HITACHI

Hitachi Systems Security Journal

VOL.15

株式会社 日立システムズ

Hitachi Systems Security Journal VOL.15

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

●ご利用条件

本文書内の文章等すべての情報掲載に当たりまして、株式会社日立システムズ（以下、「当社」といいます。）といたしましても細心の注意を払っておりますが、その内容に誤りや欠陥があった場合にも、いかなる保証もするものではありません。本文書をご利用いただいたことにより生じた損害につきましても、当社は一切責任を負いかねます。
本文書に記載した会社名・製品名は各社の商標または登録商標です。
本文書に掲載されている情報は、掲載した時点のものです。掲載した時点以降に変更される場合もありますので、あらかじめご了承ください。
本文書の一部または全部を著作権法が定める範囲を超えて複製・転載することを禁じます。

© Hitachi Systems, Ltd. 2025. All rights reserved.

「AI は最高のハッカーを超えられるか ?」
問いを変えれば、未来が見える !!

デイビッド・ブラムリー インタビュー

2016 年、DARPA 完全自動サイバー防御競技会 (CGC) で優勝した自律システム「Mayhem」。開発を率いたカーネギーメロン大学のデビッド・ブラムリー教授は、技術の商業化に挑み、学術界と産業界の狭間で奮闘してきた。「AI は世界最高のハッカーを打ち負かせるか ?」—CODE BLUE 2025 の基調講演に登壇した教授は「それは間違った質問だ」と答える。真の問いは「実社会で AI ハッキングを信頼できるか」であり、その答えは失敗をどう扱うかにあるという。ソフトウェアセキュリティの第一人者が、20 年以上の研究と実践から導き出した、AI が進むべき道とは。

取材・文 = 斉藤 健一／通訳 = エル・ケンタロウ／撮影 = 松沢 雅彦

「AIは世界最高のハッカーを打ち負かせるか？」という問いの意味

斉藤（以下 S ）：基調講演を拝見しました。こちらの理解は断片的なものになりますが、高度なコンピューター科学の話題を、わかりやすく話してくださっていたのではないかと思っています。

デイビッド・ブラムリー（以下 D ）：「AIは世界最高のハッカーを打ち負かせるか?」私はこの質問をよく受けます。講演では、このようにさまざまな形で注目される複雑な話題を、より簡単に説明したいと考えています。

S 講演では最高のハッカーの1人としてGeoHot(ジョージ・ホッツ）氏※1を紹介し、彼のハッキング実績の1つとして、初代iPhoneのジェイルブレイクを取り上げました。

D 初代iPhoneが登場したのは2007年のことです。デバイスとして非常に洗練されていますが、米国では通信キャリアがAT&Tに限定されていました。当時の彼は17歳。iPhoneを使いたいけれどキャリアは変更したくない。そこで、彼はiPhoneの方を変えることにしたのです。

S はい。当時のことはよく覚えています。ジェイルブレイクに成功した彼は、自身のブログやYouTubeなどで公表していました。

D GeoHotは状況を逆転させました。ハッカーの行動をとったのです。彼は、機械の仕様に従うのではなく、機械を自分の意図どおりに動かしたのです。この視点に立つと、「AIは世界最高のハッカーを打ち負かせるか?」という質問は、私には間違っているように思えるのです。仮にAIがGeoHotをハッキングで打ち負かしたとしても、本来の質問は「実社会でAIによるハッキングを信頼できるか?」になると思います。

S 質問が間違っているというのは興味深い洞察です。もう少し具体的に説明いただけますか。

D 仮にAIが世界最高のハッカーと同等の能力を持っていたとしても、40%の確率で誤った情報を出すとわかっていて、果たしてそれでも信頼できるか? という話です。能力は同等かもしれませんが、AIは頻繁に嘘をつくのです。

S なるほど。より理解しやすくなりました。

AIを改善する3つのループ

S 講演では、ご自身の経験を交えながら、AIを改善するための方策を示されていました。

D 何十年もこれに取り組んできた後、実社会でAIハッキングを信頼できるようにするために正しく理解すべき3つの核心があると思います。私はこれをループベースの思考と呼んでいます。正しく理解すべき3つのことがあります。

S はい。順に説明をお願いします。

D 1つ目は自律性ループです。AIセキュリティシステムの中心となるモデルであり、テスト、失敗、学

●デイビッド・ブラムリー（David Brumley）
Mayhem.Security（旧称：ForAllSecure）CEO、Bugcrowdチーフ AI・サイエンス・オフィサー、カーネギーメロン大学教授を務めるアプリケーションセキュリティの第一人者。2012年カーネギーメロン大学のCTFチームである「PPP」のメンバーらとForAllSecureを起業。2016年には、DARPA（米国国防高等研究計画局）が主催したCyber Grand Challengeで優勝を果たす。次世代の育成にも情熱を注いでおり、無償のコンピューターセキュリティ教育プログラム「picoCTF」を立ち上げている。

※1 **GeoHot（ジョージ・ホッツ）：**
https://ja.wikipedia.org/wiki/%E3%82%B8%E3%83%A7%E3%83%BC%E3%82%B8%E3%83%BB%E3%83%9B%E3%83%83%E3%83%84

習、再テスト、そしてそれを何度もループして繰り返すのです。

S 2016 年に DARPA が主催した Cyber Grand Challenge（CGC）※2 では、デイビッドさんが率いるカーネギーメロン大学チームの Mayhem が優勝を飾りました。講演では、自律性ループの重要性は、この大会での経験に起因しているとのことですが。

D Mayhem が競技で優勝できた決定的な理由は、攻撃力や防御力ではなく、セキュリティパッチが実際の運用環境で問題なく機能することを保証する品質保証能力にありました。場合によっては、ヘタなセキュリティパッチを当てるよりも、パッチを適用しない方がよいという判断ができたのです。つまり、自律的なセキュリティには自律的なテストが不可欠であり、システムが適切に動作するかという展開の最大の障壁をクリアしたことが大きな理由になっていると考えています。

S 自律性ループが CGC の勝因にもつながっていると伺い、とても興味深いと思いました。

D 2 つ目が採用ループです。これは、技術を顧客の実際のニーズと価値観に合わせて調整し、導入可能にするプロセスです。

S 採用ループに関しては、航空宇宙会社での事例が紹介されていました。

D はい。航空宇宙会社で Mayhem を展開しようとした時の初期の失敗エピソードです。私たちは顧客に対して技術主導で適応を要求するなど、セキュリティの専門家視点で価値を提案していました。ですが、実際には、技術の優位性だけでは不十分で、顧客の声に耳を傾け、顧客の価値基準に合わせて製品の価値を再定義する柔軟性が採用の成否を分けるのです。

S セキュリティに限らず、すべての製品・サービスに通じる話ですね。

D 航空宇宙会社の例では、セキュリティを「航空宇宙会社のアセンブリの 80% のコードをカバー」と具体的に言及したことがきっかけとなり、両社の関係が好転しはじめたのです。

S 顧客の言語で価値を語ることの重要性ですね。

CODE BLUE 2025 でのブラムリー氏の講演動画やプレゼンテーション資料は、2026 年 1 月以降に公式サイトにアーカイブされる予定

また、講演では顧客がいる場所で会うことの重要性にも言及されていました。

D 3 つ目は、改善ループです。これは、人間と AI が意図的な練習を通じて継続的にスキルを向上させる学習サイクルです。意図的な練習には「1. 特定のスキルに焦点を当てること（明確な学習対象）」「2. 特定の目標設定（達成すべき具体的なゴール）」「3. タイムリーなフィードバック（即座の結果確認）」「4. 繰り返し（反復による定着）」「5. 難易度の段階的な上昇（より高難易度問題へ）」という 5 つの要素があります。

S 講演でこの部分を聞いていて、まさしく人間のハッカーがスキルアップしていく過程と同じだと感じました。

D そのとおりです。そして、これらの 5 つの要素すべてを提供する理想的な訓練環境が CTF 競技です。人間にも AI にも同じ学習フレームワークを適用できます。熟達は天才的な指導者からではなく、構造化された反復練習から生まれるのです。

S AI と CTF の両方に精通しているデイビッドさんならではの洞察だと思います。

D 今回の講演の根底には「失敗」からどのように学ぶかというテーマがあります。CGC で Mayhem

※ 2 **Cyber Grand Challenge（CGC）**：米国防高等研究計画局（DARPA）が 2016 年に開催した世界初の完全自動サイバー防御競技会。AI システムが人間の介入なしにソフトウアのぜい弱性を自動検出・修正し、ネットワーク防御能力を競った。
https://www.darpa.mil/research/programs/cyber-grand-challenge

ブラムリー氏によれば、Mayhem が CGC で優勝できた理由は、攻撃力や防御力ではなく、セキュリティパッチを自律的にテストする品質管理能力にあったという

が勝利できたのは、ヘタな修正パッチを当てるよりも修正パッチを適用しない方がよいという判断ができたからです。企業への採用では、初期の失敗を認め、それを乗り越えることが重要でした。また、CTF においては、失敗を活用して学習することが不可欠です。

S なるほど。これまで伺った 3 つのループには「失敗」というキーワードが共通していたのですね。

D 実世界で AI ハッキングを信頼するためには、失敗を可視化し、制御し、私たちの目標と整合させる必要があるのです。

ソフトウェアのバグをなくすという挑戦

S 今回のインタビューにあたって、デイビッドさんのプロフィールを調べていたのですが、カーネギーメロン大学の資料に「ソフトウェアのバグをこの世からなくしていきたい」というビジョンが示されていました。これまで伺ってきた講演の話題と重複する部分もあるかもしれませんが、この目標に向けて、どのような道のりを歩んでこられたのでしょうか。

D 私たちカーネギーメロン大学の研究チームの目標は、プログラムのバグを自動的に発見・検証し、それを世界規模で実現することでした。ただ、研究を進める中で大きな課題に直面したのです。

S どのような課題だったのですか？

D バグ発見に関する研究は多数存在しますが、非常にノイズが多いという問題です。SBOM や静的解析といった既存ツールでは、誤検出率が約 50% に達します。つまり、発見されたバグの半分は実際には問題ではないということです。

S それでは実用性に欠けますね。

D そのとおりです。そこで私たちが立てた中心的な問いは、「ハッカーのようにバグを発見し、証明することはできるか？」でした。ハッカーはエクスプロイトによってバグの証明を行います。これが、私たちが数十年にわたって取り組んできたテーマなのです。

S なるほど。その研究はどのように進められたのでしょうか。

D 約 10 年間、トップレベルのセキュリティカンファレンスで論文を発表し続けてきました。私はそれを「ギリシャ文字の羅列」と呼んでいます。数式ばかりで、厳密な科学的手法と大量の実験を行っていました。DARPA CGC で私たちがワクワクしたのは、純粋に学術的な枠組みを超えて、公正な評価を受けられるチャンスだったということです。

S CGC での優勝後、そのまま大学に戻って研究を続けるという選択肢もあったと思いますが。

D まさにそうです。論文を書き続けるという選択肢もありました。しかし、私がより興味を持ったのは、研究された技術やスキルをいかに市場に持っていくかということでした。

S それは研究者としては大きな転換点ですね。

D 日本でもおそらく同様だと思いますが、学会や学術界で議論される技術的な内容が、結果的に市場に出ることなく、研究者の中だけで生き続けるケースが多くあります。そうではなく、それをいかに市場に向けて、日々の生活の中に溶け込むような形で紹介できるかということに非常に興味を持ったのです。

S 研究成果を実社会に届けることへの強い思いが、Mayhem の商業化につながったわけですね。

D そのとおりです。学術的な厳密性を保ちながら、実用的な価値を提供する。それが私たちのめざした道であり、私の講演の核心でもあります。

学術界と市場の狭間で

S 研究内容を市場に持って行ったときの、学術界での反応はいかがでしたか。技術移転に対して何か反応などはありましたか。

D 学術界は最先端の技術を追求することを重視しています。ただ、実用化の課題については「それは単なるエンジニアリングの問題だ」と、やや軽く扱われる傾向があります。

S 実用化よりも理論が優先されるということですね。

D ただ、これはある意味で自己防衛なのかもしれません。「なぜ実社会で受け入れられないのか」という本質的な問いに向き合うことを避けているように感じます。科学的な内容と同じくらい、この問いは重要だと思うのです。

S Mayhem が市場での実績を積み重ねる中で、学術界の見方に変化はありましたか？

D 残念ながら、大きな変化は見られません。学術界では論文を書くことが評価の中心で、実用化への取り組みが報われにくい構造があります。

S なるほど。評価の仕組みの問題なのですね。

D 興味深いのは、産業界にも似た課題があることです。産業界は「学術界は実践的な問題に向き合っていない」と感じ、学術界は「解決策はあるが、まだ実装されていないだけだ」と考えています。

S 互いに理解が不足しているということでしょうか。

D 両者がもう少し歩み寄れればと思います。学術界が実装の大切さを認め、産業界が基礎研究の価値を理解する。そうした相互理解があってこそ、真の技術革新が生まれると信じています。

AI × CC への関与

S AI × CC（AI Cyber Challenge）[※3] に関わるきっかけは何だったのでしょうか？

D AI × CC は CGC とは異なる課題でしたが、私たちが CGC で優勝していたこともあり、専門家として招かれました。

実世界で AI のハッキングを信頼するためには、失敗を可視化し、制御し、私たちの目標と整合させる必要がある、とブラムリー氏は強調する

S 具体的にはどのような役割を担われたのですか？

D 主にスコアリングシステムの設計を担当しました。AI × CC は防御に特化しており、CGC のように攻撃と防御の全領域を扱うものではありませんでした。

S スコアリングシステムで特に重視された点は何でしょうか？

D 重要な課題の 1 つが、パッチの品質評価でした。コンピューターにぜい弱性のパッチを作成させると、オープンソースでは多くの場合、そのパッチが人間によって却下されます。なぜなら、編集箇所が多すぎて、変更が大きすぎるからです。

S もう少し具体的に説明いただけますか。

D 例えるなら、記者が AI に記事の編集を頼んだら、全文書き直されてしまったようなものです。必要最小限の修正ではなく、過剰な変更をしてしまうのです。そこで私たちは、AI を正しい方向に導くためのスコアリングアルゴリズムの設計を支援しました。

S AI × CC に関わった期間はどれくらいですか？

※3 **AI × CC（AI Cyber Challenge）**：DARPA と ARPA-H が主催する、重要インフラやオープンソースソフトウェアを保護する革新的な AI システム開発競技会。準決勝は 2024 年、2025 年の決勝戦は DEF CON 33 の会場で開催された。
https://www.darpa.mil/research/programs/ai-cyber

2025年11月、Bugcrowdによる Mayhem.Securityの買収のニュースが報じられた。ニューヨークにあるナスダック本社ビルのディスプレイには買収を歓迎するコメントが表示された
（写真提供：デイビッド・プラムリー氏）

D プロジェクト自体が短期集中型で、研究内容が中心となっています。今回のスコアリングシステムに関しては、約 1 年間かけて設計しました。

S 今回の CODE BLUE ではスペシャルセッションとして、AIxCC の上位入賞チームのメンバーによるパネルディスカッションが行われる予定です。どのような内容になりそうでしょうか。また、このセッションが実現した経緯についても教えてください。

D コミュニティが非常に小さく、みんなお互いに知り合いなので、自然と「一緒にやろう」という話になりました。AI × CC の運営として、参加者たちにスポットライトを当てることは非常に重要だと考えています。そうすることで、次世代が育つための道しるべを示すことができると思っています。

Bugcrowd との統合による新たな可能性

S インタビューの数日前、Mayhem Security が Bugcrowd に買収されたというニュースが流れてきました。Bugcrowd の一員となることで、これまで取り組んでこられたことが、どこまで世界を広げることができるのか教えていただけますか。

D 今回の Bugcrowd による買収は、私たちにとって非常に魅力的なものでした。その理由は、私たちは優れたツールを持っていますが、顧客側には、それを十分に活用できるほどの成熟度や人財リソースが必ずしもそろっていないという課題があったからです。

S ツールと人財を組み合わせることで、より実践的なソリューションになるということですね。

D はい。そして重要なのは、私たちは決して AI がエンジニアやハッカーを置き換えるとは考えていないということです。むしろ、優秀なハッカーをより強力にするための支援ツールとして AI を位置づけています。Bugcrowd との統合により、世界中の優秀なハッカーたちをサポートできるようになると考えています。

S 本日は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 2026.01

文＝日立システムズ

ランサムウェア事案にみるレジリエンスの重要性

【概要】：2025 年、複数の組織がランサムウェア攻撃を受け、事業への影響が長期化し、サプライチェーンへも波及する事案が相次いだ。被害企業の公表内容から、従来の侵入防御対策では進化するサイバー攻撃への対応が困難であることが改めてわかる。そのため、被害を想定し、影響を最小限に抑え、事業を迅速に回復する「レジリエンス」の向上が必要となる。その施策の 1 つとして、BCP の実効性を確認する訓練を実施し、迅速な事業回復体制を整備しておくことが重要である。

【内容】：多くの組織では、アセスメント、EDR 導入、ペネトレーションテストなどの侵入防御中心の対策を実施している。しかし、昨今のランサムウェア攻撃事案から、従来対策では進化するサイバー攻撃への対応が困難であることが改めてわかる。当社のインシデント対応サービスで対応した事犯でも同様の状況が見て取れる。

このような状況において、侵入防御中心の対策だけでなく、被害を前提に影響を最小限に抑え、事業を迅速に回復する能力「レジリエンス」の向上が急務となっている。この「レジリエンス」向上に資する活動として BCM（事業継続マネジメント）があり、この中で作成するのが、いつ、誰が、何を、どうするかを定めた具体的な計画書である BCP（事業継続計画）である。BCM の実効性を高めるには、BCP を事前作成することに加え、BCP を利用した定期的な訓練が必要となる。ランサムウェア事犯を想定した訓練としては、経営陣を含む事業継続を目的とした訓練が考えられる。例えば、ランサムウェアの被害によりシステムでの受注・出荷業務ができない場合、応急復旧として手作業で受注・出荷を進め、サプライチェーンへの影響を軽減する。この応急復旧には経営陣による判断が必要となる。BCP の実効性を確認するため、事前に訓練を実施し、迅速な経営判断を行える体制へと見直すことが重要となる。技術的な訓練としては、バックアップの正常性確認が重要である。

バックアップは計画だけでなく、手順の正確性や再現性を実機で確認しておかないと、実際の場面で失敗し、システム停止が長期化する可能性がある。特に複数システムが相互連携する環境では、個々のバックアップが正常に復元できても、データの不整合、旧バージョン依存によるエラー、時刻同期の不一致による認証障害など、復旧後に問題が生じやすい。さらに、ランサムウェア被害ではバックアップ自体の改ざん・汚染の有無や、攻撃者による不正プログラムが仕込まれていないことを確認する安全性確認も不可欠である。このため、復旧は本番環境に戻す前に別環境でシステムの安全性確認を行う必要があり、システム復旧に時間を要する可能性がある。迅速なシステム復旧には「ゴールデンイメージ」を準備・定期更新し、データバックアップに頼らない再構築手段も並行して検討しておく必要がある。この考え方は、米 CISA（Cybersecurity and Infrastructure Security Agency）が公開するランサムウェアガイドにも記載されている※ 1。

昨今のランサムウェア事案は、従来の侵入防御中心の対策だけではサイバー攻撃を完全に防げない現実を示し、企業が「レジリエンス」を向上する必要性を再認識させた。「レジリエンス」向上施策の 1 つとして、年 1 回程度、想定シナリオに基づく全社的な訓練・演習を継続的に実施し、浮き彫りになった課題をもとに BCP を見直すことが重要となる。また、経営層が訓練に参加することで、インシデント発生時の意思決定者と方法を明確にしておくことが重要である。加えて、外部の専門機関との連携方法や広報対応手順を事前に準備することで、実際のインシデント発生時の混乱を最小限に抑えられる。

［情報源］ ※ 1 https://www.cisa.gov/stopransomware/ransomware-guide

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

1. 環境構築編

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「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・分析する方法を整理します。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**

本稿「プロキシサーバーログ調査 1. 環境構築編」では、プロキシサーバーについての解説を行い、続編のログ分析に備えて VirtualBox を用いたプロキシサーバー、クライアントの環境構築を行います。

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. プロキシサーバーと Squid の概要

2.1 プロキシサーバー

プロキシサーバーは情報の受信者と送信者の間で通信を中継するサーバーです。プロキシサーバーにはフォワードプロキシとリバースプロキシという 2 つの用途があります。

フォワードプロキシとしてのプロキシサーバーは、クライアント（利用者）側のネットワークに配置されクライアントの代理として外部のサーバーにアクセスを行います。クライアントはプロキシサーバーにリクエストを送り、プロキシサーバーが外部の Web サーバーなどへ通信して得たレスポンスをクライアントに返却します。

図 1 フォワードプロキシの例

フォワードプロキシにより以下のようなメリットがあります。

• セキュリティ向上：

プロキシサーバーでアクセス制御や有害サイトのブロックを行い、内部ネットワークのクライアントを保護できます。また、プロキシ経由に限定しインターネットへの直接のアクセスを防ぐことで、ログ監査が行いやすくなります。

• アクセス速度向上：

一度取得した Web コンテンツをプロキシサーバーがキャッシュすることで、同じデータへの再アクセス時に高速な応答が可能となり、インターネット回線の帯域節約に寄与します。

• 匿名性の確保：

プロキシサーバーを利用することにより、クライアントの IP アドレスが隠ぺいされ、外部にはプロキシサーバーの情報のみが見えるため、クライアントの匿名性が高まります。

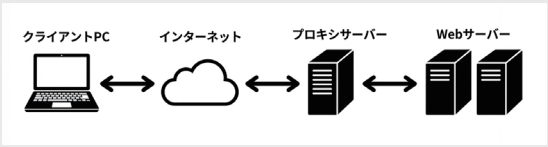


図 2 リバースプロキシの例

リバースプロキシにより以下のようなメリットがあります。

• セキュリティ向上：

インターネットから Web サーバーを隠蔽し、不正アクセスや DDos 攻撃 (分散型サービス拒否攻撃) に晒されるリスクが減少します。また、クライアントからリクエストを受け取った際にクライアントには元の URL が表示されたまま、バックエンドサーバーには URL を書き換えて送信することができます (URL リライト)。この仕組みにより、内部構造を秘匿にできます。

• 負荷分散：

クライアントからのアクセスを複数のサーバーへ分散させてシステムの安定性とパフォーマンスを向上させます。

• キャッシュ機能：

Web サーバーからの応答を保存して代理で応答することで、クライアントからの要求への応答速度の向上および Web サーバーの負荷軽減が図れます。

• SSL 終端：

リバースプロキシで暗号化された HTTPS 通信を復号し、バックエンドとは HTTP 通信を行うことで、バックエンドサーバーの負荷を減らします。

また、プロキシサーバーが配置されている内部ネットワークに攻撃者が侵入した場合、攻撃者がインターネットへ通信を行う際にはプロキシサーバーを経由する必要があります。そのため、プロキシサーバーのログを調査することで、不審な IP、URL への通信やクラウドサービスへのアクセスがわかる可能性があります。

2.2 Squid

プロキシサーバーの実装としては Squid が代表的です。Squid は Linux 上で動作する高性能な OSS で、フォワードプロキシやリバースプロキシとして利用できます。Squid は HTTP/HTTPS や FTP などのプロトコルに対応し、きめ細かなアクセス制御（ACL）や豊富なログ機能を備えています。企業や組織での Web アクセス管理にも広く使われており、今回の環境構築でもこの Squid を使用します。

3. 準備

3.1 CentOS の準備

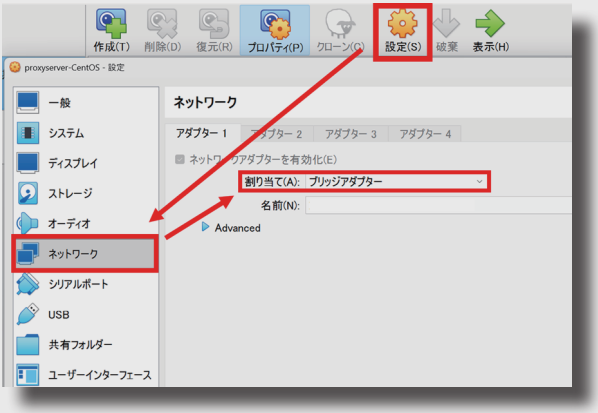
今回、検証で利用する CentOS を準備します。

CentOS は、「Vol.50 Let's try HDD 保全！ 1. 準備編」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（スナップショット機能を活用してください）。
また、はじめて作成する方は、本誌バックナンバー※を参考に CentOS の準備をお願いします。
なお、本稿では、仮想マシン（以下、「VM」）を「proxyserver-CentOS」と呼称します。

※ Vol.50　https://www.hitachi-systems.com/-/media/report/specialist/hj/download/2023_hj50.pdf

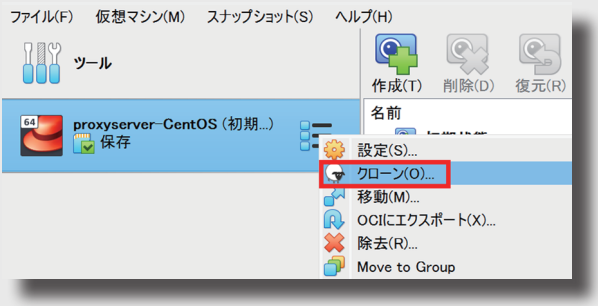
3.2 CentOS のネットワーク接続

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、「ブリッジアダプター」となっていることを確認します。



3.3 クローンの作成

プロキシサーバー「proxyserver-CentOS」を基として、クライアント「client-CentOS」を作成します。「proxyserver-CentOS」のスナップショットを右クリックし、「クローン」を選択します。



クローン作成のナビゲーションが表示されます。名前に「client-CentOS」と入力し、MAC Address Policy で「すべてのネットワークアダプターで MAC アドレスを生成」を選択します。その他は初期値でかまいませんので「次へ」を随時選択しクローンを作成します。

クローン作成後、3.2 と同様の手順で「client-CentOS」のネットワーク設定が「ブリッジアダプター」となっていることを確認します。

3.4 ホスト名の設定

2 台の CentOS を利用するにあたり、画面上でどちらの VM を使用しているかを明確にするために、ホスト名を変更しておきます。

「proxyserver-CentOS」を起動し次のコマンドを実行します。

```
# hostnamectl set-hostname Proxy
# exit
```

再度ログインし、以下のとおりホスト名が変更されていれば完了です。

```
[root@Proxy ~]#
```

同様に、「client-CentOS」を起動し次のコマンドを実行します。

```
# hostnamectl set-hostname Client
# exit
```

再度ログインし、以下のとおりホスト名が変更されていれば完了です。

```
[root@Client ~]#
```

以降の説明でコマンド操作を示す際は、操作する VM を [VM:Proxy]、[VM:Client] のようにホスト名で明示します。

3.5 ネットワーク設定

検証を行うにあたり、IP アドレスの設定および通信テストを行います。
[VM:Proxy,Client] 両方で次のコマンドを実行し、IP アドレスを確認します。

```
# ip a
```

以下が実行結果です。

```
[root@Proxy ~]# 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 08:00:27:82:40:dd brd ff:ff:ff:ff:ff:ff
    inet 172.20.10.11/28 brd 172.20.10.15 scope global dynamic noprefixroute enp0s3
       valid_lft 3207sec preferred_lft 3207sec
    inet6 240a:61:4150:1a87:a00:27ff:fe82:40dd/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe82:40dd/64 scope link noprefixroute
```

実行結果の中の赤枠の部分が割り当てられている IP アドレスです。
本稿では説明を明確にするために、以下の IP アドレスを追加します。

・プロキシサーバー [VM:Proxy]：192.168.10.100
・クライアント [VM:Client]：192.168.10.101

[VM:Proxy] で次のコマンドを実行し、プロキシサーバー用 VM に IP アドレスを追加します。

```
# ip addr add 192.168.10.100/24 dev enp0s3
```

再度「ip a」コマンドを実行し、以下の出力結果のとおり、IP アドレス 192.168.10.100 が追加されていることを確認します。

```
[root@Proxy ~]# ip addr add 192.168.10.100/24 dev enp0s3
[root@Proxy ~]# 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 08:00:27:82:40:dd brd ff:ff:ff:ff:ff:ff
    inet 172.20.10.11/28 brd 172.20.10.15 scope global dynamic noprefixroute enp0s3
       valid_lft 3463sec preferred_lft 3463sec
    inet 192.168.10.100/24 scope global enp0s3
       valid_lft forever preferred_lft forever
```

[VM:Client] で次のコマンドを実行し、クライアント用 VM に IP アドレスを追加します。

```
# ip addr add 192.168.10.101/24 dev enp0s3
```

再度「ip a」コマンドを実行し、以下の出力結果のとおり、IP アドレス 192.168.10.101 が追加されていることを確認します。

```
[root@Client ~]# 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 08:00:27:82:40:dd brd ff:ff:ff:ff:ff:ff
    inet 172.20.10.11/28 brd 172.20.10.15 scope global dynamic noprefixroute enp0s3
       valid_lft 2507sec preferred_lft 2507sec
    inet 192.168.10.101/24 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 240a:61:4150:1a87:a00:27ff:fe82:40dd/64 scope global dadfailed tentative noprefixroute
       valid_lft forever preferred_lft forever
```

「ip addr add」コマンドで追加した IP アドレスは再起動すると元に戻ります。

次に、両方の VM を起動した状態で互いの VM に向けて「ping」を実行し、VM 間の疎通確認を行います。
[VM:Client] で次のコマンドを実行します。

```
# ping -c 3 192.168.10.100
```

以下のような結果 (以下、「正常 ping 応答結果」) が出力されていれば VM 間の通信が正しく行われています。

```
[root@Client ~]# ping -c 3 192.168.10.100
PING 192.168.10.100 (192.168.10.100) 56(84) bytes of data.
64 バイト応答 送信元 192.168.10.100: icmp_seq=1 ttl=64 時間=1.17ミリ秒
64 バイト応答 送信元 192.168.10.100: icmp_seq=2 ttl=64 時間=2.21ミリ秒
64 バイト応答 送信元 192.168.10.100: icmp_seq=3 ttl=64 時間=0.922ミリ秒

--- 192.168.10.100 ping 統計 ---
送信パケット数 3, 受信パケット数 3, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.922/1.433/2.205/0.555 ms
```

次に、両方の VM で「www.google.com」への「ping」を実行し、VM からインターネットへの疎通確認を行います。
[VM:Proxy、Client] 両方で次のコマンドを実行します。

```
# ping -c 3 www.google.com
```

両方の VM で以下のような正常 ping 応答結果が出力されていれば、インターネットへの接続が正しく行われています。

```
[root@Client ~]# ping -c 3 www.google.com
PING www.google.com (142.250.207.36) 56(84) bytes of data.
64 バイト応答 送信元 nrt13s55-in-f4.1e100.net (142.250.207.36): icmp_seq=1 ttl=109 時間=63.0ミリ秒
64 バイト応答 送信元 nrt13s55-in-f4.1e100.net (142.250.207.36): icmp_seq=2 ttl=109 時間=60.5ミリ秒
64 バイト応答 送信元 nrt13s55-in-f4.1e100.net (142.250.207.36): icmp_seq=3 ttl=109 時間=44.6ミリ秒

--- www.google.com ping 統計 ---
送信パケット数 3, 受信パケット数 3, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 44.607/56.027/62.992/8.139 ms
```

以上の手順より、プロキシサーバー側の VM とクライアント側の VM が互いに通信でき、かつインターネットとも通信できる環境の構築が完了しました。

参考：正常 ping 応答結果が出力されなかった場合
原因の切り分けのため、[VM:Proxy、Client] 両方で次のコマンドを実行します。

```
# ping -c 3 8.8.8.8
```

・正常 ping 応答結果が出力された場合
IP アドレスを指定したインターネットへの接続は行うことができています。
DNS による名前解決が正しく機能していない可能性が考えられますので、CentOS の DNS 設定を確認してください。

・正常 ping 応答結果が出力されなかった場合
インターネットへの接続が正しく行われていない可能性があります。
再度、3.2. 節を参照しながら VirtualBox の設定 (ブリッジアダプター設定等) を確認してください。
それでも解決しなかった場合は、ホスト OS(使用している PC) 自体のネットワーク接続が正常に

行われていない可能性があるため、Web ブラウザーなどでネットワークに接続できるかを確認し、接続できない場合はホスト OS 自体のネットワーク設定を行ってください。

4. プロキシサーバーの設定

続いて、プロキシサーバー用 VM 上に Squid をインストールし、プロキシサーバーとして利用できるよう設定を行います。

4.1 Squid のインストールと起動

[VM:Proxy] で次のコマンドを実行し、Squid のパッケージをインストールします。

```
# yum –y install squid
```

インストールが成功すると、Squid の実行ファイルや設定ファイルがシステムに展開されます。
続いて、[VM:Proxy] で次のコマンドを実行し、Squid サービスを起動します。

```
# systemctl start squid
```

また、[VM:Proxy] で次のコマンドを実行し、Squid サービスの自動起動設定を行います。

```
# systemctl enable squid
```

以下のような実行結果となります。

```
[root@Proxy ~]# systemctl enable squid
Created symlink /etc/systemd/system/multi-user.target.wants/squid.service
→ /usr/lib/systemd/system/squid.service.
```

[VM:Proxy] で次のコマンドを実行し、Squid のサービスの状態を確認します。

```
# systemctl status squid
```

以下の出力結果の中で、赤枠の箇所が起動状態、黄枠の箇所が自動起動設定状態を表します。それぞれ、起動状態が「active (running)」であれば Squid が起動しており、自動起動設定状態が「enabled」であれば自動起動設定が有効になっています。

```
[root@Proxy ~]# systemctl status squid
● squid.service - Squid caching proxy
     Loaded: loaded (/usr/lib/systemd/system/squid.service; enabled; preset: disabled)
     Active: active (running) since Mon 2025-10-27 16:15:04 JST; 3h 15min ago
       Docs: man:squid(8)
   Main PID: 2762 (squid)
      Tasks: 3 (limit: 22974)
     Memory: 23.9M (peak: 24.1M)
        CPU: 1.491s
     CGroup: /system.slice/squid.service
             ├─2762 /usr/sbin/squid --foreground -f /etc/squid/squid.conf
             ├─2764 "(squid-1)" --kid squid-1 --foreground -f /etc/squid/squid.conf
             └─2765 "(logfile-daemon)" /var/log/squid/access.log
```

4.2 ポート開放

CentOS Stream 9 では「firewalld」(ファイアウォールサービス) がデフォルトで有効になっています。
Squid は標準では TCP 3128 番ポートで待ち受けを行うため、クライアントからプロキシサーバーに接続できるよう 3128 番ポートを明示的に開放する必要があります。

[VM:Proxy] で次の 2 つのコマンドを実行し、3128 番ポートへのアクセスを許可するルールを追加します。

```
# firewall-cmd --add-port=3128/tcp --permanent
# firewall-cmd --reload
```

各「firewall-cmd」コマンド実行後に「success」と出力されれば、設定が正常に反映されています。

[VM:Proxy] で次のコマンドを実行し、現在の zone(public) に開放されているポートの一覧を確認します。

```
# firewall-cmd --list-ports
```

以下のように「3128/tcp」という表示が確認できれば、3128 番ポートの開放は完了です。

```
[root@Proxy ~]# firewall-cmd --list-port
3128/tcp
```

参考：firewalld における zone

「firewalld」には「zone」という概念があり、ネットワークインターフェイスごとに異なるセキュリティポリシーを適用できます。
例えば、同じサーバーに複数のネットワークアダプター（LAN 用 , 管理用など）が存在する場合、それぞれに異なる zone(public,internal,trusted 等) を割り当てることで、通信の許可範囲を柔軟に制御できます。
zone の設定ミスに起因したトラブルを防ぐためにも、「firewall-cmd --get-active-zones」などのコマンドを利用し、意図した zone にインターフェイスが正しく割り当てられているかを確認することが重要です。

5. クライアントの設定および疎通確認

環境構築の仕上げとして、クライアントがプロキシサーバー経由で通信を行うよう設定を行い、プロキシサーバー経由での Web サイト接続を試みます。

クライアント側のプロキシ設定
現在の環境ではクライアントからプロキシサーバーを経由せずに外部に HTTP/HTTPS 通信ができてしまうため、クライアント側のファイアウォール (以下、「FW」) 設定によりプロキシ必須の環境を構築します。「firewalld」において OUTPUT 通信を制御する際は、ダイレクトルールで設定を行います。
[VM:Client] で次の 6 つのコマンドを実行し、ダイレクトルール設定の追加および反映を行います。

```
# firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 0 -m
conntrack
 --ctstate ESTABLISHED,RELATED -j ACCEPT
# firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 0 -o lo
-j ACCEPT
# firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 0 -p tcp
-d 192.168.10.100 --dport 3128 -j ACCEPT
# firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 1 -p tcp
--dport 80 -j REJECT
# firewall-cmd --permanent --direct --add-rule ipv4 filter OUTPUT 1 -p tcp
--dport 443 -j REJECT
# firewall-cmd --reload
```

各「firewall-cmd」コマンド実行後に「success」と出力されれば、設定が正常に反映されています。
[VM:Client] で次のコマンドを実行し、ダイレクトルール設定を確認します。

```
# firewall-cmd --direct --get-all-rules
```

以下の結果が出力されていれば、クライアントはプロキシサーバーを経由しない限り外部への HTTP/HTTPS 通信ができない設定となっています。

```
[root@Client ~]# firewall-cmd --direct --get-all-rules
ipv4 filter OUTPUT 0 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
ipv4 filter OUTPUT 0 -o lo -j ACCEPT
ipv4 filter OUTPUT 0 -p tcp -d 192.168.10.100 --dport 3128 -j ACCEPT
ipv4 filter OUTPUT 1 -p tcp --dport 80 -j REJECT
ipv4 filter OUTPUT 1 -p tcp --dport 443 -j REJECT
```

[VM:Client] で次のコマンドを実行し、Google サイトへの疎通確認を行います。

```
# curl --noproxy "*" www.google.com --head
```

以下のように、「接続がタイムアウトしました」または「Connection refused」と表示されていれば、プロキシサーバー経由でない通信を正常に FW で拒否することができています。

```
[root@Client ~]# curl --noproxy '*' www.google.com --head
curl: (28) Failed to connect to www.google.com port 80: 接続がタイムアウトしました
```

次に、クライアントからプロキシを利用するために、環境変数「http_proxy」の設定を行います。
[VM:Client] で次のコマンドを実行します。

```
# export http_proxy=http://192.168.10.100:3128
```

「192.168.10.100:3128」の部分は、プロキシサーバーの IP アドレスと待ち受けポート番号です。本環境変数を設定することで、curl、yum などのコマンドや、プロキシ対応のアプリケーションにおいて自動的に指定のプロキシ経由で通信を行うようになります。なお、本設定は現在のシェルセッションに対してのみ有効な設定となります。

5.2 プロキシ経由での通信テスト

実際にクライアントから外部サイトへアクセスし、プロキシ経由で通信が行われるかを確認します。
[VM:Client] で次のコマンドを実行し、Google サイトの HTTP ヘッダー情報を取得します。

```
# curl www.google.com --head
```

以下のように、HTTP レスポンスヘッダーの中に「HTTP/1.1 200 OK」「Via: 1.1 Proxy」と表示されていれば、プロキシサーバー経由で Google サイトへ接続できています。

```
[root@Client ~]# curl www.google.com --head
HTTP/1.1 200 OK
Content-Type: text/html; charset=ISO-8859-1
```

～～中略～～

```
X-Cache: MISS from Proxy
X-Cache-Lookup: MISS from Proxy:3128
Via: 1.1 Proxy (squid/5.5)
Connection: keep-alive
```

不要なポートはセキュリティ上開放しておくべきでないため、プロキシを長期間使用しない場合には、待ち受けとして設定した TCP 3128 番ポートを閉じることを推奨します。
[VM:Proxy] で次の 2 つのコマンドを実行します。

```
# firewall-cmd --remove-port=3128/tcp
# firewall-cmd --reload
```

各「firewall-cmd」コマンド実行後に「success」と出力されれば、設定が正常に反映されています。

[VM:Proxy] で次のコマンドを実行し、現在の zone(public) に開放されているポートの一覧を確認します。

```
# firewall-cmd --list-port
```

以下のように出力結果に「3128/tcp」という表示がなければ、3128 番ポートが閉じられています。

```
[root@Proxy ~]# firewall-cmd --list-port
3128/tcp
```

以上でプロキシサーバー利用環境の構築は完了となります。
続編に備えて、両方の VM のスナップショットを作成しておきましょう。

6. おわりに

今回はここまでとなります。

「1. 環境構築編」ではプロキシログを調査する前段階として、プロキシサーバーの構築手順を取り上げました。

今回、構築した環境では、「/var/log/squid/access.log」にプロキシアクセスログが記録されます。続編では、このログを用いてアクセス履歴を分析します。

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

※本カタログに記載されている会社名、製品名は、それぞれの会社の登録商標または商標です。
※本カタログに記載されている内容、仕様については、予告なく変更する場合があります。
※本製品を輸出する場合には、外国為替および外国貿易法ならびに、米国の輸出管理関連法規などの規制を御確認の上、必要な手続きをお取りください。なお、ご不明な場合は、当社営業にお問い合わせください。

Printed in Japan