HITACHI

Hitachi Systems Security Journal

VOL.11

株式会社 日立システムズ

Hitachi Systems Security Journal VOL.11

TABLE OF CONTENTS

ワークショップを通じて潜在的な脅威を広く認知させ
攻撃・防御それぞれの視点からセキュリティ向上の議論を行なう

ヒューバート・リン インタビュー

今回は、ファイアウォールの制限を回避する新たな C2 通信実装の「Saucepot C2」を紹介するとともに、開発者のヒューバート・リン氏のインタビューをお届けする。「Saucepot C2」は、CODE BLUE 2025 で開催されたワークショップの教材に使われたソフトウェア（概念実証）だ。従来のポートノッキングを応用し、送信元ポートを制御して正規のトラフィックに攻撃者の通信を紛れ込ませることで、検知をきわめて困難にするステルス性を備えている。 ワークショップでは、レッドチームとブルーチーム双方の視点を持つ参加者が集まり、実戦的な演習が行われたというが、その開発の狙いや今後の脅威への対策について、開発者のリン氏に話を伺った。

取材・文＝吉澤 亨史／通訳＝エル・ケンタロウ／撮影＝松沢 雅彦／編集＝斉藤 健一

ポートノッキングの手法を応用して ファイアウォールをすり抜ける

2025 年 11 月 18 日、CODE BLUE 2025 においてヒューバート・リン氏によるワークショップ「Whispers Through the Firewall: Data Exfiltration and C2 with Port Knocking（ファイアウォールを潜り抜けるささやき：ポートノッキングを用いたデータ流出と C2 通信）」が行なわれた※1。

1 回 2 時間のワークショップとなっており、この日は同じ内容で 3 回実施された。

ワークショップに参加するには通信可能な 2 台の Linux マシンが必要で、これらはローカル環境の仮想マシン、または、AWS や Azure、GCP 上で構築することもできる。テスト済みディストリビューションは Debian 11、12、Ubuntu 22.04、24.04、CentOS Stream 9、Red Hat Enterprise Linux 9 となっていた。

ワークショップのアジェンダは、「① 従来のポートノッキング」、「② ポートノッキング 2.0」、「③ Saucepot C2」、「④ 5 つの演習」であった。ポートノッキングとは、クライアントがサーバーの閉じられたポートに対して特定の接続試行シーケンス（ノック）を送信することで、通信やアクセスの成立可否を制御するステルス性の高いネットワーク手法である（MITRE ATT&CK テクニック ID：T1205.001）。

ワークショップ前の会場の様子。参加者の多さから、追加の椅子を急きょ用意することになった（吉澤撮影）

正しいシーケンスを受信すると、サーバーは動的にポートを開く、あるいは特定のアクションをトリガーすることで通信を成立させる。例えば、80 番ポートで稼働している極秘の Web サーバーがあり、デフォルトではすべてのトラフィックを遮断するファイアウォールが設定されている状況を想定する。この Web サーバーにアクセスしたいクライアントは、サーバー側であらかじめ定義されたシーケンスにしたがって、4000、7000、4000 の順に接続を試行する。サーバー側はこれによって正しいノック、すなわち事前に合意された合言葉であると

●ヒューバート・リン（Hubert Lin）

Netskope 社シニア・スタッフ脅威研究エンジニア。台湾在住。セキュリティ脅威研究者として、脅威およびぜい弱性調査、ペネトレーションテストに従事している。マルウェア収集、ハニーポット運用、システム統合など多岐にわたる領域での経験を保有し、プロセス自動化やスクリプト言語による効率化にも精通している。
技術スキルは多様であり、Perl、シェルスクリプト、Java、C、アセンブリなどのプログラミング言語や、Linux、VMware、Docker などの環境に対応している。セキュリティ分野ではファイアウォール、Metasploit、tcpdump、Wireshark などのツール活用に精通し、ネットワーク設定およびデータベース管理にも経験を有している。

※ 1 **Whispers Through the Firewall: Data Exfiltration and C2 with Port Knocking**
https://archive.codeblue.jp/2025/program/contests-workshops/whispers-th-fw/

認識し、そのクライアントに対してのみ通信を許可するよう制御を変更する。

しかし、このような従来のポートノッキングは、任意の宛先ポートに対して通信が可能な環境を前提としている。実際のサイバー攻撃では、外部のC2 サーバーに対して 4444/TCP のような通常使用されないポートが利用されることがしばしばある。その対策として、組織は通信先ポートを 80、443 のような業務上必要なものに限定している場合がある。

そこで本ワークショップで扱われる手法では、このポートノッキング手法を応用して、シーケンスに用いるポートを宛先ではなく送信元ポートとして利用する。通常、外部通信時の送信元ポートは空きポートから自動的に割り当てられるが、これを 54000、57000、54000 のように意図的に制御することで、送信元ポート番号そのものに意味を持たせたり、エンコードした情報を仕込むことができる。これによって、外向きの通信が特定ポートに制限された環境においても通信を成立させ、かつデータ転送を隠蔽することが可能となる。「Saucepot C2」は、この送信元ポートを用いた通信手法を、コマンド・アンド・コントロール用途として実装した概念実証である※2。

Saucepot C2 は、Scapy と PycURL で動作する C2 サーバーとクライアント。リン氏は例として、サービス保護、データ漏えい、C2（コマンド & コントロール）の 3 つを紹介した。続いて演習となった。

5 つの演習では、「❶ 従来のポートノッキング」、「❷ エフェメラルポートのチェック」、「❸ データ流出」、「❹ 指揮統制作戦」、「❺ 異常の観察」を行なった。なお、「❷ エフェメラルポートのチェック」は NAT 処理後もクライアントの送信元ポートが保持される適切な環境にあるかどうかを確認するもので、エフェメラルポートの悪用が機能するための重要な要件となる。以上で約 2 時間となり、残りの時間は質問にあてられた。

今回のワークショップは、リン氏が所属する Netskope グローバルによるセッション登録を経て、ノミネートされ選抜されたという。そのため Netskope の日本法人がサポートを行なった。同社はハンズオンを客先を中心に実施しているが、ハンズオンでは時間が短いという声もあることからワークショップの開催にも力を入れている。

物静かで穏やかな印象のヒューバート・リン氏だが、技術の詳細を語るときには、相手が理解できるまで何度でも説明する情熱も持っている

Netskope のソリューションエンジニアである藤宮淳氏によると、参加者はベンダーやコンサルティングのセキュリティ担当者、そしてエンドユーザーが多かったという。特に、セキュリティコンサルティングやレッドチーム活動をされている方など、セキュリティに関するグランドデザイン寄りの参加者が多かった印象だったという。参加者からは「有用な内容だった」という声や「こんな攻撃手法があるのかと新鮮だった」などの意見がある一方で、セッションにさらに多くの時間があればよかったという声も聞かれた。

次の段落からはインタビューをお届けする。

攻撃者よりも先に新たな手法を考えてユーザーに周知すれば防御につながる

吉澤（以下、Y）：本日はワークショップに参加させていただき、ありがとうございました。普段からこうした演習をされているのですか。

ヒューバート・リン氏（以下、H）：私は実践的な演習が好きなんです。それでさまざまなツールや

※ **2 Saucepot C2**　https://github.com/netskopeoss/saucepot

アイデアを試して、どのようなインスピレーションを得られるか探っています。それには自分の時間の多くを費やしています。

Y こうしたワークショップに参加する方は、企業の方が多いのでしょうか、それとも学生が多いのでしょうか。

H 参加者は企業の方が多いと思います。

Y 企業からの参加者としては、SOC や CSIRT の担当者、あるいはレッドチームの専門家が多いのでしょうか？

H ブルーチームおよびレッドチームのメンバーが多くを占めています。参加者の中には、「ブルーチームに所属していますが、レッドチームの手法を理解することで、自らのフォレンジック能力を向上させたい」とおっしゃる方もいます。

Y 今回は、従来のポートノッキング手法と Saucepot C2 の両方についてご解説いただきました。この Saucepot C2 に使われている手法は、現在、一般的に認識されているのでしょうか。

H いまだ広く使われている手法ではありません。このような手法が C2 インフラで活用される可能性を紹介したいと考えました。どのような通信が誤用または悪用され、C2 フレームワークを構築する可能性があるのかについて、広く認識していただきたいと考えています。今後、このような手法が増加する可能性も考えられるためです。

Y なるほど。では、その手法はいつ頃から存在していたのですか？

H 手法そのものは十数年前から存在しています。攻撃者はその技術を悪用して、さらにレベルを押し上げていると考えています。

Y Saucepot C2 のような攻撃手法は、監視および検知が困難であると考えられますが、この点についていかがお考えでしょうか？

H まず第一に、アプリケーション層で通常のトラフィックのように見えるため、監視が難しいのです。そして、監視することができたとしても、実際に何か問題を検出することはさらに大きな課題となるでしょう。

Y Saucepot C2 を使用した攻撃は、すでに確認されているのでしょうか？

H このような手法が実際に使用されている事例は、まだ確認したことがありません。

Y 仮に、実際に使用されたとすれば、監視および検知が困難になりますからね。

H はい。例えばログを見たとしても、特定の URL に対して複数回アクセスしていることは分かりますが、通信内容を詳細に検査しても疑わしい痕跡は認められません。また、この手法を事前に把握していなければ、攻撃の明確な異常性を特定することは困難だと思います。

Y 何か対策方法はあるのでしょうか？

H 回避策としては、すべての Web リクエストをプロキシ経由にすることが考えられます。プロキシを経由することでソースポートが再割り当てされるため、攻撃は成立しなくなります。また、私はルーター開発チームに、このような脅威があることについて警鐘を鳴らしたいと思っています。例えば、ソースポートをランダム化したり、ルーターの NAT レベルで技術的な対策を実装したりすることが考えられるのではないかと思います。

Y 実際にワークショップに参加された方々からの反応はいかがでしたか？

H 複数の有意義な質問をいただいています。技術的な質問がほとんどですが、時には具体的な手法について詳しく質問される方もいらっしゃいます。実際にソースコードを見せながら「ここが今回のハンズオンの該当部分です」と説明すると、参加者の関心がより高まるようです。特に、今回の CODE BLUE は、過去に実施したワークショップと比べて、参加者の積極性が本当に高かったと感じています。

Y 今回のワークショップは、例えば必要な環境などの事前告知がなかったように思うのですが、参加者の方々はちゃんと Linux がインストールされたノート PC を準備されていました。このことから、参加者はポートノッキングなどについて、日頃からある程度の理解と実践経験を持っているのではないかと思います。

H それは私自身も期待していたところです。講師である私の操作を視聴するだけでなく、参加者の方々が自ら実際に操作を試みることを重視していました。

Y ワークショップの成果は十分だったということですね。

H 非常に良い成果が得られたと思っています。特に、参加者の数よりも参加者との対話の質を大事にしており、その点で成功を感じ、満足しています。

普段から多くのツールに触れてアイデアを試し、そこから得た知見を実践的な演習に生かしているという。今回のワークショップもブラッシュアップして、さらに改善していきたいと語る

攻撃手法がもたらす潜在的な脅威に着目して研究を進める

Y 普段は台湾においてもこのようなワークショップを実施されているのでしょうか。

H いいえ、していません。普段は脅威研究者として、さまざまなツールや新しい手法にどのような脅威があるのかを研究しています。同時に、その過程で防御側が対応できないセキュリティギャップがないかを検証することにも、かなりの時間を費やしています。一般的に、さまざまな脅威や攻撃キャンペーンを分析する研究者は多いですが、私はそういった従来的な研究アプローチはしていません。

Y 今回のテーマでワークショップを実施することにした主な動機は何でしょうか？

H プロトコルはいろいろと存在しますが、その中には攻撃者が C2 通信に悪用する可能性のあるものもあります。こうした実態をレッドチームのメンバーに示したいという考えから、今回ワークショップという形で発表させていただきました。その目的は、セキュリティ評価の観点から演習をするときに、従来とは違うアプローチが取れることを示すことでした。一方、ブルーチームにとっては、このようなプロトコルの悪用の可能性を新たに知り、実際の攻撃を受ける前に対策を講じることができるようになります。

Y CODE BLUE では講演だけでなく、ワークショップでも激しい採択競争があると聞きます。あなたのワークショップが選定された理由についてどのようにお考えですか。

H 組織が必ずしも対策を講じていない脅威ベクターに関する実践的なセッションを提供していたこと、そしてより広範なコンテンツであったことが、人々に「参加したい」と思わせたのだと思います。

Y CODE BLUE について、どのような印象をお持ちになりましたか。

H 非常に大規模なカンファレンスだと思いますし、興味深い講演も多くありました。個人的には、ビレッジ※3 に特に興味を持ちました。さまざまな研究成果やデモ、ハンズオンが展開されていて、時間があればそれらを見て回りたかったですね。

Y ありがとうございます。引き続き、ヒューバートさんご自身のキャリアについて伺います。LinkedIn で略歴を拝見すると、ハニーポット、ぜい弱性発見、IPS シグネチャー、ペネトレーションテスト、レッドチーム活動などを研究しているとのことでしたが、そうした方向に進んだきっかけは何だったのでしょうか。

H 私はセキュリティ業界で 20 年間働いています。元々はディフェンス側の立場で研究していましたが、やり尽くしたと感じ、さらに探求したいとも思いました。そのときにレッドチームへの興味が高まってきたので、現在の会社に入る前にレッドチームに関連するさまざまな研究を始めました。その研究は今も続いています。

Y 今後の目標などあれば教えてください。

H 今、AI が非常に注目されています。今後さらに AI は進化していくと思うので、自分の研究にどう取り込んでいくかをじっくり考えていきたいですね。

Y ありがとうございました。

※ **3** CODE BLUE 2025 では、海外のコミュニティの連携して「自動車」「医療機器」「海運」「航空・宇宙」「産業用制御システム」のビレッジ（ワークショップ）が開催された。

社会のさまざまな動向を把握し、リスクの変化に対応したセキュリティ体制を構築

Hitachi Systems CSI（Cyber Security Intelligence）Watch 2026.03

文＝日立システムズ

AI エージェント時代に求められる可観測性（Observability）

【概要】 AI エージェントは従来の対話型 AI と異なり、与えられた目的に基づいて自律的に複数のシステムを横断して操作を実行するソフトウェアである。その挙動は実行時に動的に組み立てられ、同じ目的でも常に同一の操作が実行されるとは限らない。AI エージェントを説明可能に運用するには、システムの内部状態を読み解く可観測性（Observability）の確保が不可欠である。AI が主体的に判断する時代において、可観測性を設計段階から組み込むことが、AI エージェント時代のシステムの前提条件となる。

【内容】 AI エージェントとは、単に人間の指示に応答する対話型 AI とは異なる存在だ。与えられた目的を達成するために自律的にタスクを生成し、複数のシステムを横断して操作を実行するソフトウェアである。従来の自動化が事前に定義された操作を実行するのに対して、AI エージェントは状況に応じて行動計画を更新し、情報を検索・判断して次のアクションを選択する。そのため、同じ目的が与えられても、常に同一の操作が実行されるとは限らない。AI エージェントは状況に応じてどのような行動をとるかを実行時に判断する点に特徴がある。したがって、従来の管理手法ではその挙動を十分に理解することが難しい。

この特性を理解するうえで重要な概念が、可観測性（Observability）である。これはシステムが出力するテレメトリ情報から、内部状態をどれだけ正確に把握できるかを測るものだ。監視があらかじめ決められた閾値に基づいて異常を検知するのに対し、可観測性は想定外の事象が起きた際にその原因を専門知識に依存せず理解することを目的とする。すなわち、可観測性は何が起きたかを検知する仕組みではなく、なぜそれが起きたかを理解するための枠組みである。テレメトリ情報は、ログ、メトリクス、トレースの 3 つで構成される。ログは操作やイベントを記録するテキスト情報であり、AI エージェントではプロンプト、推論内容、ツール呼び出し結果を時系列で把握するために使用される。メトリクスは CPU 使用率やエラー件数などの数値データを継続的に収集し、システムの状態変化検知に利用できる。トレースは 1 つの処理がシステム内をどの順序で通って実行されたかを追う情報であり、複数のシステムをまたぐ操作の因果関係を理解するために必要だ。AI エージェントを可観測にするためには、これらの情報を文脈として読み解くことが必要である。重要なのは、可観測性は予期しない振る舞いを抑え込むための仕組みではない点である。むしろ、想定外の行動が発生することを前提に、それがなぜ起きたのかを説明できる状態を維持することが目的である。重要な視点が、AI エージェントを人間と同じ 1 つの操作主体として扱うという考え方である。どのエージェントがどの目的のもとに、どの権限を利用し、どの資産に対してどのような操作を行なったのかを時間軸に沿って追える状態を整備することで、AI エージェントの行動は説明可能となり、人間がリスクと正当性を評価できる基盤が整う。

今後、組織が AI エージェントを安全に活用するためには、可観測性を単なる運用上の工夫ではなく、設計要件として位置づける姿勢が求められる。ここでいう可観測性とは、AI エージェントの判断や行動を後から説明できる状態を支える基盤である。例えば、推論プロセスと操作ログを関連付ける仕組みや、権限管理とテレメトリ情報を結びつける設計、AI エージェントの行動を俯瞰できる可視化基盤の整備などである。これらは AI エージェントの安全性を高めるだけでなく、将来的な監査、説明責任、ガバナンス要件に対応するうえで不可欠の要素となる。AI エージェントが自律的に行動し、実際の環境へ影響を与える時代において、その振る舞いを説明できることは付加的な要件ではない。この可観測性を設計段階から確保することこそが、AI エージェント時代のシステムにおける前提条件である。

セキュリティツールを実践的に紹介する連載企画

Let's try プロキシサーバーログ調査

3. 攻撃兆候検出編

文＝日立システムズ

1. はじめに

本稿は、各種セキュリティツールなどを実践的に紹介する連載企画です。前々号より「プロキシサーバーログ調査」と題して、プロキシログを収集・分析する方法を紹介します。OSS である Squid を用いたプロキシサーバー構築、ログの出力設定から、インシデントの初期対応で求められるアクセスログの分析のハンズオンまでを順を追って解説します。特に仮想環境（VirtualBox）における実行例も交え、手を動かしながら理解できる構成とします。

1. **環境構築編**
2. **ログ分析編**
3. **攻撃兆候検出編**

「プロキシサーバーログ調査 3. 攻撃兆候検出編」では、grep 等の Linux 標準コマンドを用いたログ分析のハンズオンを体験していただき、プロキシログから攻撃の兆候を検出する方法を解説します。また、報告用の成果物作成を意識し、表計算ソフトを用いたログ分析についても解説します。

なお、本稿の安全性には留意していますが、安全を保証するものではありません。OA 端末（社内ネットワーク接続機器）で実施するのではなく、分離された回線内および機器を利用する事を推奨いたします。また、本稿で構築する環境はプロキシサーバーログ調査を目的としており、実際のプロキシサーバー運用環境の要件を満たすものではありません。

2. 準備

本稿では、攻撃痕跡を含むプロキシログ (以下、分析対象ログ) を用いて演習を行ないます。しかし実際に不正アクセスを行なってログを作成することは難しいため、本誌では生成 AI を用いて分析対象ログの生成を行ないます。

まずは、API などを用いずに生成 AI へアクセスを行なうために、GUI の CentOS 仮想マシンを作成します。

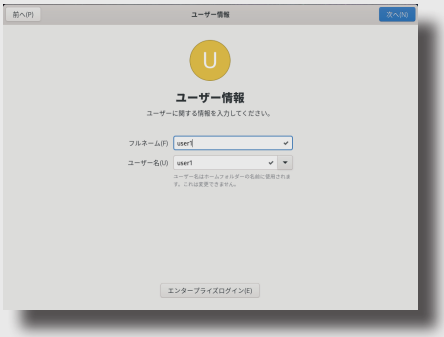
2.1 GUI 環境の準備

GUI の CentOS 仮想マシンを未作成の方は、Vol.50「Let's try HDD 保全！ 1. 準備編」※1 を参考に、下記の表に示す設定で作成してください。ISO イメージはダウンロード済みのものを使用可能です。

表 仮想マシンの変更点

変更点	変更後の値
仮想マシンの名前	GUI-CentOS
ディスクサイズ	15 GB
ソフトウェアの選択：ベース環境	サーバー：GUI 使用（Server with GUI）

インストール完了後に再起動を行ない、「セットアップ開始」をクリックしてセットアップを進めます。基本的にはデフォルト設定のまま「次へ」または「スキップ」をクリックして進めてください。ユーザー情報画面では、フルネーム欄に任意の文字列を入力し、「次へ」をクリックします。本誌では「user1」を入力しています。



パスワード画面で任意のパスワードを入力し、「次へ」をクリックします。ここで設定したパスワードは sudo 実行時などで必要になるため、メモしておいてください。

セットアップ完了画面で「CentOS Stream を使い始める」をクリックし、デスクトップ画面が表示されればセットアップは完了です。ツアーが表示された場合は「必要ありません」をクリックしてください。なお、この時点でスナップショットを作成しておくことをお勧めします。

※ 1 https://www.hitachi-systems.com/-/media/report/specialist/hj/download/2023_hj50.pdf

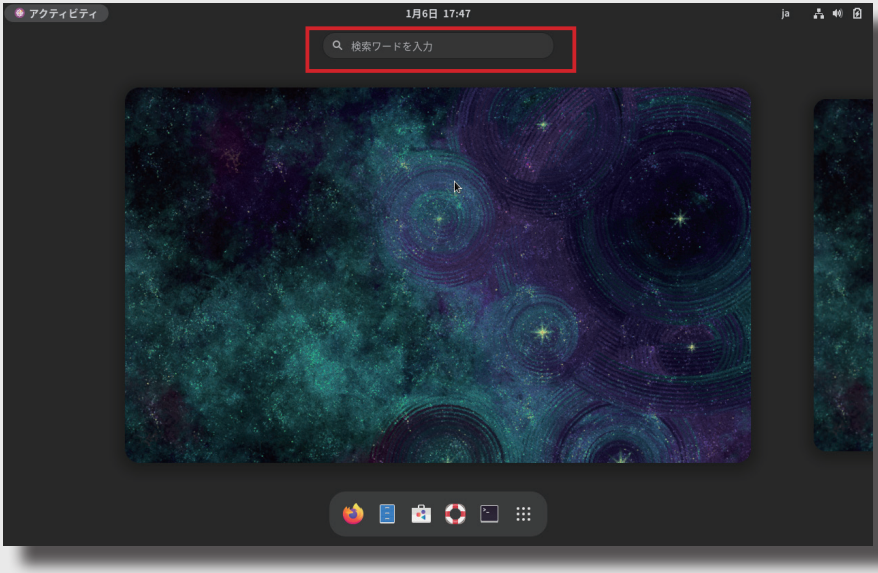
2.2 ログ生成の準備

ここでは、生成 AI を用いて分析対象ログの生成を行ないます。
ただし、同じ文章（以下、プロンプト）を入力しても、モデルのバージョンや乱数の違いによって生成されるログは異なります。そのため、以降で示す実行結果と全く同じ結果が出力されない可能性がある点をご了承ください。

ログ生成の準備を以下の手順で解説します。なお、CentOS の Firefox から本誌 PDF ファイルにアクセス可能な場合は、「6. 付録」のログをコピーし、「3. ログ保存」の手順において、ペーストすることで、同様の出力結果が得られます。この場合、ログのコピー・ペースト以外の作業は不要となります。

2.2.1 プロンプト作成

Web サイト上で直接プロンプトを入力すると誤送信が発生する可能性があるため、テキストエディターでプロンプトを作成します。
CentOS で「text」と入力して Enter キーを押すと、テキストエディターが起動します。画面が表示されない場合は、Super（Windows）キーを入力して表示させてください。



起動したテキストエディターに、次ページのプロンプトを入力してください。
Super（Windows）キー +Space で日本語入力に切り替えられます。

指示
プロキシログを 60 行考案し、1 分に 1 ログ程度の間隔で 1-30 行目、31-60 行目で分け、そのうちの 1-30 行目 (05:00-05:30) をまず出力して。

条件
- 説明なしで出力はログのコードブロックのみ
- ログ形式は Apache の Combined
- Referer,User-Agent はどちらも "-" でいい
- 60 行全体で日本時間 19/Dec/2025:05:00 から 06:00 の間のログ
- 192.168.10.102、192.168.10.103 が送信元 IP のログのリクエスト URL は、C2 サーバー example.net が多数、ファイル共有サービス upload.example.net が少数、example.net, その他のドメインが少数
- 192.168.10.102、192.168.10.103 以外が送信元 IP のログも混ぜるが、example.net, upload.example.net は含まない正常なアクセス

入力完了後、Ctrl+A、Ctrl+C の順で操作し、プロンプトをクリップボードにコピーします。

2.2.2 ログの生成（その 1）
Super（Windows）キーを入力して「firefox」と入力後、Enter キーを押すと Firefox が起動します。Firefox で任意の生成 AI サイトにアクセスし、プロンプトを送信します。

出力されたログのリクエスト URL に「example.net」と「upload.example.net」の文字列が含まれていることを確認してください。含まれていない場合は、サイトを開き直してプロンプトを再送信してください。その後、出力ログ右上の「コードをコピーする」をクリックしてコピーします。

2.2.3 ログの保存
Super（Windows）キーを入力して「terminal」と入力後、Enter キーを押すと Terminal が起動します。
次のコマンドを実行して、分析対象ログファイル「access.log」を作成します。

```
$ cat >> access.log
```

下記のように入力待ち状態となります。Ctrl+Shift+V でログをペーストした後、Ctrl+D で cat コマンドを終了してください。

```
[user1@localhost ~]$ cat >> access.log
```

次のコマンドを実行し、access.log の中身を確認します。

```
$ cat access.log
```

ペーストしたログが出力されれば、access.log への追記は正常に完了しています。

2.2.4 ログの生成（その 2）

Alt+Tab を何度か入力して Firefox をアクティブにし、生成 AI に以下の 31 ～ 60 行目用のプロンプトを送信します。出力トークン上限の関係で、複数に分けて出力しています。

> 31-60 行目 (05:30-06:00) を出力して

その後出力ログをコピーし、前項の「ログの保存」を再び行ない、access.log に追記します。
以上で、ログ準備は完了です。

3. grep コマンドによるログ分析

grep は、UNIX 系 OS で標準提供される文字列検索コマンドです。オプションや正規表現を活用することで、膨大な量のログから有用な情報を効率的に抽出できます。
本稿では、プロキシログに対して grep を適用し、簡易的なシナリオに沿って被害範囲特定などのログ分析を行ないます。

3.1 検知情報および調査目的

ある端末でマルウェア検知アラートが上がったという想定で、以下の検知情報とプロキシログをもとに調査を行ないます。

[検知情報]

- **検知端末 IP アドレス :192.168.10.102**
- **検知時刻（定期フルスキャン時刻）:2025/12/19 6:00**

プロキシログを分析することで、ネットワーク内の複数の端末からのアクセスを網羅的に把握します。本分析では、被害範囲の特定を主な目的としてプロキシログを分析していきます。

3.2 不審通信先の抽出

端末上で実行されたマルウェアは C2 サーバー（攻撃者が感染端末を制御するためのサーバー）へ通信を行なう可能性があります。そこで、マルウェアを検知した端末 192.168.10.102 の通信先 URL を抽出し、出現回数の上位からリストアップすることで C2 サーバーの疑いがあるドメイ

ンを洗い出します。
検知時刻より前の段階で感染していた可能性が高いため、検知直前の 1 時間に絞って分析を行ないます。

Terminal で次のコマンドを実行します。

```
$ cat access.log | grep '19/Dec/2025:05:' | grep '^192¥.168¥.10¥.102' | awk '{print $7}' | sort | uniq -c | sort -nr
```

コマンドの説明は以下のとおりです。

|（パイプ）：
パイプの前の出力をパイプの後のコマンドの入力に用いることができます。
cat access.log：
cat でログファイルを出力します。
grep '19/Dec/2025:05:'：
05:00-05:59 のログを出力します。
grep '^192¥.168¥.10¥.102'：
送信元 IP が 192.168.10.102 のログを出力します。先頭の ^(ハット) は行頭を意味する正規表現であり、これにより各ログ行頭の送信元 IP アドレスのみを検索できます。
awk '{print $7}'：
各ログ行の 7 番目のフィールド (空白区切りでのまとまり) を出力することにより、リクエスト URL を取り出します。出力するフィールドの数字は、ログの形式により異なります。
sort | uniq -c：
リクエスト URL の重複をカウントし表示します。その際、uniq は並んだ行でしか重複をカウントできないため、先に sort で昇順に並べ替えます。
sort -nr：
-nr オプションでカウント結果を数値として降順に並べ替えます。

本稿のプロキシログでは、下記のような出力結果となります。

```
[user1@locathost ~]$ cat access, log | grep '19/Dec/2025:05:' | grep '^192.168.10.102' | awk '{print $7}' | sort | uniq -c | sort -nr
      7 http://example.net/api/ping
      4 http://example.net/api/update
      2 http://example.net/assets/module.bin
      2 http://example.net/api/commands
      1 https://upload.example.net/file/UiopAs
      1 https://upload.example.net/file/QwErTy
      1 https://upload.example.net/file/LkJhGf
      1 https://upload.example.net/file/AbCdEf
      1 http://example.net/assets/config.json
```

出力結果から、検知端末からのアクセス回数が多いドメインのリストが得られました。次に、出現回数上位のドメインから AbuseIPDB（https://www.abuseipdb.com/）などのサイトを用いて C2 サーバーの可能性を調査します。
本稿では不審通信先調査のハンズオンは行ないませんが、実際の調査では、各サイトで判定された悪意スコアや報告件数などの情報をもとに通信先が C2 サーバーである可能性を推測します。

今回は example.net が C2 サーバーのドメインであったと仮定し、分析を進めます。

3.3 他の感染端末を抽出

次に、被害範囲の特定のために、C2 サーバーのドメインをもとに grep コマンドを実行し、C2 サーバーへ通信を行なっている他の端末の有無を調査します。
次のコマンドを実行します。

```
$ cat access.log | grep ‘example.net’ | awk ‘{print $1}’ | sort | uniq –c
```

本稿のプロキシログでは、下記のような出力結果になります。

```
[user1@localhost ~]$ cat access.log | grep 'example.net' | awk '{print $1}' | sort | uniq -c
     16 192.168.10.102
     18 192.168.10.103
      1 198.51.100.77
```

検知端末である 192.168.10.102 以外に、192.168.10.103 と 198.51.100.77 が C2 サーバーと思われるドメインへアクセスしており、マルウェア感染が疑われます。
アンチウイルスソフトの定義ファイルが古い場合やアンチウイルスソフトが無効化されている場合には、マルウェアが削除されていない可能性があります。さらなる感染拡大を防ぐためにも、これらの端末をネットワークから隔離することを検討する必要があります。

3.4 不審通信開始時刻の抽出

次に、被害範囲の特定のために、C2 サーバーのドメインをもとに grep コマンドを実行し、C2 サーバへ通信を行なっている他の端末の有無を調査します。
次のコマンドを実行します。

```
$ cat access.log | grep ‘example.net’ | awk ‘!seen[$1]++’
```

1 つの検索条件で最古、最新のログを確認する際は head コマンドや tail コマンドを用いることが多いですが、今回は「awk '!seen[$1]++'」を用いることで、送信元 IP アドレス（1 番目のフィールド）ごとの最古のログを一括で確認します。このコマンドは、指定したフィールド番号（$1）をキーとして、そのキーが初めて現れた行のみを出力しています。
また、先頭の cat を tac に変更して逆順に出力することで、最新のログを確認することも可能です。
本稿のプロキシログでは、以下のような出力結果となります。

```
[user1@localhost ~]$ cat access.log | grep 'example.net' | awk '!seen[$1]++'
192.168.10.102 - - [19/Dec/2025:05:00:03 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:01:07 +0900] "GET http://example.net/assets/config.json HTTP/1.1" 200 842 "-
198.51.100.77 - - [19/Dec/2025:05:10:00 +0900] "GET http://blog.example.net/archive HTTP/1.1" 200 2981 "-" "-"
```

2025/05/00:03 に 192.168.10.102 の端末から C2 サーバーと思われるドメインへの初めての通信が記録されていることから、該当時刻付近でのマルウェア実行が疑われます。この情報は、他のログ分析などの今後の調査における重要な手掛かりとなります。
ただし、この出力結果はあくまで調査対象のプロキシログの範囲であるため、ログが残っていない以前の時刻でも通信が行なわれた可能性があることに注意が必要です。

3.5 情報窃取の確認

攻撃者は窃取した情報を正規のクラウドストレージサービスにアップロードして持ち出す場合があります。そのため、調査の際は過去に攻撃に悪用されたことがあるサービスのドメインでの抽出が有効です。

本稿では、クラウドストレージサービスである upload.example.net と filetransfer.example.net への通信記録の有無を確認し、通信があった場合には通信を行なった端末を特定します。

次のコマンドを実行してください。

```
$ cat access.log | grep –i –e ‘upload.example.net’ –e ‘filetransfer.example.net’
```

本稿のプロキシログでは、下記のような出力結果になります。

```
[user1@localhost ~]$ cat access.log | grep –i –e ‘upload’ –e ‘filetransfer’
192.168.10.102 – – [19/Dec/2025:05:06:08 +0900] "GET https://upload.example.net/file/AbCdEf HTTP/1.1" 200 4096 "–" "–"
192.168.10.103 – – [19/Dec/2025:05:14:18 +0900] "GET https://upload.example.net/file/XyZ123 HTTP/1.1" 200 6144 "–" "–"
192.168.10.102 – – [19/Dec/2025:05:27:05 +0900] "GET https://upload.example.net/file/QwErTy HTTP/1.1" 200 4096 "–" "–"
192.168.10.102 – – [19/Dec/2025:05:36:07 +0900] "GET https://upload.example.net/file/UiOpAs HTTP/1.1" 200 6144 "–" "–"
192.168.10.103 – – [19/Dec/2025:05:44:25 +0900] "GET https://upload.example.net/file/ZxCvBn HTTP/1.1" 200 4096 "–" "–"
192.168.10.102 – – [19/Dec/2025:05:54:09 +0900] "GET https://upload.example.net/file/LkJhGf HTTP/1.1" 200 6144 "–" "–"
```

出力結果から、filetransfer.example.net へ通信を行なった端末は見つかりませんでしたが、端末 192.168.10.102 と端末 192.168.10.103 において upload.example.net への通信が行なわれたことが分かります。ただし、クラウドストレージサービスなどへの通信記録が確認できたとしても、情報窃取が行なわれたと断言はできません。パケットデータにおける通信量や MFT（Master File Table）※2 におけるファイルアクセスの記録などを確認し、情報窃取の可能性を調査する必要があります。さらに、そういった調査の上でも、具体的に窃取されたファイルなどを特定することは困難です。

本稿では通信先の一例として upload.example.net と filetransfer.example.net に絞って調査しましたが、実際にはその他の不審なドメインでも調査し、被害端末や時間帯において不審なアクセスがないかを確認する必要があります。

以上、grep コマンドを駆使したログ分析により、被害範囲特定、C2 サーバー特定、マルウェア侵入時刻の絞り込み、情報窃取確認といった成果が得られました。

次節では grep コマンドで得られた結果をもとに、他者への報告を想定した成果物を作成する方法を解説します。

4. 表計算ソフトによるログ分析

前節の grep コマンドを用いた分析は、膨大な量のログに対して高速に処理を行なうことが可能です。しかし、出力結果が分かりにくく、分析結果の可視化には向いていません。

そのため、調査では grep コマンドで必要な部分のログを抽出した後、Calc（LibreOffice）や Excel などの表計算ソフトを用いてデータ集計や成果物作成を行なうことがあります。

本節では、報告用の成果物作成を意識して、オープンソースオフィスソフト LibreOffice に含まれる表計算ソフト Calc を使用したログ分析を行ないます。

※2 **MFT（Master File Table）**：Windows でよく使われるファイルシステムである NTFS において、ファイルやディレクトリの情報を管理するテーブル。ファイル操作の痕跡を調査する際の重要な手がかりとなる。

4.1 ログデータの変換

表計算ソフトでログデータを集計・編集するために、まず次のコマンドを実行してください。スペース区切りの項目名（ヘッダー）行が記載されたテキストファイルを作成します。

```
$ echo 'SourceIP IdentUser User DateTime TimeZone Method URL
HTTPVersion StatusCode Size Referer UserAgent' > access.txt
```

次のコマンドを実行し、ログデータをテキストファイルに追記します。

```
$ cat access.log >> access.txt
```

続いて、次のコマンドを実行し、Calcをインストールします。その際にパスワードを聞かれるため、2.1 項で設定したパスワードを入力します。

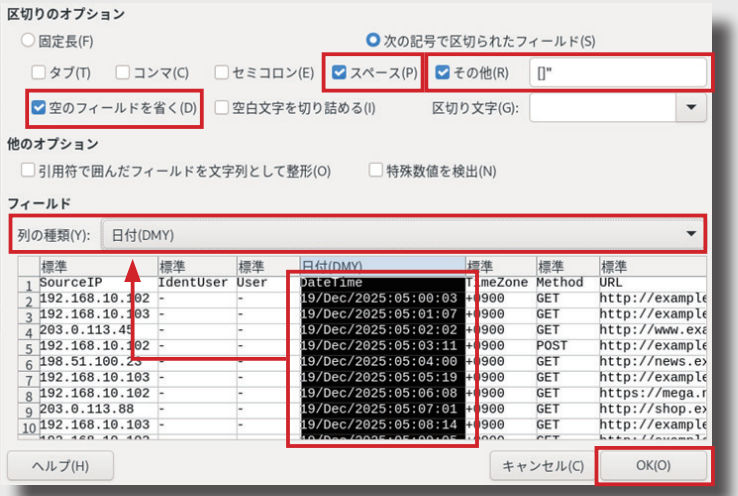
```
$ sudo yum -y install libreoffice-calc
```

次のコマンドを実行し、Calc を起動します。

```
$ libreoffice --calc access.txt
```

ただし、上記のコマンドで開かない場合は、Super（Windows）キーを入力して「terminal」と入力後、Enter キーを押して Calc を起動してください。上部タブの「ファイル」→「開く」から「access.txt」を開きます。

インポートウィンドウが開くため、「区切りのオプション」にチェックを入れ、「その他」の欄に「[]"」と入力します。その後、DateTime 列を選択して列の種類を「日付（DMY）」に変更し、「OK」をクリックしてください。

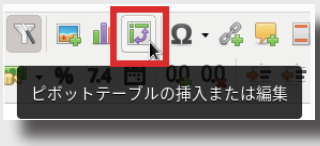


Calc では、元の DateTime 列の形式では日付として処理されません。そのため、列の種類を選択して変換することで、日付として処理が可能になります。

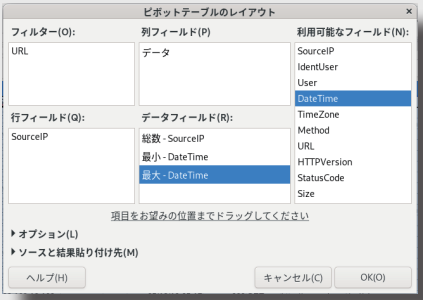
以上の操作により、ログデータを Calc で開くことができました。

4.2 Calc を用いたログ分析

本項では、C2 サーバーおよび upload.example.net にアクセスした IP アドレスをまとめた報告用成果物を作成することを目的として、ハンズオンを進めます。
まず、ピボットテーブル機能を用いてログを統計的に分析します。上部タブの中から「ピボットテーブルの挿入または編集」を選択してください。

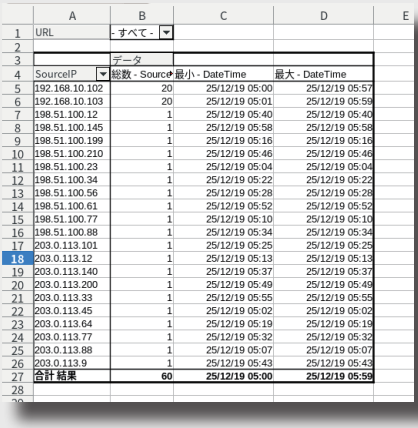


ソースの選択ウィンドウが開きます。データ範囲はデフォルトで全体が選択されているため、「現在の選択範囲」が選択された状態で「OK」をクリックしてください。
ピボットテーブルのレイアウトウィンドウが開きます。今回はアクセス先 URL ごとの SourceIP 出現回数を確認したいため、右側のフィールドを以下のように各位置にドラッグしてピボットテーブルを作成します。



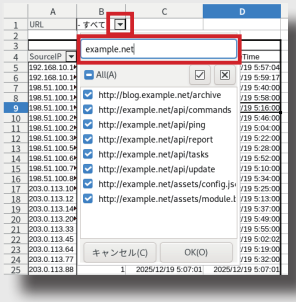
データフィールドに関しては、ドロップした段階では「合計 -SourceIP」となっています。ドロップしたフィールドをダブルクリックして、データフィールドウィンドウの計算方法で「総数」を選択し、「OK」をクリックしてください。

データフィールドに関しては、ドロップした段階では「合計 -SourceIP」となっています。ドロップしたフィールドをダブルクリックして、データフィールドウィンドウの計算方法で「総数」を選択し、「OK」をクリックしてください。



	A	B	C	D	E
1	URL	- すべて -			
2					
3		データ			
4	SourceIP	総数 - Source	最小 - DateTime	最大 - DateTime	
5	192.168.10.102	20	25/12/19 05:00	25/12/19 05:57	
6	192.168.10.103	20	25/12/19 05:01	25/12/19 05:59	
7	198.51.100.12	1	25/12/19 05:40	25/12/19 05:40	
8	198.51.100.145	1	25/12/19 05:58	25/12/19 05:58	
9	198.51.100.199	1	25/12/19 05:16	25/12/19 05:16	
10	198.51.100.210	1	25/12/19 05:46	25/12/19 05:46	
11	198.51.100.23	1	25/12/19 05:04	25/12/19 05:04	
12	198.51.100.34	1	25/12/19 05:22	25/12/19 05:22	
13	198.51.100.56	1	25/12/19 05:28	25/12/19 05:28	
14	198.51.100.61	1	25/12/19 05:52	25/12/19 05:52	
15	198.51.100.77	1	25/12/19 05:10	25/12/19 05:10	
16	198.51.100.88	1	25/12/19 05:34	25/12/19 05:34	
17	203.0.113.101	1	25/12/19 05:25	25/12/19 05:25	
18	203.0.113.12	1	25/12/19 05:13	25/12/19 05:13	
19	203.0.113.140	1	25/12/19 05:37	25/12/19 05:37	
20	203.0.113.200	1	25/12/19 05:49	25/12/19 05:49	
21	203.0.113.33	1	25/12/19 05:55	25/12/19 05:55	
22	203.0.113.45	1	25/12/19 05:02	25/12/19 05:02	
23	203.0.113.64	1	25/12/19 05:19	25/12/19 05:19	
24	203.0.113.77	1	25/12/19 05:32	25/12/19 05:32	
25	203.0.113.88	1	25/12/19 05:07	25/12/19 05:07	
26	203.0.113.9	1	25/12/19 05:43	25/12/19 05:43	
27	合計 結果	60	25/12/19 05:00	25/12/19 05:59	
28					

本ピボットテーブルでは、フィルターに設定した URL ごとに、SourceIP 出現回数（アクセス回数）、アクセス開始時刻、アクセス終了時刻の表を作成できます。
C2 サーバーのドメインにアクセスした端末の IP アドレスを集計します。URL フィルター（B1 セル）の「▼」マークをクリックし、検索欄に「example.net」と入力して Enter キーを押してください。

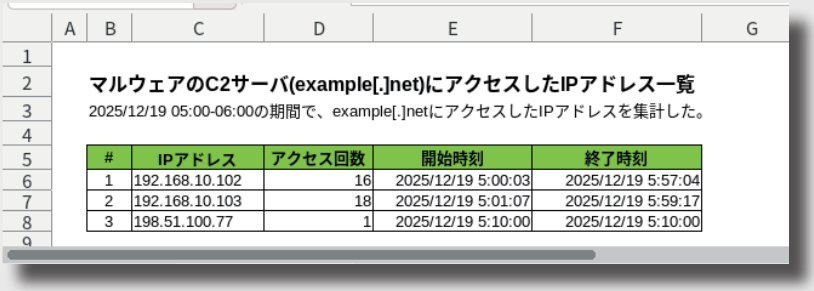


example.net にアクセスした IP アドレスごとの出現回数および時刻の表が作成されます。

	A	B	C	D
1	URL	- 複数 -		
2				
3		データ		
4	SourceIP	総数 - Source	最小 - DateTime	最大 - DateTime
5	192.168.10.102	16	25/12/19 05:00	25/12/19 05:57
6	192.168.10.103	18	25/12/19 05:01	25/12/19 05:59
7	198.51.100.77	1	25/12/19 05:10	25/12/19 05:10
8	合計 結果	35	25/12/19 05:00	25/12/19 05:59
9				

また、upload.example.net についても同様の操作を行なうことで表を作成できます。
以上の操作により、Calc を用いたログ分析および成果物作成が完了しました。
今後のために、この時点でスナップショットを作成しておくことをお勧めします。

実際の報告時は、これらの分析結果に対して説明を追記したり、セルの書式設定で整形したりするなどの編集を行ない、以下のような成果物を作成します。



マルウェアのC2サーバ(example[.]net)にアクセスしたIPアドレス一覧

2025/12/19 05:00-06:00の期間で、example[.]netにアクセスしたIPアドレスを集計した。

#	IPアドレス	アクセス回数	開始時刻	終了時刻
1	192.168.10.102	16	2025/12/19 5:00:03	2025/12/19 5:57:04
2	192.168.10.103	18	2025/12/19 5:01:07	2025/12/19 5:59:17
3	198.51.100.77	1	2025/12/19 5:10:00	2025/12/19 5:10:00

5. おわりに

今回はここまでとなります。「3. 攻撃兆候検出編」では grep を中心とした Linux コマンドを用いて、プロキシログから攻撃の兆候を検出する手順を解説しました。また、プロキシログを表計算ソフトの Calc に取り込み、報告向けの成果物作成を意識した整理方法を確認しました。

今後も新たな攻撃手法やログ分析ツールは登場し続けますが、grep などのログ分析の基本となるスキルを習得しておくことは大切です。本誌がログ分析の基本スキル習得の一助となれば幸いです。

なお、次ページには、本稿で使用したログのサンプルを掲載しています。

6. 付録（本稿「2.2.3 ログの保存」で利用できるログのサンプル）

```
192.168.10.102 - - [19/Dec/2025:05:00:03 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:01:07 +0900] "GET http://example.net/assets/config.json HTTP/1.1" 200 842 "-" "-"
203.0.113.45 - - [19/Dec/2025:05:02:02 +0900] "GET http://www.example.jp/index.html HTTP/1.1" 200 4231 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:03:11 +0900] "POST http://example.net/api/update HTTP/1.1" 200 1024 "-" "-"
198.51.100.23 - - [19/Dec/2025:05:04:00 +0900] "GET http://news.example.org/top HTTP/1.1" 200 8123 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:05:19 +0900] "GET http://example.net/api/tasks HTTP/1.1" 200 2048 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:06:08 +0900] "GET https://upload.example.com/file/AbCdEf HTTP/1.1" 200 4096 "-" "-"
203.0.113.88 - - [19/Dec/2025:05:07:01 +0900] "GET http://shop.example.com/products HTTP/1.1" 200 5321 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:08:14 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:09:05 +0900] "GET http://example.net/assets/module.bin HTTP/1.1" 200 8192 "-" "-"
198.51.100.77 - - [19/Dec/2025:05:10:00 +0900] "GET http://blog.example.net/archive HTTP/1.1" 200 2981 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:11:22 +0900] "POST http://example.net/api/report HTTP/1.1" 200 1536 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:12:09 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
203.0.113.12 - - [19/Dec/2025:05:13:00 +0900] "GET http://intranet.example.local/status HTTP/1.1" 200 734 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:14:18 +0900] "GET https://upload.example.com/file/XyZ123 HTTP/1.1" 200 6144 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:15:04 +0900] "GET http://example.net/api/commands HTTP/1.1" 200 1024 "-" "-"
198.51.100.199 - - [19/Dec/2025:05:16:00 +0900] "GET http://cdn.example.org/lib.js HTTP/1.1" 304 0 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:17:27 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:18:06 +0900] "POST http://example.net/api/update HTTP/1.1" 200 1024 "-" "-"
203.0.113.64 - - [19/Dec/2025:05:19:00 +0900] "GET http://portal.example.co.jp/login HTTP/1.1" 200 2567 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:20:15 +0900] "GET http://example.net/assets/config.json HTTP/1.1" 200 842 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:21:02 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
198.51.100.34 - - [19/Dec/2025:05:22:00 +0900] "GET http://weather.example.com/today HTTP/1.1" 200 1987 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:23:21 +0900] "GET http://example.net/api/tasks HTTP/1.1" 200 2048 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:24:10 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
203.0.113.101 - - [19/Dec/2025:05:25:00 +0900] "GET http://docs.example.org/manual.pdf HTTP/1.1" 200 10485 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:26:17 +0900] "POST http://example.net/api/report HTTP/1.1" 200 1536 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:27:05 +0900] "GET https://upload.example.com/file/QwErTy HTTP/1.1" 200 4096 "-" "-"
198.51.100.56 - - [19/Dec/2025:05:28:00 +0900] "GET http://support.example.com/faq HTTP/1.1" 200 3321 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:29:14 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:30:06 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:31:12 +0900] "GET http://example.net/assets/module.bin HTTP/1.1" 200 8192 "-" "-"
203.0.113.77 - - [19/Dec/2025:05:32:00 +0900] "GET http://www.example.co.jp/home HTTP/1.1" 200 4012 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:33:04 +0900] "POST http://example.net/api/update HTTP/1.1" 200 1024 "-" "-"
198.51.100.88 - - [19/Dec/2025:05:34:00 +0900] "GET http://media.example.org/video HTTP/1.1" 200 9123 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:35:19 +0900] "GET http://example.net/api/tasks HTTP/1.1" 200 2048 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:36:07 +0900] "GET https://upload.example.com/file/UiOpAs HTTP/1.1" 200 6144 "-" "-"
203.0.113.140 - - [19/Dec/2025:05:37:00 +0900] "GET http://shop.example.jp/cart HTTP/1.1" 200 2876 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:38:22 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:39:03 +0900] "GET http://example.net/assets/config.json HTTP/1.1" 200 842 "-" "-"
198.51.100.12 - - [19/Dec/2025:05:40:00 +0900] "GET http://blog.example.com/post HTTP/1.1" 200 3199 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:41:16 +0900] "POST http://example.net/api/report HTTP/1.1" 200 1536 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:42:08 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
203.0.113.9 - - [19/Dec/2025:05:43:00 +0900] "GET http://intra.example.local/health HTTP/1.1" 200 688 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:44:25 +0900] "GET https://upload.example.com/file/ZxCvBn HTTP/1.1" 200 4096 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:45:05 +0900] "GET http://example.net/api/commands HTTP/1.1" 200 1024 "-" "-"
198.51.100.210 - - [19/Dec/2025:05:46:00 +0900] "GET http://cdn.example.com/style.css HTTP/1.1" 304 0 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:47:18 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:48:06 +0900] "POST http://example.net/api/update HTTP/1.1" 200 1024 "-" "-"
203.0.113.200 - - [19/Dec/2025:05:49:00 +0900] "GET http://portal.example.jp/dashboard HTTP/1.1" 200 3771 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:50:21 +0900] "GET http://example.net/api/tasks HTTP/1.1" 200 2048 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:51:02 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
198.51.100.61 - - [19/Dec/2025:05:52:00 +0900] "GET http://weather.example.jp/week HTTP/1.1" 200 2210 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:53:14 +0900] "POST http://example.net/api/report HTTP/1.1" 200 1536 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:54:09 +0900] "GET https://upload.example.com/file/LkJhGf HTTP/1.1" 200 6144 "-" "-"
203.0.113.33 - - [19/Dec/2025:05:55:00 +0900] "GET http://docs.example.com/guide HTTP/1.1" 200 6890 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:56:23 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
192.168.10.102 - - [19/Dec/2025:05:57:04 +0900] "GET http://example.net/assets/module.bin HTTP/1.1" 200 8192 "-" "-"
198.51.100.145 - - [19/Dec/2025:05:58:00 +0900] "GET http://support.example.org/contact HTTP/1.1" 200 3011 "-" "-"
192.168.10.103 - - [19/Dec/2025:05:59:17 +0900] "GET http://example.net/api/ping HTTP/1.1" 200 512 "-" "-"
```

Hitachi Systems Security Journal

株式会社 日立システムズ　本社：〒141-8672　東京都品川区大崎 1-2-1　www.hitachi-systems.com

お問い合わせは

Printed in Japan