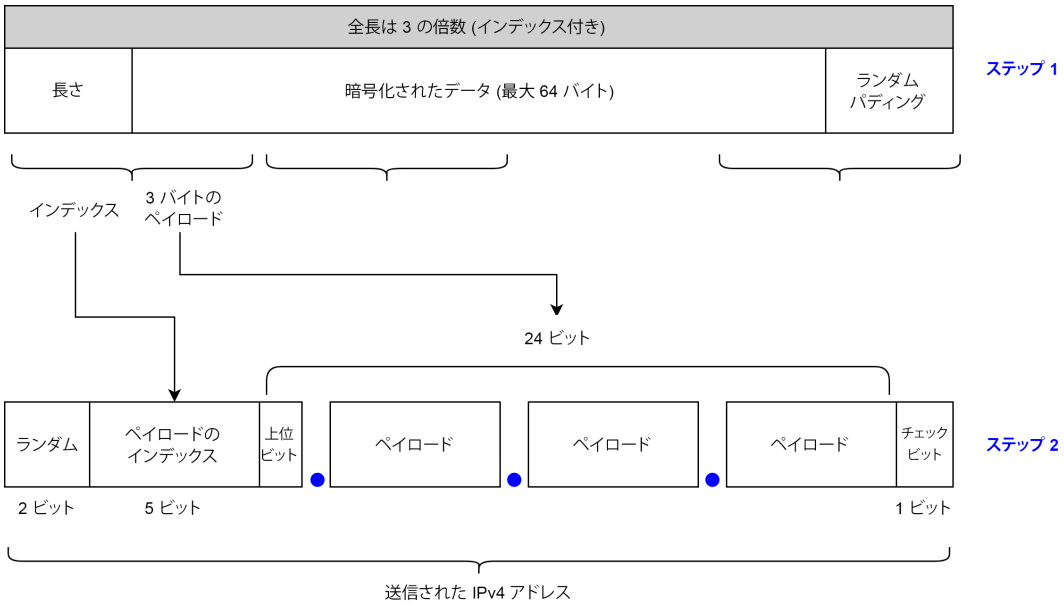

図 *3.* クライアントクエリに対する *IPv4* 応答の *Pupy* サーバーのコード化。データは、各アドレスのペイロードの *3* バイトを使用して、一連の *IPv4* アドレスにコード化されます。

前回のレポートでは、Decoy Dog の IPv4 応答の分布が驚くべきものであることに言及しましたが、これは、Pupy 応答のコード化の成果物であることがわかりました。3 つのランダムビットと増分インデックスをすべての応答データの最初のオクテットの上位 7 ビットとして使用し、結果の IPv4 アドレスが必ず特定の範囲になるようにし、それらの範囲が応答内の回答数に直接相互に関連づけるようにします。応答自体はクライアントに送信されてくるデータサイズにより決定されます。特に、最初の IP アドレスは常に 64.0.0.0/8、128.0.0.0/8、または 192.0.0.0/8 の範囲内になります。

インデックスが増加するたびに、IP アドレスの最初のオクテットの選択肢が 2 つずつシフトされます。具体的には :

- インデックスが 0 で長さが最大 64 であるため、最初の IP アドレスは 64、128、192 のいずれかで始まる。その結果、応答の最初の IP アドレスには上位 3 ビットのみが設定される。
- インデックスが 1 であり、ランダムに生成された上位 3 ビットに 2 が加算され、データペイロードの上位ビットは 0 または 1 になる可能性があるため、2 番目の IP アドレスは、66、67、130、131、194、195 のいずれかで始まる。
- 3 番目の IP アドレスは、68、69、132、133、196、197 などで始まる。

下の図 4 では、増加する応答数に対するこのアルゴリズムの結果を確認できます。特に、ヒルベルトマップを使用して、IP アドレスの最初のオクテットが、3、12、15 の回答の合計応答数とどのように相関しているかを示しています。

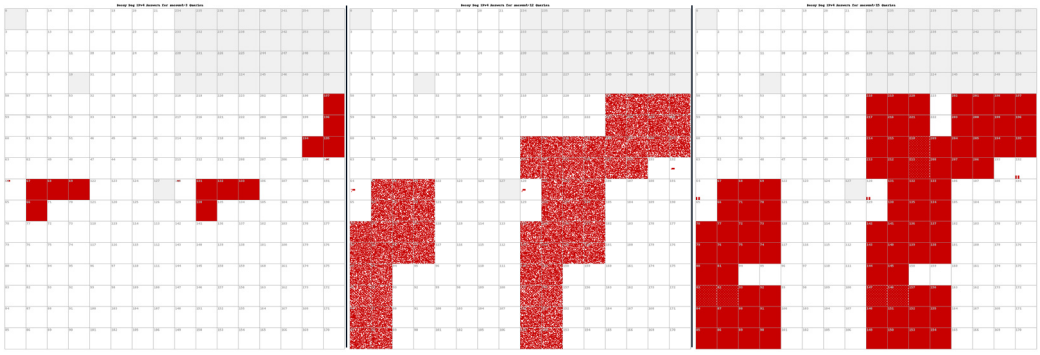

図 4. それぞれ 3、12、15 の回答を含む *Pupy* 応答における *IPv4* アドレスの分布を示すヒルベルトマップ

IPv4 アドレスの構造により、すべての応答を観察する人は誰でも、送信されたデータを再構築できます。このデータは暗号化されていますが、長さと時系列分析を使用して応答をプロファイリングできます。このタイプの分析により、このレポートで後ほど説明するように、通信に関する情報を明らかにできます。

パッシブデータ分析

Pupy の通信は強力に暗号化されていますが、パケットの復号化と追跡に必要な情報は可逆的な方法でコード化されています。DNS クエリと応答が収集されると、それらをまとめて分析して、Pupy の展開とクライアントに関する情報を導き出すことができます。一般にパッシブ DNS (pDNS とも呼ばれます) と呼ばれるパッシブ DNS の収集は、企業リゾルバ、パブリック再帰リゾルバ、ルートサーバー、TLD サーバーなど、インターネットの様々な場所で行われています。以下のセクションでは、Pupy クエリのパッシブ DNS の収集を利用して通信に関する情報を取得する方法を紹介します。

Pupy コントローラとそのクライアントに関する大量の情報をパッシブ DNS から回収できます。特に、次のような情報を回収できます。

- 一度にアクティブなクライアントのおおよその数
- サーバーとクライアントの間で発生する交換の種類
- 展開の署名 (クライアントのスリープ間隔など)
- クライアントのキー交換と活動全体のタイムライン

これらの手法を使用して、当社のサーバーと Decoy Dog サーバーからのトラフィックを分析しました。これにより、Decoy Dog が Pupy にどの程度似ているか、またサーバー同士がどの程度似ているかを理解できました。最終的に、これらの手法により、すべての Decoy Dog の展開をプロファイリングできるようになりました。使用された方法の技術的な詳細については、付録 C で詳しく説明しています。

PUPY ペイロードの署名

クライアントとサーバー間の通信の性質は、パッシブデータ分析を使用してある程度推測できます。クライアントの語彙、つまりクライアントが作成できる個別のペイロードは非常に制限されており、クライアント通信は 9 種類しかありません。2 つのタイプは同じペイロード長を共有しますが、別のタイプは複数の長さを持つことができます。脅威アクターは Pupy にカスタムイベントを作成し、ペイロード長の多様性をさらに生み出す可能性があります。

サーバーはより柔軟な語彙を備えており、単一の DNS 応答で複数のコマンドを伝達できるため、プロファイリングがより困難になります。ただし、Pupy システムの通信の大部分は、セッションの初期化、キーの交換、サーバーへのクライアントのハートビートに関連しています。サーバー通信は、クライアントリクエストの確認応答、新しいセッションを確立する必要性を含むエラーメッセージ、キー交換が大部分を占めています。

その結果、DNS クエリと応答の基礎となるペイロードの長さを使用して、様々な種類の通信の署名を作成できます。これらの署名により、一般的な保守管理活動をサーバーからの意味のあるコマンドから分離し、カスタムイベントタイプの使用を探し出せます。これらは、Decoy Dog を含む、パッシブデータで観察された Pupy クライアントとサーバーの全体的な動作をプロファイリングするために使用できます。

下の図 5 は、当社独自の Pupy データ内のクライアントクエリとサーバー応答で観察されたペイロード長のヒートマップを示しています。サーバーの長さはコマンド引数や連結されたコマンドによってさらに変動しますが、クライアント通信は明確に定義されています。通信のプロファイリングでは、チェックサムやノード情報を含む、基礎となるペイロードの長さを使用します。その結果、例えば、クライアント確認応答 (Ack) の長さは 19 バイト、サーバー Ack の長さは 6 バイトになります。付録 D には、一般的なクライアントとサーバーのペイロード長とコマンドとの関係に関する表が含まれています。

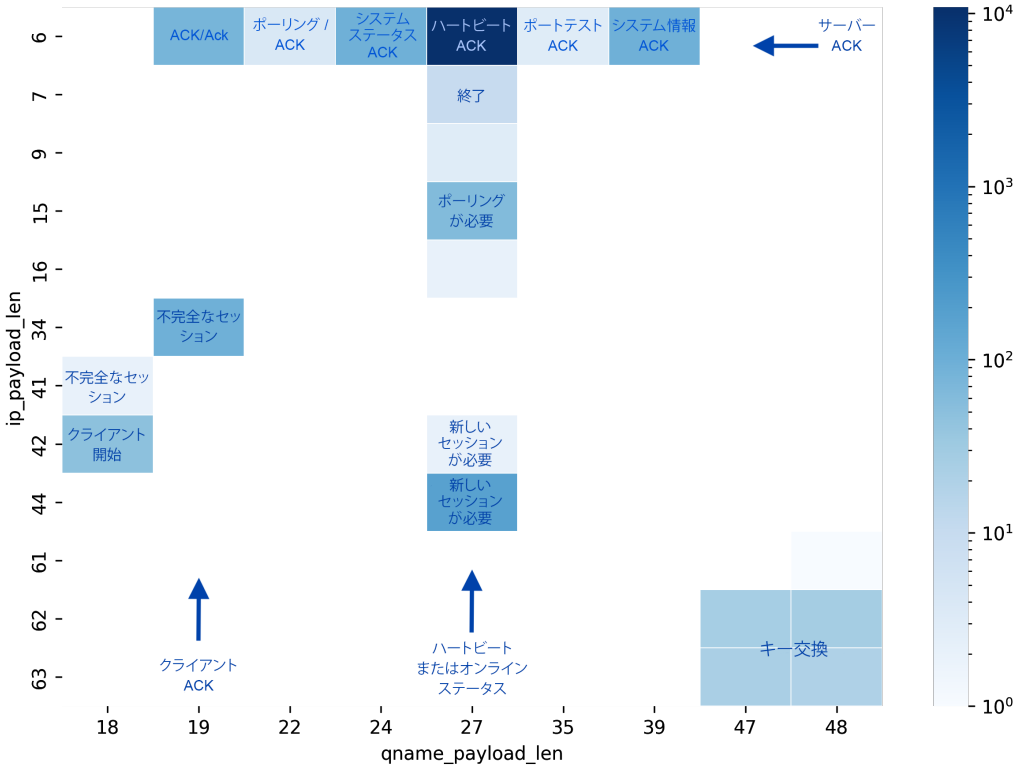

図 5. *Pupy* トラフィックで観察された一般的なペイロード長ペアの注釈付き分布。ペイロードは、クエリまたは応答で送信される暗号化されたデータです。このグラフにはサーバーからの複雑な *DNS C2* コマンドは含まれておらず、注釈のないセルは完全には識別されていません。長さはバイト単位です。

Decoy Dog

Decoy Dog の通信は、Infoblox リゾルバだけでなく、多くの公共および商用リゾルバでも観察されました。Decoy Dog の操作とツールキットが Pupy とどのように異なるかをより深く理解するために、他のパッシブ DNS を使用して独自の収集を強化しました。2022 年 3 月 29 日から 2023 年 6 月 16 日までの期間中の合計 1,500 万件を超える DNS イベントを分析しました。さらに、ネームサーバーを積極的に調査し、収集されたパッシブ DNS トラフィックを、独自の Pupy クライアントとサーバーによって生成された DNS トラフィックと比較しました。

Decoy Dog とその操作をより深く理解するために、様々な手法を使用しました。また、パブリックリポジトリ VirusTotal で見つけたサンプルをリバースエンジニアリングし、DNS の結果を検証し、その他の機能を明らかにしました。この後のセクションでは、分析の詳細を説明し、結果を紹介します。この作業の主要な部分は次のとおりです。

- Decoy Dog は Pupy ではありませんが、マルウェアの機能を大幅に拡張し、侵害されたデバイス上での永続性を確保するのに役立つ大規模なリファクタリングです。
- これは少数の脅威アクターによって操作されており、彼らは異なる TTP を使用し、2023 年 4 月のツールキットの暴露に対して異なる反応を示しています。
- 影響を受けるデバイス全体の数は少なく、1 つのコントローラ上にあるデバイスはわずか 4 台ほどです。
- 2023 年 4 月以降に登録された新しいコントローラは、当社の最初のレポートで概説されている特性を緩和するように適応されています。これには、クライアント IP アドレスへの応答を特定の場所に制限するジオフェンシングメカニズムが含まれます。
- DNS 分析は、Decoy Dog を検出するだけでなく、その使用方法を理解し、Pupy から分離するための強力なツールであることが証明されました。DNS 分析と選択的リバースエンジニアリングを組み合わせることで、Decoy Dog とそれがもたらす脅威の確実な全体像が得られます。

キー交換

前述したように、セッションはキー交換が完了し、SPI 値が設定されると開始されます。理論上、単一の暗号化セッションは無期限に継続できますが、実際には、コントローラが新しいセッションを確立するためには多数の条件が必要となります。したがって、クライアントの単一の実行インスタンスには通常、多数のセッションが存在する可能性があります。Pupy のペイロードの署名を使用して、クライアントとサーバーの間で共有キーが生成されたタイミングを特定し、各コントローラの新しい侵害またはクライアントの再起動のいずれかによる、クライアントの初期化の回数を、時間の経過とともに大まかに見積もることができます。

下の図 6 は、いくつかの Decoy Dog コントローラのキー交換のタイムラインを示しています。一部のコントローラでは、観察されたキー交換にギャップがあります。claudfront[.] の最後のキー交換は、2022 年 12 月に観察されましたが、クライアントの活動は継続しただけでなく、2023 年に増加しました。すべての一意の SPI 値の 70% 以上が 2023 年に初めて観察されました。同様に、allowlisted[.]net のコントローラでは、2022 年 12 月から 2023 年 4 月の当社の開示後まで、キー交換はありませんでした。最後に、cbox4[.]ignorelist[.]com も、キー交換が長期間行われていないことを示しており、ドメインの運用が停止する直前に少数のキー交換が行われました。脅威アクターは、DNS とは異なるトランスポートを介してキー交換を実行するようにクライアントを再構成したのではないかと考えられます。

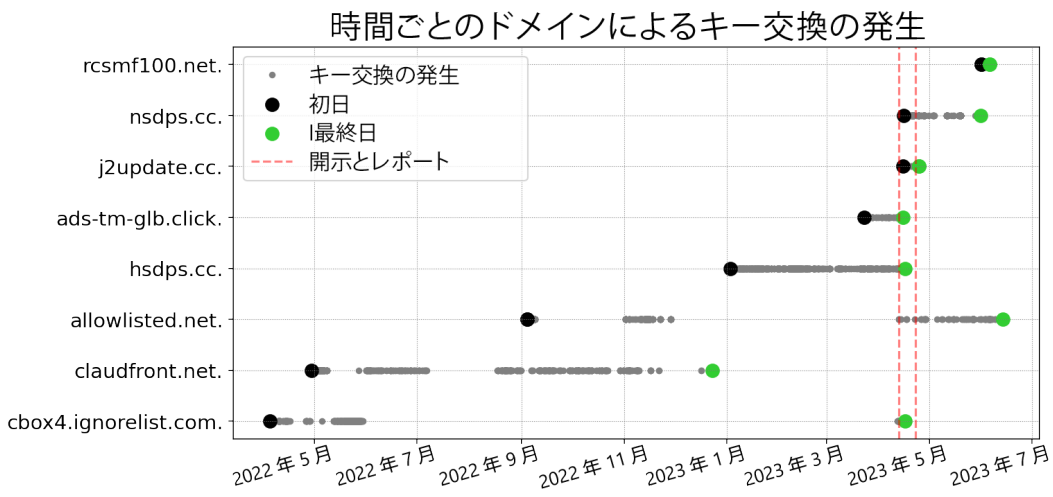

図 6. 選択した *Decoy Dog* ドメインで観察されたキー交換のタイムライン

クライアントのタイムライン

全体的なクライアント数に加えて、各コントローラが一度に維持するアクティブなクライアントの数と、クライアントがサーバーと活発に通信していた時間も決定する必要がありました。付録 C で説明されているノンス値をグループ化する方法を使用しました。この分析により、以下の図に示されているように、長期間にわたる Decoy Dog の操作についての鍵となる洞察が得られました。特に次のような点です。

- すべてのコントローラは一度に少数のクライアントを管理しており、制御しているクライアントの数は少なくとも 4 台で、すべてを合わせても 50 台未満である可能性があります。
- 元のドメインである cbox4[.]ignorelist[.]com は、大きなコントローラの 1 つであり、複数の時点でクライアントの急増が見られます。また、少数の非常に長時間実行されているクライアントも維持しています。
- 監視対象 2 番目のコントローラ、claudfront[.]net は、2023 年 2 月に活動が劇的に増加しています。
- 監視対象 3 番目のコントローラ、allowlisted[.]net は、一貫して同時に、少数のクライアントを維持しています。
- コントローラの ads-tm-glb[.]click と hsdps [。] cc は、当社の開示を受けて、クライアントを新しいコントローラに移管しました。
- Claudfront[.]net と allowlisted[.]net は、当社の開示に対応しては、運用を変更せず、cbox4[.]ignorelist[.]com は運用を停止し、hsdps[.]cc と ads-tm- glb[.]click は、クライアントを新しいドメインに移管しました。

常にクライアントの総数を見積もることは困難ですが、同時にアクティブなクライアントの数が少ないということは、これらの運用が高度に標的を絞ったものであることを示しています。また、セキュリティ事業者がこの活動を検出せず、感染したデバイスもまだ発見していない理由もこれで説明できます。感染したクライアントは非常に少数のネットワークに存在しており、明らかに DNS で C2 通信を識別してブロックできないネットワークがあります。

以下の折れ線グラフでは、単一クライアントの活動を線で表し、これをクライアントスレッドと呼びます。y 軸は、ノンスチェーンによって識別された個別のクライアントスレッドを示します。再起動またはその他の手段によって Pupy クライアントが再始動されると、新しいノンスが生成され、新しいスレッドが観察されます。一部の図では、クライアントの再始動を示す可能性がある活動に明らかな中断があります。x 軸は時間を示しています。

図 7 は、最初の Decoy Dog ドメイン cbox4[.]ignorelist[.]com のクライアント活動を示しています。最初のクライアントスレッドは 2022 年 3 月下旬に開始され、最も長いスレッドは 1 年近く続きました。このコントローラには当初少数のクライアントしか存在していませんでしたが、2022 年 5 月中旬に変化が起こり、その結果、約 40 のクライアントが同時にアクティブになったことがわかります。クライアントスレッドの同様の増加は定期的に発生し、月間増加が最も大きかったのは 2022 年 8 月でした。ただし、新しいクライアントスレッドが開始されると、他のスレッドは終了していました。年間の活動を通じて、同時にアクティブなクライアントの数は常に 50 未満であるようです。また、図 7 からは、クライアントスレッドの 4 分の 1 が 6 か月以上継続して、継続的に運用されていることがわかります。LinkedIn の投稿後にすべての通信が停止し、再び観察されていません。

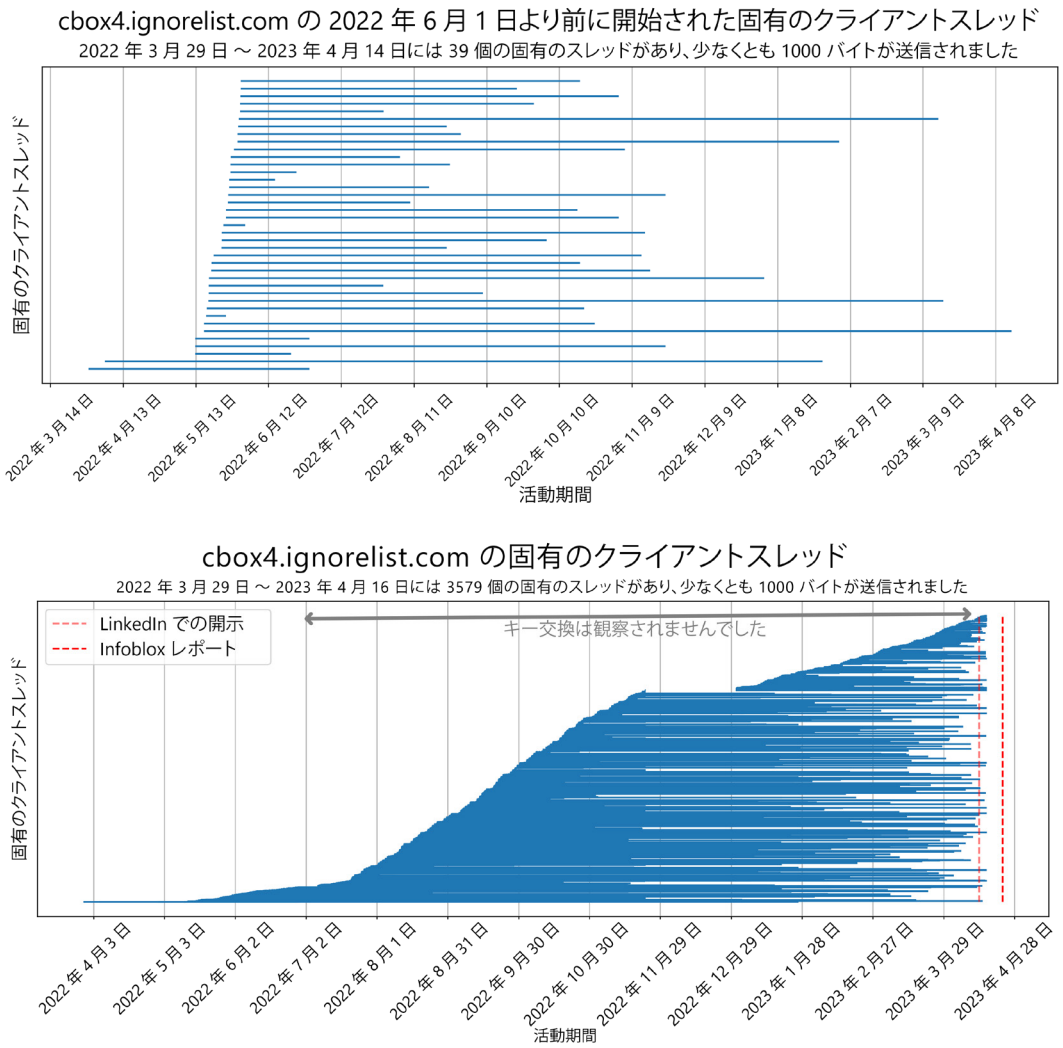

図 7. 上の図は 2022 年 6 月 1 日より前に存在していたクライアントを示し、下の図は時系列のクライアントスレッドを示しています。

claudfront[.]net の DNS 活動は、時系列的に 2 番目に出現した Decoy Dog ドメインであり、cbox4 とはかなり異なります。下の図 8 に示すように、2023 年 2 月上旬まで、このコントローラに対して同時にアクティブだったクライアントは 10 未満でした。その後、クライアント数は大幅に増加しましたが、感染が広がったというほどではありませんでした。この増加のタイミングは、2 月 13 日にコントローラドメインを含むバイナリサンプルを VirusTotal に提出する直前

です。[15] cbox4[.] と異なり、ignorelist [.] com は、当社の開示後 claudfront[.]net クエリに目立った変更はありませんでした。

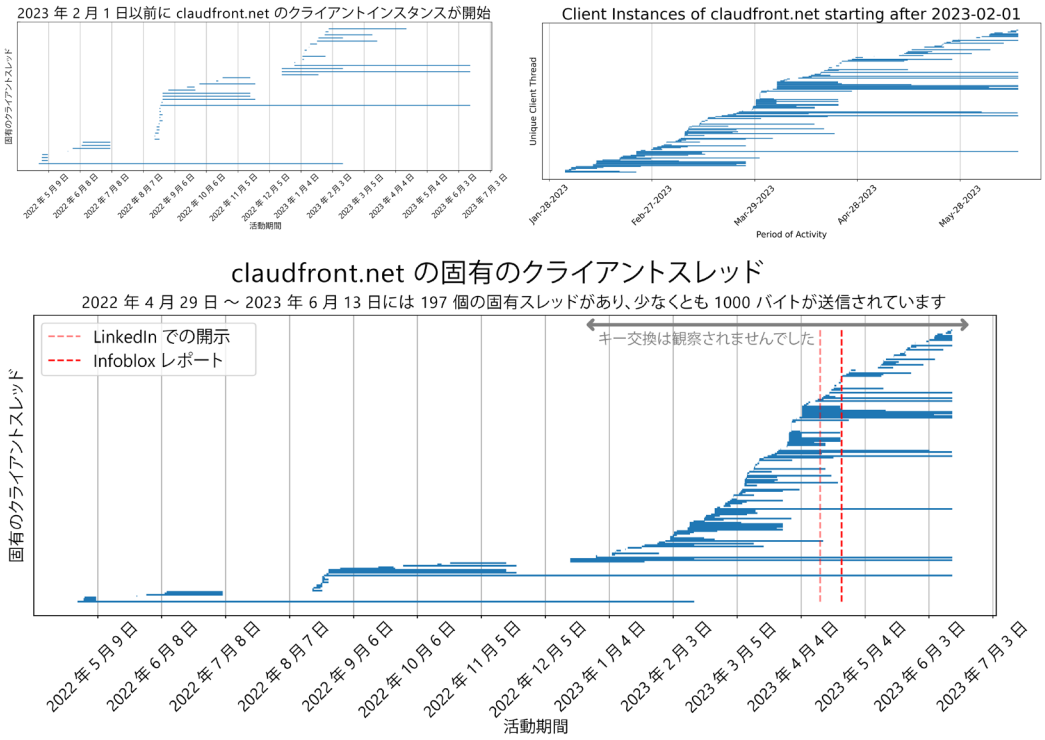

図 *8.* 時間の経過に伴うノンススレッドに基づく *claudfront[.]net* のクライアントスレッド。*2023* 年 *2* 月初旬に大きな変化があります。この変化は、個別の期間を示す個別画像が拡大表示されています。

3 番目のドメイン、allowlisted[.]net は、動作において別の変化を示しています。このケースでは、クライアントの数は常に 10 以下という少人数です。claudfront[.] net とは異なり、2023 年 2 月に変化はなく、利用可能な allowlisted[.]net を含む、allowlisted[.]net バイナリサンプルはありません。図 9 に示すように、2022 年 11 月中旬から当社の開示直後までキー交換は見られませんでしたが、これは 2023 年 4 月にクライアント活動が急激に終焉し、いくつかのスレッドが再開されるのと一致しています。

15 0375f4b3fe011b35e6575133539441009d015ebecbee78b578c3ed04e0f22568 は、最初に提出された 2023-

2月13日 07:39:55 UTC

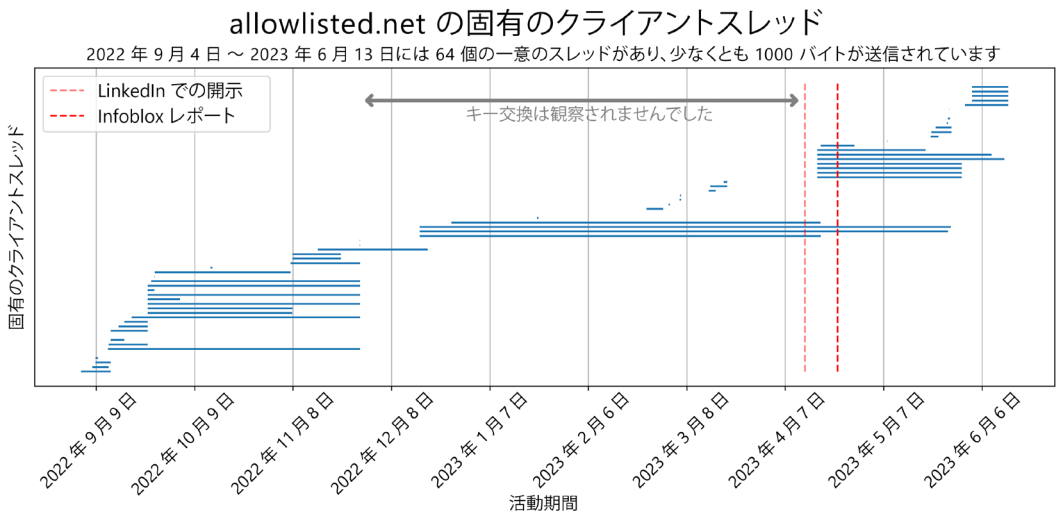

図 *9.allowlisted[.]net* のクライアントスレッド。*allowlisted[.]net* 上のクライアントの数はこれまでずっと非常に少なく、これは情報開示後も変わっていません。

最後に、hsdps[.]cc、nsdps[.]ccm、ads-tm-glb[.]click、j2update[.]cc による関連活動を観察しました。ソーシャルメディアでの開示後に hsdps[.]cc と ads-tm-glb[.]click のドメインは、運用を停止しましたが、クライアントの一部は nsdps[.]cc と j2update[.]cc のそれぞれに移管されました。これを発見できたのは、時間の経過とともにすべてのドメイン全体にノンスチェーンを作成し、あるコントローラとの通信を開始し、別のコントローラで終了するスレッドを特定したからです。[16]

新しいドメインの、nsdps[.]cc と j2update[.]cc は、ソーシャルメディアでの発表から 48 時間以内に登録されました。クライアントスレッドのグラフから、あるドメインセットが活動を停止し、他のセットが開始していることがわかります。そのほぼ直後に、コントローラはクライアントと活発に通信を開始しました。DNS 分析によるクライアント転送の発見に続き、後述するように、この変更を行うコマンドのバイナリサンプルの証拠が見つかりました。

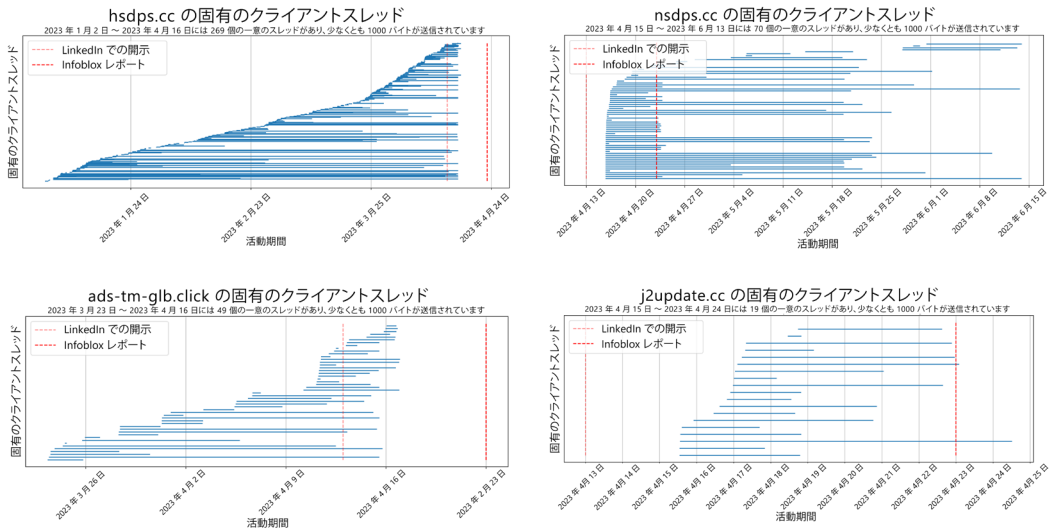

図 *10. 4* 個の *Decoy Dog* コントローラドメインのタイムラインでの比較。*hspds[.]cc* と *ads-tm-glb[.] click* のコントローラは、*Infoblox* の開示後に、通信を停止し、*nsdps[.]cc* と *j2update[.]cc* のドメインが通信を開始。また、これらのドメイン間でのクライアントの移動も確認しました。

16 32 ビットのナンスでこれがランダムに発生する確率は非常に低く、これらのドメインでは、あるコントローラから別のコントローラへのノンスの「転送」の数が多く発生しました。

最初のレポート以来、非常に少数のクライアントを持つ追加のコントローラがアクティブになるのを確認してきました。ここで示されているクライアントの動作は、当社の発表に連動しており、Decoy Dog ツールキットが複数の脅威アクターによって使用されていることを示しています。

DECOY DOG ペイロード署名

私たちは、13 か月間にわたってグローバル pDNS で観察された 1,550 万件のクエリ応答のクライアントとサーバーのペイロード長を解読しました。次に、サーバーの動作を理解するために、クライアント / サーバーのペイロードの Pupy の署名と Decoy Dog の観察データを比較しました。全体的なトラフィック分布は Pupy と一致していることがわかりましたが、明らかな違いもありました。Decoy Dog クライアントは、デフォルトの Pupy よりも大量のリクエストまたは語彙のセットを利用しています。

図 11 は、すべての Decoy Dog システムにわたるペイロード長ペアの相対分布を示しています。付録 D で詳しく説明しているように、Pupy 署名を使用すると、すぐにいくつかの結論を引き出すことができます。

- 予想された 9 個を超えるクライアントペイロードの存在。
- 当社の研究室では観察されていないサーバーのペイロード長があった。
- 通信の大部分は、セッションの維持とキー交換に関連していた。
- Decoy Dog サーバーへのクエリの大部分はエラー応答を受け取り、真のクライアントではなくサードパーティによるスキャンと一致する変動を示して、そのほとんどは、当社による発表後に発生していた。

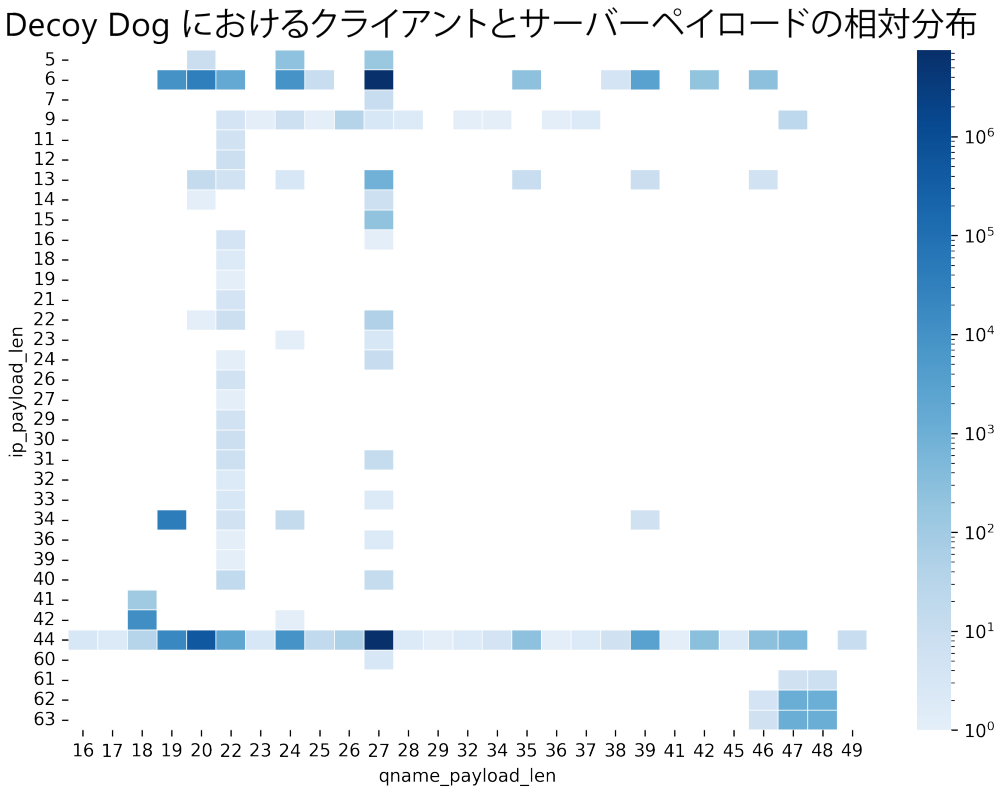

図 *11. Decoy Dog* 通信で観察された、クライアントとサーバーのペイロード長の相対的な分布

固有のクライアントペイロード長には、20、25、38、42、46 が含まれていました。これらの一部は、別のキー構成またはポーリングパラメータの変更に関連付けられている可能性があります。通信が何であったかを特定することはできませんが、バリエーションは存在しています。さらに、Pupy で観察されたものを超える追加の応答ペイロード長がありました。最も注目すべきなのは、

Decoy Dog は、サーバーペイロードは 13 バイトで、活動が活発になると時間の経過とともにこれが確認されます。このペイロードが何であるかを特定することはできませんが、クライアントへの送信に 8 バイトのデータを必要とする 1 つのコマンドと一致しています。また、5 バイトのペイロードを含む多数のサーバー応答も確認されました。これは、当社の Pupy データでは観察されなかった別の長さであり、これはクライアントへのデータ転送を必要としない単一のコマンドを示しています。下の図 12 では、Decoy Dog で見つかり、当社の　Pupy の実験では見られなかった固有のペイロードペアをまとめています。

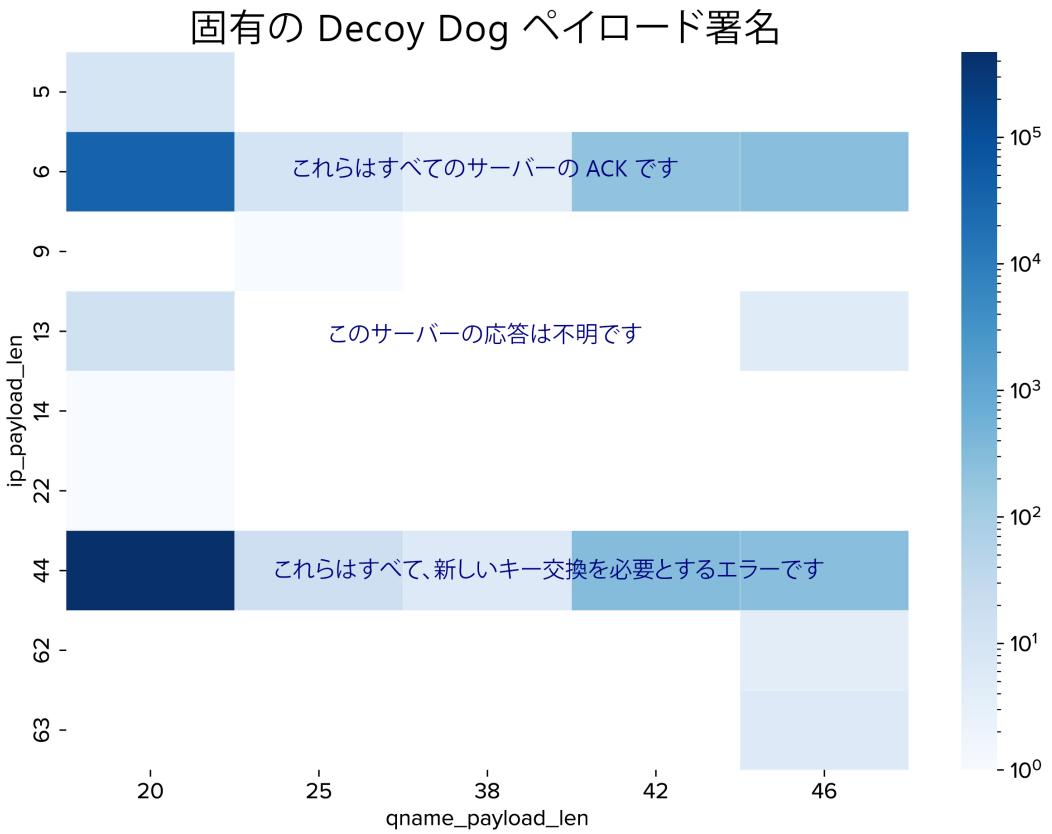

図 *12. Decoy Dog* で観察され、デフォルトの *Pupy* 通信では見つからなかったクライアントとサーバーのペイロード長ペアの要約

また、時系列を使用して、デフォルト構成への変更を特定しました。確立された Pupy セッションでは、クライアントは 30 秒ごとにチェックインします。クライアントのハートビートクエリの変動に関する統計分析を使用して、デフォルトの 30 秒に加えて、2 分と 30 分のハートビート間隔があることがわかりました。

この分析の結果、各 Decoy Dog ドメイン通信の性質を理解し、日常的な保守管理とリモートアクセスコマンドを分離できました。また、Decoy Dog サーバーのサブセット間とサブセット内で使用されている可能性のある Pupy のカスタマイズを分離することもできました。Decoy Dog のトラフィックの大部分は日常的な確認応答とエラーであり、エラー通信は Pupy の観察に基づいて予想されるものとは不釣り合いであることがわかりました。次のセクションでは、このエラー応答現象について調査した結果をご紹介します。

ワイルドカードとジオフェンシングの動作

私たちは最初の技術調査レポートで、Decoy Dog サーバーが再生された DNS クエリに応答したことを報告しました。これは依然として謎のままです。Decoy Dog が、もともと数日または数週間前に行われたクエリにいつ、どのように応答するかを理解しようとしたところ、さらに驚くべき動作が明らかになりました。Decoy Dog サーバーのいくつかは再生に応答するだけでなく、Pupy のコード化と一致するあらゆるクエリにも応答します。DNS では、これをワイルドカード応答と呼びます。通常の Pupy サーバーは NXDOMAIN または SERVFAIL 応答を返しますが、Decoy Dog サーバーは通常 15 個の IP アドレスを返します。

下の図 13 は、ランダム化されたクエリに対する応答を示しています。このケースでは、クエリ名内に「wild」と「wildcard」という単語を入れ、2 つの異なる Decoy Dog サーバーから 15 件の回答を受け取りました。応答はクエリごとに異なり、Pupy のコード化スキームに従っています。私たちの調査を通じて、Decoy Dog は期待される NXDOMAIN 応答を返すのではなく、ほとんどすべてのエラーをこの方法で処理していることがわかりました。エラー処理の詳細については、付録 E を参照してください。

```
; <<>> DiG diggui.com <<>> @ns1.rtupdates.net wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22151
;; flags: qr aa rd; QUERY: 1, ANSWER: 15, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. IN A

;; ANSWER SECTION:
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 64.88.80.242
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 131.163.188.250
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 68.221.203.220
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 198.206.187.196
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 200.37.65.250
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 75.195.241.234
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 141.67.92.44
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 142.153.85.81
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 209.92.80.161
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 147.26.100.52
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 213.83.7.105
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 150.143.51.118
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 153.171.88.194
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 219.226.5.44
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.rtupdates.net. 60 IN A 157.111.237.108

;; Query time: 150 msec
;; SERVER: 5.252.179.232#53(5.252.179.232)
;; WHEN: Sat Jun 03 15:29:11 UTC 2023
;; MSG SIZE  rcvd: 321
```

```
; <<>> DiG diggui.com <<>> @ns1.allowlisted.net wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33023
;; flags: qr aa rd; QUERY: 1, ANSWER: 15, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. IN A

;; ANSWER SECTION:
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 64.88.161.73
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 67.179.145.230
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 69.153.193.38
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 71.14.146.226
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 73.22.176.2
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 138.151.231.153
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 141.232.226.212
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 79.241.118.178
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 209.158.29.150
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 147.248.180.89
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 148.158.234.156
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 215.63.12.236
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 153.141.240.250
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 219.18.219.74
wildcard5yt1j46ra2gwossxhw1q9999.2vzxfwild3999999.allowlisted.net. 60 IN A 156.250.150.9

;; Query time: 151 msec
;; SERVER: 83.166.240.52#53(83.166.240.52)
;; WHEN: Sat Jun 03 15:31:15 UTC 2023
;; MSG SIZE  rcvd: 323
```

図 *13. 2* つの *Decoy Dog* 権威サーバーからのワイルドカード応答動作。どちらの場合も、サーバーは文字列「*wild*」と「*wildcard*」を含む同じランダム化クエリに対して、*Pupy* のコード化と一致する *15* 個の *IP* アドレスで応答しました。

さらに驚くべきことに、一部の Decoy Dog サーバーは、クライアントに代わってクエリを実行する再帰リゾルバの IP アドレスに応じて異なる応答をします。図 14 は、Decoy Dog ドメイン nsdps[.]cc へのクエリの再生を示しています。このドメインははもともと数週間前に発生しました。Yandex パブリックリゾルバ経由でクエリを作成すると、15 個の IP アドレスを含む応答を受け取りました。また、ロシアの TimeWeb パブリックリゾルバから 15 個の IP アドレスも受け取りました。しかし、私たちが試した 30 を超えるパブリックリゾルバのうち、他に応答を返したものはありませんでした。このタイプの動作はジオフェンシングと一致しており、サーバーは IP アドレスの地理的位置に基づいて DNS クエリに応答します。この動作を 2023 年 6 月に発見し、一部のサーバーはロシアの IP アドレス経由で DNS クエリを転送した場合にのみ応答し、他のサーバーは任意の場所からの適切な形式のクエリに応答することがわかりました。このタイプの選択的応答により、コントローラはロシアにいると思われるクライアントとのみ通信することが保証されます。コントローラは以前に Infoblox 再帰リゾルバからのクエリを解決していたため、この機能は公開後に追加されたことがわかっています。

```
; <<>> DiG diggui.com <<>> @77.88.8.8 qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42579
;; flags: qr rd ra; QUERY: 1, ANSWER: 15, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. IN A

;; ANSWER SECTION:
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 46 IN A 72.11.125.198
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 36 IN A 203.92.202.218
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 45 IN A 76.74.229.130
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 44 IN A 207.26.86.188
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 31 IN A 80.154.112.164
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 43 IN A 146.160.113.9
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 52 IN A 148.235.159.60
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 41 IN A 151.103.182.130
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 54 IN A 89.76.7.130
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 45 IN A 218.111.60.250
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 42 IN A 93.43.159.18
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 49 IN A 128.88.84.164
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 45 IN A 195.161.207.129
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 51 IN A 68.172.178.156
qnomwvxags5u1pawkzmkllkmuoda9999.abp11i4yk5y1fqyd4tpzm1q9.nsdps.cc. 49 IN A 199.24.240.30

;; Query time: 459 msec
;; SERVER: 77.88.8.8#53(77.88.8.8)
;; WHEN: Tue Jun 20 14:31:11 UTC 2023
;; MSG SIZE  rcvd: 335
```

```
; <<>> DiG diggui.com <<>> @74.82.42.42 hoxlgxq9.yopzgoha3rlp4pdcclosfb63yodq9999.enueh2eluu6uqnjtjpid4lq9.nsdps.cc A
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

```
; <<>> DiG diggui.com <<>> @ns2.nsdps.ns2.name hoxlgxq9.yopzgoha3rlp4pdcclosfb63yodq9999.enueh2eluu6uqnjtjpid4lq9.nsdps.c
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

図 *14. Yandex* パブリックリゾルバ、*Hurricane Electric* パブリックリゾルバ、権威リゾルバからの、再生された *Decoy Dog* クエリに対する応答の比較。これらのクエリは *Tor* ブラウザ経由で連続して実行されました。*Yandex* 経由のクエリのみが応答を受け取りました。

Pupy のデフォルトのコード化 (このテストのために余分な文字を追加しました) を使用して復号できないドメイン名に対してクエリが行われた場合、nsdps[.]cc サーバーは本質的にシンクホールである IP アドレスを返します。下の図 15 に示すように、正しく復号できないようにクエリを少し変更しました。このケースでは、172.0.0.0/8 の範囲内のランダムな IP アドレスが返されました。通常、Pupy は NXDOMAIN 応答を返します。

```
; <<>> DiG diggui.com <<>> @77.88.8.1 hoxlgxq9.yopzgoha3rlp4pdcclosfb63yodq9999.wildenueh2eluu6uqnjtjpid4lq9.nsdps.cc A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2019
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;hoxlgxq9.yopzgoha3rlp4pdcclosfb63yodq9999.wildenueh2eluu6uqnjtjpid4lq9.nsdps.cc. IN A

;; ANSWER SECTION:
hoxlgxq9.yopzgoha3rlp4pdcclosfb63yodq9999.wildenueh2eluu6uqnjtjpid4lq9.nsdps.cc. 32 IN A 172.67.132.113

;; Query time: 1695 msec
;; SERVER: 77.88.8.1#53(77.88.8.1)
;; WHEN: Tue Jun 20 23:44:46 UTC 2023
;; MSG SIZE  rcvd: 124
```

図 *15.* コントローラ *nsdps[.]cc* の無効な *Pupy* ドメイン名のクエリは、予想される *NXDOMAIN* 応答の代わりに、*172.0.0.0/8* 範囲のランダムな *IP* アドレスを返します。

この動作の一部は、クライアントからの DNS 解決の生成物として説明される場合があります。DNS でホストがクエリされると、一部のリゾルバは、将来のクエリに備えるために、関連する可能性のあるドメイン名を解決しようとします。例えば、www[.]baddomain[.]com のクエリを受信する再帰リゾルバは、www[.]baddomain[.]com に加えて baddomain[.]com の解決を試みる可能性があります。この動作は、いくつかのパブリックリゾルバを介してクライアントクエリをルーティングするときに、当社の Pupy サーバーで確認されました。

単一ラベルの応答

デフォルトでは、Pupy はクライアント通信の構造または確立された ping クエリに一致しないラベルへの受信リクエストを拒否します。ただし、上記の「特別なドメイン名の処理」セクションで説明したように、DNS アクティブ化リクエスト機能を使用すると、脅威アクターはカスタムリソースのクエリに応答するように Pupy サーバーを構成できます。グローバル pDNS ログでは、単一ラベルのサブドメインを持つクエリを特定しました。そのようなサブドメインは「m」のみであり、これらのドメインの解決はアクティベーション関数によって可能であると仮説を立てました。アクティベータハッシュ関数の性質により、これらのクエリに対して単一の静的 IP アドレスが返される必要があります。この動作が確認されたのは次の 4 つのドメインです : hsdps[.]cc、nsdps[.]cc、j2update[.]cc、ads-tm-glb[.]click。これは、他のコントローラには見られない、この一連のドメインのもう 1 つの共通の特性です。これらはそれぞれ単一の IP アドレスを返しましたが、予想していた静的 IP アドレスではなく、応答内に 104 個の一意のアドレスが見つかりました。これはデフォルトの Pupy との機能の違いを示しているようですが、目的は不明です。

バイナリサンプル分析

DNS の発見に続いて、VirusTotal で入手可能なバイナリサンプルを調べて、Pupy との違いの原因がすぐに明らかかどうかを判断しました。2 つの Decoy Dog サンプルのインポートと機能テーブルを分析することにより、Decoy Dog に埋め込まれた特有の固有の署名を特定し、これにより追加の Decoy Dog サンプルを発見できるようになりました。これらのサンプルのリバースエンジニアリングにより、Decoy Dog は Pupy とは大きく異なり、最も成熟したコードは 2 番目の開発者によって作成された可能性があるということをさらに確認しました。クライアントは Python 3.8 にアップグレードされ、いくつかの新しいトランスポート、アップグレードされた暗号化、カスタムコマンド、新しい DNS 機能が含まれています。あるコントローラ、claudfront[.]net に関連するサンプルには、他には見られない機能が含まれています。このセクションでは、いくつかの重要な発見とプロセスについて説明します。技術的な詳細については付録 F を参照してください。バイナリに関連する分析データも 当社の GitHub リポジトリに追加されます。

最初のサンプルは 2022 年 9 月にアップロードされ、他のサンプルは 2023 年にアップロードされました。そのうちの 3 件は当社の開示を受けてのものです。様々な Decoy Dog サンプルの構成を抽出して比較したところ、暗号化キーがサーバー間で異なることがわかりました。cbox4[.]ignorelist[.]com と通信したすべてのサンプルには、同じ RSA キーと SSL キーが含まれていて、異なるサンプルの存在がサーバーキーの変更に関連していないことを示しています。復号化されたキーの完全なリストは、付録 I で詳しく説明されている Github リポジトリをご覧ください。サンプル内の最も初期の SSL 証明書は 2021 年 12 月 26 日に生成され、最初に観察されたコントローラである cbox4[.]ignorelist[.]com に属しています。

重要な発見の 1 つは、Decoy Dog の Pupy クライアントにカスタムコードが含まれており、攻撃者が Java モジュールを JVM (Java 仮想マシン) スレッドに挿入することによって、実行時に Java モジュールを送信して実行できるようにするということでした。この機能は Pupy の標準バージョンにはありません。このコードはすべての Decoy Dog サンプルで見つかっており、すべてのインスタンスで同一です。既知のすべての Decoy Dog クライアントサンプルの残りのバイナリ関数は、基本の Pupy クライアントの関数と同一です。

Java モジュールを含めると、答えよりも、もっと多くの疑問が生じます。デフォルトで、Pupy はすでに高度な機能を備えており、そのまま使用できる Python モジュールの使用に対応しています。これらの機能を拡張して Python モジュールを作成することは、サーバー側の変更やクライアントバイナリの変更を必要としない簡単なプロセスです。Java モジュールを実行するた

めの Python モジュールの作成は簡単にできるでしょう。対照的に、jni.h を使用せずに実行時に Java モジュールの挿入を jni.h (または標準 Java/C API の残りの部分) を使用せずに行うのは簡単な作業ではなく、専門知識が必要です。したがって、これらの Java モジュールを追加することで、攻撃者は Python を実行していないシステム、特権のある、あるいは監視されていない Java 仮想マシンを実行しているシステム、あるいは攻撃者がファイルを作成しないことでマシンに証拠を残さないようにするシナリオをターゲットにすることができる可能性があります。

クライアントには、時間の経過とともに成熟した新しい機能もあります。クライアントソフトウェアは、Python 構成ファイルを特定のバイナリにマーシャリングすることによって作成されます。構成ファイルには、設定、通信に必要なすべてのキー (RSA、SSL 証明書、パスワードなど)、クライアント Python モジュールが含まれています。サンプル内で見つかったモジュールは、解凍されて侵害されたデバイスによって実行されますが、公開されている Pupy コードとは大きく異なります。

埋め込みモジュールを抽出して分析すると、Decoy Dog の開発とカスタム変更の興味深いストーリーが描かれます。まず、かなりの数の Pupy モジュールが Decoy Dog から単純に削除されました。おそらく、攻撃者がそれらを役に立たないと判断したためです。第 2 に、同様のサンプルでもモジュールに多数の違いがあり、場合によっては機能が大きく異なることがあります。第 3 に、多数の変更と新しい機能によって追加された複雑さにより、Pupy の開発にはかなりの時間がかかり、リソースの微調整が必要になります。さらに、Pupy のコードベースとモジュールは、Python 2.7 から Python 3.8 へ移植され、コードの品質、メモリ操作の安定性、Windows との互換性が改善しました。サンプルには、時間の経過とともに、バージョン 3 から 4 に変更しているクライアントバージョンが含まれています。最新の Pupy クライアントはバージョン 2 です。コードの成熟度や主な機能と比較して、提出日をまとめたタイムラインは以下の図 16 にある通りです。

変更されたモジュールの性質と数を分析することで、コードの成熟度の観点から、次のハッシュは、公開されている最も古い Decoy Dog バイナリであると特定できました。ad186df91282cf78394ef3bd60f04d859bcacccbcdcbfb620cc73f19ec0cec64 。このハッシュは、ネームサーバーの cbox4[.]ignorelist[.]com と通信します。Pupy と最も多くのコードを共有していますが、このサンプルは私たちのレポートが公開されてから数日後の 2023 年 4 月 27 日まで VirusTotal にアップロードされませんでした。しかし、付属の SSL 証明書によると、このサンプルは 2021 年 12 月にまで遡る可能性があります。開発者は、特定のポーリング機能、XOR 関数、新しいトランスポート、マルチスレッドネットワーク通信の完全なサポートを追加しました。興味深いことに、これまでのサンプルはすべて Linux ライブラリであるにもかかわらず、多くの新しいモジュールが特に Win32 をターゲットにしています。この実行可能ファイルでは、DNS 通信の処理を担当するコードはデフォルトの Pupy と同じです。

時間が経過すると、サンプルの cbox4[.]ignorelist[.]com との通信は、より複雑になりました。一連の 3 つのサンプル上に、SSL、TCP、UDP モジュールの完全な書き換えだけでなく、同期 HTTP (BOSH) を介して双方向ストリームを使用して通信するモジュール全体を含む、通信モジュールの数が追加されました。また、Decoy Dog の背後にいる脅威アクターは、既存のエクスプロイトと通信モジュールを Windows プラットフォームに移植するために多くのスクリプトを追加し、DNS 通信を担当する picocmd クライアントを書き直し、古いコードの質と安定性の改善を実装しました。コード内の Windows への参照は、新しい Decoy Dog 機能を含む更新された Windows クライアントの存在を示唆していますが、現在のサンプルはすべて Linux をターゲットとしています。

新しいバージョンには、マルウェアが長期間にわたって C2 サーバーと通信できない場合に、侵害されたマシンがサードパーティの DNS サーバーに接続できるようにする緊急モジュールも含まれています。このモジュールは、DGA を使用して、無料のダイナミック DNS サービス内でクライアントがクエリするドメインを選択します。これらのバージョンでは、C2 コントローラの位置を特定するためのブートストラップ、ビーコンドメインの確立、緊急サービスへの CNAME クエリの組み込みも可能です。クライアントバージョン 3 以降に見られる広範な永続化メカニズムは、金銭的な動機を持つ攻撃アクターやレッドチームによって実行されるものではなく、インテリジェンス操作に最も頻繁に関連する機能です。

最も成熟したコードは、コントローラ claudfront[.]net に接続し、AlterDnsCncDomain と CompromizedNode と呼ばれる 2 個の新しいコマンドを含みます。前述したように、クライアントのノンス値の分析により、一部の Decoy Dog アクターが当社の開示後にクライアントを新しいコントローラに移行したことが判明しました。一般に入手可能な Pupy ソースコードに基づいて、カスタムコマンドを使用せずにこれがどのように可能であるかはわかりませんでした。おそらく AlterDnsCnCDomain コマンドがこれらのクライアント移管のソースであり、したがって nsdps[.]cc に関連付けられているコントローラが最も高度なコードを使用している可能性があります。このコードが他のコードから大きく逸脱していることは、新しい開発者が関与していることを示している可能性があります。このコードには、クライアントのバージョン 4 が含まれています。

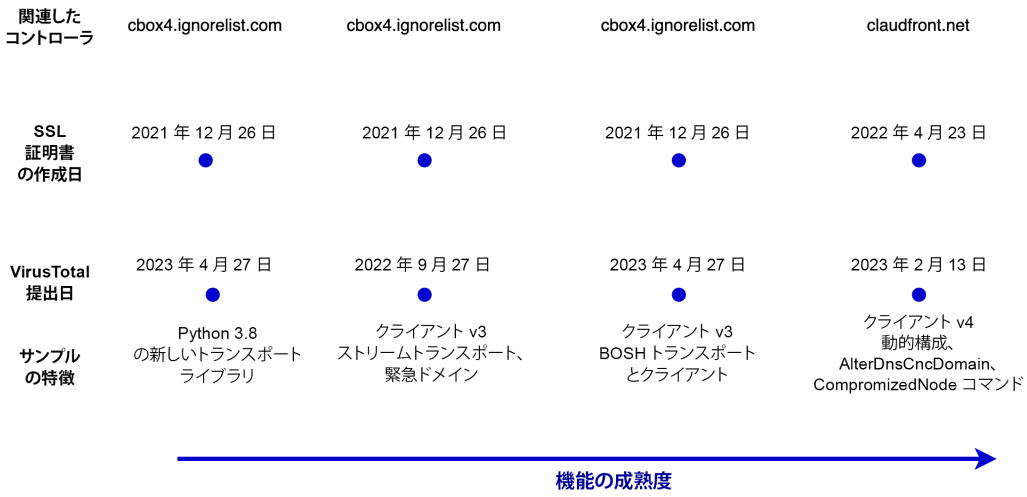

図 *16. VirusTotal Decoy Dog* 関連の提出とコードの成熟度のタイムライン

Decoy Dog の機能強化にも関わらず、Pupy のより基本的なバージョン用に開発された YARA ルールは依然としてマルウェアを検出できることは注目に値します。ただし、サンプルが既知のコードと機能から大幅に逸脱していることを検出することはできません。これにより、マルウェア研究者は、両方のタイプのマルウェアが同じルールによってフラグ付けされるため、Decoy Dog サンプルが単なる基本的な Pupy であるという誤った仮定に陥る可能性があります。このため、付録 G に Decoy Dog の新しい YARA ルールを含めました。

コントローラの比較

Infoblox は現在、21 個の Decoy Dog ドメインを追跡しています。これらの多くは観察可能な C2 活動がほとんど、またはまったくないため、現時点では開示していません。一部のコントローラはソーシャルメディアでの最初の開示後に変更され、残りは最初のレポートをリリースした後に変更されました。それらはすべて、運用を停止するか、クライアントを新しいコントローラに移動するか、レポートで説明した「ping」動作を変更するかのいずれかで対応しました。ジオフェンスを追加したケースもあります。使用されている他の TTP と連動した、これらの応答により、現時点で少なくとも 3 つの脅威アクターがツールキットを利用していると結論付けることができます。下の表 1 では、動作や類似の特性に基づいてコントローラドメインのサブセットをグループ化しています。

ドメイングループ	特徴
cbox4.ignorelist[.]com	• 最初のアクティブなドメインで、Decoy Dog ツールキットのソースと思われる • 開示後に無効化された • Araid ダイナミック DNS の使用 • ハートビート間隔が 30 秒 • ジオフェンスなし • 少なくとも 3 個の異なるクライアントソフトウェアの反復 • 2022 年 3 月下旬に初めて観察されたが、2021 年 12 月には存在していた可能性あり • クライアント v2 と v3
claudfront[.]net allowlisted[.]net maxpatrol[.]net atlas-upd[.]com	• アクティブコントローラの 2 番目のセット、2022 年 5 月から開始 • 開示後も運用を継続 • Namecheap に登録 • リモート暗号化通信が初めて確認された前の ping12.<domain> • ping 応答を NODATA 応答に変更 • ロシアの IP ホスト • ハートビート間隔は 30 秒 • ジオフェンスなし • クライアント v3 と v4 • allowlisted[.]net と claudfront[.]net 間にいくつか相違がありこれは、異なる脅威アクターを示唆している可能性あり
hsps[.]cc nsdps[.]cc j2update[.]cc ads-tm-glb[.]click	• アクティブコントローラの 3 番目のセット、2022 年 12 月に開始 • 開示後にコントローラ間でクライアントを移動 • 最初のコントローラを停止 • ハートビート間隔は 2 分と 30 分 • 開示後にジオフェンス設定 • ping 応答を単一の非ローカルループバック IP アドレスに変更 • 単一ドメインラベル (m) の使用 • おそらくクライアント v4
rcmsf100[.]net	• 2023 年 6 月に初めて観察 • allowlisted[.]net とホスティングを共有 • NODATA の ping 応答 • ジオフェンス設定あり

表 *1.* いくつかの *Decoy Dog* コントローラの比較。

INFOBLOX ネットワーク中の DECOY DOG

Infoblox は、当社のリゾルバが、Decoy Dog クエリを再生するセキュリティ事業者のスキャナーによって作動されたと判断しました。スキャナーの動作と Decoy Dog の動作の組み合わせにより、検出された信号が作成されました。インターネットスキャンは今では広く知られたビジネスになり、スキャンは今や大量のインターネットトラフィックを占めています。これは、正規のアクターと悪意のあるアクターの両方によって実行されます。最近の研究では、ダークネットによる観測を使用して、これらのスキャンの影響を把握しました。[17] ほとんどのスキャンはポートスキャンに限定され、グローバルな IP 空間全体で開いているポートを特定する試みで、環境内では他にも様々なスキャン活動があります。例えば、開いているディレクトリを検索するスキャナやオープン DNS リゾルバがあります。一部の組織はスキャン活動を完全に文書化していますが、多くの組織は文書化していません。

「攻撃的なスキャン」とは、ネットワークのパフォーマンスを低下させる可能性のある、未承認または大量のスキャン活動です。ネットワークへのサービス拒否を引き起こしたり、Decoy Dog の場合のように偽のセキュリティイベントを作成したりする可能性があります。[18] 攻撃的なスキャンは、ネットワークの所有者が活動に同意していないネットワークを犠牲にして、運用者に利益をもたらします。2023 年 4 月、Decoy Dog が検出されたネットワークのセキュリティチームは、システムが侵害されていないことを確認するために、これらの DNS クエリの根本原因を見つけるために多大なリソースを費やしました。これらのクエリは、主にファイアウォールから発信されていたため、特に憂慮すべきものであり、ファイアウォール業界はここ数か月、ファイアウォールへの攻撃に対する懸念を高めています。[19]

Decoy Dog クエリがどのようにしてリゾルバに到達したのか、そしてなぜそれらのクエリが標的のマルウェア C2 ビーコンに似た信号を引き起こしたのかは複雑に入り組んでいます。同様の活動に対する防御側の認識を支援するために、図 17 に簡単な説明と図を示します。

Infoblox が Decoy Dog DNS クエリを受信するには、顧客ネットワークが DNS プロバイダーとして Infoblox を使っている必要があります。さらに、お客様はファイアウォールなどのセキュリティ装置を備え、受信 URL フィルタリングとそのデバイスから当社のリゾルバへの DNS 転送の両方が設定されている必要があります。これらの基準だけで制限がつきます。これらが満たされると、次のシーケンスが発生します。

- スキャナーは、ネットワーク内の IP アドレスから直接マルウェア C2 のコンテンツを取得しようとし、これらの DNS C2 通信が Web コンテンツではない場合でも、これが実行される。
- セキュリティ装置はそのリクエストを傍受し、ドメイン名の解決を試みる。
- DNS リクエストは Infoblox に転送され、そこでクエリを解決して、応答を返し、ドメインがお客様によって構成された DNS ブロックリストに含まれている場合は、結果は返さない。
- ベンダーによってスキャンされているドメインが Decoy Dog またはその他のマルウェアではない場合、そのドメインは解決され、ファイアウォールのルールに応じて、Web サイトのコンテンツがスキャナーに返される。

---

17 攻撃的なインターネット全体のスキャナー: Network Impact and Longitudinal Characterization (ネットワークへの影響と縦断的な特徴づけ)、2023 年 5 月、Anand, Dainotti, Sippe, Kallitsis https://arxiv.org/pdf/2305.07193.pdf

18 https://live.paloaltonetworks.com/t5/general-topics/spurious-hits-from-the-expanse-webcrawler/td-p/447239 、最終アクセス日 2023 年 6 月 11 日

19 https://blog.talosintelligence.com/state-sponsored-campaigns-target-global-network-infrastructure/、最終アクセス日: 2023 年 6 月 11 日

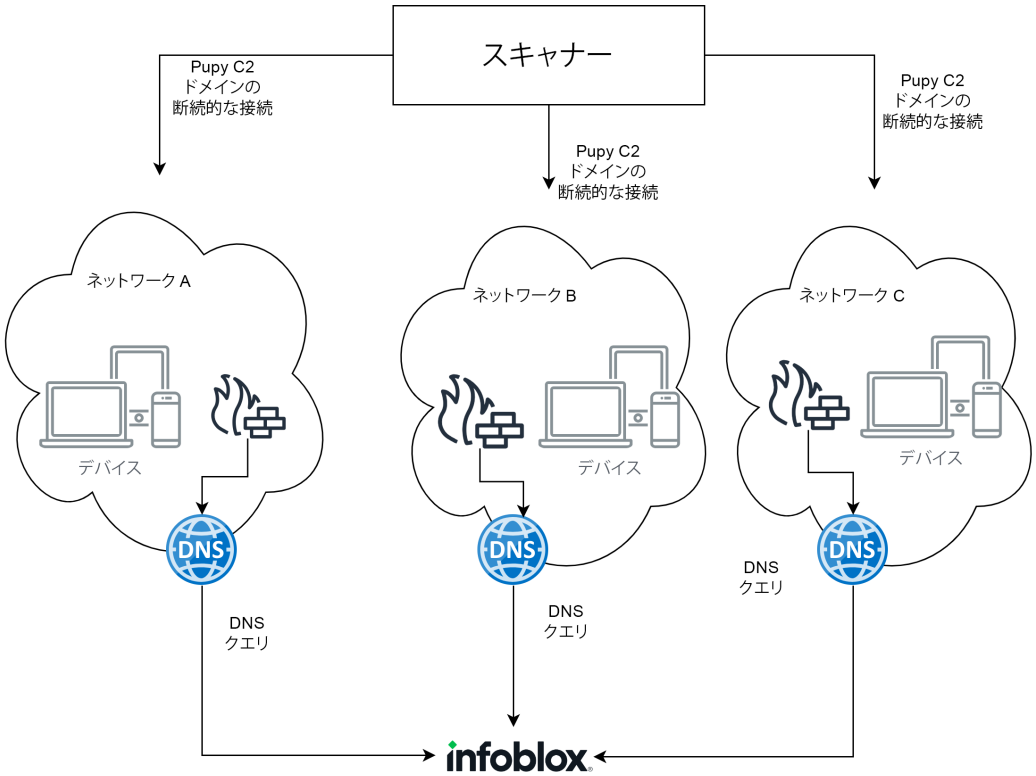

図 *17 Decoy Dog DNS C2* ドメインのクエリは、異なるネットワーク内のデバイスから *Infoblox* リゾルバに対して作成されました。これらは市販のスキャナーによって引き起こされ、断続的に作動されました。

Infoblox は、IP アドレスに既知のオープンポートがない場合でもベンダーがスキャンを実行し、一般的なポートに加えてまれなポートを使用すると判断しました。ベンダーが使用する IP アドレスとポートをどのように決定するかはわかりません。このような類の無差別な攻撃的なスキャンの結果、非常に機密性の高いデバイスが、実際は侵害されていない場合でも侵害されているように見える可能性があります。ベンダーは広範囲かつ継続的にコンテンツをスキャンしているようですが、Infoblox は上記の基準が満たされた場合にのみ DNS クエリを監視しました。その結果、ベンダーによって行われたスキャンの数は非常に多く、これは攻撃的なスキャンと一致していますが、時間の経過とともに断続的に少数のクエリのみを解決しました。このタイプの構成では、攻撃者が特定のネットワーク上で偵察を実行できるようになります。これについては付録 H で説明しています。

Infoblox Intelligence は、すべての DNS 活動の履歴を保管し、それらを使用してネットワークとグローバル DNS 内のドメイン活動を集計した統計を作成して維持します。当社はこれらの集計を使用して、マルウェアの C2 ビーコンと一致する異常な動作を含む幅広い脅威を特定します。特に、時間の経過とともに、異常な数の顧客ネットワークでクエリが発生し、データ窃取と一致するサブドメインがあり、期待される動作に比べてクエリ数が少ないドメインを探しています。これを遂行するために、私たちは数年間にわたって観察したすべてのドメインの統計と数兆件の DNS クエリを使用します。

Decoy Dog やその他のマルウェア C2 ビーコンは、一度発見されると非常に疑わしく見えますが、検出するのは非常に困難です。その性質上、DNS トラフィックには非常に変動しやすく、外れ値、つまりめったに見られず、データ窃取と一致するドメイン名の構造を持つドメインの大部分が含まれています。ただし、規定の侵入テスト活動以外では、DNS の窃取とビーコン送信は非常にまれです。さらに、侵入テストの DNS 署名は、マルウェア C2 ビーコンとはまったく異なります。Decoy Dog は、大容量システムである Pupy RAT の亜種からの DNS C2 であることが判明しましたが、トラフィックがセキュリティ事業者によってネットワークに注入されたため、目立たないビーコンであるように見えました。

リゾルバへの Decoy Dog クエリはスキャナーによって開始されましたが、Decoy Dog ネームサーバーの異常な動作により検出されました。前回のレポートで明らかになったように、Decoy Dog ネームサーバーは繰り返しのクエリに応答しましたが、それは時々断続的なものでした。これは、Pupy やその他の暗号化された通信プロトコルとは矛盾しています。さらに、コントローラが正しく形式が整っている、整形式のクエリに応答することがわかりました。複合的な動作により、システムは断続的な低量のビーコンを検出しました。ネットワーク内でのこの種のスキャンとオープン DNS 転送動作は、企業にさらなるセキュリティリスクをもたらします。外部当事者がネットワーク内部から DNS クエリを作動できるように許可することで、攻撃者はネットワークに対して偵察を行うことができます。この脆弱性については、付録 H で詳しく説明しています。

結論

Decoy Dog は明らかに深刻な脅威です。少数の脅威アクターがこのツールキットを 1 年以上使用しており、唯一文書化された検出結果は DNS データの監視によるものでした。これは高度にターゲットを絞った運用で使用されて、そのコントローラが非常に限られた数のアクティブなクライアントと対話していることしか観察されていません。私たちは Decoy Dog について多くのことを知ることができましたが、その足場を築くために使用された脆弱性が特定され、軽減されるまでには、依然として深刻な脅威であり続けるでしょう。

Decoy Dog の最初の公開後、脅威アクターは被害者のシステムへの継続的なアクセスを確保するために様々な方法で対応しました。これらの対応は、コントローラの DNS 応答動作の変更、コントローラへのジオフェンシング制限の追加、クライアントを新しいコントローラに移動することなどです。これらの適応にも関わらず、Infoblox は脅威アクターを追跡し続け、Decoy Dog とそれが Pupy RAT とどのように異なるのかについてさらに詳しく学び続けています。

Decoy Dog を作成するために Pupy に加えられた変更はかなりのものであり、これは、洗練された脅威アクターであることを示しています。これらの変更には次のものが含まれます。

- Pupy は Python 2.7 で書かれ、Decoy Dog には Python 3.8 が必要であり、Windows との互換性やメモリ操作の改善など、数多くの改善が含まれている。
- Pupy は通信での語彙がとても限られ、Decoy Dog は、複数の新しい通信モジュールの追加により、その語彙を大幅に拡張している。
- Decoy Dog は以前の DNS クエリの再生に応答するが、Pupy は応答しない。
- Pupy はワイルドカード DNS リクエストに応答しないが、Decoy Dog は応答し、それにより、パッシブ DNS で見られる解決の数が実質的に 2 倍になる。実際、Decoy Dog は、クライアントとの有効な通信の構造と一致しない DNS リクエストに応答する。
- Decoy Dog は、任意の Java コードを JVM スレッドに挿入して実行する機能を追加し、被害者のデバイス上で永続性を維持するための多数の新しいメソッドを追加する。

これらの変更による高度化により、巧妙に作成されたクエリに Decoy Dog が応答するための選択がさらに興味深いものになります。この決定は一見間違いのように見えますが、その根拠はまだ知られていない可能性があります。現時点では、Decoy Dog のまた別の謎のひとつに過ぎません。

将来、Decoy Dog をめぐるこれらの謎がさらに調査されるにつれて、防御者は次のことに留意する必要があります。

- Pupy と Decoy Dog の両方の IP は暗号化されたデータであり、通信に使用される実際の IP を表すものではないこと。マルウェアに関連する実際の IP への接続はすべて偽であること。
- DNS 応答で返される IP には意味がないが、DNS クエリと応答自体には、追跡に使用できる意味のある情報が含まれていること。ただし、通信量が少ないため、検出した通信を追跡するには長いログ履歴が必要になる。

- ツールキットのワイルドカード応答とセキュリティ事業者による積極的なスキャンの組み合わせにより、何も侵害されていないように見える場合があること。
- 被害者のマシン上の Decoy Dog クライアントを検出できる YARA ルールが利用可能で、Decoy Dog を公開されているバージョンの Pupy と区別することができること。

Decoy Dog は、DNS 脅威検出アルゴリズムのみを使用して検出されました。現在までのところ、マルウェア自体の検出とその機能の全範囲を説明する公開情報はありません。長い間検出されずに運用されてきたという事実は、業界がマルウェアベースの検出に過度に依存している場合に生じる弱点を浮き彫りにしています。DNS を検出し、対応することが現時点では Decoy Dog から防御する唯一の方法であり、被害者の脆弱性と Decoy Dog 自体が十分に理解された後でも、最良の選択肢となる可能性があります。

指標

本レポートで説明したコントローラやサンプルに関連する Decoy Dog の指標は、以下にリストアップされており、Github 公開リポジトリで入手できます。[20]

ドメイングループ	特徴
ads-tm-glb[.]click	Decoy Dog C2 ドメイン
allowlisted[.]net	Decoy Dog C2 ドメイン
atlas-upd[.]com	Decoy Dog C2 ドメイン
cbox4[.]ignorelist[.]com	Decoy Dog C2 ドメイン
claudfront[.]net	Decoy Dog C2 ドメイン
hsdps[.]cc	Decoy Dog C2 ドメイン
j2update[.]cc	Decoy Dog C2 ドメイン
maxpatrol[.]net	Decoy Dog C2 ドメイン
nsdps[.]cc	Decoy Dog C2 ドメイン
rcmsf100[.]net	Decoy Dog C2 ドメイン
13[.]248[.]169[.]48	Decoy Dog C2 ネームサーバー IP
156[.]154[.]132[.]200	Decoy Dog C2 ネームサーバー IP
194[.]31[.]55[.]85	Decoy Dog C2 ネームサーバー IP
5[.]199[.]173[.]4	Decoy Dog C2 ネームサーバー IP
5[.]252[.]176[.]63	Decoy Dog C2 ネームサーバー IP
5[.]252[.]176[.]22	Decoy Dog C2 ネームサーバー IP
5[.]252[.]179[.]18	Decoy Dog C2 ネームサーバー IP

20 https://github.com/infobloxopen/threat-intelligence/tree/main/cta_indicators

67[.]220[.]81[.]190	Decoy Dog C2 ネームサーバー IP
69[.]65[.]50[.]194	Decoy Dog C2 ネームサーバー IP
69[.]65[.]50[.]223	Decoy Dog C2 ネームサーバー IP
70[.]39[.]97[.]253	Decoy Dog C2 ネームサーバー IP
83[.]166[.]240[.]52	Decoy Dog C2 ネームサーバー IP
4996180b2fa1045aab5d36f46983e91dadeebf d4f765d69fa50eba4edf310acf	Decoy Dog バイナリ SHA256
ab8e333ef9bc5c5a7d1ed4cab08335861e150 b0639d3d0ca4c30b7def5cdccde	Decoy Dog バイナリ SHA256
ad186df91282cf78394ef3bd60f04d859bcaccc bcdcbfb620cc73f19ec0cec64	Decoy Dog バイナリ SHA256
6c8f413111f1abfee788dad4ee7cca37e0c259 7cca66d155af958c535faf55cc	Decoy Dog バイナリ SHA256
0375f4b3fe011b35e6575133539441009d015 ebecbee78b578c3ed04e0f22568	Decoy Dog バイナリ SHA256
6c8f413111f1abfee788dad4ee7cca37e0c259 7cca66d155af958c535faf55cc	Decoy Dog バイナリ SHA256
t1fde0f101c9395f39ecd16430b41041a59107 c73c904087309fb8d0e8d87e0077129f3f	Decoy Dog Telfhash 署名[21]

21 https://github.com/trendmicro/telfhash

付録 A: クライアントコマンド処理

図 18 は、本レポートで説明されたクライアントの操作サイクルを示しています。クライアントは、スリープ状態、サーバーへのポーリング状態、コマンドへの応答状態の間を繰り返し遷移します。

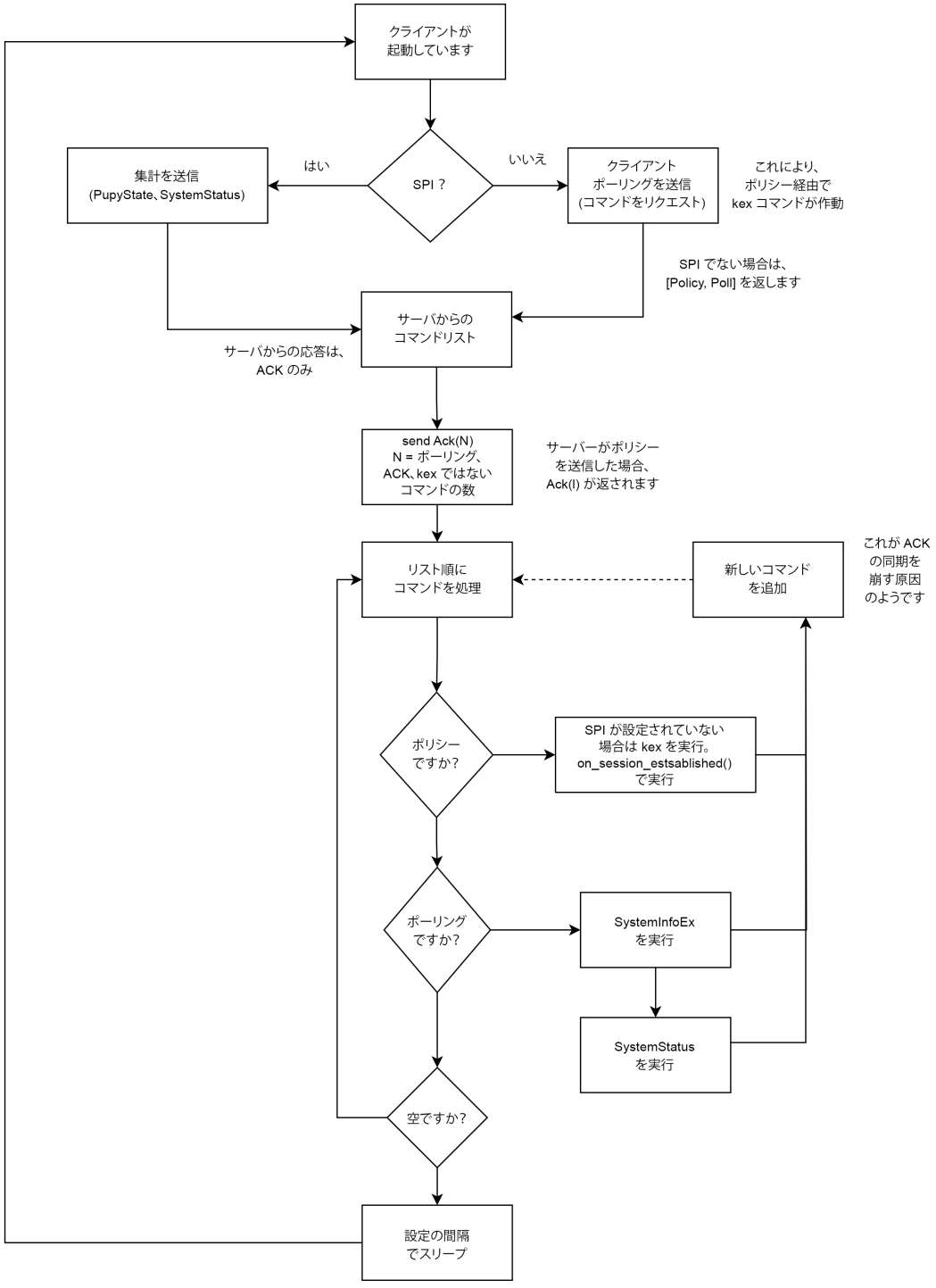

図 18. クライアントのワークフロー

付録 B: 通信ペイロードの構造

クライアントとサーバーの暗号化ペイロードの構造はまったく同じですが、その処理には違いがあります。特に、クライアントには、前述したようにデータペイロードとともにすべてのクエリに 13 バイトのクライアント情報が含まれます。

クライアントとサーバーは両方とも、受信者に送信する情報の種類に対してコマンドという用語を使用します。したがって、クライアントが起動時にサーバーに接続すると、それはクライアントコマンドとみなされます。コマンドは、クライアントまたはサーバーがデータに特定の処理を適用できるように登録されます。クライアントからこれが行われることはまれですが、1 つの通信に複数のコマンドが存在する可能性があります。

コード化と送信のために送信されるペイロードの形式は次のとおりです。

- 4 バイトのチェックサム
- 連結されたコマンドパッケージで、1 バイトのコマンド ID と可変のコマンド依存データ部分が含まれる。

ペイロードの全長は 52 バイトを超えることはできない。

付録 C: パッシブデータからのクライアントの再構築

前述したように、Pupy クエリには暗号化データと 2 つのコード化された値 (ノンス値と SPI 値) が含まれ、ある程度のセキュリティを提供し、サーバーがクライアント通信を命令できるようにします。SPI 値は、サーバー内で進行中のセッションを識別するために特に使用され、キー交換が成功した後のクエリに存在します。その結果、同じ SPI を含み、近い時間に発生するクエリは、同じクライアントからのものであることがほぼ確実です。一方、単一のクライアントには時間の経過とともに多くのセッションと多くの SPI 値が存在するため、SPI だけではクライアントを区別できません。代わりに、ノンス値を使ってクライアント通信を分離します。

クライアントが初期化されると、開始点として機能する 32 ビットのノンス値がランダムに生成されます。パケットごとに、このノンスは送信されるデータの長さだけ増加します。サーバーは、ノンスを副次的なセキュリティチェックとして使用し、クエリを受信するたびにナンスが確実に増加するようにしますが、その主な用途は、基礎となる通信を正しく復号化して解釈することです。下の図 19 に示すように、観察された一連の Pupy クエリからこれらのノンス値を復号し、一連の次のノンスを計算できます。

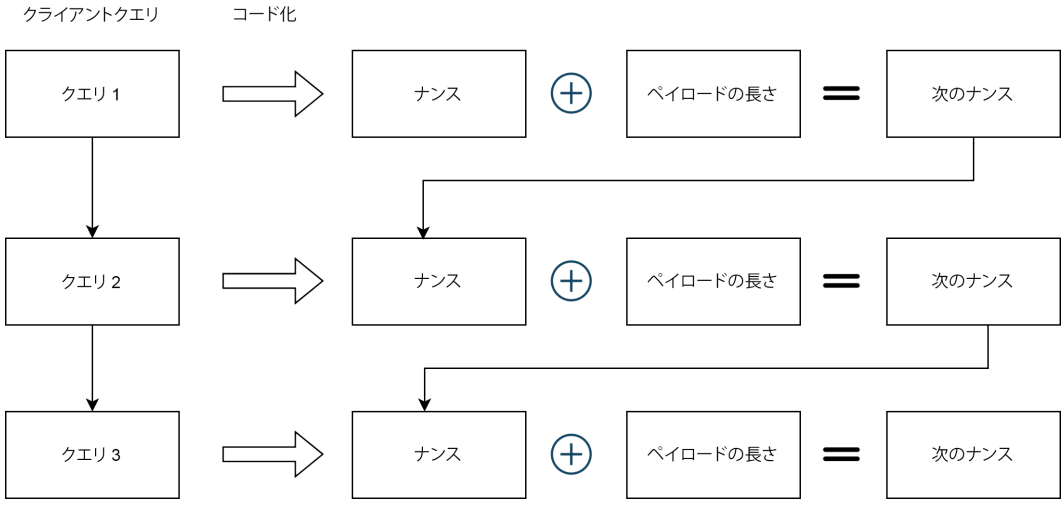

図 19. 一連の *Pupy* クエリ内の ノンス値の関係

結果として、1 つのクライアントからクエリを並べ替えることと、一連のクエリが 1 つのクライアントに属していることを確認できます。Pupy デプロイをパッシブデータの収集において、クエリが多くのクライアントから発信され、時間が重複する場合がありますが、これらの観察は、ナンスの構築により、かなり自信を持って個別のクライアント活動に分離できます。ペイロードの暗号化にはナンスが使用されるため、開発者は強力な乱数ジェネレータを使用してペイロードを作成しました。これにより、各クライアントが一意の開始のナンス値を生成するようになります。[22] ナンスは、クライアントが再起動するたびに再度生成されます。

暗号化の追加のセキュリティにより、集計された監視でクライアントを区別するメカニズムも提供されます。これを行うために、すべてのクエリについて、コード化されたノンスと次のノンス値の両方を計算します。次に、以下の図 20 に示すように、連続するノンス値を使用してクエリを連鎖させます。基礎となるデータは暗号化されたままですが、クライアントの数を推定し、その活動の長さを観察することができます。さらに、ペイロードの長さを使用し、クライアント間で時系列で比較することで、通信自体に関する情報を推測できます。

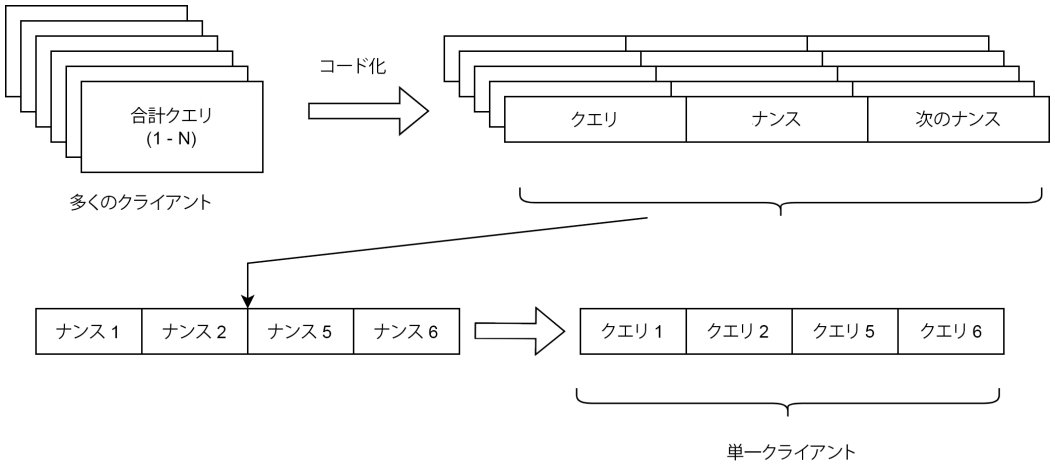

図 20. ノンス値を使用して、クエリのクライアントスレッドを観察の集合セットから分離

このタイプの窃取には、感染したクライアントの DNS リゾルバの変更とパケットの投下という 2 つの課題があります。デフォルトでは、Pupy はクライアントのデフォルトの DNS リゾルバを使用し、リゾルバの選択は脅威アクターの制御下にない可能性があります。クライアントがローミングする場合、ローカル環境に応じて異なる再帰リゾルバを使用する場合があります。エンタープライズネットワークでは、Infoblox などのベンダーの DNS インフラストラクチャを使用する場合があり、クライアントの設定に関係なく、DNS クエリがエンタープライズ再帰リゾルバを介して強制されます。[23] さらに、DNS が UDP を介して転送される場合、パケット損失は避けられません。その結果、パッシブ DNS だけですべてのクエリを観察する可能性は低く、復元されたナンスチェーンにギャップが生まれ、それはサイズにおいてかなりのギャップが生じる可能性があります。

ただし、ノンスがランダムに生成された値であるという事実を利用して、クライアントスレッドを再構築することはできます。開発者は、独立した Pupy クライアントがノンス値を共有する可能性が極めて低いことを保証する強力な乱数ジェネレータを使用しました。さらに、一度に送信できるデータは 52 バイトだけであり、ノンス値はペイロードごとに増加するため、別々に生成された 2 つのノンスチェーンが重複する可能性はほとんどありません。その結果、ノンス値を順序付けし、統計的に似ているものをグループ化することで、クライアントを分離できます。1 つのクライアントは一度に 1 つのノンスしか持たないため、任意の時点でアクティブなクライアン

22 まれに、2 つの異なるクライアントによって同じノンスが同時に生成される可能性があります。

23 Who is Answering My Queries：Understanding and Characterizing Interception of the DNS Resolution Path (DNS 解決パスの傍受についての理解と特徴づけ) Baojun Liu, et al. 2018 年 https://www.usenix.org/conference/usenixsecurity18/presentation/liu- baojun

トの数を推定できます。このレポートの本文で紹介するように、この手法は Decoy Dog クライアントのクエリチェーンを回復するのに非常に効果的であることがわかりました。

付録 D: ペイロードの署名

このセクションの表には、Pupy 通信で一般的に観察された特定のコマンドのペイロード長が含まれています。特に、すべての標準的なクライアントコマンドの暗号化されたペイロード長を提供します。サーバーのペイロードはクライアントのペイロードよりも柔軟です。最も一般的なものを以下に示します。

クライアントコマンド	ペイロードの長さ
クライアントのチェックイン (初回)	18
Ack	19
クライアントチェックイン (まれな変数)	22
システムステータス	24
オンラインステータス	27
クライアントのチェックイン (セッション中)	27
ポートテスト	35
システム情報の拡張	39
キー交換	47、48

表 2. クライアントコマンドとペイロードの長さ

サーバーコマンド	ペイロードの長さ
Ack	6
セッションが必要: ポリシー、ポーリング	42
セッションが未完了: ack、ポリシー	34
エラー: メッセージ、ポリシー、ポーリング	44
システム情報が必要: ポーリング	15
キー交換	62、63
終了	7

表 3. 共通のサーバーコマンドとペイロードの長さ

付録 E: エラー処理

Pupy には、サーバーで発生する可能性のある様々なエラーに対するカスタム処理が含まれています。ドメインが適切に復号されない、または再生されると、サーバーから NXDOMAIN 応答が返されます。以下のコードスニペットは、サーバークエリの処理を示しています。応答が返されない場合は、NXDOMAIN 応答を返します。

```
answers = self.process(qtype, qname.stripSuffix(self.domain).idna()[:-1])
klass = SUPPORTED_METHODS[qtype]


if answers:
    for answer in answers:
        reply.add_answer(RR(qname, qtype, rdata=klass(answer), ttl=600))


    if self.edns:
        reply.add_ar(EDNS0(udp_len=512))
else:
    reply.header.rcode = RCODE.NXDOMAIN
```

図 21. クライアントのクエリを処理する *Pupy* サーバーのソースコード

Decoy Dog では、サーバーから NXDOMAIN が返されるはずの多くのクライアントクエリが、代わりに応答 (通常は 15 個の IP アドレス) を返します。これは、コードの変更によるものと思われます。Decoy Dog は、内部的に、DnsCommandServerException を使って、考えられる様々なエラーに応答します。DnsCommandServerException は、発生したエラーの種類を指定し、新しいキー交換を実行してからシステム情報を送信するようにクライアントに指示する応答をクライアントに返します。このエラー処理のコードブロックを以下に示します。

```
except DnsCommandServerException as e:
    nonce = e.nonce
    version = e.version
    responses = [e.error, Policy(self.interval, self.kex), Poll()]
    emsg = 'Server Error: {} (v={})'.format(e, version)
    logger.debug(emsg)
    if node:
        node.warning = emsg
```

図 22. クライアントにエラーを返す *Pupy* サーバーのソースコード

Pupy サーバーとクライアント間の通常の通信では、既知のクライアントに有効なセッションが存在しない場合に、このタイプの例外が発生します。これは、クライアントペイロードが無効であるか、チェックサムが正しくない場合にも使用されます。それ以外の場合はすべて、結果は NXDOMAIN になります。

付録 F: バイナリサンプル分析

Pupy クライアントバイナリ

Pupy サーバーは、最初にセットアップされるときに、Pupy ライブラリファイルをコンパイルし、アーキテクチャごとに静的テンプレートファイルを作成します。これらのテンプレートファイルは圧縮され、高度に難読化され、すべての記号が削除されています。

その後、サーバー上で pupygen.py を使用してクライアントバイナリを手動で作成できます。このスクリプトは、特定の構成バイト (リモートホスト、トランスポートタイプ、デバッグフラグなど) を目的のアーキテクチャとファイルタイプに対応する静的テンプレートにマーシャリングして、C2 固有のバイナリを作成します。

Pupy クライアントバイナリは、様々な高度機能を提供し、Windows、macOS、Linux、Solaris、Android など、ほぼすべてのプラットフォームをターゲットにできます。特に、メモリ内に常駐し、サーバーとやり取りし、完全なリバースシェル機能を提供し、ファイルレスコピーを作成できます。バイナリが実行されると、バイナリは検出を回避し、プロセス強制終了手法に対する耐性を高めるために、メモリ内に自身のコピーを作成します。

Java インジェクション関数の例

Decoy Dog バイナリには、Java インジェクションに関連する多数の新しい関数が含まれています。これは、これらの関数の 1 つの例です。

```
undefined8 FUN_00105903(void)
{
  int iVar1;
  long lVar2;
  long lVar3;
  long lVar4;
  undefined8 uVar5;
  char *pcVar6;
  undefined local_20 [8];
  undefined8 local_18;

  local_18 = 0;
  if (DAT_005fbda0 == 0) {
     pcVar6 = "JVM was not loaded yet";
  }
  else {
       jvm_address = check_jvm_is_running(0);

       if (jvm_address == 0) {
       return 0;
       }
       classloader_address = find_classloader(lVar2);
       if (classloader_address == 0) {
       pcVar6 = "Preferred classloader was not found";
       }
       else {
       thread_class_address = find_jv_thread(lVar2);
       if (thread_class_address == 0) {
       pcVar6 = "Could not find Thread class";
       }
       else {
       iVar1 =
inject_in_thread(jvm_address,thread_class_address,"currentThread","()Ljava/lang/Thread;",&lo
cal_18);
       if (iVar1 == 0) {
       iVar1 = inject_in_class(jvm_address,local_18,"setContextClassLoader","(Ljava/lang/ClassLoader;)V",
                                          local_20,classloader_address);
       if (iVar1 == 0) {
          uVar5 = (*DAT_005fb748)(1);
          return uVar5;
       }
      pcVar6 = "Iteration failed";
      }
      else {
      pcVar6 = "Could not find current JVM Thread";
      }
  return 0;
```

図 *23. Decoy Dog* 関数が部分的に逆アセンブルされ、インジェクションのために現在実行中の *JVM* スレッドを見つけようとしました。

付録 G: DECOY DOG の YARA ルール

次の YARA ルールは、2023 年 7 月の時点で観察された Decoy Dog サンプルを検出するために使用できます。

```
/*
This rule only detects Decoy Dog. It was adapted from Florian Roth's Pupy Rule
original author : Florian Roth / @neo23x0
original link : https://github.com/Neo23x0/signature-base/blob/master/yara/gen_pupy_rat.yar
*/

/* Rule Set ----------------------------------------------------------------- */
import "elf"
import "pe"

rule DecoyDog_Backdoor {
   meta:
     description = "Detects Decoy Dog backdoor"
     license = "Detection Rule License 1.1 https://github.com/Neo23x0/signature-
base/blob/master/LICENSE"
     author = "Infoblox Inc."
     reference = "https://github.com/n1nj4sec/pupy-binaries"
     date = "2023-07-11"

   strings:
     $x1 = "reflectively inject a dll into a process." fullword ascii
     $x2 = "ld_preload_inject_dll(cmdline, dll_buffer, hook_exit) -> pid" fullword ascii
     $x3 = "LD_PRELOAD=%s HOOK_EXIT=%d CLEANUP=%d exec %s 1>/dev/null 2>/dev/null" fullword ascii
     $x4 = "reflective_inject_dll" fullword ascii
     $x5 = "ld_preload_inject_dll" fullword ascii
     $x6 = "get_pupy_config() -> string" fullword ascii
     $x7 = "[INJECT] inject_dll. OpenProcess failed." fullword ascii
     $x8 = "reflective_inject_dll" fullword ascii
     $x9 = "reflective_inject_dll(pid, dll_buffer, isRemoteProcess64bits)" fullword ascii
     $x10 = "linux_inject_main" fullword ascii
     $x11 = "jvm.PreferredClassLoader" fullword ascii
     $x12 = "jvm.JNIEnv capsule is invalid" fullword ascii

   condition:
     (3 of them and $x11 ) or (3 of them and $x12)
     or (uint16(0) == 0x5a4d and pe.imphash() == "84a69bce2ff6d9f866b7ae63bd70b163" and
$x11) or (elf.telfhash() ==
"t1fde0f101c9395f39ecd16430b41041a59107c73c904087309fb8d0e8d87e0077129f3f")
}
```

図 *24. Decoy Dog* サンプルを検出するための *YARA* ルール

付録 H: 明らかになったセキュリティの脆弱性

デバイスがインバウンド通信で DNS クエリを実行するように設定されている場合、デバイスは外部エンティティにその動作とリソースを部分的に制御させるようになります。[24] 特に、この構成により、偵察、オープンリゾルバ、サービス拒否攻撃への参加を可能にするための手段を脅威アクターに提供できます。DNS は複雑であるため、ベンダーとネットワークオペレータの両方がこれらのリスクを理解していない可能性があります。私たちが検出したクエリを送信したセキュリティ装置は新しい機能を搭載することを目的としていましたが、それらの機能で DNS を使用すると、ネットワークが偵察やその他の脅威にさらされる可能性があります。

外部組織に DNS クエリを提供するネットワーク内のデバイスは、オープンリゾルバとして知られています。場合によっては、デバイスは応答を返しても、様々な状況により外部 DNS クエリを完全には解決できないことがあります。いずれの場合も、このようなデバイスはネットワーク自体にリスクをもたらし、またネットワークの使用で分散型サービス拒否 (DDOS) を増幅する

24 https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA10g000000PLRaCAO、最終アクセス日 : 2023 年 6 月 11 日

リスクをもたらします。オープン DNS リゾルバのリスクは十分に文書化されており、これらのリスクのため、Infoblox を含む多くのサービス契約ではオープンリゾルバが禁止されています。

Decoy Dog クエリの場合、セキュリティ装置はオープンリゾルバではありませんでしたが、それでも外部当事者が DNS クエリを作動することを許可していました。このような構成は増幅攻撃に使用できませんが、脅威アクターが他の目的に使用する可能性があります。例えば、脅威アクターは下の図に示す、ネットワークに対して偵察を実行できます。脅威アクターはドメインを作成し、受信クエリをログに記録するように対応するネームサーバーを構成します。次に、攻撃者はスキャンメカニズムを使用して、ネットワークに接続するために調整されたドメイン名を送信します。オープンリゾルバー検索の場合、これらは DNS クエリである可能性があります。Decoy Dog の場合、HTTPS 接続でした。いずれの場合も、内部デバイスは DNS クエリを生成し、脅威アクターが制御するネームサーバーに送信します。その後、脅威アクターは、受信したクエリにドメイン名と元の IP アドレスを結び付けることができます。この種の攻撃は、試行ごとに限られた量の情報を取得しますが、後の攻撃に備えて内部ネットワークをマッピングできるよう十分に確立されたメカニズムです。

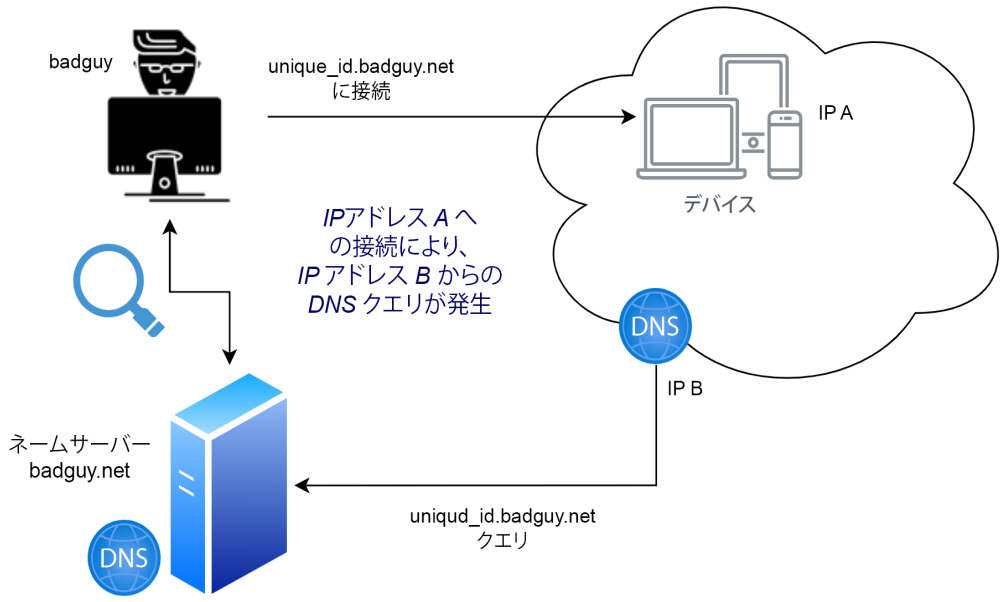

図 25. 脅威アクターは、ネームサーバーへの *DNS* クエリを作成する一意のドメイン名を作成することにより、ネットワーク上で偵察を実行。

付録 I: 調査データ A

調査の目的で、私たちは Pupy サーバーを確立し、再帰リゾルバを介してサーバーとクライアント間の通信をルーティングしました。分析のためにこれらの DNS クエリログを収集し、そのログを調査に利用できるようにしています。データには、数日間の様々な活動が含まれています。ほとんどの場合、リバースプロキシを確立することでクライアントを制御し、コマンドを SSL 経由で送信しました。これは Decoy Dog にも当てはまると思われます。ただし、サーバーからの DNS 応答を介して、利用可能なコマンドをすべて実行しました。さらに、複数のクライアントが同時にアクティブになり、多数のクライアントが再起動する期間があります。含まれる活動の範囲では、ここで説明した結果を再現できるはずです。

このデータは GitHub の公開リポジトリ infobloxopen: threat-intelligence で入手できます。[25] query-response ログには A レコードの結果が含まれ、以下のフィールドを含む csv ファイルにまとめられています。

- timestamp: Unix エポック 秒単位のクエリ時刻
- query: クライアントのクエリで送信された完全修飾ドメイン名
- response: サーバーから返された IP アドレスのセット
- client_payload_len: クエリ内のペイロードのバイト数 (ホスト情報を含む)
- server_payload_len: 応答内のペイロードのバイト数

このリポジトリには、このレポートの指標も含まれています。追加の指標は、リクエストに応じて防御側が TLP:RED 情報として利用できます。さらに、VirusTotal で入手可能なバイナリサンプルのリバースエンジニアリングから得られたデータも提供しています。これには以下が含まれます。

- 各サンプルの組み込み構成パラメーター
- 各サンプルに埋め込まれた暗号化キーとパスワード
  - » BIND_PAYLOADS_PASSWORD
  - » DCONFIG_PUBLIC_KEY (クライアント v4 のみ)
  - » DNSCNC_PUB_KEY_V2
  - » ECPV_RC4_PRIVATE_KEY
  - » ECPV_RC4_PUBLIC_KEY
  - » SCRAMBLESUIT_PASSWD
  - » SIMPLE_RSA_PUB_KEY
  - » SIMPLE_RSA_PRIV_KEY
  - » SSL_BIND_CERT
  - » SSL_BIND_KEY
  - » SSL_CA_CERT
  - » SSL_CLIENT_CERT
  - » SSL_CLIENT_KEY
- Decoy Dog バイナリを検出できる YARA ルールと TELF ハッシュ

25 https://github.com/infobloxopen/threat-intelligence

Infoblox株式会社
2〒107-0062 東京都港区南青山2-26-3
VORT外苑前I 3F

03-5772-7211
www.infoblox.com

RP-072023 v1