LLMを用いた悪意あるOSS開発者に起因するセキュリティリスクの検知手法の検討

鐘本 楊[1,a)]　荒川 玲佳[1]　上原 貴之[1]　秋山 満昭[1]

概要：悪意ある OSS 開発者に起因するセキュリティリスクが表面化している．本研究は，OSS 開発におけるセキュリティリスクをインサイダー攻撃とプロテストウェアの 2 つに分類し，これらのリスクを事前に検出する手法を提案する．インサイダー攻撃では，攻撃者が OSS 開発者に圧力をかけ，管理権限を奪取し悪意あるコードを埋め込む事象を特徴として捉え，プロテストウェアは開発者の金銭的問題あるいは政治的主張の発言を特徴として捉え，LLM を活用して開発者の発言からセキュリティリスクを事前に検出する手法を検討する．実験結果からインサイダー攻撃に関しては検知率 50.0%，誤検知率 0.07%，プロテストウェアに関しては検知率 50.0%，誤検知率 1.66%の精度で検知可能であること，またインサイダー攻撃に関しては事件が発覚する前に，プロテストウェアに関しては悪性コードが混入する前に検知可能であることを確認した．

キーワード：OSS, セキュリティリスク，インサイダー攻撃，プロテストウェア

Towards Detecting Security Risks Caused by Malicious OSS Developers using LLM

Yo Kanemoto[1,a)]　Reika Arakawa[1]　Takayuki Uehara[1]　Mitsuaki Akiyama[1]

Abstract: Security risks stemming from malicious OSS developers are becoming increasingly apparent. This study categorizes security risks in OSS development into two types: insider attacks and protestware, and proposes methods to detect these risks in advance. In the case of insider attacks, we focus on scenarios where attackers exert pressure on OSS developers to seize administrative privileges and embed malicious code. Protestware, on the other hand, is characterized by developers expressing financial issues or political statements. We propose a method utilizing LLMs to detect security risks from developers' statements in advance. Experimental results demonstrate that our method achieves a detection rate of 50.0% with a false positive rate of 0.07% for insider attacks, and a detection rate of 50.0% with a false positive rate of 1.66% for protestware. Furthermore, it was confirmed that insider attacks can be detected before the incidents are exposed, and protestware can be detected before malicious code is injected.

Keywords: OSS, Security Risk，Insider Attack，Protestware

1. はじめに

ソフトウェアは現代社会のあらゆる分野において不可欠な要素となっており，IT システムだけでなく，自動車や航空機など人命を扱う機械や日常生活に欠かせない電気や水道などの重要インフラの制御に利用されている．幅広い用途や要件に対応するため，ソフトウェアに求められる機能は増加し，その複雑性はより増している [1].

ソフトウェア開発では，開発効率化のためにサードパーティのライブラリや Open Source Software (OSS) のソースコードを再利用することが主流になってきており，その

[1] 日本電信電話株式会社 NTT 社会情報研究所
NTT Social Informatics Laboratories
3-9-11 Midori-cho Musasino-shi Tokyo
[a)] yo.kanemoto@ntt.com

動向は加速している．約 90%のソフトウェアがサードパーティのライブラリや OSS のソースコードを再利用していたという調査結果も存在する [2].

近年では，悪意ある OSS 開発者に起因するさまざまなセキュリティリスクが表面化している [3]．OSS 開発では複数の開発者がプロジェクトメンバとして集まって開発することが一般であり，オンラインでのコミュニケーションが多いため，メンバ全員が顔見知りという訳ではない．そのため，素性を知らない相手と共同で作業を行うこととなる．本研究ではこれらのセキュリティリスクの中でも喫緊の課題である二つのリスクについて注目する．一つ目は攻撃者によるソフトウェア開発者へのインサイダー攻撃に対するリスクであり，二つ目はソフトウェア開発者の心情の変化によるプロテストウェア化に対するリスクである．

インサイダー攻撃では，悪意ある OSS 開発者，つまり攻撃者が善意の OSS 開発者に対して精神的な圧力をかけ，ソフトウェアに対する改変権限を奪取することにより，悪意あるコードを自由に埋め込むことができることである．例えば，2024 年 3 月 29 日に Linux 向け圧縮ユーティリティとして広く利用されている XZ Utils にバックドアが存在していたことが発覚した [4]．この事例では複数の開発者が OSS 管理者に対して，パッチが遅いことや進捗がないことを非難し，その後管理者権限の譲渡を執拗に要求している．管理者はその要求に屈し，管理者権限を攻撃者の一人に付与し，その後ソフトウェアにバックドアが埋め込まれた．

プロテストウェアとは開発者の思想を主張をするために利用されたソフトウェアのことであり，OSS 開発者の心情によって抗議を表明するために作成される．典型的な理由としては金銭的問題に対する抗議や，戦争や犯罪に対する抗議である．プロテストウェア全てが有害という訳ではなく，ドキュメントや表示が少し変化するといった言論の自由の範囲内に収まるものもあれば，破壊的で有害な性質を持つものも存在する．例えば，2022 年 1 月に人気 OSS である colors.js および faker.js に無限ループが発生する DoS 攻撃のコードが追加されたことが発覚した [5]．原因は OSS 開発者への金銭的サポートが少ないことに対する抗議であった．この OSS 開発者は，事件が発生する数年前から「私は，コミュニティからの貢献に追いつき，年間にいくつかの安定したリリースを行うために最善を尽くしています．しかし，しばらくの間収入がなく，金銭的に非常に厳しい状況にあります.」や「これ以上，無料の仕事をしません．私に報酬を支払うか，プロジェクトをフォークしてください．これ以上フォーチュン 500 企業やその他の小規模な企業に無料で仕事を提供することはありません.」といった金銭的な問題に対する発言を行った後，抗議を示すために悪性コードが追加された．

埋め込まれた悪意あるコードを検知する手法に対する既存研究は多数存在する [6]．しかし，このアプローチは悪意あるコードが混入された後の特徴に基づいて検知を行うため，事後の発見となる．事件の発覚によりソフトウェアの開発が止まったり，急に利用できなくなる可能性もあり，OSS を利用するシステム・サービスの事業者にとって，問題の OSS に対する依存関係を解消したり，新たなソフトウェアを選定したり，することが必要となり，対処にかかる修正コストが大きいことが課題となる．

このような背景を踏まえて，本研究では悪意あるコードが埋め込まれる前に，つまり，事前にソフトウェアに潜むセキュリティリスクを検知・予測可能か検討した．ソフトウェアを開発するのは人であることから，人に着目し，開発者の言動からインサイダー攻撃のリスクやプロテストウェアにつながるリスクを事前に察知できるかに挑戦する．具体的には大規模言語モデル (LLM) を活用して，開発者の発言から特記すべき状況が起きたことを識別し，これらの状況が継続して発生している際にセキュリティリスクありとして検知する手法を提案する．

実験の結果，提案手法はインサイダー攻撃に関しては検知率 50.0%，誤検知率 0.07%，プロテストウェアに関しては検知率 50.0%，誤検知率 1.66%の精度で検知することがわかった．またインサイダー攻撃に関しては事件が発覚する前に，プロテストウェアに関しては悪性コードが混入する前に検知ことが可能であることがわかった．

2. 背景

2.1 OSS 開発における開発者の役割とコミュニケーション

OSS プロジェクトは通常，地理的に分散した開発者コミュニティによって推進されており，それぞれの開発者が異なる専門知識，背景，文化を持っている．したがって，OSS 開発における開発者の役割は単にコードの作成にとどまらず，プロジェクト全体の方向性を導くリーダーシップや，他の貢献者との協調，さらにはユーザとの対話を含む多岐にわたる．OSS 開発におけるコミュニケーションは，開発者間の協力を円滑に進めるために欠かせない要素である．分散型チームが共通の目標に向かって効率的に作業するためには，透明性の高い情報共有と，明確で一貫したコミュニケーションが必要とされる．具体的には，コードレビュー，問題追跡，ドキュメント作成，さらにはチャットツールやメーリングリストを通じた非公式な議論など，さまざまな形態のコミュニケーションが日常的に行われている．

2.2 OSS 開発者に対するインサイダー攻撃

攻撃者は OSS プロジェクトの管理者や貢献者になりすまして，他の開発者に悪意のあるコードの混入を促すことができる．このような攻撃が成功すると，攻撃者は OSS プロジェクト全体を危険にさらし，多くのユーザーに影響を

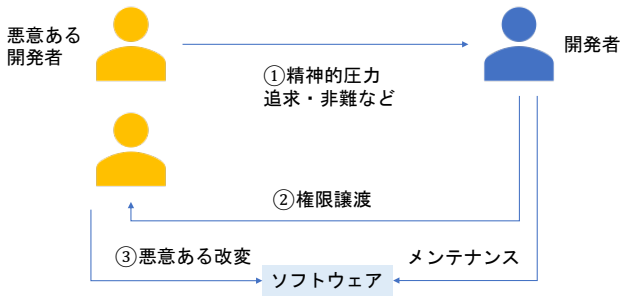

図 1: OSS 開発者に対するインサイダー攻撃
Fig. 1 Insider attacks against OSS developers

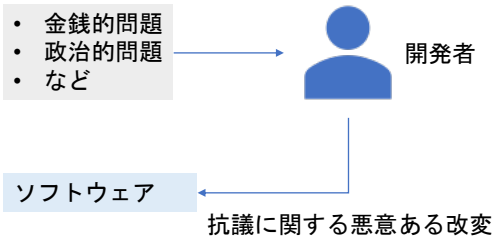

図 2: プロテストウェアの発生原因
Fig. 2 Causes of protestware

与える悪意のあるソフトウェアを拡散することができる．本研究では，このような OSS コミュニティの内部に侵入し，管理者権限で悪事を行う行為をインサイダー攻撃と呼び，このインサイダー攻撃に着目する．

2024 年 3 月 29 日には発覚した Linux 向け圧縮ユーティリティとして広く利用されている XZ Utils にバックドアが混入されたインサイダー攻撃に関する事例が存在する．図 1 にそのインサイダー攻撃の概要を示す．この事例では複数の開発者が OSS 管理者に対して，パッチが遅いことや進捗がないことを非難し，その後，管理者権限を譲渡することを執拗に要求している．管理者はその要求に屈し，管理者権限を攻撃者の一人に付与し，その後攻撃者によってバックドアがソフトウェアに埋め込まれた [4]．

2.3 プロテストウェア

プロテストウェアはソフトウェア開発者が特定の社会的あるいは政治的なメッセージを広めるために意図的に組み込むコードやソフトウェアの一種である．プロテストウェアはソフトウェア自体に抗議メッセージを表示する機能を組み込んだものや，特定の機能を制限・停止させるものが含まれる．プロテストウェアは，利用者や企業が OSS に対して不信感を抱かせる原因となる可能性がある．ソフトウェアが意図しない形で動作し，ユーザに損害を与える場合，OSS に対する信頼性や安全性が損なわれ，OSS 全体の評判に悪影響を及ぼすことが懸念される．

プロテストウェアの具体的な例として，例えば，2022 年 1 月に人気 OSS である colors.js および faker.js に無限ループが発生する DoS 攻撃のコードが追加されたことが発覚した [5]．原因は OSS 開発者に対する金銭的サポートが少ないことに対する抗議でした．2022 年 3 月 16 日においても人気 OSS である node-ipc に特定の地域において起動した際に PC 内のファイルを破壊するコードが追加されていることが発覚した [7]．原因はロシアによるウクライナ侵攻に対する抗議でした．この OSS 開発者は 3 月 10 日に「戦争が何かしら進展し，第三次世界大戦のような大規模な事態に発展し，もっと多くの人が『何か対策を取っておけばよかった』と感じるようになるか，戦争が終わってこの変更が削除されるまでの間は，影響がないバージョンを利用する自由があります.」といった皮肉的でかつ戦争に抗議する発言を行った後，3 月 16 日に抗議を示すために悪性コードが追加された．これらの事件は，OSS コミュニティ内で大きな議論を巻き起こし，プロテストウェアの倫理性や正当性に関する問題が浮き彫りとなった．

本研究では，このような特定の環境によって開発者の考え方が変化し，その開発者が提供するソフトウェアにも影響を及ぼすと考え，通常のソフトウェアをプロテストウェアに変化させてしまう開発者に焦点を当てる．図 2 に本研究で考えるプロテストウェアの発生原因の概要を示す．過去事例から，金銭的問題あるいは政治的問題に対する抗議に関するプロテストウェアが多いため，この二つの特徴に注目する．

3. 悪意ある開発者に起因するリスク検知手法

背景で述べた通り，悪意を持つ開発者に起因するセキュリティリスクは大きく二つに分類される．一つ目は攻撃者が管理者権限を持つ開発者に精神的圧力を加え，ソフトウェアレポジトリの管理者権限を奪取し，悪意あるコードを埋め込むインサイダー攻撃リスクである．二つ目は抗議などの思想によって，悪意あるコードがソフトウェアに埋め込まれるプロテストウェアリスクである．一つ目のインサイダー攻撃に対応するために複数の開発者の発言に基づき特定の状況が発生しているかを判断する．特定の状況が発生しておりかつ，段階が進行していると判断できる場合に検知を行う．二つ目のプロテストウェアに対応するため，単一の開発者の金銭的問題や政治に関する抗議的発言が継続している場合に検知を行う．

図 3 に提案手法の概要を示す．提案手法は二つの検知部に分かれており，各検知部には二つのステップが存在する．提案手法では特定の状況の発生あるいは発言の意図の判断に，LLM を活用する．LLM に対して判断に関する指示および発言の内容を入力し，特定の状況が発生しているを判断させ，状況識別結果を出力する．状況識別結果を元に，各検知部でそれぞれのリスクがあるかを分析し，検知を行う．本研究で言及する発言とは具体的にはソフトウェア開発する際に発生する commit, issue, merge request に対す

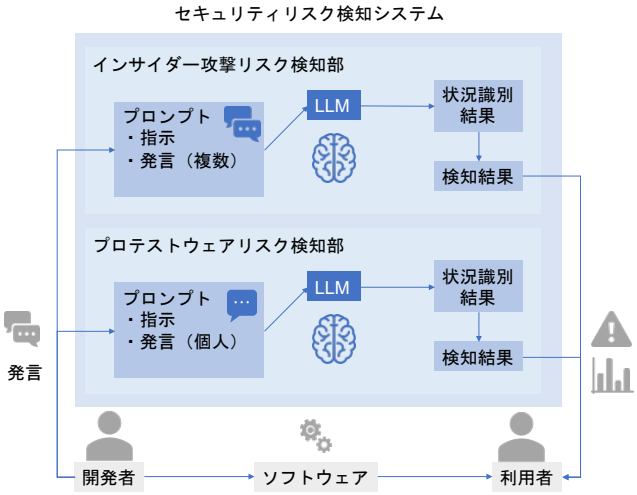

図 3: 提案手法の概要

Fig. 3 Overview of the proposed method

るコメントや，メーリングリストに投稿されるメールの文面などを指す．

3.1 インサイダー攻撃リスク検知部

ステップ 1：プロンプト生成 まず，検査対象のソフトウェアプロジェクトに関わる複数の開発者の発言を収集する．ソフトウェアに対する issue や merge request に関わる発言にはエラーログやファイル一覧といった発言の意図を汲み取るのに必要のない部分も多く含まれるため，この部分は除去する．一定期間ごとに，複数の開発者の発言から特定の状況として識別すべき観点があるかを判断させることを目的とし，発言内容，期待する回答の形式を指示した内容をプロンプトとして生成する．

以下に，本研究で使用したプロンプトの具体例を示す．

使用したプロンプトの例

```
あなたは優秀なコードの分析者です
## タスク
- 以下の発言から管理者に圧力を与える発言があるか判断しなさい
- 以下の発言から執拗な管理者への昇格要求があるか判断しなさい
- 以下の発言からセキュリティの問題があるか判断しなさい
## 制約
- 期待する回答の形式に沿って JSON 形式で出力しなさい
- 期待する回答の判断結果は選択肢から選択しなさい
- 期待する回答の判断理由は評価した理由を日本語で出力しなさい
- 期待する回答の具体例は判断の際に参考にした発言を出力しなさい
## 発言内容
  { 発言の時系列データ }
## 期待する回答
  { 期待する回答例 }
```

LLM に上記のプロンプトを入力し，分析結果を得る．

ステップ 2：検知 図 1 で紹介したように，インサイダー攻撃の攻撃には三つの進行ステージがあると考え，ステージ A, B, C と定義する．ステージ A は管理者に精神的圧力を与える段階，ステージ B は執拗に管理者への昇格要求を行う段階，ステージ C はセキュリティの問題が発生している段階，を指している．インサイダー攻撃の検知ではこのステージの進行があるかを検知する．発言から状況がステージ A → B → C の順に進んでいる場合に検知する．

3.2 プロテストウェアリスク検知部

ステップ 1：プロンプト生成 プロテストウェアの発生はソフトウェア単位ではなく，人によって引き起こされると考え，開発者単位で検知を行う．検査対象の開発者個人の発言から，インサイダー攻撃検知リスク部と同様に必要のないエラーログ等の部分を除去し，一定期間ごとに，対象開発者個人の発言から特定の状況として識別すべき観点があるかを判断させることを目的とし，発言内容，期待する回答の形式を指示した内容をプロンプトとして生成する．プロンプトのタスクとして指定する状況識別観点は以下の通りである．

- 以下の発言から金銭的な問題があるか判断しなさい
- 以下の発言から政治的な要求を行う内容があるか判断しなさい

ステップ 2：検知 図 2 で紹介したように，プロテストウェアはインサイダー攻撃と違い，すでに権限を持っている開発者自身が悪意あるコードを埋め込むことによって生まれるため，ステージ進行がないと考える．そのため，各観点において発言が複数発生している場合にプロテストウェアリスクありの開発者として検知する．実験では一定期間の発言が 2 回以上ある場合に検知するよう設定した．

4. 評価

提案手法を評価するため，インサイダー攻撃およびプロテストウェアに関する事例を元にデータセットを作成した．データセットを用いて，提案手法の検知精度および処理性能を調査した．実験では LLM として，OpenAI 社の gpt-4o および gpt-4o-mini を駆使して，発言に基づく状況識別を行った．

4.1 データセット

データセットに使用したデータ数を表 1 に示す．インサイダー攻撃に関してはプロジェクト単位で検知を行うため，プロジェクト数が単位となる．プロテストウェアに関してはプロジェクト単位ではなく開発者単位で検知を行うため，人数が単位となる．

悪性データについては過去事例に関する Web 記事からソフトウェア名を特定し，そのソフトウェアに関わる開発者の発言を収集した．発言の収集にはソフトウェア開発レポジトリサービスである GitHub における開発イベントを継続的に収集・保管するサービス GH Archive [8] を活用した．インサイダー攻撃における事例については GitHub 上に特徴となる痕跡がないため，別途，開発者らが連絡を行

表 1: データセットの概要
Table 1 Overview of the dataset

ステージ	項目	件数
インサイダー攻撃	悪性	2
	良性	7,299
プロテストウェア	悪性	14
	良性	2,957

表 2: 検知精度
Table 2 Detection accuracy

リスク	項目	精度	件数
インサイダー攻撃	検知率	50.0%	1/2
	誤検知率	0.07%	5/7,299
プロテストウェア	検知率	50.0%	7/14
	誤検知率	1.66%	49/2,957

うメーリングリストおよび GitHub とは別の開発レポジトリサービスから収集した．

良性データについては有名なソフトウェアを対象として，それに関わる開発者の発言を収集した．OSS プロジェクトの人気を表すスター数の上位 10,000 プロジェクトを収集対象とした．これらの OSS プロジェクトにおいて，すでにインサイダー攻撃の発生やプロテストウェアとなっている可能性もあるが，有名なソフトウェアであるため，その可能性は極めて低いと考える．OSS プロジェクトの中には，学習用のコンテンツやサンプルコードといったソフトウェアでないプロジェクトも存在するため，ソフトウェアの説明からソフトウェアでないプロジェクトを排除した．また，実験にかかるコスト，および精度を勘案して，発言数が多すぎる場合 (10,000 件以上) と少なすぎる場合 (100 件以下) は除外した．以上の条件から，インサイダー攻撃リスクの評価に関しては 7,299 件のプロジェクトに関わる 2,137,900 名の開発者の発言 17,714,044 件を良性データとして使用した．プロテストウェアリスクの評価に関しては 2,957 名の開発者の発言 3,352,778 件を良性データとして使用した．また実験的にかかるコストや処理時間の面から一つの発言の文字数の上限を 1024 文字とした．

4.2 検知精度

表 2 に提案手法の検知精度を示す．インサイダー攻撃に関しては検知率 50.0%，誤検知率 0.07%，プロテストウェアに関しては検知率 50.0%，誤検知率 1.66%の精度であった．誤検知率は比較的低いが，検知率は高くなく，今後の改良が必要である．検知漏れ，誤検知の理由については 5 章で議論する．

4.3 処理性能

図 4，図 5 にインサイダー攻撃リスク検知における処理時間を示す．図 4 の横軸がプロジェクトに関わる複数の開発者全体の発言数であり，図 5 の横軸がある開発者の発言数である．縦軸はプロンプトを LLM に入力する処理から検知結果が出力されるまでの処理時間である．どちらの検知部においても発言数と処理時間が比例する関係であり，10,000 発言を処理するのに概ね 1 から 5 分程度を要する．処理にかかる時間のほとんどは LLM による状況判断にかかる時間であった．

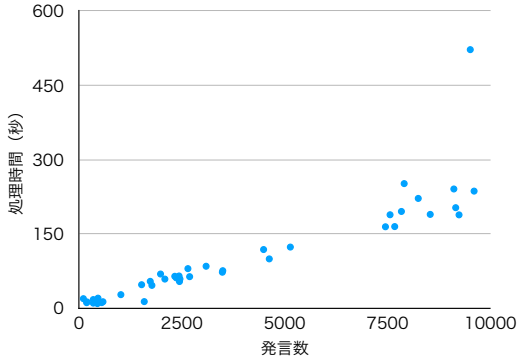

図 4: インサイダー攻撃リスク検知における処理時間
Fig. 4 Processing time of insider attack risk detection

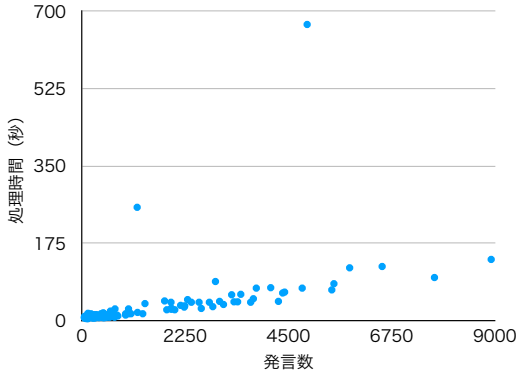

図 5: プロテストウェアリスク検知における処理時間
Fig. 5 Processing time of protestware risk detection

5. 考察

5.1 検知漏れに関する分析

インサイダー攻撃 インサイダー攻撃に対しては検知漏れが 1 件発生した．表 3 に事例における提案手法で識別した状況を示す．検知できなかった理由は攻撃者が権限昇格を狙わずに脆弱性あるコードを混入しようとしたからである．そのため，ステージ B に相当する状況が発生せず，提案手法では検知できなかった．

プロテストウェア プロテストウェアに対しては検知漏れが 8 件発生した．著者らよる発言の確認の結果，4 件についてはソフトウェア開発に関する不満に関する発言は見受けられるが，金銭的や政治的な抗議に関する発言は確認できなかった．3 件については，一般的な不満を含め，抗議を示す発言を確認できなかった．プロテストウェアに関わ

表 3: 検知漏れ事例における状況識別結果
Table 3 Situation identification results in cases of missed detection

日付	ステージ	観点	識別理由 (発言の抜粋)
2020-07-06	A	緊急性の認識	会話の中で，ある開発者が何度も早急にマージすることを求めている発言が見られる． (I realized that was a big essay of No!! Not trying to put down ideas, more just want to get this fix out so we go from not usable at all to at least somewhat usable lol. can we get this merged? Looks like we've discussed/fixed everything we can, at least in the scope of this change.)
2020-07-21	A	精神的圧力を与える発言	ある開発者が何度も早急にマージすることを求めており，貢献が歓迎されないと感じていることを示している． (I'm not coming back to this project until I see that contributions made in good faith are welcomed instead of fought every step of the way.)
2020-10-26	A	相手への強い非難	ある開発者が強い不満を表明している． (Nah man, I'm tired of this. You've been moving the goalposts for months - first it was keep the alphabetical sort, then it was keep the sort-by-time, now it's the UI is a regression.)
2021-02-10	C	セキュリティ問題	会話の後半で SQL インジェクションのリスクについて言及されている．(And actually the SQL Injection here was a security risk since it is possible for apps to send searches via an Intent, e.g. ...)

る開発者が著名な場合，発言も多く，得られる情報も多いが，そうでない開発者は自身を表す発言が少ないため，どういった問題に直面しているかや，考え方を把握できない傾向にある．1 件については，政治的抗議に関する発言を識別していたものの，継続的ではない，つまり分析期間中に一回しか発生していないことから検知することができなかった．誤検知とのトレードオフになるため，今後は閾値調整が必要になると考える．

5.2 誤検知に関する分析

インサイダー攻撃 5 件の誤検知についてその理由を調査した．多くの検知は状況識別については妥当であるものの，ステージの進行が関連するものなのか不明なものが多かった．表 4 に誤検知したある事例の状況識別結果を示す．管理者の追加を要求するが，本当に攻撃者が意図した開発者が管理者として追加されているか，あるいは，追加された管理者によってセキュリティに関わる問題が引き起こされているかは発言からだけでは不明確で，全く関連していない事象がインサイダー攻撃に関わる事象として関連づけられているため，誤って検知してしまったと考える．

プロテストウェア 49 件の誤検知についてその理由を調査した．図 6 にその内訳を示す．政治的批判を行っている，あるいは金銭的な問題を抱えている，として検知されている割合はおよそ同等であった．政治的批判の理由として，国家や政権，企業に対する批判が多かった．金銭的な問題の理由として，開発に資金援助を求めることや，より具体的にサーバや API 等を維持することができないことに言及することが多かった．また，わずかではあるが，本研究で意図する発言者が直面している金銭的な問題ではなく，サービスの有料版と無料版の違い，有料サービスが受けられていない，有償ライセンスに関する問題など，クリティカルではない金銭に関わる問題が発生し，これを特徴として検知してしまっていた．

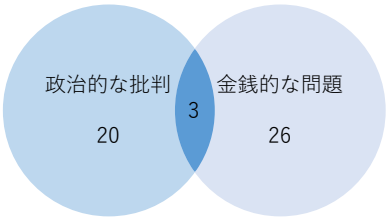

図 6: プロテストウェアリスクに対する誤検知理由の内訳
Fig. 6 Breakdown of false positive reasons for protestware risk

5.3 事前検知の可能性

インサイダー攻撃 検知した 1 件の事例について検知した日付と悪性コードが混入した日付を比較した結果，悪性コード混入前に検知には至らなかったが，事件が発覚する日付よりも前に検知できていることを確認した．図 5 に検知した際の状況識別結果を示す．提案手法ではステージ C の状況を経て検知となるため，検知した日付はステージ C イベントの発生日である 2024 年 2 月 26 日となる．このインサイダー攻撃の事例では悪性コードが混入された日付は 2024 年 2 月 24 日頃であり，混入する日付より以前には検知できていなかった．しかし，本攻撃が発覚したのは 2024 年 3 月 29 日であり，提案手法はそれよりも一ヶ月ほど以前に検知できていたため，より被害の最小化に貢献できるのではないかと考える．

プロテストウェア 検知した 7 名の開発者について検知した日付と抗議に関わるコードが混入した日付を比較し，3 名の開発者については事前に検知できていた．1 名については事件発生の約 1 年前のタイミングで検知しており，2

表 4: 誤検知事例における状況識別結果
Table 4 Situation identification results in false positive cases

日付	ステージ	観点	識別理由 (発言の抜粋)
2021-02-23	A	精神的圧力を与える発言	多くのユーザーが問題の解決を求めており，特定のエラーに対する不満が多く，開発者に対する圧力がかかっている (CRON reached the end of time, judgement day. how is this possible?)
2021-11-17	B	管理者の交代要求	プロジェクトのメンテナンスが不十分であることに対する不満があり，管理者の交代を求める声がある (You should add a hint to the main page that this project is currently unmainted. ... I'm happy to add other maintainers to the project. Clearly I haven't had enough time to maintain this project with everything else going on.)
2023-01-06	C	セキュリティの問題	脆弱性に関する具体的な指摘があり，これがプロジェクトにとって重要な問題であることが示されている (It would appear that a vulnerability has been identified in the version of luxon used in this project)

表 5: インサイダー攻撃事例における状況識別結果
Table 5 Situation identification results in insider attack case

日付	ステージ	観点	識別理由 (発言の抜粋)
2022-06-14	A	精神的圧力を与える発言	いくつかの発言から，特定のリリースや修正に対する圧力が感じられる．(With your current rate, I very doubt to see 5.4.0 release this year.)
2022-06-22	B	管理者の交代要求	いくつかの発言から，管理者の交代を求める声がある．(Why not pass on maintainership for XZ for C so you can give XZ for Java more attention?)
2024-02-26	C	セキュリティの問題	サンドボックス機能を無効化する方法が提案されている．(Hello! Thanks for the bug report and build logs. Luckily, the workaround for this is very simple. With Autotools build, you can pass the flag '–disable-sandbox' to 'configure'.)

名については事件発生の約数日前のタイミングで検知していた．検知から事件発生の期間にばらつきがあることから，提案手法が検知しても，対処にどのぐらいの猶予があるか不明のため，対処までの見立てをつけることが難しいことが今後の課題となる．

事前に検知できなかった 4 名の開発者については，事件発生後に政治的要求に関する発言が初めて発生しており，抗議に関わるコードを混入後，その意図を示す開発者が一定数いることが示唆される．そのため，発言からこういった開発者の意図を事前に把握することは困難である．

6. 関連研究

ソフトウェアに混入する悪性コードの検知 Duan ら [6] はソフトウェアのソースコードに対してメタデータ，静的解析，動的解析を用いて，パッケージレジストリの悪用を調査する手法を提案した．PyPI，Npm，RubyGems から 100 万以上のパッケージを分析し，339 個の新しい悪意のあるパッケージを発見した．

鐘本ら [9] は LLM を用いて悪性コードの検知にかかる細かい設定や検知ルールの維持にかかる負担を低減可能かを調査した．誤検知が多いことがまだ課題ではあるが，人間では発見できなかった事例や，悪性コードが難読化されていても解読して分析可能な点が示唆されている．

これらの研究は悪性コードの特徴に基づく検知であることから，検知のタイミングが必ず混入後となる．本研究は悪性コードの混入前に今後悪性コードを混入する可能性が高いタイミングを検知することを目的としている点で異なる研究と考える．

ソフトウェア開発プロセスに基づくリスク評価 OpenSSF scorecard ではセキュリティ観点のスコアに基づき OSS を評価し，OSS プロジェクトに対して改善提案を行っている [10]．OSS をセキュリティ観点のメトリクスを計測してスコアを算出し，スコアの高さがセキュリティの信頼性を示している．OpenSSF で扱う OSS の開発状況だけでなく，過去発生した脆弱性の状況も加味してセキュリティリスクを提示する手法も提案されている [11]．CHAOSS では OSS プロジェクトの健全度を定量的に測定することを目的としている．コード量やコミット回数のような典型的なメトリクスだけでなく，エレファントファクタのような開発コミュニティにどれだけ多様性があるかを評価する実験的な指標も存在する [12]．

これらの研究はソフトウェアの何か特徴あるリスクを指摘するのではなく，全体的に見てリスクの高さを判断している．本研究ではインサイダー攻撃やプロテストウェアと

いった具体的なリスクであることがわかるため，リスクを知りたい利用者にとってより対策がしやすい情報になると考えている．

ソフトウェア開発レポジトリのコミットに着目して，異常と思われるコミットを検知することでソースコードに悪意ある機能の混入等を検知する手法が存在する [13]．Ganz ら [14] はプログラムコードで隠された機能を探す代わりに，ソフトウェアの開発過程を分析し，そのバージョン履歴において悪意のある活動の手がかりを見つけることを提案している．

これらの研究ではソフトウェア開発の活動における異常性を捉えて，悪性コードの検知を目指している．しかし，異常として特徴が現れるのは悪性コードの混入時になると考えるため，事後の検知となってしまうことが予想される．本研究では開発者の発言を用いて事前に検知することを目的としているため，研究の意図が異なると考える．

ソフトウェア開発の継続性に関する評価 セキュリティリスクではない OSS コミュニティの継続性に関する研究も存在する．OSS 開発者が開発に興味を持っていることを感情分析から判断し，開発の継続性を予測する手法が提案されている [15]．Jack らは価値観に関連する議論が，OSS 開発者の離脱に与える影響を明らかにするための調査を行なった [16]．52 の OSS プロジェクトにおける価値観に関連する議論の出現頻度と，開発者の離脱率との関連を調査した．結果，敬意に関する議論が貢献者の離脱を引き起こす可能性があるため，OSS の管理者は，このような議論に対して注意を払い，適切に対処する必要があること，また，自由や社会的権力に関連する価値観の議論は，プロジェクトの主要な目標と技術作業に結びつけることで，持続的な貢献を促進する可能性があることがわかりました．

これらの研究では，本研究と同様に開発者に着目したアプローチであるが，OSS の継続性や開発者の継続性を予測するという点は本研究と異なる目的である．

7. おわりに

悪意ある OSS 開発者に起因するセキュリティリスクに対して，本研究は，これらのリスクを事前に検出する手法を検討した．インサイダー攻撃では，攻撃者が OSS 開発者に圧力をかけ，管理権限を奪取し悪意あるコードを埋め込む事象を特徴として捉え，プロテストウェアは開発者の金銭的問題あるいは政治的主張の発言を特徴とし，LLM を活用して開発者の発言からセキュリティリスクを事前に検出する手法を提案した．実験結果からインサイダー攻撃に関しては検知率 50.0%，誤検知率 0.07%，プロテストウェアに関しては検知率 50.0%，誤検知率 1.66%の精度で検知可能であること，またインサイダー攻撃に関しては事件が発覚する前に，プロテストウェアに関しては悪性コードが混入する前に検知可能であることを確認した．

ソフトウェア利用者は本研究の成果を活用することで，ソフトウェアを利用する前に，そのソフトウェアに潜むリスクをより把握できることにあり，セキュリティリスクが高いと疑われるソフトウェアの利用を事前に避けることができるメリットがある．

今後の課題は，検知精度の向上であり，事象間の関連性を加味した検知アルゴリズムや，SNS 等を含めた発言やソフトウェア開発における行動も含めた情報源の拡大を予定している．

参考文献

[1] Burkacky Ondrej, Johannes Deichmann, Stefan Frank, Dominik Hepp, and Rocha Andre. When code is king: Mastering automotive software excellence. *McKinsey Global Publishing*, 2021.

[2] SOLVING YOUR OPEN SOURCE RISK WITH SOURCECLEAR. *VERACODE*, 2018.

[3] Hossein Siadati, Terrence Brent Hernandez, and Elijah Lorenzo Tripp. DevPhish : Exploring Social Engineering in Software Supply Chain Attacks on Developers. *arXiv*, 2024.

[4] Russ Cox. Timeline of the xz open source attack. *https://research.swtch.com/xz-timeline*, 2024.

[5] Liran Tal and Assaf Ben Josef. Open source maintainer pulls the plug on npm packages colors and faker, now what ? *Snyk Blog*, 2022.

[6] Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages. In *NDSS*, 2021.

[7] Liran Tal. Alert : peacenotwar module sabotages npm developers in the node-ipc package to protest the invasion of Ukraine. *Snyk Blog*, 2022.

[8] GH Archive. *https://www.gharchive.org*, 2024.

[9] 鐘本楊, 荒川玲佳, 秋山満昭. LLM を用いたソースコードのリスク検知手法の検討. 信学技報, 2024.

[10] OpenSSF. OpenSSF Scorecard - Security health metrics for Open Source. *https://github.com/ossf/scorecard*, 2024.

[11] 葛野弘樹, 矢野智彦, 山内利宏. オープンソースソフトウェアに対するセキュリティリスク指標の提案と評価. コンピュータセキュリティシンポジウム, 2022.

[12] CHAOSS. Metrics and Metrics Models. *https://chaoss.community/kb-metrics-and-metrics-models/*, 2024.

[13] Danielle Gonzalez, Thomas Zimmermann, Patrice Godefroid, and Max Schafer. Anomalicious : Automated Detection of Anomal ous and Potentially Mal icious Commits on GitHub. In *ICSE-SEIP*, 2021.

[14] Tom Ganz, Inaam Ashraf, Martin Härterich, and Konrad Rieck. Detecting Backdoors in Collaboration Graphs of Software Repositories. In *CODASPY*. Association for Computing Machinery, 2023.

[15] 山下一寛, 亀井靖高, 鵜林尚靖. 感情分析による OSS プロジェクト中断の予測に向けた調査. In *SES*, 2016.

[16] Jack Jamieson and Naomi Yamashita. Predicting open source contributor turnover from value-related discussions : An analysis of GitHub issues. In *ICSE*. Association for Computing Machinery, 2024.