HITACHI

Hitachi Systems Security Journal

VOL.72

株式会社 日立システムズ

Hitachi Systems Security Journal VOL.12

TABLE OF CONTENTS

●はじめに

本文書は、株式会社日立システムズ の公開資料です。バックナンバーは以下の Web サイトで確認できます。
https://www.hitachi-systems.com/report/specialist/index.html

鉄道信号システムのセキュリティ上の盲点を突く

ダビド・メレンデス インタビュー

取材・文＝斉藤 健一／通訳＝エル・ケンタロウ／撮影＝卯月 梨沙

東京で開催されたサイバーセキュリティ国際カンファレンス CODE BLUE 2024 において、「レガシー鉄道信号システムの悪用」と題した講演が行われた。発表の焦点は、スペインで広く導入されている鉄道信号システム「ASFA」（Anuncio de Señales y Frenado Automático）。ASFA は、線路側に設置されたビーコンが特定の周波数で列車に信号情報を送信し、運転席に表示する仕組みとなっている。運転士の見落としや誤認といったヒューマンエラーを防ぐことを目的に導入され、中央の管制から列車位置を把握できない区間などで重要な役割を果たしてきた。

しかし講演者のダビド・メレンデス氏は、この ASFA に運用上の盲点を発見した。技術資料を基に解析を重ね、自ら試作したデバイスを用いて、悪意ある第三者が偽の信号を列車に送信できる可能性を実証したのだ。規模こそ小さな実験だったが、われわれが日々利用する鉄道の安全性を脅かすサイバーリスクの存在を具体的に実証した意義は大きい。

鉄道信号システム ASFA に潜むぜい弱性

斉藤（以下 S ）：講演の中で紹介された「ASFA」という鉄道信号システムは、国際的な規格なのでしょうか。

ダビド・メレンデス（以下 D ）：ASFA はスペイン独自のシステムですが、その基盤となる技術は世界各国の鉄道でも共通して利用されています。

S CODE BLUE 事務局による講師への事前インタビュー※ では、「研究を始めたきっかけは鉄道が好きだから」と語られていました。では実際に、この研究を進めるにあたって、どのような資料や情報源を調査されたのでしょうか。

D もともと見つけた資料には、いくつかの仕様要件のようなものが記されていました。例えば「このデバイスと列車側の装置、線路上のビーコンは、この周波数で通信する必要がある」といった具合に、複数の周波数やその密度に関する条件が示されていたのです。そこで私は、電子工学の視点から「もし自分が設計するならこう組み立てるだろう」と考え、資料には書かれていない部分を推測して補いながら理解を深めていきました。

S 講演では「今回来日できなかった研究所のパートナーがいる」とおっしゃっていましたが、その方との役割分担や、一緒に研究を始めることになった経緯についてお聞かせください。

D 資料の研究は、もともと 5 年前に自分ひとりで始めたのですが、その過程でパートナーとなるガブリエラ・ガルシアさんと出会いました。彼女は、僕が取り組んでいる研究をより幅広い人に理解してもらえるように翻訳的な役割を担うのが得意です。さらに優れているのは、先ほど触れた資料に含まれるギャップを埋め、全体の思考プロセスを支援してくれる点で、そうした強みを生かして一緒に研究を進めてきました。

S わかりました。今回の研究は、中央のコントロールが管理していない「ダークテリトリー」をターゲットにしているとのことでした。例えばスペインの場合、中央のコントロールセンターが実際に管理しているのは国全体のどのくらいの部分で、そして今回の研究が応用できるのはそのうちどのくらいなのか、お聞かせいただけますか。

D 私たちが「ダークテリトリー」と呼んでいるのは、あくまで導入的な枠組みとして設定したトピックです。実際には中央コントロールの有無にかかわらず攻撃は成立し得るため、管理されていない領域だけに限った話ではありません。

S なるほど、中央コントロールの有無にかかわらず攻撃が成立し得るというのは意外でした。

D もともとガブリエラさんが「このテーマなら多くの人に理解してもらえるだろう」と考えて選んだ経緯もあり、基礎的な段階にとどまっていますが、扱っている技術は幅広く応用でき、今後さまざまな領域に展開できる可能性があります。その意味で、この研究の示唆するところは非常に大き

●ダビド・メレンデス（David Melendez）

サイバーセキュリティとハードウェアハッキングの分野で 12 年以上の経験を持つセキュリティリサーチャー。DEFCON、BlackHat、RootedCON など、世界中の著名なカンファレンスで画期的な調査を発表してきた実績を持つ。

また、ドローンの開発者でもあり、ドローンを使ったサイバーセキュリティ研究における革新的なアプローチを紹介する書籍「Hacking with Drones」を上梓している。技術の限界に挑む情熱を持ち、組込みシステムのセキュリティと機能性を向上させる新たな方法を常に模索している。

※ **CODE BLUE 事務局による講師への事前インタビュー**（インタビューに加えて、講演写真や講演動画・スライドのリンクも提供）
https://note.com/code_blue/n/n3ce83a364858

いと考えています。

S この信号システムの問題は、列車とビーコンとの間でやりとりされる通信に認証機構がないことが原因なのではないかと思いました。この点はいかがでしょうか。

D ご指摘のとおりです。公開されている仕様書だけでここまでできてしまうということは、裏を返せば認証機構がないため、極端な話、どんな情報でも送り込めてしまう可能性があります。場合によっては、列車を停止させるような信号を発信することさえあり得るのです。

S 自動車のハッキングにおいても、当初は命令が認証されていなかったことがありました。それと似たような状況なのではないかと思いました。

D 自動車のハッキングと比べると、攻撃はおそらく容易です。理由は、ここではデジタルデータのやり取りがなく、特定の周波数によるアナログ信号だけで完結しているためです。つまり、自動車の CAN バスのように、プロトコル上でデジタル信号を交換しているわけではなく、純粋にアナログ的な仕組みとして成り立っています。さらに、このビーコンには Wi-Fi リレーを接続することで、リモートから通信できる可能性もあります。

S デバイスを制作するときに、いちばん苦労したポイントは何ですか。

D いちばん大変だったのは、アンテナの長さを決めることでした。周波数自体は分かっていたものの、それに適した長さを正確に特定するのは非常に難しかったのです。仕様書に「何メートル」と明記されているわけではないため、実際にアンテナを巻いてはテストし、再び巻いてはテストするという実験を繰り返すしかありませんでした。

S 試作にかかった期間は、どれほどでしたか。

D 検討に約 3 カ月、テストには約 10 日を要しました。資料に掲載されていた写真から概寸はわかっていたので、「この部品数であれば大体これくらいの大きさだろう」と見当をつけました。さらに、ビーコンの写真を参考にしながら、おそらくこのくらいのサイズだろうと予測して作業を始めました。

S 写真から大きさの見当をつけて作業されたのですね。

D また、資料を見ると、コイルが 2 つあり、それぞれが電車側に向いていました。ということは、

メレンデス氏の講演動画やスライドなどは CODE BLUE 公式サイトのアーカイブページからもアクセスできる
https://archive.codeblue.jp/2024/results/results/#result-6

ビーコンは両方のコイルに同時に信号を送る必要があるはずです。そう考えると、単に 2 つのコイルを並べるだけでは不十分で、コイル自体を大きくしなければ伝播範囲は広がらないだろうと予測していました。

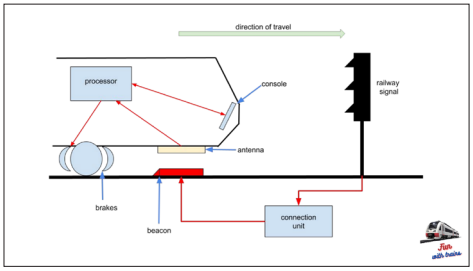
S 実際に動かしてみたときのことを教えてください。

D ガブリエラさんと 2 人で実験に取り組んでいたとき、信号を見て「今ここで周波数のピークが上がった」と気づいた瞬間がありました。目標の数値が出たときには、さまざまな感情が一度に押し寄せました。「本当にできた」という達成感、「ようやくたどり着いた」という安堵感、そして同時に「使い方を誤れば危険にもなり得る」という緊張感です。

S まさに、「ひらめきの瞬間」であったのと同時に、危うさも意識されたのですね。この研究はこれからも続けていくおつもりですか。

D はい。今後は、高速列車や現在運行している電車にも対応できる仕組みがあるのではないかと考えています。おそらく、同様のシステムがすでにデジタル化されて存在していると思います。ただ、サイバーセキュリティの観点から見ると、従来からのぜい弱性がそのまま引き継がれている可能性があるのではないかと感じています。

線路に敷設されたビーコン（写真左）と列車に設置されたアンテナ（写真右：赤い円の囲み部分）［メレンデス氏の講演スライドより］



ASFA の構成図［メレンデス氏の講演スライドより］

メレンデス氏らが制作したデバイス、列車側のアンテナ（写真上）と線路側のビーコン（写真下）

S 今回の研究をこうしたカンファレンスで発表されるのは、これまでにもあったのでしょうか。

D はい。これまでに米国の DEFCON とスペインの RootedCON で発表しています。

S そのときの参加者の反応はいかがでしたか。

D DEF CON では非常に好意的に受け止められ、参加者からも良い反応を得ることができました。一方で、スペインで発表した際には、会場にいた一部の人たちから、意図とは関係のない質問が次々と投げかけられる場面もありました。

S 自国の鉄道システムに関する発表ですから、快く思わず、話をそらしたり、時間を浪費させようとしたりする人がいても不思議ではありませんね。ところで、この研究は本業の合間の時間を使って取り組まれたのでしょうか。

D 普段はサイバーセキュリティに関する R&D を行っているため、その合間の時間で今回の研究を進めていました。理論的な検討を重ねる中で、いざ試作品を作ろうという段階になったとき、「家の中を長期間ワイヤーで散らかしたままにはできない」と考えました。そこで「ここからここまで」と期間を区切り、その間に集中して取り組んだことが、研究成果につながったのです。

低レイヤーへの挑戦
可視化で拓く新しい学び

S 差し支えなければ、普段のお仕事についても教えていただけますか。

D 個人の研究ですので、所属する組織については伏せたいと思います。私自身はもともと組み込み系のソフトウェア開発に携わっており、その流れでハードウェア系のサイバーセキュリティにも関わるようになった、というのが日々の仕事の内容です。

S CODE BLUE の講師プロフィールや今回の講演を拝見して、ハードウェアの低レイヤー分野を得意とされているのだと感じました。その分野をめざすようになったきっかけや理由を教えていただけますか？

D むしろ私の考え方は逆です。最初に何かに興味を持つと、それを理解するためにはどこまで深掘りする必要があるのか、どのレベルまで踏み込まなければならないのか、そうした観点で取り組んできました。上位レイヤーのことを学ぶうちに、理解を深めるためには低レイヤーも勉強せざるを得ませんでした。つまり、もともと低レイヤーに特別な関心があったのではなく、自分が興味を持った事柄を理解するために低レイヤーを学ぶ必要があった、ということです。

S サイバーセキュリティには、リバースエンジニアリングやバイナリー解析といった上位レイヤー

「難しい」と思われがちな低レイヤー技術も実物デバイスを制作し可視化することで「難しくない」と理解を深める啓発活動を行って行きたいと語るメレンデス氏

寄りの興味深い取り組みが数多くあり、サービスに近い技術も対象とされています。一方、低レイヤー領域は「難しく、ハードコアな人が挑むもの」と見られがちです。しかし、近年は利用できる技術も増え、以前ほど敷居は高くありません。ご自身は好奇心から低レイヤーを避けるのではなく積極的に飛び込み、挑戦してきたと語られました。その研究姿勢やアプローチを形づくっている要素は何か、お考えを伺いたいと思います。

D おっしゃるとおりだと私たちも考えています。そこで、あえてこうしたデバイスを持ち込み、実際に触れてもらうことで理解を深められるよう可視化しました。形にしたものを提示することで、「思ったほど難しくない」と感じてもらえるようにしたのです。一般的に「低レイヤー」と聞くと、魔法のように難解なものと思われがちですが、実際に物理的なデバイスを目にすれば、それほど複雑ではないと理解してもらえるはずです。私たちは、こうした活動を通じて啓発を進めていきたいと考えています。

S サイバーセキュリティの分野にいると、このような物理デバイスに触れる機会は多くありません。だからこそ、こうした取り組みは、非セキュリティ分野の人々にとって理解を身近に感じられるきっかけになると思います。さらに、学生をはじめとするエンジニアにとっても、学びの良い教材になるでしょう。

D 低レイヤーは、サイバーセキュリティにおける最後の大きな障壁の 1 つではないかと思います。その背景には、サイバーセキュリティの分野から入った人にとって、この領域が非常に高いハードルに映るという事情があります。ブラックボックスの中で魔法のような仕組みが動いているように見えることもあり、利用者の側からすると「大丈夫、ハードウェアというバリアがあるからセキュリティも問題ない」と、つい安易に依存してしまいがちなのです。

S 昔、OT のセキュリティでもエアギャップ（ネットワークに接続されていない）があるから、大丈夫だと過信していた時代がありましたが、USB メモリーなどによる攻撃を受ける結果となりました。今回の研究も同様に、常識のベールが剥がしていくものだと考えます。

D 確かに、すべてのことには大きな問題が伴います。例えば Web サーバーであれば、バグバウンティを通じて報告を受け、修正するコストは比較的低く抑えられます。しかしハードウェアデバイスの場合は、置き換えや再設計、再構築が必要になることも多く、そのコストははるかに高くつきます。そうした意味で、私たちはハードウェアの方がより困難になると考えています。

S 本日は、興味深い話をいろいろと伺うことができました。ありがとうございました。

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 2025.10

文＝日立システムズ

警察庁が公開したランサムウェア復号ツールの検証

【概要】：2025 年 7 月 17 日、警察庁はランサムウェア「Phobos/8Base」により暗号化されたファイルを復元するツールを公開した。本稿では、この復号ツールの対応範囲を確認するため、Phobos の亜種を含む 6 つの実検体で検証を行い、その結果を報告する。

【内容】：警察庁はランサムウェア「Phobos/8Base」により暗号化されたデータを復号するツールを公開した。FBI の協力を得て開発され、世界規模のランサムウェア対策として提供された。本ツールは警察庁の Web ページからダウンロード可能で、復号対象と復号先を指定して実行するだけの簡単な操作で利用でき、UI も分かりやすかった。

警察庁は 2024 年 2 月、Lockbit により暗号化されたファイルの復号ツールを公開した。当社では 5 つの検体で復号可否を検証したが、すべて失敗した。対象とする Lockbit のバージョンが異なっていた可能性がある。

本稿では、「Phobos/8Base」復号ツールの復号範囲を確認するための検証結果を報告する。まず、Windows 11 の仮想環境を構築し、暗号化対象となるダミーファイル 6 つを用意した。暗号化後、ガイドラインに記載の拡張子 3 種（8base、faust、Elbie）、記載のない Phobos 拡張子 1 種（eight）、および Phobos ベースのランサムウェア 2 種（2023、FUNNY）を検証した。

検証結果を表に示す。6 検体中 4 検体で復号に成功した。ガイドライン記載の 3 種に加え、記載外の拡張子「eight」も復号可能だった。一方、Phobos ベースの CrySIS および Dharma では復号に失敗した。両検体は命名規則が他と異なり、規則に沿って名称を変更しても「Error: Target file not found」と表示された。このことから、命名規則が復号鍵導出に関わる要素の 1 つだが、他にも条件があると推測される。

今回のツールは、Lockbit 用ツール公開時と異なり、警察庁から実例動画や使用ガイドラインが提供されている。このことから、より汎用的かつ実用的なツールであると考えられる。復号範囲が広いのは、国際捜査で押収したサーバーや逮捕者から得た情報を活用して開発されたためと推測される。なお、本ツールは復号後のファイル完全性を保証せず、破損や欠損が残る可能性がある。バックアップからの復旧を基本とし、復号後のファイルに悪意あるものが含まれていないか確認するなど、使用時は十分な注意が必要である。

表 復号に関する検証結果一覧

検体の種類	暗号化ファイルの命名	拡張子	対応拡張子への記載有無	復号可否
8Base	{ 元のファイル名 }.id[{8 文字のランダムな英数字 } ー {4 桁の数字 }].[{ メールアドレス }].{ 拡張子 }	.8base	有	○
Phobos	同上	.faust	有	○
Phobos	同上	.Elbie	有	○
Phobos	同上	.eight	無	○
Dharma	{ 元のファイル名 }.id-{8 文字のランダムな英数字 }.[{ メールアドレス }].{ 拡張子 }	.2023	無	×
CrySIS	同上	.FUNNY	無	×

[情報源] https://www.npa.go.jp/bureau/cyber/countermeasures/ransom/phobos.html

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 Linux 調査コマンド

1. ネットワーク編

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。今回より「Linux 調査コマンド」と題して、Linux に標準的に搭載されているコマンド群を用いて、ネットワーク通信・プロセス・スケジューラ・履歴などの側面から情報を収集・分析する方法を整理します。対象とするコマンドは **ss, lsof, ps, ip, crontab, history, date, who** であり、どれもインシデント対応において重要な役割を果たします。

調査者はこれらのツールを通じて早期の状況把握をするとともに、痕跡の保存・再構築といった対応を行なう必要があります。本稿では、各コマンドの基本的な使い方から、実際の調査現場を想定した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. ネットワーク編
2. プロセス・ファイル監視編
3. システム管理編

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ことを推奨します。

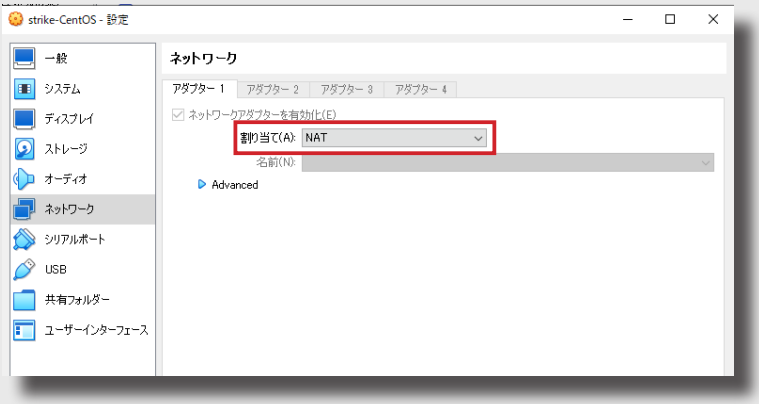
2. 事前準備

2.1 CentOS の準備

今回、検証で利用する CentOS を準備します。
CentOS は、「Let's Try HDD 保全！ 1. 準備編」（本誌 Vol.50）にて作成していますので、作成済みの方はそちらをご利用ください（すばやく初期状態に戻せるように、スナップショットを有効活用してください）。

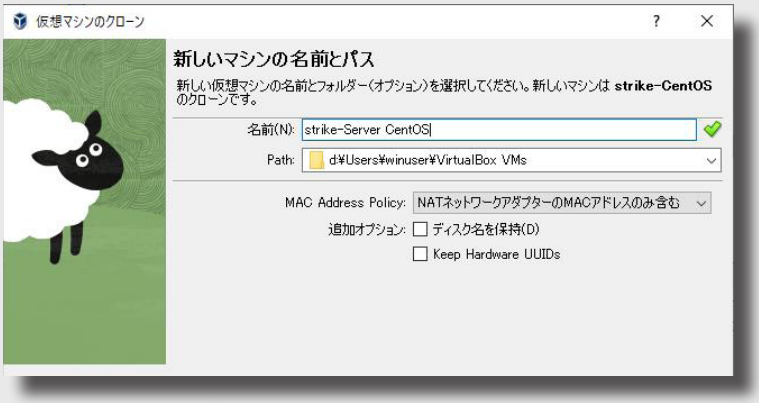
2.2　CentOS のネットワーク接続

CentOS のネットワーク接続を確認します。
「設定」→「ネットワーク」→「アダプター１」の割り当てが、「NAT」
となっていることを確認します。



2.3　クローンの作成

ip コマンドのハンズオン時に仮想環境同士の通信を行ないます。
その際、もう１台 CentOS 環境が必要になるため、下記手順に沿ってクローンを作成してください。
ハンズオンは外部との通信は行わず内部通信を利用するため、以前講義で作成していただいたものを再利用していただいてもかまいません。
手順：「クローン」→「次へ」→「次へ」→「完了」
新しくクローンした場合、現在作成済みの仮想環境とは名前が異なる任意の名前を設定してください。ここでは「strike-Server CentOS」とします。
クローン作成に関して「Let's Try ぜい弱性検証＋緩和策適用！ 1．ぜい弱性（Log4j）体験編」（本誌 Vol.60）を参考にしてください。



3. ネットワーク関連コマンド

Linux システムにおけるネットワーク調査やフォレンジック調査では、**ss** コマンドと **ip** コマンドの理解と活用が欠かせません。これらのコマンドは、システムの通信状況やネットワーク設定を把握するための重要な手段です。
特に、侵害対応や証拠保全では、早期かつ正確な情報収集が求められるため、これらのコマンドの基本操作をマスターしておく必要があります。本稿では、初心者でも理解できるように基礎から応用までを解説し、実際に手を動かしながら学べるハンズオンも紹介します。

3.1. ソケットの状態確認する ss コマンド

ss（**socket statistics**）コマンドは、Linux におけるソケットの状態を確認するためのツールであり、TCP/UDP ポートの待ち受け状態や確立済みの接続、通信中のプロセス情報などを確認することができます。

3.1.1 基本的なコマンド操作

基本的な使用方法やよく用いられるオプションを紹介し、ネットワーク接続状況の把握における **ss** の有用性を確認します。

・すべての接続を表示する

```
# ss -a
```

・LISTEN 状態のポートを表示する

```
# ss -l
```

・TCP 接続のみを表示する

```
# ss -t
```

・プロセス情報も表示する

```
# ss -p
```

3.1.2 ハンズオン

3.1.1 で確認したコマンドを用いて下記の観点でハンズオンを実施します。

① 現在 LISTEN しているポートを調べる
② 特定のポート（例：22 番ポート）に関する情報を探す
③ 不審なプロセス（例：異常に多くの接続を持つプロセス）がないかを確認する

現在開放されているポートを確認します。
下記コマンドを入力し、Enter を押します。

```
# ss -atu
```

-a：すべてのソケットを表示　-t：TCP　-u：UDP

また、”grep” を利用することで特定の Port に絞った表示も可能になります。

```
[root@localhost ~]# ss -atu
Netid       State       Recv-Q      Send-Q           Local Address:Port           Peer Address:Port
udp         UNCONN      0           0                127.0.0.1:323               0.0.0.0:*
udp         ESTAB       0           0          10.0.2.15%enp0s3:bootpc           10.0.2.2:bootps
udp         UNCONN      0           0                   [::1]:323                   [::]:*
tcp         LISTEN      0           128               0.0.0.0:ssh                 0.0.0.0:*
tcp         LISTEN      0           128                  [::]:ssh                    [::]:*
```

今回は、SSH を利用した通信に絞って表示しています。

```
# sudo ss -ap | grep :ssh
# sudo ss -anp | grep :22
# ss -ntp | grep -E 'dport=22|sport=22'
```

-n オプションを付けることで、名前解決をせずポート番号を数値で表示させることができます。HTTPS（443 番ポート）や SSH（22 番ポート）における外部との通信は、正規の用途がある一方で、攻撃者の活動の痕跡である可能性もあるため注意が必要です。例えば、HTTPS を利用した不審な IP アドレスへの通信が継続的に発生している場合、マルウェアが C2（Command & Control）サーバーと通信している疑いがあります。特に、通信先が不自然なドメイン名であったり、国外の無関係な IP であったりする場合には、定期的な通信パターンやデータ送受信の内容も含めて詳細な分析が求められます。

```
[root@localhost ~]# sudo ss -ap | grep :ssh
tcp   LISTEN 0      128                                0.0.0.0:ssh                0.0.0.0:*     users:(("sshd",pid=779,fd=3))

tcp   LISTEN 0      128                                   [::]:ssh                   [::]:*     users:(("sshd",pid=779,fd=4))
```

一方、SSH への外部からの接続が確認された場合には、さらに深刻な脅威である可能性があります。SSH は通常、管理目的で利用されるため、外部からの接続は不正アクセスの試みを示しているかもしれません。特に、パスワード総当たり攻撃（ブルートフォース）や、何らかのぜい弱性を突いた侵入が行われていた場合、攻撃者が内部ネットワークに足場を築いてしまっている可能性があります。

3.1.3　ss コマンドの調査利用

調査の際、以下の観点から **ss** コマンドは重要と言えます。

- 外部への異常な通信先（IP アドレスやポート）
- 内部で異常なリスニングポートの存在
- 通信に関与しているプロセスの特定（PID との突き合わせ）

また、証拠保全においても利用できます。
ネットワークの状態は揮発性の高い情報とされ、ネットワークの遮断・隔離、マシンのシャットダウン、時間経過などで情報が失われます。そのため、インシデント対応中には、調査対象システムの通信状態を可能な限り早くファイルとして記録し、後から詳細に分析できるよう備える必要があります。

例：**`ss -ap > /tmp/ss_snapshot_$(date +%F_%T).log`**

タイムスタンプ付きのファイル名で保存しておくことで、後の時系列分析や証拠提出にも活用できます。

3.2. ネットワーク構成の確認・設定を行なう ip コマンド

続いて、Linux におけるネットワーク構成の確認・設定を行なう **ip** コマンドを確認します。
ip コマンドは従来の ipconfig や route の代替として広く利用されています。

3.2.1 基本的な操作

ip コマンドを利用した基本的な操作を確認します。

・IP アドレスの確認

```
# ip addr
```

・ルーティングテーブルの確認

```
# ip route
```

・ネットワークインターフェイスの状態確認

```
# ip link
```

新たなアドレスの追加の例（後述のハンズオンにて確認）

```
# sudo ip addr add 192.168.1.100/24 dev eth0
```

3.2.2　自分の IP アドレス設定を確認

現在、ネットワークインターフェイスに割り当てられている IP を表示します。
下記コマンドを入力し入力し Enter を押してください。

```
# ip address
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d4:57:cc brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 79557sec preferred_lft 79557sec
    inet6 fe80::a00:27ff:fed4:57cc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

表 ip address の出力例

項目	説明
2:00:00	インターフェイス番号（識別用。通常は気にしなくて OK）
eth0:	インターフェイス名。ここでは「eth0」つまり有線 LAN アダプターを指します
<BROADCAST,MULTICAST,UP, LOWER_UP>	インターフェイスの状態を示すフラグ群： ・BROADCAST = ブロードキャスト可能 ・MULTICAST = マルチキャスト可能 ・UP = インターフェイスが有効化されている ・LOWER_UP = 物理層（ケーブルなど）も接続されている
mtu 1500	MTU(最大転送単位)。一度に送れるパケットの最大サイズ（通常は 1500 バイト）
inet 192.168.1.100/24	IPv4 アドレスとサブネットマスク（/24 は 255.255.255.0 に相当）
brd 192.168.1.255	ブロードキャストアドレス。このネットワーク内の全ホスト宛に送信する際に使用されます
scope global	アドレスのスコープ（有効範囲）。global はインターネット上でも有効なアドレスを意味します
dynamic	この IP アドレスは DHCP など動的な手段で割り当てられたことを示します
eth0(再表示)	この IP アドレスが所属するデバイス（ここでは「eth0」）を再確認しています

次にルーティングテーブルの確認を行ないます。
どのネットワークにある相手にデータを送るとき、どのインターフェイスを通るかを確認できます。

```
# ip address
```

```
default via 10.0.2.2 dev enp0s3 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
```

default via 10.0.2.2 dev enp0s3　：デフォルトゲートウェイ

他に default が存在しないか、外部への不審なルートがないか確認してください。

3.2.3　外部サーバーとの通信 (ローカルで再現)

外部サーバーとの通信がどのように見えるかを実際に確認していきます。**ip,ss,lsof** を利用した VirtualBox 上の CentOS 同士のネットワーク検証を行ないます。

・検証準備

検証を行なう前に「2. 事前準備」で用意した CentOS 環境を追加で 1 つ作成してください。

本検証では、**lsof,netcat** を利用します。後述しますが、CentOS9 Stream では標準でインストールされていませんので、後述の解説にしたがってダウンロードしてください。
すでにダウンロード済みか確認する場合は下記コマンドを実行した結果を確認します。

```
# which nc
```

存在する場合と存在しない場合、それぞれの出力結果は以下の図のとおりです。

存在する場合

```
[root@localhost ~]# which nc
/usr/bin/nc
[root@localhost ~]# which lsof
/usr/bin/lsof
```

存在しない場合

```
[root@localhost ~]# which lsof
/usr/bin/which: no lsof in (/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin)
[root@localhost ~]# which netcat
/usr/bin/which: no netcat in (/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin)
```

・ダウンロード方法

isof,nc のダウンロードは下記のコマンドを実行してください

```
# yum install -y nc
```

```
# yum install -y lsof
```

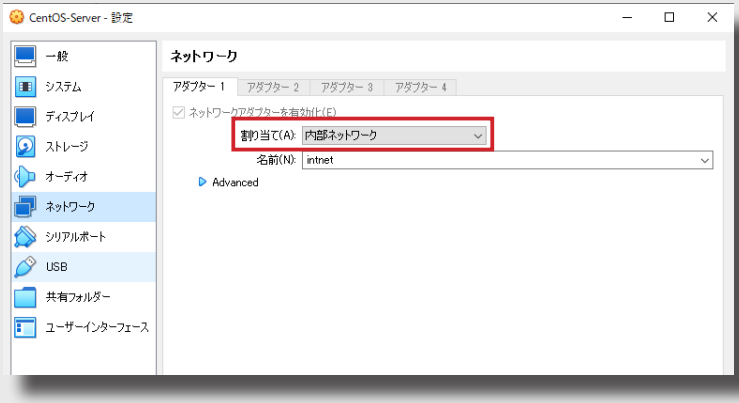
ダウンロード後、下記のように表示されれば完了しています。

```
Installing dependencies:
 libtirpc                       x86_64                  1.3.3-9.el9              baseos                    94 k

Transaction Summary
================================================================================
Install  2 Packages

Total download size: 333 k
Installed size: 826 k
Downloading Packages:
(1/2): libtirpc-1.3.3-9.el9.x86_64.rpm          366 kB/s |  94 kB     00:00
(2/2): lsof-4.94.0-3.el9.x86_64.rpm             638 kB/s | 239 kB     00:00
--------------------------------------------------------------------------------
Total                                           427 kB/s | 333 kB     00:00
CentOS Stream 9 - BaseOS                         35 kB/s | 1.6 kB     00:00
Importing GPG key 0x8483C65D:
 Userid     : "CentOS (CentOS Official Signing Key) <security@centos.org>"
 Fingerprint: 99DB 70FA E1D7 CE22 7FB6 4882 05B5 55B3 8483 C65D
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : libtirpc-1.3.3-9.el9.x86_64                            1/2
  Installing       : lsof-4.94.0-3.el9.x86_64                               2/2
  Running scriptlet: lsof-4.94.0-3.el9.x86_64                               2/2
  Verifying        : libtirpc-1.3.3-9.el9.x86_64                            1/2
  Verifying        : lsof-4.94.0-3.el9.x86_64                               2/2
```

今回、２台利用するため２台ともにダウンロードをお願いします。
検証実行前に VirtualBox のネットワーク設定を確認してください。
「設定」→「ネットワーク」→「アダプター 1」→割り当てを「内部ネットワーク」としてください。



今回利用する環境どちらとも内部ネットワークに変更してください。
また、はじめに作成した仮想環境をクライアント、クローンで作成した環境をサーバーと考えてください。

・クライアント側での IP アドレスの確認
クライアント側で **ip a** を入力し、自身の IP を再度確認してください。

```
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d4:57:cc brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.20/24 brd 192.168.56.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed4:57cc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

検証で利用する IP アドレスは赤枠になります。
「3.2.2　自分の IP アドレス設定を確認」(p15) の時点では筆者の IP アドレスは「10.0.2.15」となっていましたが、検証にあたって IP アドレスを変更いたしました。
10.0.2.x/24 は VirtualBox の「NAT」ネットワークであり、外部インターネットへの接続は可能ですが、仮想マシン同士の直接通信ができません。
そのため、仮想環境同士と通信が可能なホストオンリーネットワーク「192.168.56.x」に変更が必要です。

・サーバー側での IP アドレスの変更・確認
また、以前作成した環境をクローンで作成した場合、同じ IP アドレスが設定されているため、通信が実施できません。

そのため、サーバー側で新たに IP アドレスを設定する必要があります。
下記手順に沿って、IP アドレスを設定してください。
筆者はクライアント側を「192.168.56.20/24」、サーバー側を「192.168.56.30/24」としています。

① IP アドレスの変更

```
# sudo ip addr add 192.168.56.30/24 dev enp0s3
```

② IP アドレスの確認

```
# ip addr show dev enp0s3
```

ip コマンドを利用した IP アドレスの追加は再起動すると元に戻ります。ハンズオン終了後、手動で削除する場合は以下のコマンドを実行してください。

```
# sudo ip addr del 192.168.56.30/24 dev enp0s3
```

上記方法で設定できなかった場合は、以下の手順でコマンドを実行してください。

```
# nmcli connection modify enp0s3 ipv4.addresses 192.168.56.30/24
# nmcli connection modify enp0s3 ipv4.method manual
# nmcli connection up eno0s3
```

ハンズオン終了後、設定した IP アドレスを戻すには以下の手順でコマンドを実行してください。

```
# sudo nmcli connection modify enp0s3 ipv4.method auto
# sudo nmcli connection modify enp0s3 -ipv4.addresses
```

実施後、サーバー側で **ip a** を実行し IP アドレスが変更されているか確認してください。

```
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ac:65:b8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.30/24 brd 192.168.56.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feac:65b8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

確認後、**ping** コマンドが有効か実行してください。

```
# ping -c 3 192.168.56.30
```

下記のように実行できていれば問題ありません。

```
[root@localhost ~]# ping -c 3 192.168.56.30
PING 192.168.56.30 (192.168.56.30) 56(84) bytes of data.
64 bytes from 192.168.56.30: icmp_seq=1 ttl=64 time=0.581 ms
64 bytes from 192.168.56.30: icmp_seq=2 ttl=64 time=1.73 ms
64 bytes from 192.168.56.30: icmp_seq=3 ttl=64 time=0.608 ms

--- 192.168.56.30 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.581/0.974/1.734/0.537 ms
```

・サーバー側での netcat の設定とクライアント側からの接続

netcat（nc）は、TCP/UDP 接続を手軽に行えるコマンドラインツールで、ポートスキャン、デバッグ、ファイル転送などに利用されます。
セキュリティ検証や通信テストに使われ、リスナー、クライアント双方の役割を持てる柔軟なツールです。

まず、サーバー側で待受け設定を行ないます。

```
# nc -l 8080 &
```

次に、クライアント側で接続を行ないます。

```
# nc 192.168.56.30 8080
```

接続に成功すると、サーバー側とクライアント側がメッセージをやり取りできる状態になります。
成功しなかった場合、ポート 8080 が開いてないと思われます。
検証実施のため、一時的に開きます (内部ネットワークのため、外部から通信は行ないません)。
サーバー側で下記コマンドを実行して、ポート 8080 がどのような状態か確認してください。

```
# sudo firewall-cmd --list-all
```

確認後、下記コマンドを入力し、実行します。
なお、「--permanent」を付加して永続的な設定にする必要はありません。

```
# firewall-cmd --add-port=8080/tcp
```

下記コマンドを実施して、設定を即時反映させてください。

```
# sudo firewall-cmd --reload
```

設定後、再びサーバー・クライアントの待受け、接続を実施してください。
クライアント側またはサーバー側で入力した文字がチャットのように双方向で表示されれば問題ありません。

下記コマンドを入力すると、一覧で表示されます。

```
# ss -atp
```

・クライアント側

・サーバー側

「ESTAB」で通信が成功しており、プロセス情報や通信相手の IP アドレスが確認できます。「webcache」と表示されているのは、**ss** コマンドが 8080 →「webcache」と名前解決を行っているためです。

ss -antp と入力すると、名前解決されずポート番号で表示することができます。

3.2.4 ip コマンドの調査利用

① 不審な外部通信があると疑われた場合

・**ip addr** → 攻撃者が不正な NIC/IP を追加していないか

・**ip route** → 通信経路が VPN 経由や外部トンネル経由になっていないか

・**ip link** →　無効 NIC が急に UP になってないか

② 内部からの情報通信が疑われる場合

・**ip route** で外部向けのゲートウェイが社内ネットワーク外になっていないか

・**ip neigh** で普段と異なる MAC アドレスが見られないか

インシデント調査時の注意点

・ライブ状態の情報は揮発性が高いため、調査時にはコマンド実行後すぐに記録を残す必要があります (ログに保存、出力をコピーするなど)。

・ip コマンドの出力は通信経路・対象が直接わかるものではないが、状況として重要な情報となります。

4. おわりに

今回はここまでとなります。

本稿では、Linux の基本的なネットワークおよび接続情報に関するコマンドである **ss** と **ip** を中心に、実運用や調査における活用法を網羅的に取り上げました。

次回は、プロセス・ファイル監視編として、**ps, lsof ,date** を確認します。

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