IPA

2022 年度 未踏 IT 人材発掘・育成事業 採択案件評価書

1. 担当 PM

竹迫 良範（株式会社リクルート　データプロダクトユニット　ユニット長）

2. クリエータ氏名

赤松 宏紀（大阪大学大学院情報科学研究科 博士前期課程）
大迫 勇太郎（大阪大学大学院情報科学研究科 博士前期課程）

3. 委託金支払額

2,736,000 円

4. テーマ名

内部処理分析を基にした Web アプリケーションのセキュリティ SaaS の開発

5. 関連 Web サイト

- Phrude ティザーサイト：https://phrude.com/
- Phrude ドキュメントページ：https://phrude.com/docs/intro
- Phrude デモサイト：https://demo.phrude.com/
- Phrudeの検証で用いたPython Flask製のデモ用Web アプリ：https://github.com/phrude/flask-demoapp

6. テーマ概要

近年、インターネットを介した多岐にわたるサービスが広く提供されるようになり、その中でもサービス提供にあたって媒体として Web アプリケーションが広く用いられている。しかし、実装上の不備や実装に際し、利用しているライブラリの脆弱性などにより、Web アプリケーションにおいは様々なセキュリティリスクが存在している。本プロジェクトでは、Web アプリケーションに対する攻撃を検知する方法として内部処理分析を用いる SaaS システム「Phrude」を開発した。Phrude の特徴は、プロファイラによって Web アプリケーションの内部で実行された処理を記録し攻撃に特有の処理を発見できることにある。Phrude では稼働している Web アプリケーションに内部で実行された処理を記録するためのプラグインであるプロファイラを導入し、そこから得た内部処理

情報から攻撃に特有の処理を発見することで、既存のセキュリティ対策ツールでは検知及び対策ができないセキュリティ侵害を検知する。

8. 開発目標

Web アプリケーションに対する攻撃事例は様々なものがあるが、Phrude ではインジェクションかつ任意の処理を挿入する攻撃の検知を目指した。任意の処理を挿入する攻撃では、実行されている Web アプリケーションと同等の権限による任意の操作が可能であるため、

- Web アプリケーションが扱う任意の情報（個人情報、他のコンピューティングリソースへのアクセスキー、秘匿しているパラメータ等）の流出
- Web アプリケーションが実行されている環境そのものの計算資源を使用した暗号通貨のマイニング
- アクセスが許可されている他の計算資源への攻撃の展開
- Web アプリケーション自体の書き換えによって利用者に情報を入力するような誘導

などの悪用が行われる可能性があり、インジェクションに分類される攻撃事例の中でも任意の処理を挿入する攻撃の悪用による影響は特に大きいため、これらの攻撃の検知を目指すことを開発目標にした。

7. 採択理由

Web アプリケーションのセキュア開発現場では、開発プロセスの中で CI と連携してデプロイ前にコードの静的解析や自動の脆弱性検査を実施したりするが、本番環境の攻撃データを元に自動で事前検査するようなツールはまだ普及していない。

本提案では、本番環境にプロファイラを仕込んで、外部からの攻撃を受けたときのプログラムの関数呼び出しの履歴情報を元に異常を分析し、WAF の攻撃検知に応用している。そしてそれらのデータを CI に組み込む自動検査用の Fuzzing ツールに還元することによって、Fuzzing の精度向上と時間短縮が期待でき、理想的なセキュア開発サイクルを実現する。提案者が過去に開発した OSS の Fuzzing ツール shfz を一部元にして開発するが、様々なプログラミング言語に対応したプロファイラの開発や、WAF の検知・遮断機能、解析エンジンの精度向上、各種マネジメント機能の統合開発など数多くの新しい開発作業が見込まれる。

本番環境の攻撃データからの分析結果を開発現場のプロセスの中に還元することによって、既存のパターンマッチやルールベースでは通り抜けてしまう攻撃への早期対処ができる安全な未来の実現を期待したい。

9. 進捗概要

内部処理分析という新たな仕組みを用いたセキュリティ SaaS システム Phrude（Profiling History-based Runtime Detection System）を開発した。

(1) Web アプリケーションの内部処理を収集するプロファイラ
(2) ルールに基づく攻撃検知機構
(3) ダッシュボード

の 3 点を含む、Phrude のシステム構成図を図 1 に示す。

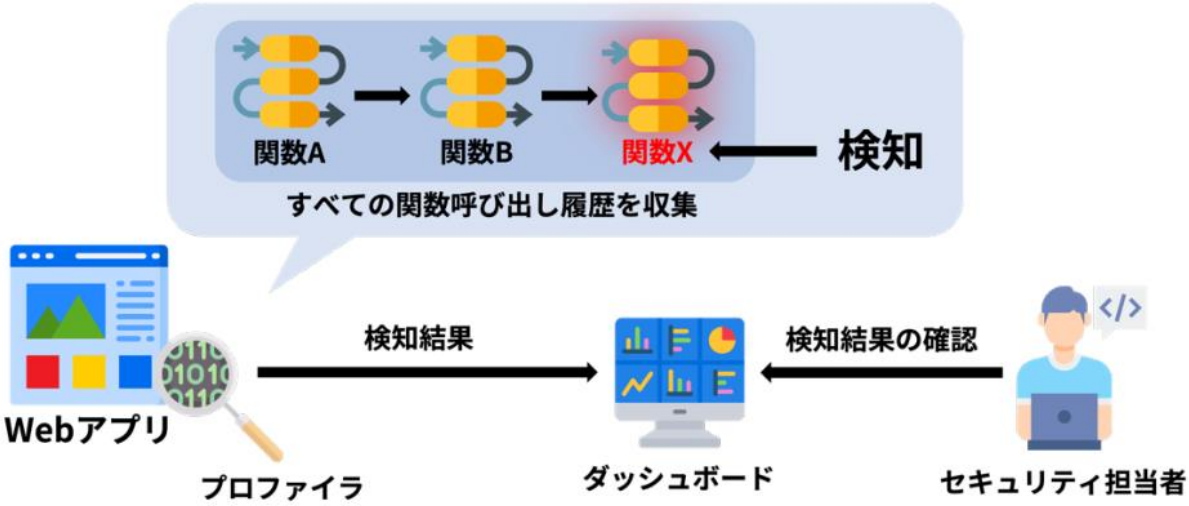

図 1：Phrude のシステムの構成図

Web アプリケーションのセキュリティ担当者がまず Web アプリケーションにプロファイラを導入し内部処理を収集する。ここで収集される内部処理は図 2 で示されているように実行された関数名・関数が含まれるプログラムファイル・関数呼び出しの引数情報・呼び出し元関数をプロファイラからなる関数呼び出し履歴であり、Web アプリケーションに対して処理を挿入する攻撃があった際、攻撃による処理が記録される。そしてプロファイラとともに実装されている攻撃検知機構が動作し攻撃を検知する。プロファイラで収集した内部処理である関数呼び出し履歴に対し、事前に設定した攻撃の処理としてよく用いられる関数が指定されたルールに一致するような処理が実行されたか判定することで攻撃を検知する。

```
...
関数名 :  call
ファイル : /usr/local/lib/python3.10/site-packages/jinja2/runtime.py
引数 :  {
    "args": "('CLOUD_SECRET_KEY',)",
    "kwargs": "{}",
    "_Context__obj": "CLASS:method-wrapper",
    "_Context__self": "CLASS:Context“
}

関数名 :  getenv
ファイル :  /usr/local/lib/python3.10/os.py
引数 :  {
    "key": "CLOUD_SECRET_KEY",
    "default": null
}
...
```

図 2：Phrude が収集する内部処理の例

攻撃が検知された場合、検知した処理の関数名・関数が含まれるプログラムファイル・関数呼び出しの引数情報・呼び出し元関数をダッシュボードに送信し、Web アプリケーションのセキュリティ担当者が確認する。図 3 で示しているように、記述ルールではパラメータ name はルール名、パラメータ func は検知する関数名、パラメータ file は検知する関数が含まれるプログラムファイル名となっている。次のパラメータ level は検知の重要度を示し、指定できる値は info、warn、alert の 3 つの段階となっている。最後のパラメータ value では、特定の値を指定することで検知結果の中でそれを取り上げて表示することができる。

図 3 左の例では取り上げる値の名称 key に対して、関数の引数を示す $func_args の中で getenv 関数において取得する環境変数名が格納されている key パラメータを指定している。図 3 右の検知結果では指定されたルール名が表示されるとともに、パラメータ value で指定した getenv 関数において取得する環境変数名が格納されている key パラメータの値である CLOUD_SECRET_KEY が取り上げて表示されていることがわかる。

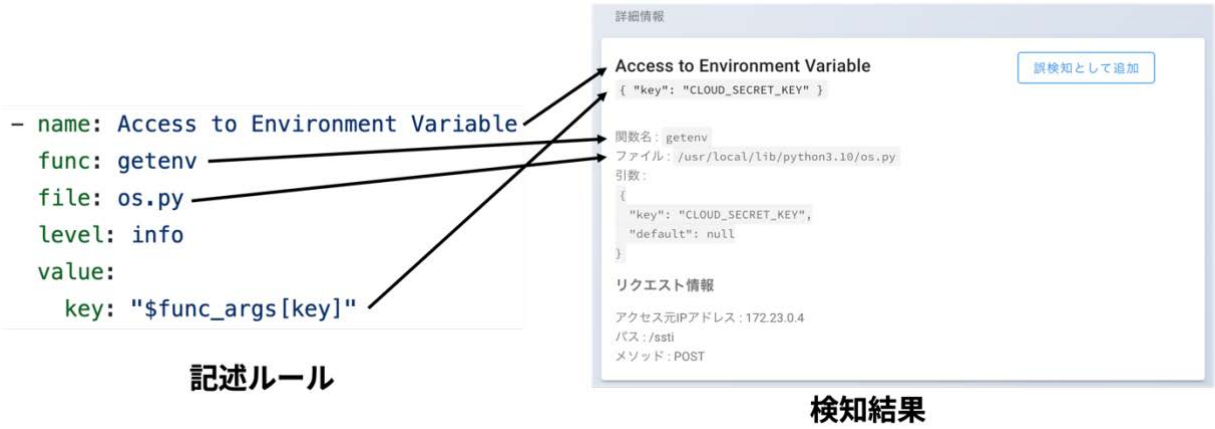

図 3：Phrude の記述ルールを検知結果の対応例

10. プロジェクト評価

従来の攻撃検知手法として、WAF やシステムコールによる攻撃検知が挙げられる。これらの攻撃検知手法と今回開発した Phrude に対して、検知が得意な攻撃手法の比較を表 1 に示す。

表 1：既存攻撃検知手法と Phrude が検知を得意とする攻撃手法の比較

攻撃手法	WAF	システムコールによる攻撃検知	Phrudeによる攻撃検知	Phrudeによる原因追跡
Cross Site Scripting	○	×	×	×
SQL Injection	○	×	×	×
OSSのサプライチェーン攻撃	×	△	○	○
任意の処理を挿入するWAFの検知を回避した/未知の攻撃	×	△	○	○

WAF は、特定の文字列や文字パターンを見つけることで検知をしているため、既知の典型的な攻撃に対しては効果を発揮するが、検知されうる特定の文字列や文字パターンを無くす細工をした HTTP リクエストによる攻撃については検知ができない。また、システムコールによる攻撃検知は新規プロセスの作成などのシステムコールを発行する処理からなる攻撃を検知できるものの、環境変数の取得などのシステムコールを発行しない攻撃に対しては検知ができない。Phrude では、これらの攻撃検知技術では防げない攻撃を内部処理の分析によって検知することができ、相補的な関係にあると言える。Phrude の導入

により、既存のセキュリティ対策ツールでは検知不可能な Web アプリケーションに対する攻撃を新たに検知できるようになったと評価できる。

11. 今後の課題

プロファイラによる内部処理の取得と攻撃検知における処理の負荷が高いことが課題である。現在の実装では同量のリクエストに対して 10〜20 倍程度の処理能力の割り当てが必要となっており、実サービスへの運用導入での一つの大きな障壁となっている。現在の実装では、内部処理を取得する処理に Python の標準の機能の settrace API を利用していることが大きく影響しているが、今後 Python 言語処理系自体に内部処理の取得と攻撃検知における処理を組み込むことによって大幅な高速化を実現することを期待したい。それが実現できれば商用レベルでの活用も見込まれる。