IMES DISCUSSION PAPER SERIES

情報セキュリティ・シンポジウム（第23回）の模様：
オープン・ソース・ソフトウェアのセキュリティ

Discussion Paper No. 2023-J-9

IMES

INSTITUTE FOR MONETARY AND ECONOMIC STUDIES

BANK OF JAPAN

日本銀行金融研究所

〒103-8660 東京都中央区日本橋本石町 2-1-1

１．はじめに

日本銀行金融研究所・情報技術研究センター（Center for Information Technology Studies：CITECS）は、2023 年 3 月 3 日、「オープン・ソース・ソフトウェアのセキュリティ」をテーマとして、第 23 回情報セキュリティ・シンポジウムを開催した。

オープン・ソース・ソフトウェア（Open Source Software: OSS）とは、ソースコードが公開され、誰もが自由に利用や修正、再配布が可能なソフトウェアのことである。OSS は、通常、技術者らの自発的な協力によって開発・保守されている。また、一般的に OSS は、ばらつきはあるものの透明性の高い検証が積み重ねられていることから、高いソフトウェア品質が期待できる。さらに、OSS は、現在、金融分野に限らずあらゆる分野で、ソフトウェア開発に欠かせない存在となっている。近年では、機械学習（人工知能）をビジネスに活用する動きが拡がっているが、こうした機械学習においても OSS が使用されている。

OSS とそれ以外の商用ソフトウェア[1]等は、その本質においては同じである。もっとも、ソースコードの透明性、開発形態、ライセンスと利用の様態などにおいては、さまざまな違いがある。ユーザ企業が OSS を安全に活用していく際には、こうした OSS の特色を踏まえたセキュリティ対策を講じることや、ライフサイクルに応じた適切な脆弱性対応を実施することが望ましい。

こうした観点から、今次のシンポジウムでは、金融業界において OSS を安全に活用していくことを展望し、OSS を活用するメリットや脆弱性対応のあり方について、関連する分野の専門家や実務者を招き、講演とパネル・ディスカッションを行った。

当日は、情報セキュリティ技術に関心のある金融機関関係者、研究者、システム開発・運用に携わる技術者などを中心に約 150 名がオンラインで参加した。本稿では、次頁で示す当日のプログラムに沿って、講演およびパネル・ディスカッションの概要を順に紹介する（以下、敬称略、文責：日本銀行金融研究所）[2]。

---

[1] 本稿においては、「商用ソフトウェア」は、OSS 以外のソフトウェアのうち、企業が営利目的で開発し、そのソースコードが公表されていないものを指す。

[2] 文中の講演者やパネリストの所属と肩書きは、シンポジウム開催時点のものである。また、本稿に示された意見はすべて発言者たち個人に属し、その所属する組織の公式見解を示すものではない。また、本シンポジウムでの講演の資料等については、日本銀行金融研究所の当該サイト（https://www.imes.boj.or.jp/jp/conference/citecs/23sec_sympo. html）を参照されたい。

【第 23 回情報セキュリティ・シンポジウムのプログラム】

- 講演１「OSS の世界観とセキュリティ」
  一般社団法人　JPCERT コーディネーション・センター[3]　理事　真鍋敬士
- 講演２「OSS のセキュリティ～金融業界の視点から～」
  一般社団法人　金融 ISAC[4]　専務理事　鎌田敬介
- 講演３「機械学習と OSS のセキュリティ」
  神戸大学　教授　小澤誠一
- パネル・ディスカッション
  ・パネリスト：真鍋敬士、鎌田敬介、小澤誠一
  ・モデレータ：日本銀行金融研究所　企画役　菅　和聖

２．講演１「OSS の世界観とセキュリティ」

真鍋は、OSS の概念や起源、背後にある思想、社会における位置づけの変遷、脆弱性対応の流れについて、次のとおり発表した。

（１）OSS の概念と OSS コミュニティの文化

「オープン・ソース・ソフトウェア」という言葉は、オープン・ソース・イニシアティブ（Open Source Initiative: OSI）[5]が 1998 年頃に公表した宣言の中で規定された（Open Source Initiative [2007][6]）。この宣言では、以下に記載のとおり、OSS に求められる 10 個の要件が掲げられており、ソースコードやその派生物の取扱いに関する留意事項等が定められている[7]。

1. 再頒布の自由を認めること
2. ソースコードを同梱して頒布すること
3. 派生ソフトウェアの頒布を許可すること

---

[3] JPCERT（Japan Computer Emergency Response Team）コーディネーション・センターは、情報セキュリティ対策の支援（情報収集・分析、分析結果の公表、関係組織の調整等）を中立かつ技術的な立場から行う一般社団法人。

[4] 金融 ISAC は、日本の金融機関がサイバー・セキュリティに関する協働活動を行うことを目的とする一般社団法人。

[5] OSI は、オープン・ソースの定義とソフトウェア・ライセンスのレビューなどを通じて OSS のエコシステムの基盤を整備することを目的とする米国の非営利法人。1998 年に設立された。

[6] 2023 年 5 月の執筆時点で、バージョン 1.9（最終更新日は 2007 年）となっている。

[7] ここに記載した OSS に求められる 10 個の要件は日本銀行金融研究所による仮訳であり、正確な文言については、引用元である OSI のウェブ・ページを参照されたい。

4. 作者のコードの完全性を確保する場合に限り頒布を制限できること
5. 特定の個人やグループに対する差別の禁止
6. 特定の利用分野（fields of endeavor）に対する差別の禁止
7. 被頒布者に対するライセンスの分配（distribution）が平等であること
8. 特定製品のみに有効なライセンスの禁止
9. 他のソフトウェアを制限するライセンスの禁止
10. ライセンスが個別技術に依存せず技術中立（technology-neutral）であること

歴史を遡れば、OSS の定義は、「デビアン（Debian）社会契約」（Debian Project [1997]）という声明にルーツを持つ[8]。この声明では、OSS を単に「フリー・ソフトウェア（free software）」と呼んでいたが、その後、フリー・ソフトウェアの概念は、世間一般における社会・経済活動との親和性が高まるように徐々に調整されていった。その結果として整理されたものが今日の OSS の概念である。

OSS コミュニティの文化を端的に表す言葉は、1990 年代に流行した「UTSL（Use The Source, Luke）」[9]である。UTSL とは、ソフトウェアに関するさまざまな問題を解決するために、ソースコードをコミュニティ内で共有しながら、必要な都度活用しようという考え方である。

（２）ソフトウェアの社会的位置づけの歴史的変遷

20 世紀の人々は、ソフトウェアを「情報処理の知見が集まって具現化した収穫物」であり、社会全体の「資源」として捉えていた。そして、ソフトウェアそのものではなく、それを生み出す事業や経済活動を重視する考え方が主流であった。こうした考え方に立つと、商用ソフトウェアと OSS は、以下のような対立軸でとらえることができる。

商用ソフトウェアは、知的財産として、作製した企業によって独占的に利用され、当該企業によってメンテナンスされる。そのセキュリティについては、ソースコードが秘匿されることによって安全性が保たれている。これに対して、OSS は、誰もが利用可能な共有財産であり、エコシステム（OSS コミュニティ）によってメンテナンスされる。セキュリティ面については、公開されたソースコー

---

[8] デビアン社会契約は、デビアン（Debian）の開発を行うデビアン・プロジェクトが公表したものである。デビアンは Linux ディストリビューションの 1 つである。Linux は、世界で最も普及しているオープン・ソースのオペレーティング・システムであり、その枢要部分（Linux カーネル）に付随するさまざまな周辺ソフトウェアをパッケージにしたものが Linux ディストリビューションである。Debian は Software in the Public Interest, Inc. の登録商標である。Linux は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標である。

[9] 鎌田より、補足説明として、UTSL の由来は米国の有名な SF 映画のセリフの一部である「Use the force, Luke」であるとの解説があった。

ドが透明性の高い検証に耐えることによって安全性が保たれている。すなわち、「誰でも見ることができて、検証できるようにしてあるからこそ本当に安全である」との考え方に依拠している。

OSS のエコシステムでは、ソフトウェアを中核として、それに付随するさまざまな活動が展開される。こうした活動には、例えば、ソフトウェアの開発、販売、展開、サポート、およびこれらに関連するビジネスが含まれる。すべてのソフトウェアに対してすべてのコミュニティ・メンバーがアクセスすることができ、その結果としてソフトウェア・コミュニティが形成される。20 世紀の OSS コミュニティは、商用ソフトウェアと比べると比較的規模は小さかったが、自由な雰囲気が漂っていた。

21 世紀に入ると、ソフトウェアの位置づけが「社会の資源」から「社会インフラの担い手」へと様変わりし、OSS はその中核に位置するようになった。社会インフラは、さまざまなシステムが相互運用性を確保しながら構築される。このため、社会インフラには、ソフトウェアの仕様はもちろん、ソースコード自体もオープンであることが求められる。OSS は、こうした時代の要請に適うものであった。

（3）OSS を取り巻く状況

1998 年頃になると、国家レベルでの OSS への取組みが活発となった。日本では、特許庁が知的財産上のリスクなどに関する調査研究報告書（PwC コンサルティング合同会社［2020］）を公表した。また、OSS を利用する際の注意事項をまとめた「Opensource for ALL」（特許庁・内閣府［2019］）などの各種パンフレットも発刊された。経済産業省からは、OSS の利活用にかかる課題をまとめた報告書（経済産業省商務情報政策局サイバーセキュリティ課［2022］）が公表された。この報告書は、企業の経営者を主な対象としたものであり、法的リスクやレピュテーション・リスクが取り上げられている。

米国では、連邦政府が OSS に積極的にコミットする姿勢を示しており、OSS を積極的に活用していくという方針とその意義を「Federal Source Code Policy」（Office of Management and Budget, and Executive Office of the President [2016]）として公表している。そのうえで、自身で開発した OSS を Code.gov[10]というプラットフォーム上で公開する活動などを行っている。

（4）OSS の脆弱性

シノプシス社（Synopsys, Inc.）[11]という民間企業では、OSS のセキュリティ・

[10] https://code.gov/

[11] シノプシス社は、半導体の集積回路や電子機器の回路の設計を自動化するソフトウェア

リスクに関する分析レポート[12]を毎年公表している。2022 年の同社のレポートでは、2,400 個を超えるソフトウェアを対象とするコードベース[13]に関する調査結果が発表され、97%のソフトウェアに何らかの OSS が含まれていたとの報告がなされた。また、2,097 個のソフトウェアを対象としたセキュリティ面からの調査においては、約 8 割のソフトウェアが、過去 4 年間以上にわたり開発活動実績のない OSS を含んでいたことが判明した。さらに、調査対象の約 8 割のソフトウェアが、1 個以上の既知の OSS 脆弱性を含んでいたことも報告された。

一般論として、ソフトウェアの脆弱性には、仕様、設計、実装に関するものがあり、これらの脆弱性は、ソフトウェアにおける何らかの不備（バグ）によって生じる。OSS の場合、こうした脆弱性が、サプライ・チェーンの問題によってより深刻化する傾向にある。例えば、ハッシュ関数値を計算するソフトウェアにおいて、計算アルゴリズムの実装に何らかの脆弱性がある場合や、計算アルゴリズムそのものが危殆化する（安全性が保証されなくなる）場合には、そのソフトウェアを組み込んだすべてのシステムが危険に晒されることになる。

（５）脆弱性によるリスクへの対応

こうしたリスクへの対応策としては、ソフトウェアを常に安全な状態（最新版）に保つという、パッチ・マネジメントや脆弱性のライフサイクル・マネジメントが主流となっており、この 20 年間、こうした状況に大きな変化はない。

ただ、最近では、OSS のセキュリティ対策に有用な道具として、ソフトウェア構成管理表（Software Bill of Materials: SBOM、エスボム）が注目されている。SBOM は、ソフトウェアがどのようなパーツから成り立っているかを表形式で整理するものであり、2018 年に、米国商務省電気通信情報局（National Telecommunications and Information Administration: NTIA）のアラン・フリードマン（Allan Friedman）博士によって提唱された。いわゆる食品の成分表示のソフトウェア版ととらえるとイメージしやすいのではないか。NTIA では、SBOM をソフトウェアの透明性確保に向けた有効な手段の 1 つと位置づけたうえで、その活用方法を検討した。その結果、NTIA は、2021 年に、SBOM に最低限含めるべき情報を規定したガイダンスを発表した（National Telecommunications and

---

（Electronic Design Automation）等を開発および販売する企業。

[12] 2022 年の「オープン・ソース・セキュリティ&リスク分析（OSSRA：Open Source Security and Risk Analysis）レポート」の公表に関するニュースリリースが以下の URL で公表されている。
https://www.synopsys.com/ja-jp/japan/press-releases/2022-05-19.html

[13] コードベース（codebase）とは、ソフトウェアを構築するために使用されるソースコードの集まりを指す。ここでのソースコードは、一般的には、開発者の手によって直接書かれたものを指し、開発ツールによって自動生成されたものは含まない。

Information Administration [2021])。

（６）OSS におけるセキュリティ・インシデントの事例

OSS がさまざまなソフトウェアに組み込まれ、幅広く用いられるようになったことの当然の帰結として、OSS の脆弱性の影響範囲が広がっている。例えば、ウェブ・アプリケーション・フレームワークの Apache Struts[14]、ウェブ・ページを組み立てるフレームワークである WordPress[15]とその拡張機能、暗号化通信に用いられる OpenSSL[16]は、世界中のさまざまな場面で利用されている。これらのソフトウェアに脆弱性が発見されると、その影響は世界中に及ぶことになる。

Apache Struts の事例では、脆弱性に対するソースコードの修正内容（脆弱性情報）が開発者向けの公開サイトに誤って掲載された結果、その情報を糸口に攻撃が仕掛けられ、大量の個人情報が流出した。このように、OSS の修正対応が完了する前に脆弱性情報が公開されると、それが直ちに悪用される可能性がある。これは、OSS の透明性に起因するセキュリティ・リスクである。OSS については、開発情報は広く公開する一方、脆弱性情報は慎重に管理する必要がある。

（７）脆弱性対応のステップ

ソフトウェアの脆弱性対応には複数のステップがある（以下の図を参照）。最初のステップは、脆弱性の発見者がその情報を適切に開発者に伝えることである。もっとも、面識のない発見者からの報告を、開発者が信用するとは限らない。このため、JPCERT コーディネーション・センターや情報処理推進機構[17]がそうした情報の伝達や脆弱性対応の仲介を行っている。

脆弱性に関する情報が当該ソフトウェアの開発者に伝えられると、開発者は修正パッチを開発する。次に、修正パッチを当てたバージョンのソフトウェアがリリースされる。修正パッチ適用済みのソフトウェアがリリースされると、世間一般では「脆弱性が修正された」と認識されることが多い。

---

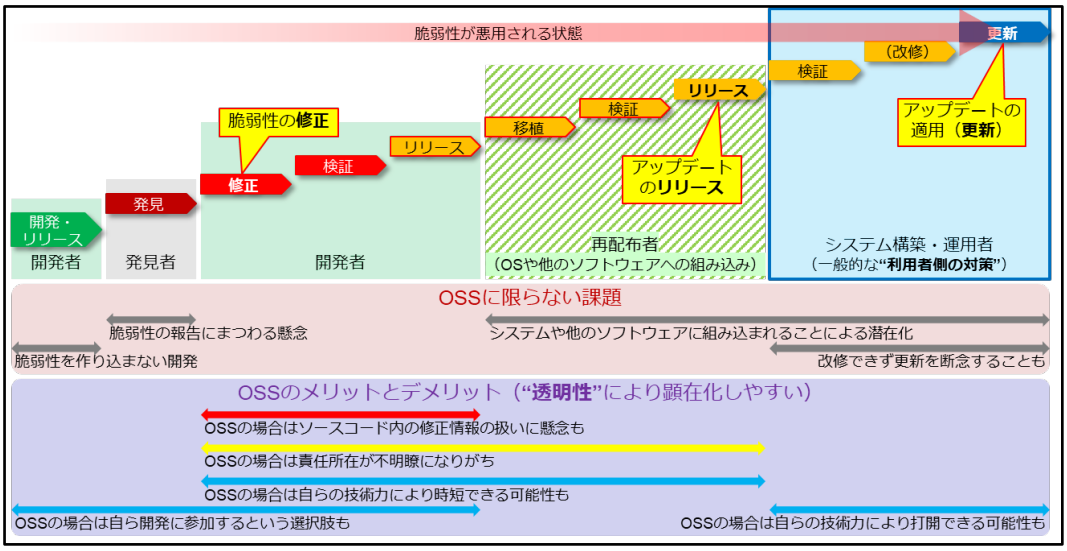
14 Apache および Apache Struts は、Apache Software Foundation の登録商標または商標である。https://struts.apache.org/

15 WordPress は、WordPress Foundation の登録商標である。https://wordpress.org/

16 OpenSSL は、米国およびその他の国における OpenSSL Software Foundation の登録商標である。https://www.openssl.org/

17 情報処理推進機構（IPA）は経済産業省の IT 政策を人材面と技術面から支えることを目的に設立された独立行政法人。

図．脆弱性対応のステップ



資料：真鍋氏の講演資料

ただ、実際には、修正パッチ作成の対象となったソフトウェアが他のソフトウェアに組み込まれているケースが多く、脆弱性対応は継続される。修正パッチ適用済みのソフトウェアを組み込んでも動作に問題がないかなどを検証し、問題がないことを確認してはじめて修正後のバージョンを取り込むこと（アップデートのリリース）になる。OSS の場合、重層的にソフトウェアが組み込まれていることが珍しくないため、検証とリリースが重層的に行われる。その過程で、システム・インテグレーター[18]が、リリースされたソフトウェアについて、ソフトウェア単体の動作と、それをシステムに移植した際のシステムの動作をそれぞれ検証する。

最終的には、システムのオーナーが、修正されたソフトウェアを自身のシステムに矛盾が生じないように組み込む責任を負う。ここまで完了してようやくシステムが脆弱性から解放されることとなる。脆弱性の発見からシステム対応の完了までには相応の期間を要するため、脆弱性が存在していたソフトウェアの修正版が正式に公開された後も、攻撃の被害がしばらくの間は続く可能性がある。ただし、このリスクは OSS に限った話ではなく、ソフトウェア一般に通底する。OSS に固有の特徴としては、技術力さえあれば、あるいは、その技術力を調達する資金力さえあれば、自力で修正して問題を回避するチャンスが生まれるということである。

18 システム・インテグレーターは、情報システムの企画、構築、運用などの業務をシステム・オーナーから一括して請け負う企業を指す。

３．講演２「OSS のセキュリティ～金融業界の視点から～」

鎌田は、金融 ISAC が中心となって実施している共助の取組みを中心に、金融機関における OSS の脆弱性対応について以下のとおり発表した。

（１）金融機関と OSS の関係

金融機関において OSS が話題にのぼる文脈は 2 つある。1 つ目は脆弱性対応において、広範囲に影響する致命的な脆弱性が発見された場合である。2 つ目は、OSS のツールを使ってログ解析やマルウェア解析などを行う場合である。ここでは、前者（脆弱性対応）に焦点を当てる。

まず、OSS と金融機関の関係について整理する。金融機関の多くは、システムの開発および提供をシステム・インテグレーターに委託している。現代において、OSS を一切使用せずにソフトウェア開発を実現できる企業は、システム・インテグレーターも含めて存在しない。こうした状況のもとでは、多くの金融機関においても、事実として OSS を利用せざるを得ないが、それを意識的あるいは積極的に利用しているケースは稀である。セキュリティ面をみても、OSS に特化した脆弱性対応を行っている金融機関は多くはない。しかし、金融機関が直面している脆弱性管理においては、OSS に特有の傾向や要素が現れている。

（２）脆弱性対応が難しい理由

一般論として、脆弱性対応の難度は高く、脆弱性に関する情報が公表されたときに、その脅威度を迅速に判断できる企業は、金融機関を含めて多くはない。これにはさまざまな理由がある。例えば、公に発信される情報の中に詳細な技術情報が含まれていない、情報が公表されるまでにタイムラグがある、公表情報に詳細な技術情報は含まれているが、企業側のセキュリティ担当者にその分野に詳しい人がいない、といった理由が挙げられる。

仮にこれらの課題がクリアされている場合であっても、そもそも脆弱性の脅威度判定が難しい理由として 3 点指摘できる。１つ目の理由は、共通脆弱性評価システム（Common Vulnerability Scoring System: CVSS）[19]の限界である。すなわち、CVSS のスコアは、技術的な脅威度に関する指標ではあるが、それが必ずしも金融機関にとっての脅威度を表すものではないということである。

２つ目の理由は、脅威度を正確に判定するには、脆弱性に加えて広範な情報技術に関する知見が要求されるという点である。例えば、ネットワーク、ウェブ・アプリケーション、データベースなどで使用されている OSS の脆弱性が発見さ

[19] CVSS は、情報システムの脆弱性に関するオープンで汎用的な評価手法である。CVSS を用いると、脆弱性の深刻度を同一の基準のもとで定量的に比較できる。

れた場合、セキュリティ面の知識に加えて、それぞれの分野の知識が要求される。企業のセキュリティ担当者が、ネットワークからデータベースまでの知識をすべてカバーするのは至難である。

3 つ目の理由は、環境的な要因が脅威度判定を難しくしている点である。深刻な脆弱性が現れたとき、企業は、自社サービスへの影響範囲を特定し、影響を受けるシステムが保有する情報や同システムを停止した場合の影響を把握する必要がある。その際、影響を受けるシステム単体ではなく、そのシステムが配置されている環境まで勘案することが重要である。例えば、影響を受けるシステムがイントラネット[20]上に配置されている場合、企業は、脅威度はさほど高くないとの判断を下す可能性が高い。しかし、最近では、自社が直接的に対応できないソフトウェアのサプライ・チェーンを通じてもたらされる脅威についても勘案しなければならない。こうした環境の変化や複雑化により、OSS に限らず脆弱性対応の難度は上がっている。

こうした中、金融 ISAC では、会員コミュニティの中でコレクティブ・インテリジェンス（collective intelligence）の取組みを行っている。すなわち、深刻な脆弱性情報が公表されると、まず、ある金融機関がその解釈や対応方法を提示し、これを受けて他の金融機関は、自社で行った検証結果をコミュニティ内に情報共有する。このように、複数の金融機関が互いの知見を重ね掛けするように持ち寄るかたちで、業界全体で情報分析を進めている。

（3）OSS の脆弱性の特徴

OSS の脆弱性は、商用ソフトウェアと比較して大まかに 3 つの特徴がある。1 つ目の特徴は、脆弱性に関する詳細な技術情報が広く出回りやすい点である。OSS のソースコードは公開されていることから、OSS の脆弱性やその修正に必要な技術情報は、開発主体が公表するまでもなく、誰でもソースコードを見て確認・分析することができる。実際、注目度の高い脆弱性情報が出たときには、技術者や研究者がその技術情報をブログなどで解説することがある。これらの技術情報は、攻撃者にとっても、攻撃手段を探るうえで有効な手がかりとなる。このため、OSS については、PoC コード（proof of concept code）[21]が作成されやすく、そこからエクスプロイト・コード（exploit code）[22]の開発を経て、マルウェ

---

[20] イントラネットは、企業などの限られた範囲内でアクセス可能なネットワークを表す。内部（intra）とネットワークを意味するネット（net）を組み合わせた造語である。

[21] PoC コードは、脆弱性が実際に悪用可能であることを証明するための検証用プログラム・ソースコードである。脆弱性実証コードとも呼ばれる。

[22] エクスプロイト・コードは、脆弱性を突く攻撃への悪用を前提としたプログラム・ソースコードである。

アの流通にまで至る流れが生じやすい。

2 つ目の特徴は、OSS の脆弱性が発見されてから、当該 OSS が組み込まれた製品の脆弱性が最終的に解消されるまでのタイムラグが長いことである。OSS の脆弱性情報が公表されると、まず、ベンダーがその情報を認知し、自社製品の修正対応を開始する。そこからさらに、金融機関が自社のシステムに修正パッチを当てるまでの期間は、商用ソフトウェアにパッチを当てるケースと比較して長くなる傾向がある。

3 つ目の特徴は、著名な OSS に関する脆弱性が 1 つ発見されると、当該 OSS の他の脆弱性が短期間のうちに発見される傾向があることである。こうした傾向は、脆弱性が見つかった OSS について、多くの人が未発見の脆弱性がまだ残っているのではないかと疑い、集中的に調査を行うために生じると考えられる。

1 つ目の特徴でも述べたとおり、ソースコードが公開されている OSS では、脆弱性情報をもとに自力で OSS を修正することも可能であり、技術力のある人材や企業にとって、ソースコードを修正する難度はさほど高くない。

（４）海外の金融機関の事例

このため、海外の先進的な金融機関では、著名な OSS 開発者を直接雇用している例もある。例えば、オープン・ソースのウェブ・サーバ・ソフトウェアである Apache HTTP Server[23]（以下、単に Apache と呼ぶ）の開発者を自社のエンジニアとして雇用しているケースでは、雇用主である企業は、当該エンジニアに対して、会社の通常業務に加えて、Apache の開発も業務として認めている。こうした背景には、OSS が著名であればあるほど、その開発に携わっているエンジニアも優秀であるため、当該 OSS の開発を業務として認めなければ、雇用契約に応じてもらうことが難しいという事情もある。この点、雇用契約さえ締結できれば、Apache についての脆弱性情報が公表されたときに、当該エンジニアの知見を活かすことができる。すなわち、修正パッチが一般に公開されるよりも前から、企業は修正対応を行うための体制整備に着手することが可能となる。また、優秀な OSS エンジニアは、通常業務においても非常に優秀であるため、そうした意味でも会社への高い貢献度が期待できる。

このように、OSS エンジニアを雇用している企業は、社会的にも重要なソフトウェアを開発しているエンジニアを雇用することによって、当該エンジニアを経済的に支えると同時に、そのエンジニアが OSS の開発を推進することを通じて、OSS コミュニティにも貢献していることになる。実際、海外の企業では、社会全体を支えるという社会貢献の意識を持っている企業が多い一方、日本の

---

[23] Apache HTTP Server は、Apache Software Foundation の登録商標または商標である。https://httpd.apache.org/

企業には、こうした考え方を持つに至っているところはまだまだ少ない。OSS の基本思想は「使いたい人が自由に使う」というものではあるが、わが国の企業の中には、OSS について、単に「無料で使えるもの」とのイメージを抱いている先も少なくない。今後は、海外企業のように、OSS コミュニティに対して貢献するとの考え方が拡がっていくことを期待したい。

（５）SBOM の先にあるもの

OSS のユーザ企業は、OSS に特化した脆弱性対応を行うかを考慮すべきである。そうした観点からも注目を集めている SBOM については、先進的な企業ほど取組みを活発化させている。もっとも、ソフトウェア・カタログの管理では、カタログに掲載される情報が標準化されておらず、相互運用が困難であるといった問題を抱えている。

そうした課題を乗り越えて仮に SBOM による OSS の一覧が作成できたとしても、その次に課題となるのは、OSS の安全性指標の開発・標準化である。安全性指標が確立されれば、企業が OSS を導入するか否かを判断する際の重要な考慮要素になると考えられる。例えば、オープン・ソース・セキュリティ・インデックス（open source security index）というウェブ・ページ[24]では、OSS コミュニティの活動状況（ユーザ数、開発者数、開発のアクティブ度合いなど）を評価してランキング形式で表示している。ユーザ数や開発が活発な OSS ほど、脆弱性が発見された場合も素早く対応できると想像される。

４．講演３「機械学習と OSS のセキュリティ」

小澤は、機械学習モデルの開発における OSS の利用状況、サプライ・チェーンや機械学習に特有のセキュリティ・リスクについて次のように発表した。

（１）オープンな基盤のうえで発展する機械学習

機械学習モデルの開発では、訓練前のモデルが訓練データを学習することによって自動的に機能を獲得する。機械学習モデルの開発において訓練データを用意することは、通常のソフトウェア開発におけるプログラムの作成に相当するものと捉えることができる。

機械学習モデルの開発は、オープン・アクセスの基盤のうえで成立している。機械学習モデルの構造と学習アルゴリズムのソースコードが、Github[25]などのリ

[24] https://opensourcesecurityindex.io/

[25] https://github.com/

ポジトリ（repository）[26]において公開され、それらをユーザが検証したうえで利用している。訓練済みの機械学習モデルや訓練データそのものも、リポジトリに保存され、誰でも利用可能となっていることが多い。特に、大規模な事前学習済みモデルは、その訓練に膨大な計算リソースが必要となるため、大手 IT 企業が提供するモデルをそのまま利用することが標準となっている。

こうしたオープン・アクセスの強みを活かした仕組みが、プラットフォームと呼ばれる統合環境において提供されている。プラットフォームとは、モデル選択や、データのクレンジング（洗浄）と前処理、モデルの本番環境への配置（デプロイ）を担う開発・運用向けの基盤である。

（２）機械学習の脆弱性

誰もが利用可能（オープン）なプラットフォームは、機械学習モデルの開発を促進する一方、リスクももたらしている。こうしたリスクをもたらす機械学習の脆弱性には、ソフトウェアに共通のものと、機械学習に特有のものが存在する。

ソフトウェアに共通する脆弱性は、プログラムのバグ等に起因するものである。プラットフォームは、大量の OSS が組み込まれた巨大なソフトウェアである。一般的にプラットフォームの信頼性は高いとされるものの、OSS に含まれる不具合が脆弱性をもたらしうる。例えば、リポジトリから取得した Docker コンテナ [27]に、脆弱性を持つコンパイラが含まれていると、これを用いて開発した機械学習モデルにも脆弱性をもたらしうる。また、公開されている訓練データや機械学習モデルのデータ自体が改ざんされて広く流通すると、サプライ・チェーンに対するリスクとなる。

機械学習に特有の脆弱性としては、敵対的サンプル攻撃やバックドア攻撃に対する脆弱性が挙げられる。敵対的サンプル攻撃は、入力データにノイズを加えることで、深層学習モデルから不正な振る舞いを引き出すというものであり [28]、

---

26 リポジトリは、プログラム・ソースコードやデータを保管・共有するためのサービスを表す。バージョン管理などの高度な機能を備えたオンライン・サービスもある。こうしたサービスは、OSS も含めた複数人でのソフトウェアの共同開発では標準的に利用される。

27 コンテナ（container）は、OS 上で動作するアプリケーションに加えて、そのアプリケーションを動作させるために必要となるライブラリやミドルウェアをまとめたソフトウェアのパッケージである。システム基盤（コンテナ・エンジン）上では依存関係を持たない独立したソフトウェアとしてコンテナを取り扱えるため、自由に追加、削除、他のシステムへの移植が可能である。「コンテナ」の名称は、運輸業界において、ものを独立した定型の箱（コンテナ）に詰めて、輸送の便を図るイメージに由来しているとされる。Docker は、Docker 社が開発するコンテナを開発、配置、実行するプラットフォームである。Docker は、Docker, Inc.の米国およびその他の国における商標または登録商標である。

28 例えば、機械学習モデルに画像認識のタスクを処理させる状況を考える。今、機械学習モデ

機械学習モデルがもともと持っている脆弱性を悪用している。

バックドア攻撃は、通常の入力データに対しては正常に動作するが、攻撃者が指定した特徴（トリガー）を持つ入力データに対して不正に動作する付加的な機能（バックドア）を機械学習モデルに埋め込む攻撃のことを指す[29]。この攻撃は、訓練データを汚染することによって、機械学習モデルに新たな脆弱性を埋め込むことで実行される。例えば、大量のデータに正解ラベルを付与して訓練データを作成する業務をアウトソーシングした場合、悪意のある委託先企業によって訓練データが汚染されるようなケースが考えられる。

また、訓練データや機械学習モデルのデータがサプライ・チェーン上で汚染されて、それが広く拡散するようなケースも考えられる。特に、汎用性が高く、高機能な事前学習済みモデルは、多数のシステムに組み込まれる可能性がある。オープンな環境では、機械学習モデルの開発は促進されるが、共有されているリソースが攻撃された場合には、それに依存するすべての機械学習モデルが被害の対象となりうる。

例えば、自動車の自動運転システムでは、GPS（Global Positioning System）やカメラ・センサーの情報がクラウド・サービスに送信され、クラウド・サービスは、自動車に対して、時々の環境に応じた車体制御情報を返信する。こうした通信を繰り返す過程において、センサーの情報を改ざんする攻撃、標識に細工を施して不正な動作を実行させる攻撃、道路標識を敵対的サンプルに変更する攻撃が行われる可能性がある。また、クラウド・サービスから自動車に返送される情報を改ざんする攻撃や、悪意のあるデータを付加する攻撃も考えられる。訓練データが汚染された場合には、これらの攻撃を通じて車体の制御が乗っ取られるというリスクもある。

（３）脆弱性への対応

機械学習モデルの脆弱性への対策として、機械学習のセキュリティ（security for AI）が大きな研究分野となっており、現在も発展途上にある。

敵対的サンプル攻撃に対しては、敵対的サンプルを訓練データに付加する敵対的学習（adversarial learning）が直截な対策となる。また、機械学習モデルが必要のない機能まで獲得することがないように不要なパラメータを削減する防御

ルがパンダの画像データの入力に対して、正しくパンダであると判定できているとする。ここで、パンダの画像データに人の目には見えない微小なノイズを加えると、高い確率でテナガザルであると誤認させることができる。こうした入力データを敵対的サンプルという。

[29] 例えば、監視カメラの画像から人物を判定および認証する場合を考える。このとき、サングラスなどの特定の装飾物（トリガー）を身に着けることによって、権限のない攻撃者が監視カメラによる認証を突破し、管理者権限を奪取するバックドアを仕掛ける攻撃が考えられる。

的蒸留（defensive distillation）という対策が考案されている。このほか、複数のモデルの平均をとるモデル・アンサンブル（model ensemble）という対策もある。

バックドア攻撃に対しては、委託先企業を慎重に選定することや、訓練データや事前学習モデルを信頼できる場所から取得するといった基本的な対策を徹底することが重要である。技術的には、トリガーや汚染データを検知する手法が考案されているが、本講演では詳細は割愛する。

広く普及している機械学習モデルには、上述した機械学習に特有の脆弱性が既に組み込まれている可能性もある。今後は、不正な振る舞いや脆弱性を検知する研究も重要になっていくと考えられる。

５．パネル・ディスカッション

パネル・ディスカッションでは、金融業界の OSS との関わり方、自助と共助の切り分け、機械学習のセキュリティに関する課題や留意点について議論を行った。その概要は以下のとおりである。

（１）金融機関と OSS との関わり方

まず、モデレータの**菅**は、金融機関が OSS コミュニティとどのように関わっていくべきかをパネリストに問うた。

真鍋は、まず、「OSS コミュニティ」という言葉の意味に関する世間一般での理解が、本来の意味とは乖離していると説明した。世間一般では、OSS コミュニティは、ソフトウェア開発を担うプログラマで構成される開発コミュニティと理解されることが多い。しかし、OSS コミュニティの本来の意味は、ソフトウェアを中心に形成されるコミュニティ全体のことを指し、OSS に関わる活動を行うあらゆる主体を包含するとしたうえで、金融機関がソフトウェアを自ら開発する行為や、対価を支払って他の事業者も巻き込みながら開発する行為も、そのソフトウェアが OSS を含んでいれば、OSS コミュニティでの活動の一環といえると説明した。

さらに、具体例として、オープン・ソースのリレーショナル・データベースである PostgreSQL[30]の事例を取り上げ、OSS コミュニティによるサポートが終了したバージョンの OSS であっても、事業会社による継続的なサポートが有償で提供されていることを紹介した。こうした OSS コミュニティの活動を補完するサービスに対価を支払うことも、OSS コミュニティへの貢献である。また、こうした OSS に関する有償サポートを、どのような事業会社であっても提供できる

[30] PostgreSQL は、PostgreSQL Community Association of Canada のカナダおよびその他の国における登録商標または商標である。https://www.postgresql.org/

点が、OSS の長所であると指摘した。

鎌田は、金融機関が主体的に特定の OSS の導入の是非を判断するケースは稀であり、委託先である IT ベンダーが OSS の組み込みを提案し、その提案を金融機関が承諾するかたちで OSS を導入することが多いと説明した。セキュリティ評価についても、金融機関自身ではなく、システムの保守に責任を負う事業者や、その事業者と金融機関の間に入っているシステム・インテグレーターが評価を行うケースが大半であると指摘した。一方、ある OSS に致命的な脆弱性が発見された場合、金融機関は自社システムの中で当該 OSS を利用している箇所がないかを悉皆的かつ慎重に調査し、利用が確認された場合には、委託先である IT ベンダーと協働して対応を進めることが多いと説明した。

（２）金融機関における自助と共助

萱は、金融機関として脆弱性対応について、どこまでを自助、どこからを共助で行うのが望ましいか、その切り分けをどのように行えばよいかをパネリストに問うた。

真鍋は、基本的な考え方として「まずは自助、次に共助」で対応することが金融機関には望まれると述べた。そのうえで、セキュリティ対策において本質的に重要なことは「伝わるべきことが、伝わるべきところに、きちんと伝わる」仕組みを構築することであり、これを効率的に行う観点から自助と共助をどう切り分けるかを考えるべきであるとの見方を示した。ソフトウェア製品については、設計、開発、ブランド化、販売などの各工程を担う企業同士が複雑に絡み合いながら最終製品が提供される。このため、場合によっては、ソフトウェア製品に関する重要な脆弱性情報が、システム・インテグレーターや金融機関まで届かないことがある。こうしたときに、必要な情報の入手先を自分で探せば自助になるし、コミュニティと協力して探せば共助になる。また、SBOM の作成を強制して、問題の所在を機械的に割り出すといった自動化の方法もある。このように、自助、共助、自動化は、脆弱性に関する情報を適切な場所に伝えるための方法論の違いとして整理できるとの考え方を示した。

次に、**真鍋**は、脆弱性情報の伝達を支援し、脆弱性対応の迅速化を促進する取組みとして、JPCERT コーディネーション・センターの活動を紹介した。例えば、ソフトウェア A とソフトウェア B が同根の脆弱性を抱えているとした場合、一方が脆弱性情報を早期に公表すると、他方の対応が完了していない間に攻撃されるといった事態が発生しうる。こうした事態を未然に防ぐためにも、JPCERT コーディネーション・センターの活動が有用であると強調した。

JPCERT コーディネーション・センターの活動について、**鎌田**は、次のように補足した。2000 年代前半、脆弱性を発見した研究者がベンダーに直接報告して

も相手にされなかったため、自らが管理するブログに脆弱性に関する情報を公開したところ、対策が確立していない段階から攻撃が始まるといった状況が頻発した。こうした状況を是正するために、脆弱性を発見した者が情報を正しくベンダーに伝える仕組みを作ろうと、日米英の三カ国で連携して始めた取組みが、JPCERT コーディネーション・センターにおける脆弱性情報の仲介活動の枠組みである。こうした取組みは日本独自のものとまでは言えないものの、しっかりと制度運用できているのは日本の特徴であると説明した。

また、**鎌田**は、共助には限界があるため、金融 ISAC においても「まずは自助が基本」との考え方をとっていると述べ、共助と自助の切り分けを次のように説明した。世の中の脆弱性情報を収集し一般論の範囲で脅威度を判定し、ソフトウェア更新などの緊要性を判定するところまでは共助で対応できる。その際、脆弱性が深刻であると判断された場合には、個々の金融機関において、開発担当グループとセキュリティ担当グループが連携して、自社環境で影響を受ける場所を特定し、パッチ適用の是非や所要時間を見積もったうえで、最悪の場合にはサービスを停止するといった選択肢の検討を行う。こうした自社でしか対応できない部分については、必然的に自助での対応になる。ただし、共助コミュニティへの質問・相談等を通じて、自社の問題解決を促進できることもあるため、共助は自助にも活用できる。

（３）脅威度判定はなぜ難しいのか

菅は、脆弱性の自社システムに対する脅威度を判定することが容易でない理由や望ましい対応についてパネリストに問うた。

真鍋は、自社において脅威度を適切に判断するうえで必要な情報が十分に伝わっていないことを指摘した。単純な不具合情報のケースを考えてみると、不具合が自社システムに及ぼす影響を見極めるには、当該システムに関するさまざまな内部情報が必要となる。具体的には、システムを構成する各ソフトウェアの詳細やバージョン情報、それらのシステムへの組み込み方などであり、こうした情報を伝達・収集することの難しさが脅威度判定の難しさにつながっているとの見方を示した。

また、**真鍋**は、情報の氾濫が脅威度判定を難しくすることもあると指摘した。情報過多により適切な情報の選定ができない、あるいは、選定するために必要な情報がない場合がある。例えば、多くの人にとって身近な表計算ソフトウェアである Microsoft 社の Excel[31]であれば脆弱性との関連を相応に意識できるが、WordPress や Apache Struts などについては、脆弱性の影響を的確に認識できない

[31] Microsoft Excel は、Microsoft Corporation の米国およびその他の国における登録商標または商標である。

といったことがある。そうした人の眼前に Open SSL の脆弱性の情報が流れてきても、自社システムとの関連を認識できないといった事態になる。こうしたものも、情報が適切に伝わっていないケースに該当すると解説した。

鎌田は、そもそも脆弱性に対する理解そのものが難しいと指摘した。その原因として、自分で実際にシステムを攻撃した経験のある人が少ないからであるとの見方を示した。多くの人にとって脆弱性に対する理解は、攻撃の実体験に基づかない概念レベルのものにとどまっており、攻撃を受けた際のシステムの挙動に関して正確に理解している人は稀である。こうした理解を得るためには、試験用の環境を用意するなどして、実際に攻撃を模擬体験することが重要であると述べた[32]。

（４）機械学習におけるセキュリティと「共助」

菅は、機械学習におけるセキュリティについても、OSS のアナロジーとして捉えられるか（脆弱性に関する情報が発見者から開発者に適切に届けられ、エコシステムの中で修正されるのか）、パネリストに問うた。

小澤は、機械学習における脆弱性は、OSS のようにオープンなエコシステムによって管理されているのではなく、AI プラットフォームなどを供する個別企業によって管理されている。このため、脆弱性が発見された場合には、モデルの開発元企業が相談窓口を設けて対策を講じているのが実態ではないかとの見方を示した。そのうえで、機械学習モデルの脆弱性を発見すること自体が難しいことに加え、発見した脆弱性の性質を適切に理解することも容易ではないと指摘した。そうした中、既存モデルの検証を行い、脆弱性の発見を担う主体はセキュリティ・カンファレンスに参加する大学や企業の研究者となっており、彼らが「自助」として行っている研究や学会運営に関する活動が、ある意味で機械学習分野のセキュリティにおける「共助」として機能しているともいえる。ただし、脆弱性を埋め込まれたモデルが普及したあとで、これを修復する仕組みは整備されておらず、課題は残ると指摘した。

（５）機械学習における脆弱性診断

小澤は、通常のソフトウェアの脆弱性診断では、既知の攻撃コードをシステムに入力して攻撃が成立するか否かを判定するのが一般的であると説明した。そのうえで、こうした手法が確立していない機械学習モデルの脆弱性診断については、通常のソフトウェアとは異なる考え方のもとで独自の手法を確立していくことが重要であるとし、次のように付言した。

---

32 実際に利用している環境に攻撃を行うと不正アクセス禁止法に抵触するため、自分で試験環境を用意することが必須である。

機械学習モデルの場合、例えば宗教や思想などに関連したセンシティブな話題に対する回答が、倫理的に答えてもよい内容か否かを確認する方法がある。実際にきわどい質問をすると、その質問には回答できないとか、法令違反になる可能性があるとの回答が返ってくることが多い。しかし、法律については考慮しない前提を置いているため、答えても法令違反にはならないとの補足説明を加えると、回答が返ってきてしまうことがあるなど、現時点でチャットボットは意外と説得に応じてしまうことが判明している。

また、**小澤**は、チャットボットは、さまざまな質問に対してもっともらしい回答をする汎用性はあるが、決まった問いに対していつもまったく同じ答えを返すことを目的とした定型的な道具ではないと述べた。そのうえで、こうしたチャットボットの汎用性は、倫理などに関する脆弱性にもつながっており、そうした脆弱性を解消するために厳格な制御を試みると、ルールベースのシステムに対する機械学習の優位性を失うおそれがある。このように、機械学習モデルの管理の厳格さと脆弱性にはトレードオフの関係が存在する中、現時点では、悪意のない開発者が機械学習モデルを訓練することをもって管理の正当性を担保していくしかないとの考え方を示した。

（６）機械学習におけるデータ汚染への対策

オープン・アクセスで誰でも利用できる機械学習モデルや訓練データについては、サプライ・チェーンのリスクのほか、広く利用されている訓練データが汚染されるリスクや、機械学習モデルに脆弱性が埋め込まれるリスクが懸念される。こうしたリスクを踏まえて、**菅**は、事前学習モデルの脆弱性や訓練データの汚染の有無を検証する方法や、汚染等が判明した場合の対応について問うた。

まず、**小澤**は、訓練データや機械学習モデルの正当性を検証する技術は十分には確立されておらず、それ自体が先端的な研究テーマであると指摘した。そうした現状において、金融機関では、機械学習モデルに特有のセキュリティ・リスクに対して、その脅威度のレベルに応じて、モデルの開発・運用環境や用途を限定するといった対応をとっている。例えば、不正送金を検出するという用途では、機密性の高い決済情報を含む訓練データは厳格に管理されていることから、データが汚染されるリスクは僅少である。さらに、インターネットと接続していない閉じた環境下であれば、仮に機械学習モデルが不正な振る舞いを行っても直接悪影響を受ける範囲は相当限定されると説明した。

さらに、**小澤**は、将来的には、どのような訓練データを使って機械学習モデルが作成されたかを証明する仕組みが必要であり、既に信頼が確立している事前学習済みモデルをチューニングする場合には、こうした証明は比較的容易と考えられると説明した。また、自身が作成したモデルが信用に足ることを他者に対

して証明するのは容易ではないが、機械学習モデルの正当性を電子透かし技術などを使いながら証明する仕組みも研究されており、今後の重要な研究課題の 1 つであると述べた。

以　上

【参考文献】

経済産業省商務情報政策局サイバーセキュリティ課、「OSS の利活用及びそのセキュリティ確保に向けた管理手法に関する事例集」、2022 年（available at https://www.meti.go.jp/press/2022/05/20220510001/20220510001-1-2.pdf）

特許庁・内閣府、「Opensource for ALL」、2019 年（available at https://www.jpo.go.jp/resources/report/takoku/document/zaisanken_kouhyou/2019_06_2.pdf、2023 年 4 月 21 日）

PwC コンサルティング合同会社、「デジタル化、IoT 化時代におけるオープンソースソフトウェアに係る知財リスク等に関する調査研究報告書」、2020 年（available at https://www.jpo.go.jp/resources/report/takoku/document/zaisanken_kouhyou/2019_06_1.pdf、2023 年 4 月 21 日）

Debian Project, "Debian Social Contract, Version 1.0," Bits from Debian, 1997 (available at https://www.debian.org/social_contract.1.0.en.html、2023 年 4 月 20 日).

National Telecommunications and Information Administration, "The Minimum Elements For a Software Bill of Materials (SBOM)," 2021 (available at https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom).

Office of Management and Budget, and Executive Office of the President, "Federal Source Code Policy: Achieving Efficiency, Transparency, and Innovation through Reusable and Open Source Software," 2016 (available at https://www.whitehouse.gov/wp-content/uploads/legacy_drupal_files/omb/memoranda/2016/m_16_21.pdf、2023 年 4 月 21 日).

Open Source Initiative, "The Open Source Definition (Version 1.9)," 2007 (available at https://opensource.org/osd/、2023 年 4 月 21 日).

The White House, "Executive Order on Improving the Nation's Cybersecurity," 2021, (available at https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/、2023 年 5 月 2 日).